16 +

XXVIII Всероссийская (с международным участием) молодежная научно-практическая конференция Нижневартовского государственного университета

г. Нижневартовск, 8–9 апреля 2026 г.

Часть 2. Техника и технологии

г. Нижневартовск
2026

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Нижневартовский государственный университет»

XXVIII Всероссийская (с международным участием) молодежная научно-практическая конференция Нижневартовского государственного университета

Часть 2

Техника и технологии

Нижневартовск, 8–9 апреля 2026 года

Под общей ред. А.А. Кулагина

Нижневартовск
НВГУ
2026

ББК 72я43 **16+**
Д 25

Печатается по решению Ученого совета
ФГБОУ ВО «Нижневартовский государственный университет»

Под общей редакцией: Кулагин Андрей Алексеевич, доктор биологических наук, профессор.

Редакторы: А.М. Яковлева, И.С. Анцева, Д.В. Вилявин.

Д 25 **XXVIII Всероссийская (с международным участием) молодежная научно-практическая конференция Нижневартовского государственного университета (г. Нижневартовск, 8–9 апреля 2026 г.)** / Под общей ред. А.А. Кулагина. Ч. 2. Техника и технологии. Нижневартовск: Издательство НВГУ, 2026. 297 с.

ISBN 978-5-00047-750-2

В сборник материалов включены статьи участников XXVIII Всероссийской (с международным участием) молодежной научно-практической конференции Нижневартовского государственного университета, прошедшие рецензирование.

Издание адресовано специалистам-практикам, педагогическим работникам, научным сотрудникам, аспирантам, магистрантам и студентам.

ББК 72я43

Подписано в печать 26.05.2026
Формат 60×84/8
Гарнитура Times New Roman. Усл. печ. листов 15,18
Электронное издание. Объем 10,02 МБ. Заказ 2367

Издательство НВГУ
628615, Тюменская область, г. Нижневартовск, ул. Маршала Жукова, 4
Тел./факс: (3466) 24-50-51, E-mail: red@nvsu.ru, izdatelstvo@nggu.ru

СОДЕРЖАНИЕ

ПРОГРАМИРОВАНИЕ. WEB-ТЕХНОЛОГИИ. СЕТИ

ИСКУСТВЕННЫЙ ИНТЕЛЛЕКТ. РОБОТОТЕХНИКА. БЕСПИЛОТНЫЕ ЛЕТАТЕЛЬНЫЕ АППАРАТЫ. ЗАЩИТА ИНФОРМАЦИИ

ПРОГРАМИРОВАНИЕ. WEB-ТЕХНОЛОГИИ. СЕТИ

Абдусалямова М.В.
Нижневартовский государственный университет
г. Нижневартовск, Россия

РАЗРАБОТКА ANDROID-ПРИЛОЖЕНИЯ ДЛЯ АВТОМАТИЗАЦИИ УХОДА ЗА КОМНАТНЫМИ РАСТЕНИЯМИ

Еще недавно образ дачника был неразрывно связан с усталостью, мозолями от тяпки и обязательными банками с солеными огурцами. Слово «сад» в сознании многих до сих пор рисует портрет бабушки или мамы, с любовью копающихся в грядках. Но сегодня все изменилось. Современное поколение, которых мы привыкли видеть с телефонами в руках, вдруг начали массово выкладывать в социальные сети фотографии микрозелени на подоконнике. Данное увлечение получило название «гарденинг», и за этим стоит не просто мода, а целая философия.

Столкнувшись с «цифровым перегрузом» и перенасыщением искусственным контентом в социальных медиа, молодое поколение демонстрирует интерес к чему-то настоящему, осязаемому и живому. Это своего рода «детокс» от виртуальной реальности, попытка обрести спокойствие и контроль над чем-то простым и прекрасным в хаотичном мире. В современных квартирах комнатные растения перестали быть просто украшением подоконников. Они стали самым доступным и эффектным способом преобразить пространство, расставить акценты и добавить жизни в минималистичный интерьер. При этом они выполняют не только эстетическую функцию: они улучшают микроклимат квартиры, очищая воздух от токсинов и увлажняя его, что особенно ценно в условиях плохой экологии крупных городов.

Но здесь возникает главная проблема. Поколение Z – поколение быстрых решений и коротких видео. Выращивание растений, наоборот, требует усидчивости, регулярности и внимания к деталям. Кто-то просто забывает полить цветы в суматохе будней. Другие, начитавшись советов из интернета, заливают их водой. А кто-то только хочет заниматься садоводством, но не понимает с чего ему начать. Информации в сети огромное количество, но она разрозненная, противоречивая, и становится непонятно, кому верить. И это проблема не только новичков: даже опытные садоводы могут запутаться в нюансах ухода за новым видом растения. Например, при покупке редкого экземпляра из другой климатической зоны может потребоваться специфический режим влажности или особый состав почвы, о котором никто не знает. Человеческий фактор очень коварен – мы можем искренне любить свои растения, желать им добра, но в итоге красивая идея разбивается о суровую реальность, и растения гибнут.

Очевидно, что нужен инструмент, который облегчит уход за «зелеными питомцами» и структурирует знания. В современном мире лучший формат такого помощника – мобильное приложение. Оно всегда под рукой, способно напомнить о поливе и подкормке и дать проверенный совет именно тогда, когда он нужен. Цель данной статьи – разобраться, каким

должно быть идеальное приложение для садоводов, чтобы помочь растениям выжить, а их владельцам – не потерять радость от общения с природой.

Первое над чем стоит задуматься при создании приложения – это его пользовательский интерфейс, то есть то, с чем будет взаимодействовать пользователь на своем экране – кнопки, текст и изображения. Всегда нужно делать интерфейс интуитивно понятным и красочным, чтобы он мог с первого взгляда удерживать внимание пользователя и заставлять его изучать приложение до самого конца [1]. Логичнее всего сделать главным экраном сам виртуальный «сад». Его задача – удобно и наглядно показывать все растения пользователя. Все зеленые «питомцы» будут располагаться в виде аккуратной сетки.

Чтобы добавить новое растение, пользователю достаточно нажать на кнопку «плюс» и внести базовые данные: вид растения и его фото. Позже можно будет заполнить и дополнительные сведения – даты последнего полива и подкормки. Система не даст забыть о важных делах: если цветку требуется вода или удобрение, рядом с его названием сразу появится небольшой значок-напоминание. Это позволяет видеть текущие задачи буквально на одном экране (рис. 1). Помимо этого также будут направлены уведомления на смартфон пользователя, что позволит не забывать о приложении, даже когда оно выключено.

Если пользователь захочет подробнее изучить состояние конкретного растения, нужно просто нажать на его карточку. Откроется экран с подробной информацией. Там будет видно, сколько дней растение уже растет и на какой стадии развития находится. Для планирования ухода предусмотрен небольшой встроенный календарь – с его помощью удобно отмечать прошедшие события и ставить задачи на будущее (рис. 2).

Рис. 1. «Мой сад»

Рис. 2. Карточка растения

Чтобы превратить рутинный уход за растениями в увлекательный процесс, будет добавлена игровая механика. Суть её будет заключаться в том, что за своевременный уход (полив и подкормку в нужные сроки) растение переходит на новую стадию роста. Это будет

выражаться визуально – семечко превращается в росток, росток превращается в маленький кустик, кустик начинает цвести, а цветок со временем увеличивается в размерах или меняет окраску. Игровая форма здесь занимает ключевую роль в удержании внимания пользователя. В психологии это называется эффектом прогресса и немедленным вознаграждением. Когда человек видит, что его регулярные усилия (в данном случае полив) приводят к конкретному визуальному результату – растение выросло, мозг получает порцию дофамина – гормона удовольствия.

Без применения данной механики приложение было бы обычным списком задач: «полей кактус», «удобри фикус». Отсутствие игровых элементов приведет к снижению вовлеченности, и пользователь утратит интерес к приложению. Но когда уход за цветами приобретает цель, например, вырастить как можно больше растений до максимального уровня, а процесс, где каждый вовремя сделанный шаг приближает человека к некоторому «выигрышу», превращается в игру, то даже взрослый человек захочет заняться садоводством. Такой подход, при котором неигровые элементы превращаются в игровую механику, называется геймификацией. Это не про создание полноценной игры, а про умелое использование её элементов для повышения мотивации и удержания внимания пользователя [5].

Однако мало просто создать напоминание – важно удержать интерес пользователя. Помимо продуманной игровой механики, ключевым элементом «вовлечения» становится цветовая гамма приложения. В нашем случае выбор палитры очевидно остановился на оттенках зеленого. Они напрямую ассоциируются с растениями, создавая правильный эмоциональный фон и подсознательно поддерживая желание человека заботиться о своем маленьком саде. В будущем, в качестве акцентных цветов можно использовать приглушенные тона земли (терракотовый, коричневый) или цветы (нежно-желтый, лавандовый), чтобы интерфейс не был монотонным [2].

Помимо интерактивного сада, в приложении будет два важных раздела, которые превратят его в настоящего помощника для цветовода. Они рассчитаны на тех, кто хочет не просто ухаживать, но и лучше понимать свои растения.

Первый раздел – «Энциклопедия растений». Это обширная база знаний, в которой пользователь сможет найти подробное описание любого цветка из своей коллекции или того, которого он только планирует завести. В нем будут не только стандартные данные, но и интересные факты, которые делают общение с природой глубже. Например, будет рассказано о символическом значении растений во флористике: какие цветы считаются символом достатка, какие оберегают дом, а что принято дарить в знак признательности.

Второй раздел – «Советы по уходу». Этот раздел предлагает практические решения для конкретных ситуаций. Здесь будут собраны рекомендации по обустройству места для растений: как подобрать идеальный горшок, какой тип грунта предпочитает тот или иной вид, как организовать подсветку в темное время года. Советы помогут новичку избежать типичных ошибок, а опытному пользователю – открыть для себя новые лайфхаки по уходу. Эти два раздела вместе создают эффект присутствия опытного наставника, который всегда под рукой.

По данным аналитической компании «F+ tech | Марвел», российский рынок смартфонов окончательно закрепляется за Android: его доля достигла 89,6%, тогда как iOS довольствуется лишь 10,4%. (https://clck.ru/3SFLhL) Приведенное соотношение еще раз подтверждает, что платформа Android является наиболее перспективной операционной системой для популяризации приложения.

Главным инструментом для создания приложений на данной ОС остается Android Studio. Эта среда разработки предлагает разработчикам всё необходимое: от умного редактора кода и визуального конструктора интерфейсов до мощного отладчика. При этом, несмотря на профессиональный функционал, среда отличается интуитивно понятным интерфейсом и простотой освоения, а самое важное – она абсолютно бесплатно [3].

Следующий важный этап – выбор языка программирования. Основным языком для разработки под Android является Kotlin. Его выбор обусловлен рядом преимуществ: современный и лаконичный синтаксис упрощает написание и чтение кода, а особенности языка способствуют снижению количества ошибок, что повышает общую эффективность разработки [4].

Что касается хранения информации, оптимальным решением видится использование SQLite. Это стандартная встроенная СУБД для Android, которая позволяет создавать реляционные базы данных для хранения и обработки структурированной информации. Навигация в приложении будет построена таким образом, что из тематических разделов пользователь сможет переходить к просмотру статей. Сами статьи планируется реализовать в формате HTML, интегрируя текст и графические элементы, при этом все файлы будут размещаться непосредственно в базе данных.

Подводя итог вышесказанному, можно сделать вывод, что современное движение и домашнее садоводство – это не временное увлечение и не мода, а отражение глубинной потребности человека в связи с живой природой в условиях технологичного мира. Однако разрыв между желанием иметь пышный сад и реальными навыками по уходу за ним может быть огромным. Предложенная в статье концепция мобильного приложения призвана стать мостом, который будет соединять искреннюю любовь к растениям и знания, необходимые для их процветания.

Именно сочетание продуманного интерфейса, элементов геймификации для поддержания интереса и обширной базы знаний способно превратить рутинную обязанность в увлекательное хобби, которое человек не забросит спустя некоторое время. Разрабатываемое приложение не просто будет напоминать о поливе, а будет обучать и вдохновлять пользователя заниматься полезным хобби, получая чувство удовлетворения от результатов своего труда.

Выбор распространенной платформы и внедрение современных технологий разработки гарантирует надежность и доступность приложения для большого количества пользователей. В конечном счете, создание такого цифрового помощника – это вклад в экологию городской жизни, помогающий молодым людям создавать свой маленький, но настоящий зеленый мир прямо у себя дома.

Литература

1. Агеева А.Д., Петросян Л.Э. Дизайн интерфейсов (UI) и пользовательский опыт (UX) // Вестник науки. 2024. Т. 1, № 6(75). С. 1354-1366.

2. Лавренова Т.В., Филиппенко В.А., Неделько А.Е. О цветовых решениях при проектировании графического интерфейса сайта // Молодой исследователь Дона. 2018. № 3(12). С. 67-71.

3. Ларин С.Э., Белаш В.Ю. Сравнительный анализ инструментов разработки мобильных приложений на Android // Тенденции развития науки и образования. 2024. № 107- 8. С. 168-170.

4. Рахматуллин Т.Г. Kotlin как язык программирования будущего // Инновации и инвестиции. 2022. № 11. С. 313-316.

5. Шергазиева М.С., Ибраева Н.А., Садырбаева А.Б. Геймификация в современном образовании: перспективы и преимущества // Вестник Иссык-Кульского университета. 2024. № 57. С. 250-255.

Бабуров Ф.В., Шабанов В.И.
Московский государственный технический университет гражданской авиации
г. Москва, Россия

ПРИМЕНЕНИЕ ИНФОРМАЦИОННЫХ ПРОДУКТОВ В АВИАТРАНСПОРТНОЙ ОТРАСЛИ

Гражданская авиация – это многогранный и технологически сложный сектор экономики, требующий от своих сотрудников высокой степени ответственности. Авиаперевозки пассажиров и грузов включают в себя ряд технических процессов и операций в рамках системы гражданской авиации, направленных на организацию и осуществление перевозок. Иными словами, воздушное движение, конечный продукт всех видов деятельности, представляет собой не единое целое, а интегрированную систему, состоящую из трех основных организационно-технических компонентов: авиакомпаний, аэропортов и систем управления воздушным движением [8].

Динамичное промышленное развитие невозможно без развития современных технологий. Российский авиационный сектор обладает не только возможностями в области авионики, но и кадровым потенциалом, имеющим опыт в цифровизации и работе со сложными цифровыми технологиями, что позволяет осуществить цифровую трансформацию авиастроения и всей авиационной промышленности [5].

Целью настоящего исследования является анализ использования современных информационных технологий в авиатранспортной отрасли, определение показателей эффективности информационных технологий. Для достижения цели исследования представляется необходимым решить следующие задачи:

- Провести анализ основных, глобальных цифровых технологий, используемых в различных отраслях экономики, с исследованием предоставляемых ими возможностей;
- Определить аспекты эффективности информационной системы;
- Оценить преимущества создания единого цифрового пространства (кластера) внутри отрасли.

Для достижения цели данного исследования использовался метод анализа – метод научного познания, заключающийся в рассмотрении любого объекта по отдельным составляющим. В сфере авиаперевозок данный метод актуален, так как позволяет изучить каждый этап процесса перевозки, и на основе полученных данных оптимизировать весь этот процесс в совокупности.

Цифровые технологии объединяют в едином цифровом пространстве специалистов – конструкторов, как и обслуживающих грузовые и пассажирские авиаперевозки, что позволяет на экосистемах цифровых платформ ввести единые нормы и регламенты, сокращая время по взаимодействию и оформлению документов участников цифровой платформы. Данные технологии предоставляют возможность применять искусственный интеллект, а также проводить мониторинг эксплуатируемых летательных аппаратов. В рамках разрабатываемого национального проекта «Экономика данных и цифровая трансформация государства» перед

авиаотраслью поставлены задачи по обеспечению технологического суверенитета России в данной стратегической сфере, в том числе в области информационной безопасности [2].

В настоящее время выделяют несколько глобальных цифровых технологий, которые применяются в различных отраслях экономики, в том числе и авиатранспортной отрасли.

Рассмотрим применение цифровых технологий в различных технологических процессах авиатранспортной отрасли [3].

1. Управление воздушным движением. Описание: Автоматизированная зависимая система наблюдения и вещания (ADS-B). Преимущества: Наземные авиадиспетчеры и пилоты получают более точную информацию о погоде и навигации и могут отслеживать перемещения воздушных судов. Также может применяться для автоматизированного управления беспилотными летательными аппаратами (БПЛА).

2. Обслуживание клиентов. Технология: 1. Большие данные. Обработка больших массивов данных. 2. Искусственный интеллект (ИИ). Преимущества: 1. Извлечение релевантной информации из больших потоков данных. Использование контекста и семантического значения (семантический поиск). 2. Беспилотные летательные аппараты. Эта технология оцифровывает документацию на борту самолета и преобразует ее в электронную форму, значительно сокращая время подготовки перед взлетом. Быстрый анализ запросов и жалоб пассажиров может повысить качество обслуживания.

3. Уберизация. Технология: Цифровая платформа. Эта технология основана на автоматизированной системе связи между участниками в интегрированной информационной среде. Преимущества: Снижение стоимости авиаперевозок за счет устранения посредников, ответственных за подготовку самолетов и пассажиров к полетам. Благодаря увеличению доходов и улучшению обслуживания пассажиров авиакомпании могут повысить свою прибыль.

4. Персонализация. Технологии: Цифровые технологии и информационные системы на борту и в аэропортах для улучшения качества обслуживания пассажиров. Преимущества: Персонализация услуг на борту: 1) В аэропорту: Бесконтактное предполетное обслуживание с использованием биометрических данных; безопасность, точность и надежность информации, гарантированные нейронными сетями; цифровые паспорта и посадочные талоны; интеграция чат-ботов или голосовых помощников на веб-сайте аэропорта или в мобильном приложении; роботизированные помощники для предоставления дополнительной информации и т. д. 2) На борту: Wi-Fi для полностью цифрового путешествия; цифровые развлекательные системы; чат-боты или голосовые помощники для предоставления питания и других цифровых услуг, что позволяет авиакомпании покрывать свои расходы за счет увеличения доходов. [3].

5. Повышение эффективности обслуживания и мобильности. Технология: цифровые планшеты для использования на борту. Технология персональных электронных устройств управления полетом для руководителей, пилотов и бортпроводников. Преимущества: оптимизирует план полета и информацию управления воздушным движением для авиадиспетчеров, автоматизирует технические и навигационные расчеты. Улучшает обслуживание на борту, информируя всех пассажиров о питании, трансферах и других рейсах.

Заменяет бортовые записи компьютерными записями. Повышает точность и скорость расчета траектории полета и навигации, обеспечивая безопасное и эффективное обслуживание на борту.

6. Охрана здоровья и безопасность. Технология: бесконтактная (NFC) цифровая технология. Давайте поговорим о проверке информации. Преимущества: Безопасность полетов: 1) В аэропорту: бесконтактная оплата, бесконтактная регистрация, автоматизированный доступ в залы ожидания, автоматизированная посадка с использованием биометрических и электронных посадочных талонов, отслеживание багажа, уведомление о задержках и т. д. 2) На борту: обслуживание пассажиров и экипажа, мобильная бесконтактная оплата, выделенный бортпроводник и автоматизированные системы для повышения качества обслуживания и безопасности пассажиров и экипажа.

7. Биометрия, пограничный контроль и багаж. Технологии: Biometric identification of passengers. Технология идентификации пассажиров и багажа на авиатранспорте. Преимущества: Внедрение цифровой идентификации человека по его персональным биологическим данным (оболочка глаза, отпечаток пальца, голос и др.) Применяется для идентификации и обеспечения безопасности. Позволяет пассажиру сдавать багаж без привлечения персонала, а в перспективе и у себя дома. Приводит к ускорению и повышению точности регистрации пассажиров и багажа на рейс. При регистрации появляется возможность получить данные о местонахождении багажа, доступные и самому пассажиру.

8. Повышение эффективности авиационного сектора. Технологии: Цифровые технологии для управления сетевыми и инфраструктурными ресурсами. Эти технологии интегрируют инфраструктурные сети и информационные системы для создания единой цифровой среды обработки данных. Преимущества: Прогнозирование времени прибытия самолетов и количества пассажиров. Это сокращает время наземного обслуживания, увеличивает продолжительность полетов и снижает затраты. Оптимизация систем поддержки аэропортов с учетом пиковых нагрузок. Это повышает прибыльность авиакомпаний.

9. Повышение эффективности ИТ-служб. Технологии: Цифровые технологии. Виртуальные и удаленные технологии для IT-сферы. Преимущества: кибербезопасность и раннее выявление вреда жизни и здоровью персонала и авиапассажиров. Благодаря виртуализации центра обработки данных обеспечивают дополнительную степень защищенности конфиденциальной информации. Сокращают затраты на администрирование, допуская работу удалённо [7].

Таким образом, цифровизация позволяет улучшить качество предоставляемых услуг, снижая при этом производственные издержки. Кроме того, эти направления будут определять отраслевую конкурентоспособность, обеспечивая новые возможности, как на традиционных, так и на перспективных рынках авиационной техники [3].

В настоящее время, в связи с имеющимся санкционным режимом, в отрасли внедряются отечественные информационные разработки, что вызывает следующие трудности.

1. Необходимость модернизации. Тестирование основных ИТ-систем началось в 2019 году, но на тот момент системы не соответствовали всем функциональным требованиям

российских авиакомпаний. Санкции, наложенные на разработчиков программного обеспечения, задержали работы по модернизации.

2. Передача данных и безопасность. Передача больших объемов данных представляет собой значительные сложности, поскольку необходимо свести к минимуму ошибки и потери. Кроме того, передача и хранение этих данных сопряжены с риском несанкционированного доступа.

3. Совместимость с существующими системами. Переход на новое программное обеспечение может вызвать проблемы совместимости с существующим ПО и потребовать дополнительных ресурсов для интеграции.

4. Адаптация бизнес-процессов. Многим авиакомпаниям необходимо реструктурировать и адаптировать свои бизнес-процессы к новому программному обеспечению.

5. Обучение персонала. Как уже упоминалось, обучение персонала имеет решающее значение для реструктуризации бизнес-процессов и внедрения новых систем.

6. Соответствие стандартам. Национальные системы должны полностью соответствовать российским и международным стандартам авиационной безопасности. [12]

На основании проведенного авторами аналитического исследования, можно выделить основные аспекты эффективности цифровой технологии:

- операционная эффективность,
- финансово-экономическая эффективность,
- эффективность обеспечения безопасности,
- клиентская эффективность,
- технологическая эффективность и использование данных.

Исходя из проведенного анализа вопроса внедрения информационных технологий в технологические процессы авиатранспортной отрасли, представляется целесообразным предложить создание единого цифрового пространства (кластера), информацию которого смогут использовать уже существующие технологии каждого участника системы – авиакомпании, аэропорты, авиаремонтные предприятия [11].

Создание такого пространства позволит:

- синхронизировать государственный контроль над всеми смежными секторами авиационной отрасли в соответствии с целями и задачами стратегических документов;
- интегрировать и стандартизировать все компоненты ИТ-систем и программного обеспечения;
- обеспечить обработку больших объемов данных;
- обеспечить безопасный доступ к этим данным для всех заинтересованных сторон;
- интегрироваться с существующей продуктовой экосистемой региона;
- значительно снизить эксплуатационные расходы;
- ускорить развитие отечественных технологий в авиационной отрасли и повысить конкурентоспособность на мировых рынках [3].

Использование цифровых продуктов превращает получаемые при их работе данные из побочного продукта деятельности в стратегический актив, который позволяет авиакомпаниям

принимать обоснованные решения, повышать конкурентоспособность и устойчивость в цифровую эпоху, а реализация данного проекта станет катализатором развития всей отрасли гражданской авиации [1; 8].

Все цифровые технологии, рассмотренные в данной работе, при дальнейшем развитии будут оказывать значительное влияние на процесс организации и осуществления перевозок воздушным транспортом, начиная от предполетного обслуживания ВС, заканчивая выдачей багажа. Каждая внедренная информационная технология помогает сокращать расходы авиакомпании на каждом этапе авиаперевозки, тем самым приводя к экономическому росту и укреплении позиций на рынке логистики.

Литература

1. Аврамчиков В.М., Тимохович А.С., Рожнов И.П. Цифровая трансформация в авиационной отрасли: возможности и перспективы // Вестник евразийской науки. 2024. Т. 16, № 3. С. 2.

2. Аладьев А.А. Цифровизация объектов аэропортовой инфраструктуры и деятельности авиакомпаний как единой системы функционирования отрасли гражданской авиации // Современная наука и молодые учёные: сборник статей Международной научно-практической конференции (Пенза, 17 февраля 2020 года). Пенза: изд-во «Наука и Просвещение», 2020. С. 101-105.

3. Алексеева М.М., Боброва А.И., Большедворская Л.Г. Авиационные компетенции в современных реалиях. Москва, 2025. 260 с.

4. Белякова Г.Я., Аврамчиков В.М. Специфика и особенности цифровой трансформации авиаотрасли // Вестник Томского государственного университета. 2024. № 68. С. 273-292.

5. Брусникин В.Ю., Гаранин С.А., Глухов Г.Е. Оптимизация процесса обмена информацией между авиапредприятиями в рамках единого информационного пространства // Научный вестник ГосНИИ ГА. 2017. № 17(328). С. 27–33.

6. Власова А.В., Зайцев М.И. Внедрение IT-решений в производственные процессы аэропорта // Актуальные аспекты развития гражданской авиации (Авиатранс-2025): Материалы XIV Всероссийской научно-практической конференции (Ростов-на-Дону, 16–21 июня 2025 года). Ростов-на-Дону: изд-во Московский государственный технический университет гражданской авиации, 2025. С. 147-155.

7. Голобоков В.А., Ступина А.А., Рожнов И.П. Формирование подсистемы информационного обслуживания управляющего автоматизированного комплекса // Системы управления и информационные технологии. 2022. № 2(88). С. 55–60.

8. Жуков В.Е. Управление социально-экономическим развитием: инновационный и стратегические подходы // Роль и значение цифровых платформ в гражданской авиации: Материалы Национальной научно-практической конференции (Гатчина, 22 декабря 2023 года). Гатчина: изд-во Государственный институт экономики, финансов, права и технологий, 2024. С. 64-68.

9. Степаненко А.С., Большедворская Л.Г. Особенности формирования транспортно-логистических цепочек // Гражданская авиация на современном этапе развития науки, техники и общества: Сборник тезисов докладов Международной научно-технической конференции, посвященной 100-летию отечественной гражданской авиации (Москва, 18–19 мая 2023 года). М.: изд-во ИД Академии имени Н. Е. Жуковского, 2023. С. 508-510.

10. Степаненко А. С. Глобальные технологические тренды и их применение в гражданской авиации // Гражданская авиация на современном этапе развития науки, техники и общества: Сборник тезисов докладов участников Международной научно-технической конференции, посвященной 45-летию Университета (Москва, 18–20 мая 2016 года). М.: изд-во Академия имени Н.Е. Жуковского, 2016. С. 261.

11. Сушко О.П., Юрченко М.В. Внедрение CRM технологий для повышения эффективности деятельности авиатранспортного предприятия // Вызовы современности и стратегии развития общества в условиях новой реальности: Сборник материалов XXXVIII Международной научно-практической конференции (Москва, 30 сентября 2025 года). М.: изд-во АНО ДПО «Университет ИТБО», 2025. С. 232-241.

12. Танковид В.В., Веригина Г.М. Переход российских авиакомпаний на отечественное программное обеспечение: проблемы и перспективы // Вестник Национального института бизнеса. 2024. № 2(54). С. 197-205.

Бегеев А.С., Вьюкин Д.А.
Нижневартовский государственный университет
г. Нижневартовск, Россия

ПРОЕКТИРОВАНИЕ ВИРТУАЛЬНОЙ ЛАБОРАТОРИИ ДЛЯ ИССЛЕДОВАНИЯ ЛИНЕЙНЫХ ЭЛЕКТРИЧЕСКИХ ЦЕПЕЙ НА ОСНОВЕ 3D-ДВИЖКА JMONKEYENGINE

С учётом стремительного роста технологий и цифровизации, современное инженерное образование ищет новые пути. Главная задача сегодня – не просто внедрить дистанционный формат обучения, а сделать его качественным и полезным для будущих инженеров. Традиционные физические стенды обладают рядом ограничений: высокая стоимость оборудования, необходимость обслуживания, ограниченный доступ в удаленном формате и потенциальная опасность при работе с электричеством для начинающих. Виртуальные лаборатории становятся эффективным инструментом, позволяющим студентам получать практические навыки без привязки к месту и времени. Как отмечается в современных исследованиях, виртуальные лаборатории позволяют студентам изучать различные научные дисциплины, проводя эксперименты и практические занятия, не выходя из дома [5, с. 375]. Кроме того, их использование способствует развитию навыков работы с современным оборудованием в интерактивной среде, что особенно актуально для студентов, совмещающих учебу с работой [4, с. 17].

Существующие программные решения для моделирования электрических цепей можно разделить на две категории. Первая – 2D-симуляторы. Они просты в использовании, но не дают пространственного представления о реальной сборке цепи. Вторая категория – профессиональные пакеты, которые обладают широкими возможностями, но являются платными, закрытыми и сложными для модификации. Таким образом, существует потребность в открытом, расширяемом и наглядном инструменте для обучения электротехнике. Выбор правильного уровня детализации 3D-моделей в такой лаборатории является ключевым фактором, влияющим на когнитивную нагрузку обучающегося и производительность системы, что подтверждается исследованиями в области оптимизации виртуальной образовательной среды [3, с. 117-120].

Целью настоящей работы является разработка виртуальной лаборатории для анализа линейных цепей. Для достижения поставленной цели необходимо решить следующие задачи: формализация модели цепи, разработка архитектуры, реализация интерфейса и проведение вычислений.

Научная новизна. Предложен подход к интеграции математического ядра расчета цепей с 3D-движком jMonkeyEngine, обеспечивающий наглядность процессов сборки и измерения. В отличие от существующих решений, разработанная система базируется на открытой Java-платформе, что обеспечивает кроссплатформенность и возможность дальнейшего расширения функционала.

Анализ предметной области

Объектом моделирования выступают линейные электрические цепи, параметры элементов которых не зависят от величин токов и напряжений. Теоретическую базу анализа составляют фундаментальные законы электротехники. Закон Ома устанавливает пропорциональную зависимость между током и напряжением: $I = U/R$. Первый закон Кирхгофа гласит, что алгебраическая сумма токов в любом узле цепи равна нулю. Второй закон Кирхгофа утверждает, что в замкнутом контуре алгебраическая сумма ЭДС равна алгебраической сумме падений напряжений.

Типовыми пассивными элементами линейных цепей являются резистор, характеризующийся активным сопротивлением R с линейной вольт-амперной характеристикой $U = I \cdot R$; конденсатор с емкостью C, создающий в цепи переменного тока емкостное сопротивление Xc = 1/(2πfC); катушка индуктивности с индуктивностью L, создающая индуктивное сопротивление XL = 2πfL. Для цепей переменного тока используется метод комплексных амплитуд, где каждый элемент характеризуется комплексным сопротивлением (импедансом): активным Żr = R, индуктивным Żl = jX_l и емкостным Żc = $-jXc$, что позволяет единообразно описывать фазовые соотношения между токами и напряжениями.

Анализ существующих инструментов для моделирования электрических цепей выявил определенную дилемму. С одной стороны, такие решения, как Multisim компании National Instruments, обладают мощным расчетным аппаратом и обширными библиотеками компонентов, однако являются закрытыми, и их приобретение затруднено или невозможно для российских образовательных учреждений в связи с санкционными ограничениями. С другой стороны, популярные онлайн-симуляторы, включая Falstad Circuit Simulator (рис. 1), предлагают удобный интерфейс и наглядную анимацию, но ограничиваются двумерной визуализацией, не дающей пространственного представления о реальном монтаже схемы (https://clck.ru/3SViFm). Отечественная программа Micro-Cap, несмотря на широкие возможности схемотехнического моделирования, также ориентирована на двумерное представление и не обеспечивает наглядности сборки цепи в пространстве. Tinkercad Circuits, предлагающий трехмерное представление макетной платы, имеет примитивную графику, ограниченную расширяемость и требует постоянного подключения к сети Интернет. Тем не менее, даже ограниченное использование виртуальных сред в учебном процессе, как показывает анализ, способствует более глубокому усвоению материала [2, с. 266].

В связи с этим в качестве базовой платформы для разработки был выбран открытый трехмерный движок jMonkeyEngine, реализованный на языке Java (https://clck.ru/3SViDw). Данный выбор обусловлен рядом преимуществ: кроссплатформенностью, обеспечивающей работу на операционных системах Windows, Linux, macOS и Android; использованием Java, что существенно упрощает интеграцию с математическими библиотеками и написание расчетного ядра; наличием развитого графа сцены, системы освещения и средств пост-обработки; поддержкой загрузки трехмерных моделей в распространенных форматах; возможностью создания как десктопных, так и веб-приложений; а также наличием активного сообщества и открытой документации. Интеграция цифровых инструментов, подобных

jMonkeyEngine, в образовательный процесс является важным шагом в современной парадигме обучения, позволяя не только визуализировать, но и аппроксимировать экспериментальные данные [1, с. 269].

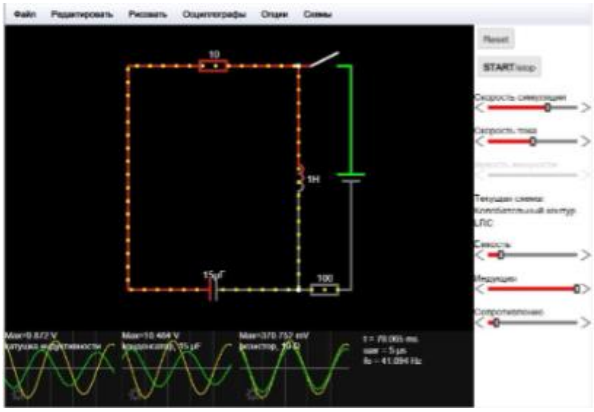


Рис. 1. Falstad Circuit Simulator [https://clck.ru/3SViFm]

Формализация модели цепи

Для программной реализации электрическая цепь представляется в виде неориентированного графа $G=(V,E)$, где вершинами V являются узлы соединения компонентов (пины), а ребрами E – сами компоненты, характеризуемые типом и импедансом. Информация о каждом компоненте хранится в структуре с полями: уникальный идентификатор, тип, списки левых и правых пинов со ссылками на соответствующие узлы графа сцены, расчетные значения тока и напряжения, а также логический признак включения в цепь. Для представления топологии цепи в целом используется структура, содержащая общий импеданс, списки последовательно соединенных компонентов и коллекцию групп параллельно соединенных элементов.

Анализ топологии выполняется в несколько этапов. На основе созданных пользователем проводов строится граф соединений, после чего проверяется замкнутость цепи наличием пути между полюсами источника. Для каждого компонента устанавливается факт включения в цепь путем проверки соединения его пинов с графом. Поиск всех возможных путей между полюсами источника позволяет классифицировать компоненты: присутствующие во всех путях считаются последовательными, встречающиеся лишь в некоторых – параллельными. Расчет общего импеданса выполняется суммированием для последовательных соединений и суммированием обратных величин для параллельных. Завершающим этапом вычисляются токи и напряжения по закону Ома. Данный подход корректен для цепей со строго последовательно-параллельной структурой. Для анализа сложных схем произвольной конфигурации требуется применение метода узловых потенциалов или метода контурных токов.

Архитектура программного комплекса

Приложение построено по модульному принципу, что обеспечивает гибкость, масштабируемость и удобство сопровождения. В состав программного комплекса входят следующие основные компоненты. Ядро приложения представляет собой главный класс,

наследующий *SimpleApplication* из библиотек jMonkeyEngine (https://clck.ru/3SViDw). Модуль трехмерной сцены обеспечивает загрузку моделей, настройку освещения, управление камерой и применение эффектов пост-обработки. Модуль ввода обрабатывает события мыши, включая клики и наведение, реализует подсветку объектов и управляет процессом создания соединений. Модуль данных хранит информацию о компонентах, их коннекторах и текущих соединениях. Модуль расчета выполняет анализ графа цепи и вычисляет электрические параметры на основе описанного ранее алгоритма. Модуль визуализации данных отвечает за отображение результатов измерений и состояния цепи на экране.

Взаимодействие модулей организовано следующим образом. При наведении пользователем курсора на коннектор модуль ввода определяет целевой объект посредством механизма лучей, после чего передает информацию модулю трехмерной сцены для активации подсветки. При последовательном клике на два коннектора модуль ввода фиксирует выбор и передает данные модулю данных для создания нового проводного соединения. После создания или удаления провода модуль данных уведомляет расчетный модуль об изменении структуры графа. Расчетный модуль перестраивает топологию цепи и вычисляет новые значения токов и напряжений. Полученные результаты передаются в модуль визуализации данных для отображения на экране и, опционально, в модуль трехмерной сцены для изменения цвета компонентов в зависимости от протекающего тока.

Модульность архитектуры обеспечивает возможность независимого развития и модификации отдельных компонентов системы. Событийная модель взаимодействия гарантирует реактивность пользовательского интерфейса. Использование графа сцены jMonkeyEngine существенно упрощает управление трехмерными объектами.

Реализация 3d-интерфейса

Ключевой задачей 3D-интерфейса является точная идентификация объектов под курсором мыши, что решается использованием встроенного в jMonkeyEngine механизма лучей. Виртуальное пространство организовано в виде трёхмерной сцены лабораторного стенда, которая загружается из внешних файлов, что позволяет изменять окружение без модификации программного кода. Для оптимальной производительности используются модели в нативном формате *j3o*, обеспечивающем быструю загрузку и хранение предварительно рассчитанных уровней детализации с готовыми материалами. В рабочих процессах, требующих частого обновления ассетов из сторонних редакторов, применяется современный формат *glTF (GLB)*, поддерживающий сжатие текстур, PBR-материалы и анимацию при прямой загрузке в движок (рис. 2). Дополнительно обеспечивается совместимость с форматами *OBJ* и *FBX* для импорта моделей из различных CAD-систем и пакетов трёхмерного моделирования.

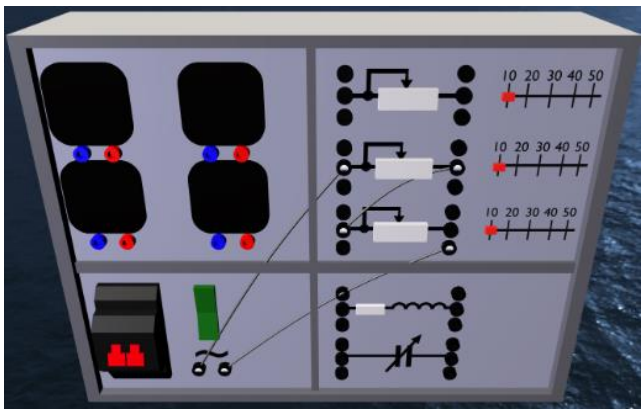


Рис. 2. Модель, загруженная в формате .glb (составлено автором)

Проведение вычислений и тестирование

Для апробации разработанного прототипа был проведен вычислительный эксперимент на схеме (рис. 3) с последовательным соединением двух резисторов сопротивлением 10 Ом каждый при напряжении источника 12 В. В виртуальной среде была собрана замкнутая цепь путем последовательного соединения коннекторов компонентов левой кнопкой мыши, после чего расчетный модуль автоматически выполнил анализ топологии. Теоретический расчет для последовательного соединения дает общее сопротивление 20 Ом, ток 0,6 А и падение напряжения на каждом резисторе 6 В, что соответствует классическим методам анализа цепей постоянного тока. Полученные программным комплексом значения полностью совпали с теоретическими, что подтвердило корректную работу механизма распознавания топологии, алгоритма расчета общего сопротивления и процедуры распределения напряжений на элементах цепи.

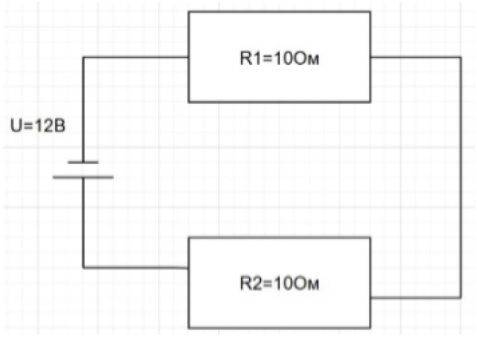


Рис. 3. Схема последовательного соединения двух резисторов (составлено автором)

В ходе исследования представлен прототип виртуальной лаборатории для анализа линейных электрических цепей, в котором предложен подход к интеграции графовой модели цепи с трехмерным движком jMonkeyEngine, обеспечивающий наглядность процессов монтажа, недостижимую в классических 2D-симуляторах. Дальнейшее развитие системы предполагает интеграцию более сложных методов расчета, расширение библиотеки компонентов и проведение педагогического эксперимента для оценки эффективности предложенного подхода в сравнении с традиционными формами обучения, что является общей практикой при внедрении цифровых инноваций в образование [1, с. 269].

Литература

1. Джалмухамбетова Е.А., Можарова А.В., Элькин П.М., Ракин Г.В. Интеграция цифровых инструментов в лабораторные работы по естественнонаучным дисциплинам // Актуальные решения проблем водного транспорта: IV Международная научно-практическая конференция (29-30 мая 2025 года). 2025. С. 268-270.

2. Масленников А.Э. Оценка эффективности использования виртуальных лабораторий в организации лабораторных занятий по физике при дистанционном обучении // Инновации в естественно-научном образовании: материалы XVII Всероссийской научно-методической конференции (г. Красноярск, 27 ноября 2025 года). Красноярск, 2025. С. 262-266.

3. Пилип П.А. Оптимизация процесса обучения в виртуальных лабораториях в зависимости от уровня детализации 3D-моделей // Система педагогического образования – ресурс развития общества: Осенняя научная сессия КГПУ им. В.П. Астафьева. Красноярск, 2025. С. 117-120.

4. Прохорчук Е.Н. Практическая подготовка студентов на занятиях по методике обучения биологии в педагогическом вузе // Инновации в естественно-научном образовании: материалы XVII Всероссийской научно-методической конференции (г. Красноярск, 27 ноября 2025 года). Красноярск, 2025. С. 15-20.

5. Ручкина А.С. Виртуальные лаборатории: практическое обучение в условиях удаленного образования // Инновации в образовательном процессе: сборник трудов Международной научно-практической конференции (г. Чебоксары, 24 апреля 2025 года). Вып. 23. Чебоксары: Политех, 2025. С. 374-376.

Башкаев И.В., Березин Д.О.
Уральский государственный экономический университет
г. Екатеринбург, Россия

АНАЛИЗ СЕМАНТИЧЕСКОГО И ЛЕКСИЧЕСКОГО ПОДХОДОВ К ПОИСКУ КНИГ ПО БИБЛИОГРАФИЧЕСКИМ ДАННЫМ НА БАЗЕ ВЕКТОРНОЙ БД QDRANT

В современных библиотечных системах поиск информации по библиографическим данным играет ключевую роль в обеспечении доступа к ресурсам. Традиционные лексические методы поиска, основанные на точном совпадении слов или фраз в полях записи (таких как название, автор или ключевые слова), демонстрируют высокую эффективность в сценариях с четко сформулированными запросами. Однако они часто оказываются недостаточными при наличии синонимов, опечаток или контекстуальных вариаций, что приводит к потере релевантных результатов и снижению полноты поиска. В этом контексте семантические подходы, использующие векторные представления текстов для оценки смыслового сходства, предлагают альтернативу, позволяющую учитывать неявные связи между запросом и данными [2, с. 12-15]. Такие методы, реализованные на базе векторных баз данных, таких как Qdrant, интегрируют модели машинного обучения для генерации эмбеддингов [3, с. 37-38], что потенциально повышает качество поиска в разнородных библиографических коллекциях.

Необходимость сравнительного анализа обусловлена растущим объемом цифровых библиотечных ресурсов и требованиями пользователей к более интуитивным системам. Лексический поиск, воплощенный в системах вроде ИРБИС64+, остается стандартом в многих учреждениях благодаря своей простоте и скорости, но не справляется с семантической неоднозначностью. В противоположность этому, семантический поиск на Qdrant с фильтрами по году и автору может улучшить релевантность результатов, но требует оценки его преимуществ и недостатков в реальных условиях. Цель настоящей работы – провести сравнительный анализ этих подходов на основе типовых запросов к библиографическим данным, выявив сценарии, в которых семантический метод превосходит лексический, и наоборот. Для достижения цели решаются следующие задачи:

1. описание архитектур систем;
2. разработка методики сравнения с использованием метрик точности и полноты;
3. анализ результатов и формулировка рекомендаций по применению.

Для проведения анализа были выбраны две системы, представляющие лексический и семантический подходы к поиску по библиографическим данным. Лексический метод реализован в библиотечной автоматизированной системе ИРБИС64+, широко используемой в российских библиотеках, а семантический – в разработанной авторами системе на базе векторной базы данных Qdrant (https://qdrant.tech/documentation). Ниже приведено описание принципов работы каждой системы.

Система ИРБИС64+ представляет собой комплексное решение для автоматизации библиотечных процессов, включая каталогизацию и поиск по библиографическим записям. Поисковый модуль опирается на лексический подход, который подразумевает сопоставление запроса с содержимым полей записи на основе точного или частичного совпадения строк.

Запрос может включать ключевые слова, фразы или комбинации с использованием логических операторов (AND, OR, NOT), а также масок для учета вариаций (например, с использованием символов подстановки).

Разработанная система семантического поиска использует векторную [1, с. 114-116] базу данных Qdrant, развернутую в контейнере Docker для обеспечения изоляции и масштабируемости. Библиографические записи предварительно декодируются с извлечением ключевых полей (название, авторы, год, ключевые слова, предметные рубрики), после чего текст нормализуется и преобразуется в векторные представления с помощью модели эмбеддингов multilingual-e5-large из библиотеки Sentence Transformers [5]. Эта модель генерирует 1024-мерные векторы, учитывающие семантическое сходство текстов на основе косинусного расстояния. Поиск выполняется путем преобразования пользовательского запроса в вектор и сравнения его с векторами записей в Qdrant, возвращая топ-N результатов по релевантности. Дополнительно реализованы фильтры по году, фамилии автора и ключевым словам. Архитектура включает FastAPI для API-интерфейса, обеспечивающего запросы с параметрами, и интеграцию с Qdrant для хранения точек (points) с payload, содержащим метаданные.

На рисунке 1 показана схема системы (процесс от декодирования записи до поиска).

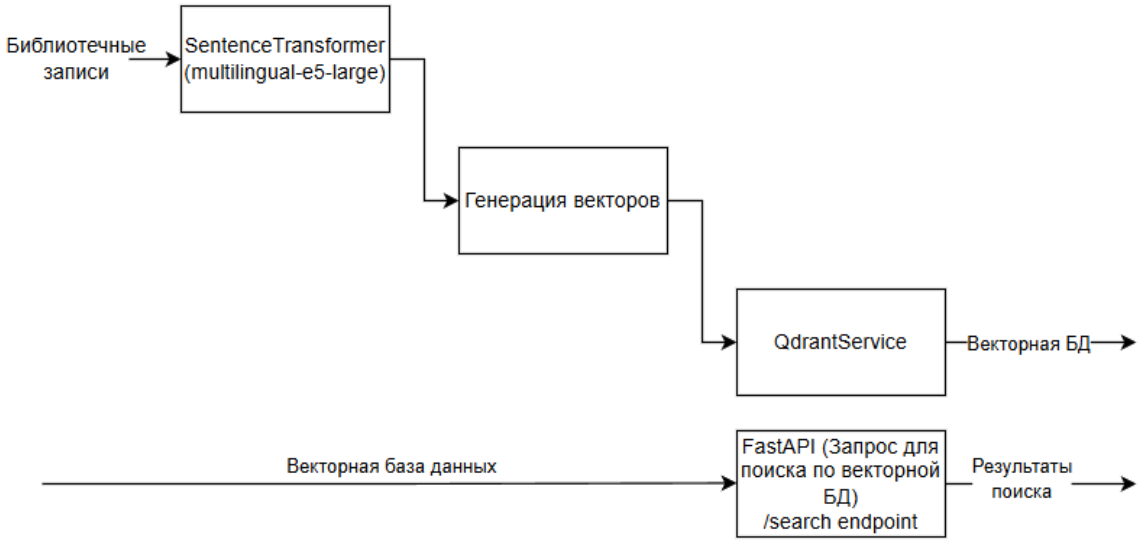


Рис. 1. Упрощённая архитектура семантической поисковой системы на базе Qdrant

Этот подход позволяет учитывать смысловые связи, такие как синонимы или тематическая близость, но требует больше вычислительных ресурсов для генерации эмбеддингов и может быть чувствителен к качеству исходных данных.

Векторная база данных Qdrant использует алгоритм Hierarchical Navigable Small World (HNSW) (www.qdrant.tech/indexing/hnsw-index) для индексации и поиска по плотным векторам. HNSW представляет собой многослойный ориентированный граф, в котором каждый вектор является узлом, а связи между узлами отражают близость в пространстве эмбеддингов [4]. Верхние слои графа содержат небольшое количество узлов с длинными связями для быстрой навигации, нижние – плотные локальные связи для точного уточнения

ближайших соседей. Поиск начинается с верхнего слоя и последовательно спускается вниз, выполняя жадный обход по ближайшим соседям на каждом уровне (www.pinecone.io/hnsw). Такой подход обеспечивает сублогарифмическую сложность поиска O(log N) при высоком уровне точности (recall > 0.95). В реализации авторов применяются стандартные параметры Qdrant: m = 16 (максимальное число связей на узел), ef_construct = 100 (при построении индекса) и hnsw_ef = 128 (при поиске) (www.qdrant.tech/benchmarks). На рисунке 2 представлена схема работы алгоритма HNSW – на каждом последующем слое происходит уточнение, позволяющее найти наиболее близкие к искомому значения, не перебирая все возможные варианты, а перемещаясь от большего к меньшему.

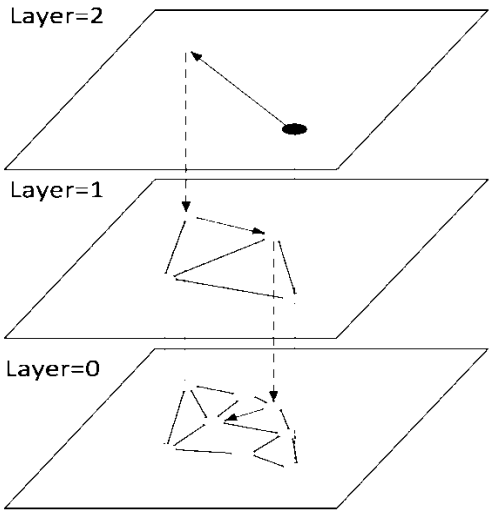


Рис. 2. Процесс поиска по графу HNSW (многослойная навигация от грубого к точному приближению ближайших соседей)

Благодаря использованию данного подхода Qdrant демонстрирует высокую производительность при выполнении поисковых запросов. Согласно официальным бенчмаркам Qdrant (обновлённым на 2024 год) и независимым сравнениям, типичные значения задержки (latency) для поиска по векторам на датасетах порядка 1 млн записей составляют 3–5 мс в среднем (p50), 5–10 мс на уровне p95 и до 15–40 мс на p99 в зависимости от конфигурации индекса HNSW (www.habr.com/articles/338360), размера батча и аппаратных ресурсов. В сценариях с оптимизацией под высокую пропускную способность (RPS – requests per second) Qdrant достигает 1000–1200+ запросов в секунду при сохранении высокой точности (recall/precision около 0.99). В реальных тестах на VPS или одиночном узле с HNSW-индексом задержка поиска стабильно лежит в диапазоне 15–40 мс (p95), что значительно ниже, чем у многих альтернатив (например, Weaviate в аналогичных условиях показывает 25–70 мс, а в некоторых конфигурациях до 100+ мс).

Такие показатели делают Qdrant подходящим для приложений реального времени, где требуется низкая задержка даже при одновременных запросах от нескольких пользователей. В нашей реализации (локальный Docker-контейнер на стандартном оборудовании университета) время отклика на поисковый запрос обычно укладывается в 50–200 мс с учётом этапа генерации эмбеддинга запроса моделью Sentence Transformers и сетевого

взаимодействия через FastAPI. Это существенно быстрее, чем наблюдаемые задержки в 10–15 секунд при сложных запросах в ИРБИС64+ на той же выборке данных.

Сравнение подходов проводилось на единой тестовой выборке библиографических записей (около 10 000 единиц), извлеченных из реальной библиотечной базы и индексированных в обеих системах для обеспечения сопоставимости. Выборка включала разнообразные типы документов: монографии, статьи, видео экспорты, с охватом тем от гуманитарных наук до техники.

Для анализа были отобраны 7 запросов, представляющих типичные сценарии библиотечного поиска:

1. точное название книги;
2. запрос с синонимами (например, "машинное обучение" вместо "искусственный интеллект");
3. тематический запрос без ключевых слов;
4. запрос с фильтром по автору;
5. запрос с опечаткой;
6. комбинированный запрос с синонимами и фильтром;
7. абстрактный тематический запрос.

Данные типы запросов выбраны для оценки способности систем справляться с вариативностью пользовательских формулировок.

Для оценки результатов было использовано ограничение на топ-10 результатов, что позволяет сосредоточиться на наиболее релевантных записях, минимизируя влияние шума.

Таблица 1

Сравнение результатов поиска по тестовым запросам

№	Запрос	Оценка ИРБИС64+	Оценка разрабатываемой системы
1	Сократ	Найдено 4 из 4 точных записей	Найдено 4 из 4 точных записей
2	Машинное обучение и нейронные сети	Найдены только те записи, где в названии или ключевых словах есть «машинное обучение» или «нейронные сети»	Найдено 10/10 записей, связанных ИИ
3	Книги про то, как компьютеры понимают человеческий язык	Найдено 1/10	Найдено 10/10 записей, близких по смыслу к запросу
4	Искусственный интеллект, Назаров	Найдено 3/10	Найдено 10/10 записей, связанных с ИИ у данного автора
5	Алгаритмы сартировки и поиска	Найдено 0/10 релевантных записи	Найдено 10/10 записей, близких по смыслу (учтена опечатка)
6	Глубокое обучение компьютерного зрения, 2015+	Найдено 1/10	Найдено 10/10 записей, близких по смыслу к запросу с фильтром после 2015 года
7	Как защитить данные от утечек в облачных системах	Найдено 3/10 релевантных записи	Найдено 10/10 записей, близких по смыслу к запросу

Анализ таблицы демонстрирует, что при точном и строго формализованном запросе №1 обе системы демонстрируют идентичный результат. Во всех остальных случаях разработанная система на базе Qdrant существенно превосходит ИРБИС64+ по количеству найденных релевантных записей.

Полученные данные демонстрируют явное преимущество семантического поиска на базе Qdrant в ситуациях, когда запрос пользователя сформулирован неточно, использует синонимы, описательный язык или содержит опечатки. Модель multilingual-e5-large позволяет учитывать смысловую близость текстов даже при отсутствии общих слов, что приводит к значительному росту полноты поиска.

Среди ограничений векторного подхода следует отметить более высокую вычислительную сложность. Генерация эмбеддинга запроса и сравнение векторов требуют дополнительных ресурсов по сравнению с лексическим поиском. Однако в реальных условиях эксплуатации ИРБИС64+ часто демонстрирует заметно большее время отклика (до 10–15 секунд на сложные запросы в крупных базах), тогда как поиск в Qdrant (с учётом этапа эмбеддинга) укладывается в 50–200 мс на стандартном домашнем оборудовании без графических ускорителей. Это делает разработанную систему конкурентоспособной не только по качеству, но и по скорости.

Выводы исследования указывают на перспективность семантического поиска с ранжированием для тематических, синонимичных и неточных формулировок. Такой механизм способен одновременно обеспечить высокую точность и существенно повысить полноту поиска в библиотечных системах.

Литература

1. Томилов Н.А., Туров В.П., Бабаянц А.А., Платонов А.В. Метод хранения векторных представлений в сжатом виде с применением кластеризации // Научно-технический вестник информационных технологий, механики и оптики. 2024. № 1. С. 112-117.

2. Шалагин Н.Д. Обзор алгоритмов семантического поиска по текстовым документам // International Journal of Open Information Technologies. 2024. № 9. С. 11-21.

3. Ivanouski A., Vector embeddings for enhanced efficiency in full-text search with artificial intelligence technologies // Universum: технические науки. 2024. № 12 (129). С. 34-39.

4. Malkov Y., Ponomarenko A., Logvinov A., Krylov V. et al. Approximate nearest neighbor algorithm based on navigable small world graphs // Information Systems. 2014. С. 61-68. https://doi.org/10.1016/j.is.2013.10.006

5. Wang L., Yang N., Huang X. et al. Text Embeddings by Weakly-Supervised Contrastive Pre-training // arXiv preprint arXiv:2212.03533. 2022.

Вайс Ф.В.
Нижневартовский государственный университет
г. Нижневартовск, Россия

ОНЛАЙН-ГАЛЕРЕЯ КАК СРЕДСТВО СОХРАНЕНИЯ НАЦИОНАЛЬНЫХ КУЛЬТУРНЫХ РЕСУРСОВ

«Искусство есть средство выражения человеческих умений и мастерства в творчестве, продуктом которого являются различные произведения, среди которых выделяют: картины (живопись) и скульптуру» [2].

Современные художники активно пользуются онлайн-платформами для продвижения своих произведений и формирования аудитории. «Платформы предоставляют для современных авторов прекрасные возможности популяризации декоративного искусства и продвижения результатов творческого труда. Активное продвижение в интернет-пространстве позволяет художнику закрепить свой персональный бренд, тем самым придать своему искусству неповторимые черты, увеличив его ценность» [4].

Онлайн-галерея – это платформа, где пользователи могут увидеть и изучить произведения искусства.

В сфере современного искусства можно выделить различные типы галерей, каждый из которых выполняет уникальные функции и задачи. Галереи выступают не только как площадки для демонстрации произведений, но и как центры взаимодействия между художниками, коллекционерами, критиками и широкой аудиторией. В зависимости от своей миссии, структуры и организационного подхода, галереи можно классифицировать на несколько категорий, включая коммерческие, некоммерческие, институциональные и временные выставочные пространства. Каждая из этих категорий имеет свои особенности и специфику, что позволяет удовлетворить разнообразные потребности и интересы в области искусства и культуры.

Третьяковская галерея

https://in.gallerix.ru

«Главная цель этого проекта – повсеместная популяризация русского искусства и желание ознакомиться с его наиболее яркими проявлениями» [1]. Для достижения этой цели был разработан специализированный каталог, предоставляющий пользователям возможность осуществлять поиск произведений по различным параметрам, включая жанровые категории, персоналии художников и исторические эпохи.

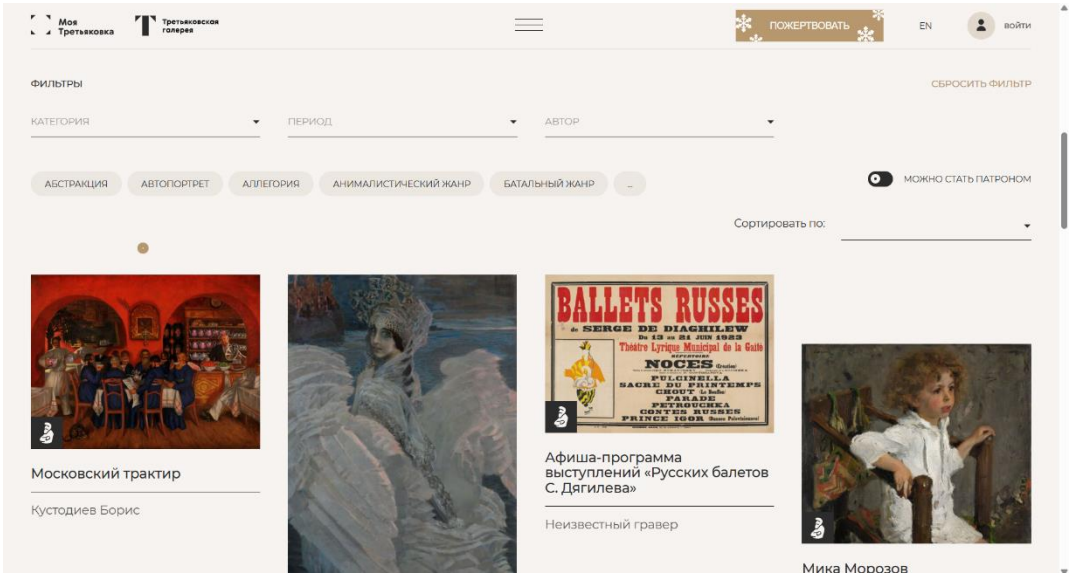


Рис. 1. Главная страница Третьяковской галереи

SAMPLER ART

https://sample-art.com

Платформа помогает молодым художникам и коллекционерам, создавая условия для увлекательного и доступного знакомства с искусством.

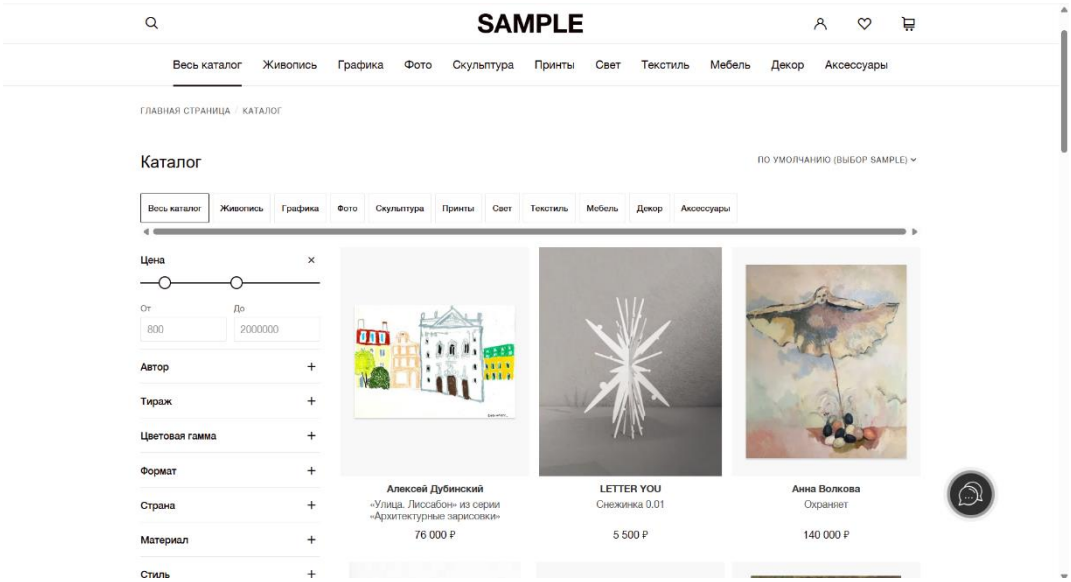


Рис. 2. Главная страница SAMPLE

Gallerix

https://in.gallerix.ru

Это онлайн-платформа, на которой представлено свыше 300 000 художественных работ, доступных для покупки. Она также включает элементы социальной сети: ленты с произведениями, профили художников, форум и статьи с советами по выбору картин.

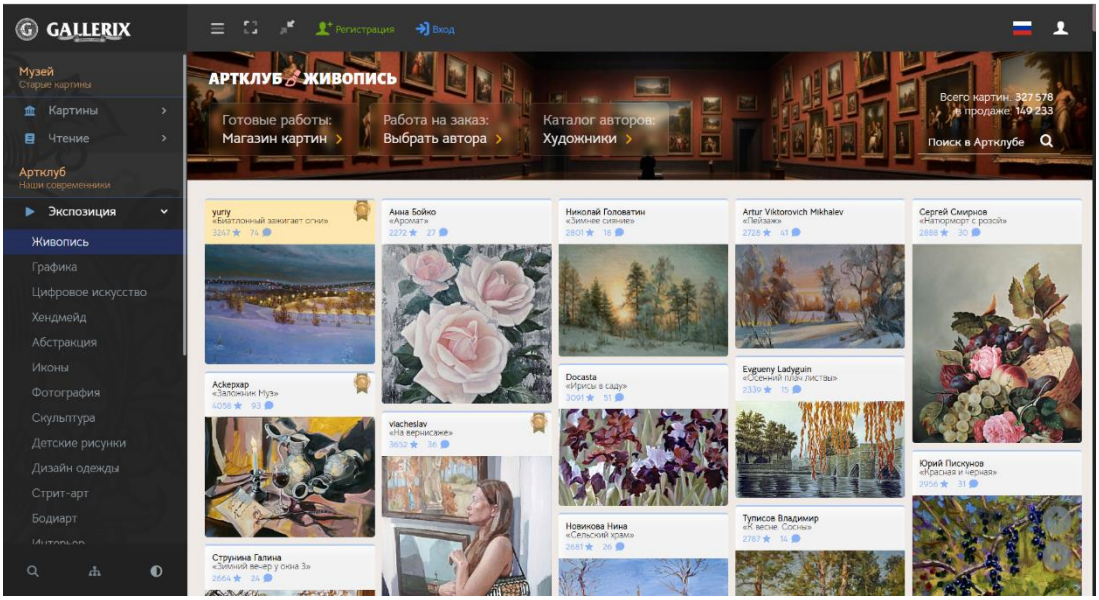


Рис. 3. Главная страница Gallerix

ОБЪЕДИНЕНИЕ

https://obdn.ru/selections/obdn-kontur-2025

Страница об участии сообщества «Объединение» в ярмарке графики «КОНТУР» в Нижнем Новгороде, где представлены работы художниц Полины Майер, Екатерины Герасименко и Светланы Цепкало. Это анонс офлайн-мероприятия, а не постоянная онлайн-галерея.

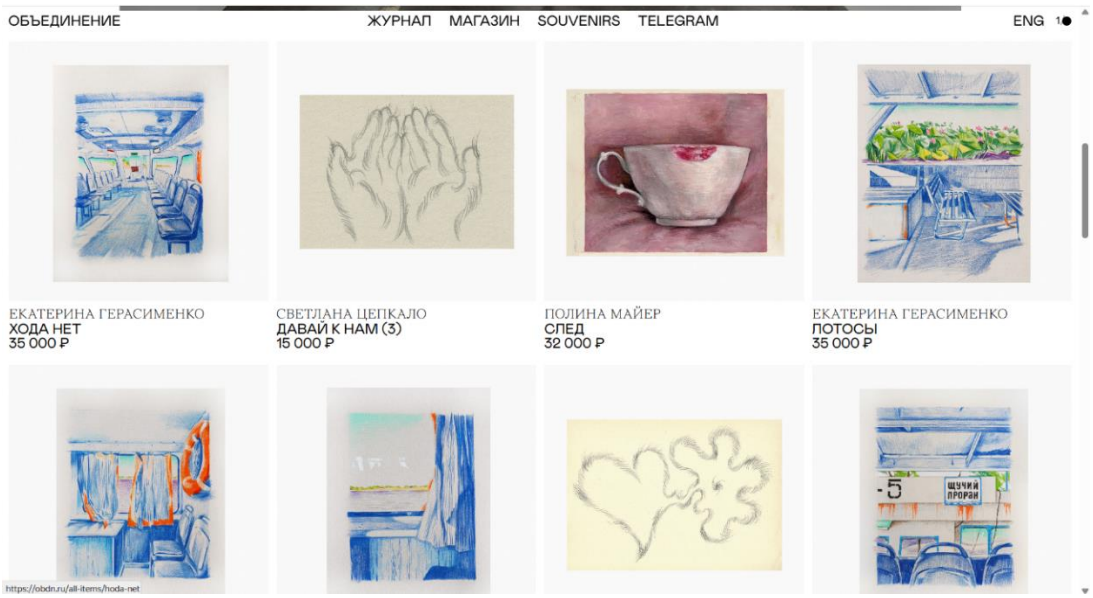


Рис. 4. Главная страница ОБЪЕДИНЕНИЕ

Все три платформы, несмотря на общую направленность на искусство, имеют разные цели и функционал, что позволяет каждой из них выделяться в цифровом арт-пространстве.

- Третьяковская галерея предлагает виртуальную коллекцию произведений искусства, направленную на сохранение культурного наследия и просвещение.

- **SAMPLER ART** – магазин современного искусства, где показывают тренды и помогают узнать о нём через экспертов и сообщество.
- **Gallerix** – большая ярмарка, где можно увидеть работы художников, пообщаться с ними и показать свои работы.
- **ОБЪЕДИНЕНИЕ** – виртуальный выставочный проект для привлечения внимания к художникам и их реальным выставкам.

Онлайн-галереи предлагают множество преимуществ для всех участников арт-рынка.

Для художников:

- **Рост популярности**. Представление работ способствует увеличению известности художника.
- **Упрощение торговли**. Цифровые технологии упрощают процесс управления продажами и улучшая взаимодействие с клиентами.
- **Прямые продажи**. Онлайн-платформы позволяют художникам общаться с покупателями напрямую, без участия посредников.
- **Всемирная выставка.** Художники получают шанс представить свои работы широкой аудитории, не ограничиваясь рамками местных галерей.

Для коллекционеров:

- **Сообщество единомышленников**. Онлайн-галереи становятся местами, где ценители искусства могут общаться, делиться своими впечатлениями и открывать для себя новых художников.
- **Онлайн-аукцион**. Галереи проводят онлайн-аукционы, где можно приобрести картины и другие произведения искусства.
- **Удобство покупок**. Онлайн-просмотр и покупка произведений искусства упрощают процесс и избавляют от лишних хлопот.
- **Прозрачность сделок**. Информация о произведениях, их истории и экспертных оценках повышает прозрачность и снижает риски.
- **Богатый ассортимент**. Онлайн-галереи предоставляют доступ к обширной коллекции произведений, независимо от того, где находится коллекционер.

Для любителей искусства:

- **Культурное обогащение**. Виртуальное знакомство с искусством расширяет кругозор и понимание различных стилей.
- **Доступность**. Онлайн-галереи позволяют людям из разных стран наслаждаться произведениями искусства.
- **Интерактивность**. Современные цифровые технологии открывают новые горизонты для организации выставок, превращая их в интерактивные пространства с виртуальными экскурсиями, видеоматериалами, аудиозаписями и другими мультимедийными возможностями.
- **Поддержка молодых талантов**. Онлайн-платформы помогают начинающим художникам заявить о себе.

Функции онлайн галерей

Галереи, несмотря на их многообразие, обладают рядом общих функций, которые играют ключевую роль в их функционировании и позволяют им эффективно реализовывать свои задачи в разнообразных контекстах.

Демонстрационная функция

Важна для любой галереи, обеспечивает высокое качество отображения искусства и предлагать удобный, интуитивно понятный интерфейс для просмотра.

Присуща всем платформам, от музейного каталога Третьяковской галереи до динамичной ленты Gallerix.

Коммуникативная функция

Главная задача создать группу людей со схожими взглядами. Платформы для общения и личные страницы участников способствуют активному обмену мнениями и дискуссиям на разные темы.

Наиболее ярко проявляется в Gallerix через соцсети и форумы, а также в SAMPLER ART через закрытые мероприятия.

Информационная функции

Предоставление информации об искусстве через статьи, образовательные программы и обзоры для просвещения аудитории. Также лекции и рассказы углубляют знания в этой области.

Третьяковская галерея акцентирует научную достоверность и просвещение через экспертные описания. SAMPLER ART и Gallerix делают это доступно через статьи и контекст. Проект «ОБЪЕДИНЕНИЕ» дает базовые сведения.

Коммерческая функция

Платформа выполняет функции, аналогичные магазину, маркетплейсу или аукциону, ориентированного на продажу произведений искусства.

В SAMPLER ART и Gallerix продажа искусства, а в ОБЪЕДИНЕНИИ – через офлайн-выставки.

Галерея не включает в себя все функции, каждая платформа использует лишь некоторые из них, в зависимости от своих задач.

Онлайн галерея для молодых художников, ориентированная на привлечение внимания и расширение целевой аудитории.

В качестве основы для разработки можно рассмотреть следующие технологии.

Для создания онлайн-галерей и виртуальных экспозиций применяются такие технологии, как HTML, CSS и JavaScript.

Специализированные фреймворки и библиотеки, такие как Three.js и A-Frame, предоставляют комплексные решения для создания интерактивных 3D-графиков и виртуальных миров.

Современные онлайн-галереи используют разнообразные мультимедийные технологии, такие как видео, аудио, анимация и интерактивные элементы. Например, библиотека Howler.js предназначена для работы со звуком в веб-приложениях. Кроме того, платформа GSAP

(GreenSock Animation Platform) позволяет создавать сложные и динамичные мультимедийные проекты.

В процессе разработки пользовательского интерфейса применяются передовые фреймворки JavaScript, такие как React и Vue. Эти инструменты обладают мощным функционалом и гибкостью, что позволяет создавать интерактивные и высокопроизводительные веб-приложения.

«BootStrap – это фронтэнд фреймворк, который предоставляет готовые компоненты и стили для быстрого создания адаптивных кроссбраузерных веб-приложений. Использование Bootstrap значительно ускоряет процесс разработки за счёт того, что множество рутинных операций, связанных со стилизацией страниц, можно выполнить в считанные минуты без необходимости писать много CSS-кода.» [3]

«Django – это высокоуровневый веб-фреймворк, позволяющий разработчикам быстро создавать сложные веб-приложения. Он был разработан с акцентом на автоматизацию рутинных задач и обеспечивает множество встроенных функций, что позволяет сосредоточиться на разработке.» [5]

Литература

1. Куприянова Т.Г., Азаматова Г.Б. Стратегия успеха государственного издательства (по материалам Третьяковской галереи) // Известия высших учебных заведений. Проблемы полиграфии и издательского дела. 2023. № 1. С. 39-42.

2. Мирошниченко В.К. Компьютерная графика и Современное искусство // Современное искусствоведение: теоретические концепции и художественные практики: Материалы Всероссийской (с международным участием) научно-практической конференции (Улан-Удэ, 11–12 ноября 2022 года). Улан-Удэ: Восточно-Сибирский государственный институт культуры, 2023. С. 19-22.

3. Фролов С.И. Разработка веб-приложения «список дел» с использованием bootstrap и Django // XXVII Всероссийская студенческая научно-практическая конференция Нижневартовского государственного университета (Нижневартовск, 09–10 апреля 2025 года). Нижневартовск: Нижневартовский государственный университет, 2025. С. 378-383.

4. Шелюгина О.А., Беляева Ю.П. Продвижение проектов в области декоративного искусства на веб-платформах // Труды молодых ученых Алтайского государственного университета. 2021. № 18. С. 98-101.

5. Яремчук А.В. Django и Flask: как выбрать фреймворк для веб-проекта // Организационно-методические аспекты повышения качества образовательной деятельности и подготовки обучающихся по программам высшего и среднего профессионального образования: Сборник статей VI Всероссийской (национальной) научно-методической конференции (Пенза, 28–29 октября 2024 года). Пенза: Пензенский государственный аграрный университет, 2024. С. 477-480.

Валентюкевич О.Е.
Нижневартовский государственный университет
г. Нижневартовск, Россия

РАЗРАБОТКА ВЕБ-САЙТА ДЛЯ АЭРОПОРТА ГОРОДА НИЖНЕВАРТОВСКА

В условиях активной цифровизации транспортной инфраструктуры наличие информативного, удобного и технологичного веб-сайта является ключевым инструментом взаимодействия аэропорта с пассажирами. Сайт позволяет предоставлять актуальную информацию о рейсах, услугах, правилах и контактах, а также обеспечивает основу для внедрения дополнительных онлайн-сервисов. В данной статье представлена разработка веб-сайта для аэропорта города Нижневартовска, основанная на современном стеке технологий с применением контейнеризации [3; 5] для обеспечения стабильности, масштабируемости и удобства развертывания. Для реализации проекта был выбран следующий набор технологий:

- Frontend: HTML5, CSS3, JavaScript – для построения интерфейса, обеспечения адаптивности и интерактивности.
- Backend: Node.js с фреймворком Express – для создания серверной логики и API [1].
- Шаблонизация: EJS (Embedded JavaScript) – для динамической генерации HTML на стороне сервера с возможностью повторного использования компонентов.
- Контейнеризация: Docker – для упаковки приложения и зависимостей в изолированные контейнеры, что упрощает развертывание и обеспечивает единообразие среды выполнения [3].

Приложение состоит из нескольких сервисов, каждый из которых работает в отдельном Docker-контейнере. Вспомогательные микросервисы (Node.js + Express) – Обрабатывают запросы по дополнительным данным, таким как суточное расписание полетов, сезонное расписание и т. д. Такое разделение на микросервисы позволяет масштабировать каждый компонент независимо и повышает отказоустойчивость системы [4] (рис. 1). Архитектурный подход является гибридным [2]:

- SSR (Server-Side Rendering) реализован с использованием EJS в качестве основного шаблонизатора, который выступает в роли мастер-страницы (masterPage), обеспечивая единообразное отображение шапки (header) и подвала (footer) на всех страницах сайта.
- CSR (Client-Side Rendering) применяется для динамических компонентов, таких как онлайн-табло или блок услуг.

Данные для этих компонентов хранятся на сервере в формате JSON и могут обновляться администратором без изменения клиентского кода, что упрощает поддержку и редактирование контента. Главная цель при проектировании была создать максимально простой инструмент, с минимальными зависимостями (от библиотек, фреймворков и т. д.) и легкой масштабируемостью. Поскольку проект на данном этапе лишь черновой вариант, в дальнейшем вполне возможны изменения в архитектурных подходах, дизайнерских решениях и так далее. Далее будут представлены схемы взаимодействия.

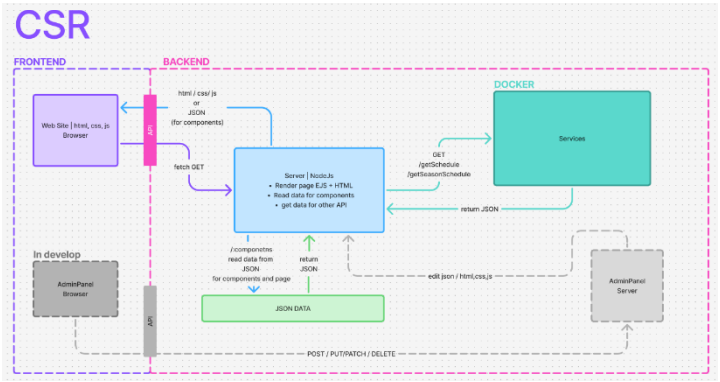


Рис. 1. Архитектура проекта сайта аэропорта

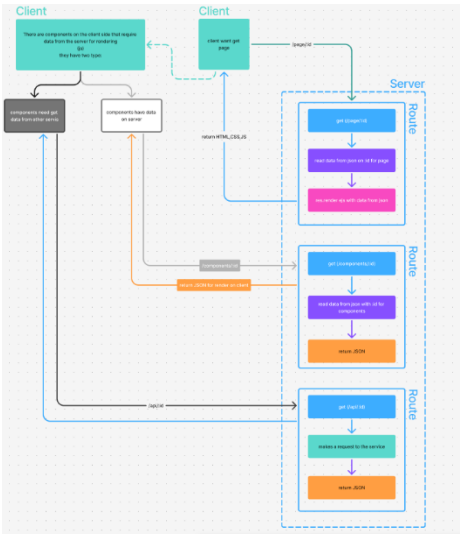


Рис. 2. Пример логики взаимодействия client/server

В веб-приложении реализована гибкая система обработки клиентских запросов и динамического рендеринга компонентов. При обращении клиента на сервер настроено несколько маршрутов (routes), через которые клиент может получать данные о страницах, данные для компонентов и осуществлять взаимодействие со сторонними сервисами (рис. 2). Рассмотрим механизм обработки запросов страниц на примере кода сервера. Когда клиент отправляет запрос на адрес http://nv-aero.ru/home (например, запрашивая домашнюю страницу), сервер выполняет следующую последовательность действий:

- Получение параметра запроса: сервер принимает в качестве параметра идентификатор (id) страницы, которую необходимо найти и отобразить.
- Загрузка структуры страниц: сервер считывает всю структуру страниц сайта, которая хранится в JSON формате (рис. 4). Представляет собой плоскую структуру, описание всех страниц сайта с их свойствами: идентификаторами, названиями, метаданными, списками компонентов и статусом видимости.
- Кэширование структуры (модуль в разработке): для оптимизации производительности реализован механизм кэширования структуры страниц. Это позволяет избежать повторного чтения и парсинга файлов структуры при каждом запросе.

- Фильтрация страниц: сервер фильтрует загруженную структуру по заданному идентификатору и дополнительному условию – страница должна быть активной (видимой для пользователей).
- Возврат структуры для рендеринга: найденная структура страницы передается в шаблонизатор EJS для генерации HTML-кода. EJS использует мастер-шаблон (masterPage), который обеспечивает единообразное оформление всех страниц сайта (рис. 3).

```
import { readData } from "./helpFunction/readData.js";

export const renderPageById = (req, res) => {
    try {
        const { id } = req.params;
        const allPages = readData('data/pages.json');
        const page = allPages.pages.filter(c => c.isActive).find(p => p.id === id);

        if (!page) {
            return res.status(404).render("error", { message: "Страница не найдена" });
        }

        res.render(`masterPage`, {
            title: page.Name,
            content: page.content,
            description: page.description,
            owner: page.owner || ''
        });
    } catch (error) {
        console.log("ERROR function renderPageById: ", error);
        res.status(500).json({ error: error.message });
    }
};
```

Рис. 3. Листинг кода на сервере для отрисовки страницы

```
{
  "pages": [
    {
      "id": "home",
      "Name": "Главная",
      "owner": "",
      "content": "home.html",
      "description": "Основная страница сайта",
      "isActive": true,
      "createdAt": "",
      "updatedAt": "2025-10-21T10:57:50.582Z",
      "child": []
    },
    {
      "id": "passengers",
      "Name": "Пассажирам",
      "owner": "home",
      "content": "passengers.html",
      "description": "Основная страница сайта",
      "isActive": true,
      "createdAt": "",
      "updatedAt": "2025-10-21T10:57:50.582Z",
      "child": []
    },
```

Рис. 4. Структура хранения страниц для сайта

После того как сервер отдал HTML-страницу, начинается процесс динамического рендеринга компонентов на стороне клиента. На каждой странице присутствуют собственные компоненты. В случае со страницей с id:home такими компонентами являются:

- Новости – блок актуальных новостей аэропорта
- Сервисы – список доступных услуг
- Табло – динамическое расписание рейсов

Выбранный подход – разделение рендеринга страницы на серверную и клиентскую части – имеет несколько важных преимуществ [2]:

1. Упрощение серверной структуры: сервер отвечает только за базовую структуру страницы, что делает его код более чистым и понятным.
2. Читаемый формат кода: компоненты реализованы на чистом JavaScript и HTML, без использования сложных шаблонизаторов типа EJS или виртуального DOM. Это облегчает

понимание и поддержку кода, хотя и приводит к дополнительной нагрузке на клиенте из-за необходимости рендеринга на стороне браузера.

3. Гибкость обновления: компоненты могут независимо обновлять свои данные без перезагрузки всей страницы.

Рассмотрим подробнее логику компонента на примере компонента «Сервисы» – услуги аэропорта:

1. Инициализация места для рендеринга: в HTML-коде страницы оставляется специальный контейнер для отображения компонента, мы просто даем уникальный id любому тегу html, где захотим рендерить компонент (рис. 5).

```
<section class=" h-screen  px-20 py-10  bg-[#cdd5dd]" id="serversNav">
    <h2 class="text-xl md:text-4xl  font-bold text-center mb-12 text-[#014e6c]">Сервисы </h2>
    <div id="servers" class="  grid  md:grid-cols-2 grid-cols-1 h-1/2  gap-6"></div>
</section>
```

Рис. 5. Инициализация контейнера для сервисов аэропорта

2. Запрос данных компонента: после загрузки страницы каждый компонент самостоятельно запрашивает данные для отрисовки с сервера через специальные API-маршруты. Например, для компонента «Сервисы» запрос отправляется по адресу http://nv-aero.ru/pages/componentsInfo/servers (рис. 6).

```
import { baseCardDarck, baseCardLight } from '/template/cardsBaseT.js';

(async function () {
  const response = await fetch('/pages/componentsInfo/servers');
  const datas = await response.json();
  const serversData = datas.serversData;
  const container = document.getElementById('servers');

  container.innerHTML = `
  <article class="space-y-5 flex flex-col h-full">
 <ul class="flex sm:flex-row  flex-col gap-5 h-full">
  <li class="w-full">
    ${baseCardDarck(serversData[0])}
  </li>
  <li >${baseCardLight(serversData[1])}  </li>
 </ul>
    <ul class="flex md:flex-row  flex-col gap-3 h-full">
      <li >${baseCardLight(serversData[2])}  </li>
      <li >${baseCardLight(serversData[3])}  </li>
      <li >${baseCardLight(serversData[4])}  </li>
    </ul>
    <article class="flex h-full w-full">
      ${baseCardDarck(serversData[6])}
    </article>
  </article>

    <article class="h-full w-full">
    ${baseCardDarck(serversData[5])}
    </article>
    `
})()
```

Рис. 6. Листинг кода компонента servers на странице home

3. Рендеринг компонента на клиенте: Получив данные от сервера, клиентский JavaScript:

- Парсит JSON-ответ.
- Генерирует HTML-структуру на основе полученных данных (рис. 7).
- Вставляет сгенерированный HTML в заранее подготовленный контейнер.
- Настраивает интерактивные элементы (кнопки, фильтры, сортировку).

```
export function baseCardLight(component) {
  return `
    <a href="${component.link ? component.link : "#"}" >
      <div class="
        relative text-black hover:text-white transition all duration-500
        bg-white rounded-x-full h-full
        overflow-hidden group cursor-pointer
      ">
        <div class="relative z-10 p-6 h-full flex flex-col">
          <h3 class="text-xl font-bold mb-3 group-hover:text-white transition-colors duration-300">
            ${component.title}
          </h3>
          ${component.description ? `
            <p class="text-gray-600 flex-grow group-hover:text-white/90 transition-colors duration-300">
              ${component.description}
            </p>
          ` : ""}
        </div>

        <div class="
          absolute inset-0 transition-all duration-700
          opacity-0 group-hover:opacity-100
          transform translate-y-5 group-hover:translate-y-0
          w-1/2
          left-1/2
        ">
          <img
            src="${component.icon}"
            alt="${component.title}"
            class="w-full h-full object-cover transform  group-hover:scale-100 transition-transform duration-1000"
          />
        </div>

        <div class="absolute inset-0 bg-[#0146c] opacity-0 group-hover:opacity-90 transition-opacity duration-500"></div>
      </div>
    </a>
  `
}
```

Рис. 7. Листинг кода светлого дизайна компонента servers на странице home

Такой архитектурный подход, сочетающий серверный рендеринг базовой структуры страницы и клиентский рендеринг динамических компонентов, обеспечивает оптимальный баланс между скоростью загрузки, интерактивностью и удобством поддержки кода [4].

Далее рассмотрим дизайн. Основные модули сайта. Главная страница: Информационный дашборд с виджетом онлайн-табло вылетов/прилетов и их статусом на данный момент. Динамическое расписание рейсов (рис. 8).

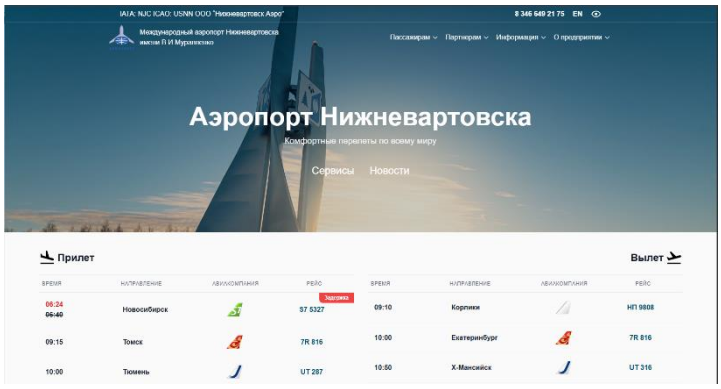


Рис. 8. Стартовая страница сайта аэропорта города Нижневартовска

Выпадающий список: для каждого раздела соответствующий набор информационных компонентов (рис. 9).

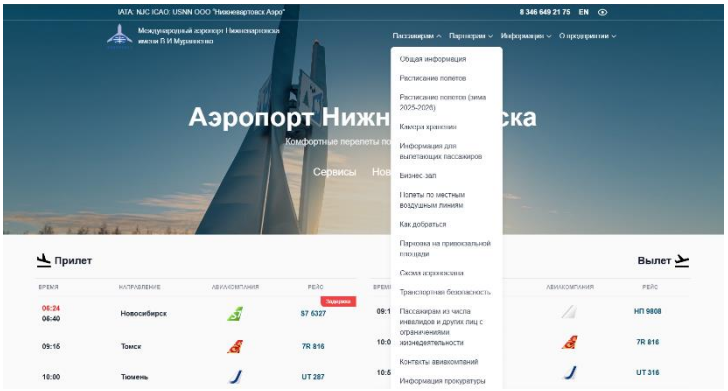


Рис. 9. Список разделов сайта аэропорта

Услуги аэропорта: Структурированный раздел с описанием услуг (парковка, VIP-залы, магазины), включая возможность онлайн-бронирования (рис. 10).

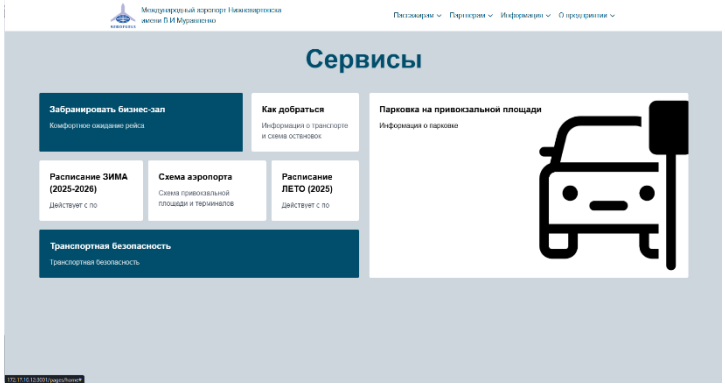


Рис. 10. Блок сервисов (услуг) сайта аэропорта города Нижневартовска

Новости: Блок актуальных новостей аэропорта (рис. 11).

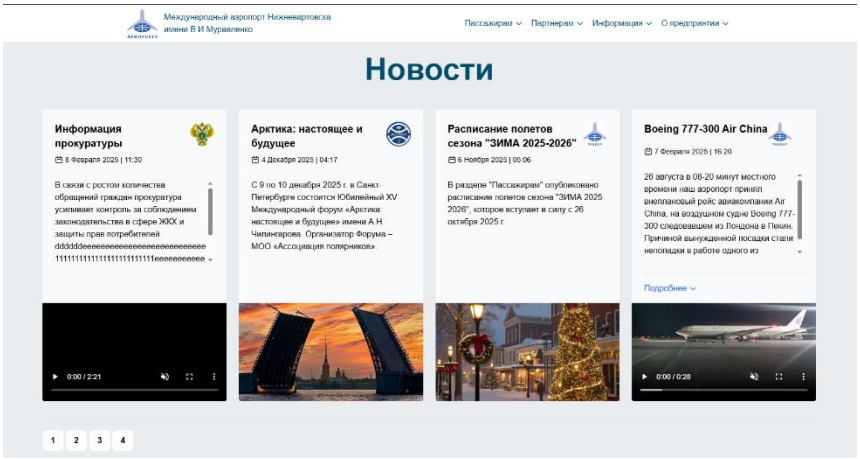


Рис. 11. Блок новостей сайта аэропорта

В заключение, разработанный прототип веб-сайт для аэропорта Нижневартовска представляет собой современное, производительное и масштабируемое решение, которое полностью соответствует потребностям пассажиров и сотрудников. Использование гибридного архитектурного подхода (SSR/CSR) в сочетании со стеком HTML/CSS/JS, EJS, Node.js и Docker [1-3] позволяет обеспечить высокую скорость разработки, удобство поддержки и отказоустойчивость системы. Гибкая микросервисная архитектура обеспечивает возможность дальнейшего расширения функциональности без переписывания существующего кода [5].

Литература

1. Иванов А.В. Современные подходы к разработке веб-приложений на Node.js // Вестник информационных технологий. 2023. № 4. С. 45-52.

2. Коваленко П.И. Гибридные подходы к рендерингу в современных веб-приложениях // Информационные системы и технологии. 2024. № 1(45). С. 23-31.

3. Петрова Е.С. Контейнеризация приложений с использованием Docker: практический опыт // Труды международной конференции «Информационные системы 2024». М.: Изд-во МГУ, 2024. С. 112-118.

4. Смирнов Д.К. Адаптивный дизайн и юзабилити транспортных веб-ресурсов // Интернет-маркетинг. 2022. № 6. С. 34-40.

5. Федоров М.И. Микросервисная архитектура в веб-разработке // Программная инженерия. 2023. Т. 14, № 2. С. 78-85.

Ван Юйбинь
Елецкий государственный университет им. И.А.Бунина
г. Елец, Россия

СПЕЦИАЛИЗИРОВАННОЕ ВЕБ-ПРИЛОЖЕНИЕ ДЛЯ РЕАЛИЗАЦИИ ОНЛАЙН-ОПРОСОВ

Актуальность разработки веб-приложений, адаптированных под конкретные задачи и условия эксплуатации обусловлено тем, что большинство действующих приложений ограничены в функциональности бесплатных версий и часто не отвечают специфическим требованиям пользователей, особенно в части безопасности данных, интеграции с внутренними системами или кастомизации интерфейса.

При выборе технологий для разработки веб-приложения для онлайн-опросов учитывались такие факторы, как производительность, удобство разработки, масштабируемость, безопасность и поддержка сообществом. В результате анализа был выбран стек технологий, включающий Node.js (бэкенд), Vue.js (фронтенд) и MySQL (база данных).

Бэкенд: Node.js

Node.js – среда выполнения JavaScript, построенная на движке V8 Google Chrome. она позволяет запускать JavaScript-код вне браузера, что делает ее идеальной для разработки серверной части веб-приложений [1].

Причины выбора:

1. Высокая производительность – использование неблокирующего ввода-вывода позволяет обрабатывать большое количество одновременных запросов без значительного увеличения ресурсов сервера.
2. Единый язык программирования – JavaScript для фронтенда и бэкенда упрощает разработку и интеграцию между компонентами.
3. Обширная экосистема пакетов – npm предоставляет доступ к огромному количеству готовых библиотек (Express.js, Passport.js, Joi).
4. Масштабируемость – легкая горизонтальная масштабируемость путем добавления новых серверов.
5. Активная поддержка сообществом – регулярные обновления и исправление уязвимостей.

В качестве веб-фреймворка для бэкенда был выбран Express.js – минималистичный и гибкий инструмент для создания RESTful API и обработки HTTP-запросов.

Фронтенд: Vue.js.

Vue.js – прогрессивный фреймворк для создания пользовательских интерфейсов. В отличие от монолитных фреймворков, он можно интегрировать постепенно или использовать для разработки полностью новых приложений [4].

Причины выбора:

1. Простота освоения – интуитивно понятный синтаксис и хорошо структурированная документация.

2. Гибкость – возможность использования только необходимых компонентов, снижение размера итогового приложения.

3. Высокая производительность – использование виртуального DOM минимизирует операции с реальным DOM.

4. Компонентный подход – создание переиспользуемых компонентов упрощает разработку и поддержку.

5. Поддержка сообществом – активное развитие и множество дополнительных библиотек (Vue Router, Vuex, Vuetify).

В качестве системы управления базами данных (СУБД) был выбран MySQL. Это одна из самых популярных реляционных СУБД с открытым исходным кодом, обладающая высокой надёжностью, производительностью и отличной поддержкой со стороны сообщества [2].

Проектирование базы данных

База данных состоит из таблиц (ключевые поля выделены), представленных в таблице 1.

Проектирование API

Для взаимодействия между клиентским и серверным слоями разработано RESTful API с эндпоинтами (ключевые запросы) [3], представленных в таблице 2.

Таблица 1

Основные таблицы базы данных веб-приложения

Таблица	Поля
users	id (PK), email (UNIQUE), password, name, role, created_at, updated_at
surveys	id (PK), title, description, creator_id (FK → users.id), start_date, end_date, status, created_at, updated_at
questions	id (PK), survey_id (FK → surveys.id), text, type, order, created_at, updated_at
options	id (PK), question_id (FK → questions.id), text, order, created_at, updated_at
responses	id (PK), survey_id (FK → surveys.id), user_id (FK → users.id), question_id (FK → questions.id), answer_text, created_at
response_options	id (PK), response_id (FK → responses.id), option_id (FK → options.id), created_at

Детальное проектирование реляционной модели данных Таблица «users» хранит учётные записи всех участников системы (табл. 3).

Таблица 2

Ключевые эндпоинты RESTful API приложения

Категория	Эндпоинт	Метод	Описание
Аутентификация	/api/auth/register	POST	Регистрация нового пользователя
	/api/auth/login	POST	Вход в систему
Управление опросами	/api/surveys	GET	Получение списка опросов
	/api/surveys	POST	Создание нового опроса
	/api/surveys/:id	PUT	Обновление опроса
	/api/surveys/:id	DELETE	Удаление опроса
Результаты	/api/surveys/:id/results	GET	Получение результатов опроса
	/api/surveys/:id/results/export	GET	Экспорт результатов в CSV/Excel

Таблица 3

Таблица «users»

Поле	Тип	Ограничения	Описание
id	SERIAL	PK	Уникальный идентификатор
email	VARCHAR(255)	UNIQUE, NOT NULL	Электронная почта
password_hash	VARCHAR(255)	NOT NULL	Хеш пароля (bcrypt)
full_name	VARCHAR(100)	NOT NULL	Полное имя
role	VARCHAR(20)	DEFAULT 'user'	Роль: 'user' или 'admin'
created_at	TIMESTAMP	DEFAULT NOW()	Дата регистрации

Примечание: Пароль никогда не хранится в открытом виде. При регистрации клиент отправляет пароль, сервер генерирует соль и создаёт хеш с помощью алгоритма bcrypt. При входе в систему процедура повторяется, и полученный хеш сравнивается с сохранённым. Таблица «surveys» описывает метаданные каждого опроса (табл. 4). Таблица «questions» содержит все вопросы, входящие в опросы (табл. 5).

Таблица 4

Таблица «surveys»

Поле	Тип	Ограничения	Описание
id	SERIAL	PK	Идентификатор опроса
title	VARCHAR(255)	NOT NULL	Название
description	TEXT		Описание
author_id	INTEGER	FK → users.id	Автор (ссылка hausers)
is_active	BOOLEAN	DEFAULT false	Статус публикации
starts_at	TIMESTAMP		Дата начала
ends_at	TIMESTAMP		Дата окончания
created_at	TIMESTAMP	DEFAULT NOW()	Дата создания

Таблица 5

Таблица «questions»

Поле	Тип	Ограничения	Описание
id	SERIAL	PK	Идентификатор вопроса
survey_id	INTEGER	FK → surveys.id	Принадлежность к опросу
text	TEXT	NOT NULL	Текст вопроса
type	VARCHAR(20)	NOT NULL	Тип: 'text', 'single', 'multiple', 'likert'
is_required	BOOLEAN	DEFAULT true	Обязательный ли вопрос
order_index	INTEGER	DEFAULT 0	Порядок отображения

Таблица «options» представляет собой варианты ответов для вопросов с выбором (табл. 6).

Таблица 6

Таблица «options»

Поле	Тип	Ограничения	Описание
id	SERIAL	PK	Идентификатор варианта
question_id	INTEGER	FK → questions.id	Связь с вопросом
text	VARCHAR(255)	NOT NULL	Текст варианта
order_index	INTEGER	DEFAULT 0	Порядок в списке

Таблица «responses» фиксирует факт отправки ответа на опрос (табл. 7).

Таблица 7

Таблица «responses»

Поле	Тип	Ограничения	Описание
id	SERIAL	PK	Идентификатор ответа
survey_id	INTEGER	FK → surveys.id	Опрос
respondent_id	INTEGER	FK → users.id	Респондент (может быть, NULL для анонимных)
submitted_at	TIMESTAMP	DEFAULT NOW()	Время отправки

Таблица «answers» хранит конкретные ответы на каждый вопрос (табл. 8).

Таблица 8

Таблица «answers»

Поле	Тип	Ограничения	Описание
id	SERIAL	PK	Идентификатор ответа
response_id	INTEGER	FK → responses.id	Связь с фактом отправки
question_id	INTEGER	FK → questions.id	Вопрос
option_id	INTEGER	FK → options.id	Выбранный вариант (для выбора)
text_value	TEXT		Текстовый ответ (для текстовых вопросов)

Такая структура позволяет гибко обрабатывать все типы вопросов и обеспечивает целостность данных за счёт внешних ключей. Например, невозможно создать ответ на несуществующий вопрос или привязать вариант к чужому опросу.

Пример SQL-запроса для получения результатов

Для генерации отчёта по опросу используется следующий запрос:

```
sql
SELECT
  q.text AS question,
  o.text AS option,
  COUNT(a.id) AS vote_count
FROM questions q
JOIN options o ON q.id = o.question_id
LEFT JOIN answers a ON o.id = a.option_id
WHERE q.survey_id = 42
GROUP BY q.id, o.id
ORDER BY q.order_index, o.order_index;
```

Этот запрос демонстрирует эффективность нормализованной модели: он легко расширяется для поддержки фильтрации по дате, респонденту или другим критериям.

Среда разработки

Перед началом разработки была настроена среда:

1. Установка Node.js, npm и Vue.js CLI.
2. Установка MySQL и создание базы данных по проектировке.
3. Установка необходимых пакетов:

– Бэкенд: Express.js, mysql2, jsonwebtoken, bcryptjs, joi.

– Фронтенд: Vue Router, Vuex, Axios, Vuetify, Chart.js.

4. Настройка Git для контроля версий.

Реализация бэкенда

Основные шаги:

1. Создание основного файла сервера (server.js), настройка Express.js и middleware (обработка JSON, CORS, логирование).
2. Подключение к MySQL через драйвер mysql2 (файл config/db.js).
3. Реализация модуля аутентификации: хеширование паролей (bcryptjs), создание JWT-токенов.
4. Реализация модулей управления опросами, вопросами и вариантами ответов (CRUD-операции).
5. Реализация модуля обработки ответов: валидация данных, сохранение в базу.
6. Реализация модуля аналитики: агрегация данных, формирование отчетов, экспорт в CSV/Excel.
7. Тестирование API с использованием Postman.

Реализация фронтенда

Основные шаги:

1. Создание проекта Vue.js с Vuetify для адаптивного интерфейса.
2. Реализация компонентов аутентификации: формы регистрации, входа и восстановления пароля.
3. Реализация компонентов администратор:

– Список опросов с фильтрацией и кнопками управления.

– Форма создания/редактирования опросов (добавление вопросов и вариантов ответов).

– Отображение результатов в виде графиков (Chart.js) и экспорт.

4. Реализация компонентов участника:

– Список активных опросов с поиском.

– Форма для прохода опросов (элементы ввода в зависимости от типа вопроса).

– История пройденных опросов и результаты.

5. Настройка маршрутизации (Vue Router) с ограничением доступа по роли.
6. Тестирование адаптивности и удобства использования.

Интеграция и тестирование

1. Настройка прокси в Vue.js для решения проблемы кросс-доменных запросов.
2. Интеграция фронтенда с бэкендом через Axios (обработка токенов, передача данных).
3. Комплексное тестирование:

– Функциональное тестирование: проверка всех функций и сценариев.

– Тестирование удобства использования: оценка интуитивности интерфейса.

– Тестирование производительности: проверка скорости обработки запросов.

Фрагмент листинга 1. App.aue

```
<template>
  <div id="app">
```

```
<el-container>
  <!-- Верхняя навигация -->
  <el-header>
    <div class="header-content">
      <div class="logo">
        <h2>Система онлайн-опросов</h2>
      </div>
      <div class="nav-menu">
        <el-menu
          :default-active="activeIndex"
          mode="horizontal"
          :ellipsis="false"
          @select="handleSelect"
        >
          <el-menu-item index="home" v-if="!isLoggedIn">
            <el-icon><House /></el-icon>
            <span>Главная</span>
          </el-menu-item>
```

Разработанное приложение может быть использовано образовательными учреждениями, компаниями и государственными органами для эффективного сбора данных и анализа результатов онлайн-опросов.

Литература

1. Берг М. Веб-разработка с применением Node.js и Express: практическое руководство / пер. с англ. Санкт-Петербург: Питер, 2023. 416 с.

2. Власовец А. MySQL. Основы программирования. М.: ДМК Пресс, 2021. 320 с.

3. Фримен Э. RESTful Web APIs: проектирование и разработка. М.: ДМК Пресс, 2022. 400 с.

4. Цицкин М. Разработка SPA с помощью Vue.js 3. М.: ДМК Пресс, 2023. 352 с.

Дворниченко Е.С.
Нижневартовский государственный университет
г. Нижневартовск, Россия

РАЗРАБОТКА САЙТА ДЛЯ КАФЕДРЫ ИНФОРМАТИКИ И МПИ

Управление образовательным учреждением сложный и многоступенчатый процесс в котором учувствуют как обучающиеся, так преподаватели, и родители.

Важным является применение современных информационных систем как один из элементов для решения управленческих задач. Сайт образовательного учреждения являясь сложной системой включает в себя много плановые задачи такие как информирование участников образовательного учреждения о различных научных и учебных мероприятий; Использование различных инструментов для педагогов и обучающихся; Инструменты для дистанционного взаимодействия всех участников образовательного процесса. Сложность сайта как компонента образовательного процесса требует использование современных инструментов и методик его разработки [5].

Разработка сайта началась с выбора средств разработки. Были выбраны [1]: HTML – язык разметки сайтов, CSS – каскадные таблицы стилей, PHP – язык, для backend логики, MySQL – реляционная система управления базами данных, JavaScript – язык для работы во frontend части сайта, в качестве редактора кода был выбран Visual Studio Code.

На главной странице необходимо размещать информацию, которую могут часто искать, но при этом она должна быть емкой, краткой и понятной для быстрого понимания [4].

Сайт оформлен в фирменном цвете университета – глубоком темно-синем, который формирует ощущение стабильности. Верхнюю часть страницы занимает панель с названием кафедры и основными разделами, обеспечивая удобный доступ к информации. Акцент ярко-синего цвета, используемый для интерактивных элементов – кнопок, ссылок и иконок придает интерфейсу современный вид. Дополнительные светлые оттенки в фонах и карточках смягчают контраст, делая восприятие информации комфортной для глаз.

Для обеспечения удобного доступа к актуальной информации о деятельности кафедры, учебных программах, преподавательском составе и новостях, сайт построен по модульному принципу и структурирован по вертикальной структуре, что позволяет быстро находить нужную информацию и хорошо в ней ориентироваться [3].

Была реализована возможность выкладывать лекции и другие материалы по учебным дисциплинам преподавателем, что позволяет обеспечить удобный и быстрый доступ к информации для студентов. А также для удобства проверки знаний пройденного материала добавлена возможность проведения тестирования преподавателем по дисциплине или занятию [2].

Первыми на сайте пользователя встречают новости (рис. 1). Для просмотра всего списка новостей можно нажать на надпись: «Больше новостей». Реализован отбор по типу новостей (рис. 2).

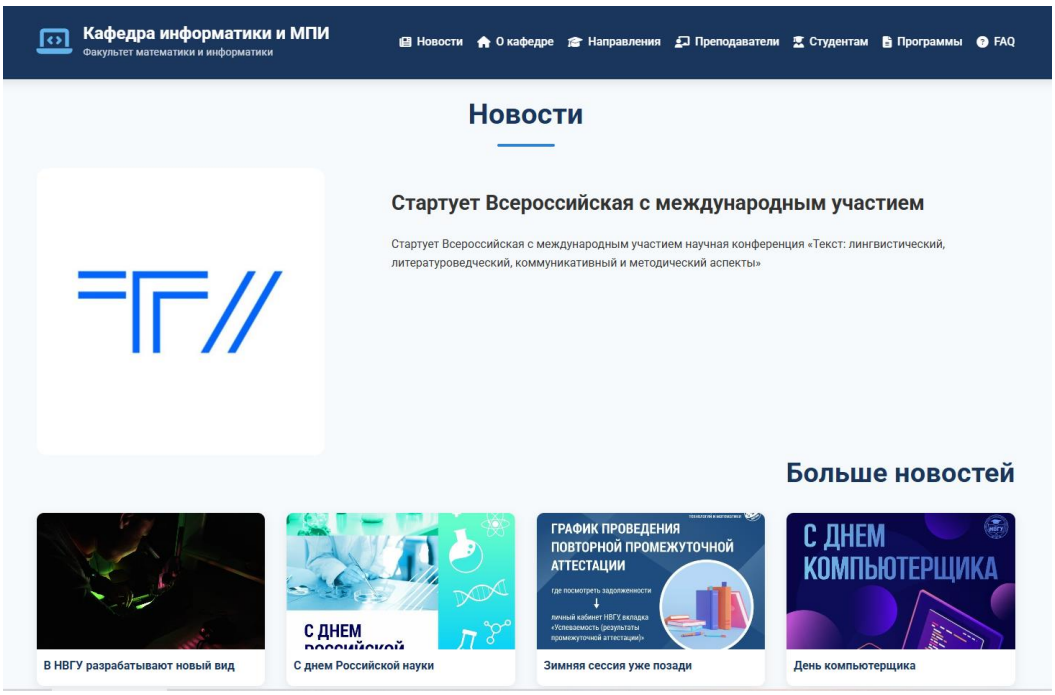


Рис. 1. Новости

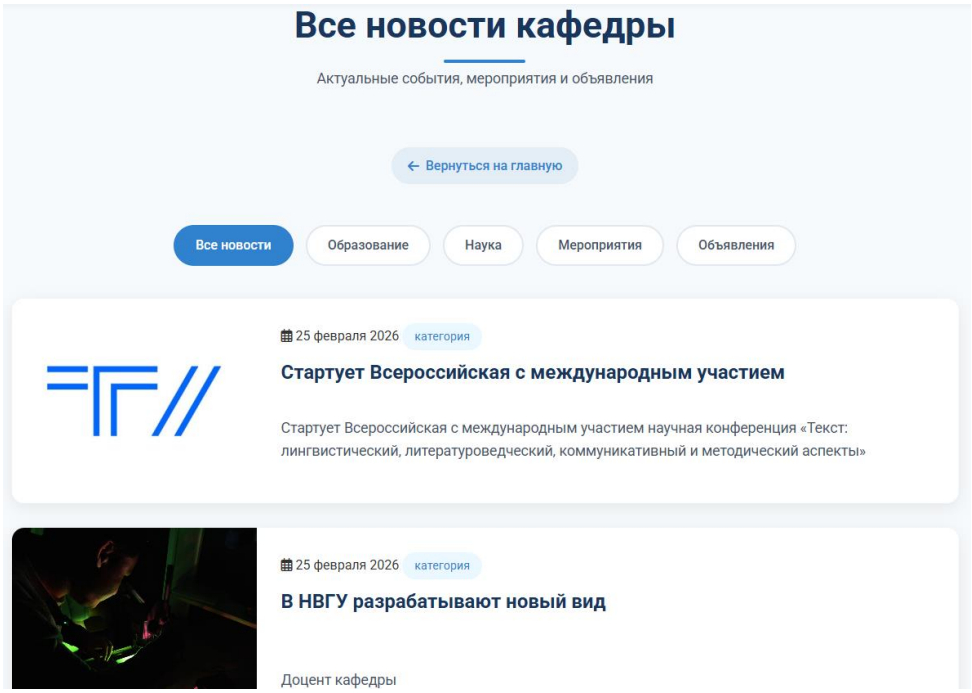


Рис. 2. Все новости кафедры

При выборе интересующей новости откроется страница с отображением полной новостной информацией (рис. 3).

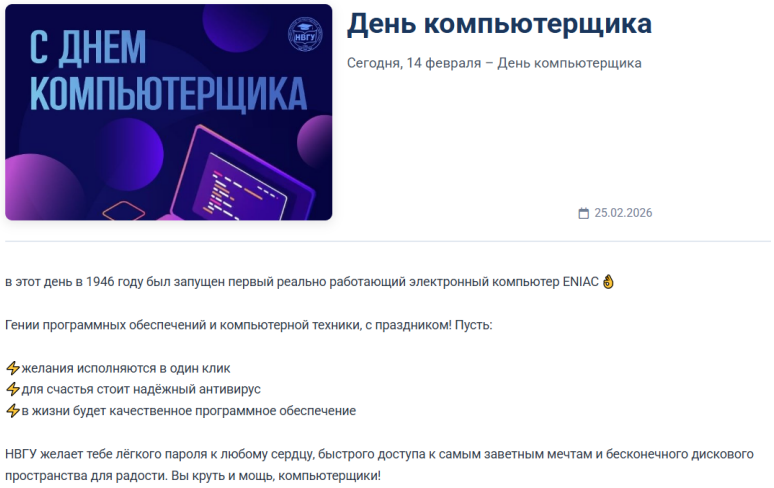


Рис. 3. Просмотр новости

После новостей идет раздел «О кафедре», в нем содержится описание кафедры (рис. 4).

Рис. 4. Раздел «О кафедре»

На сайте также представлены направления подготовки, реализован отбор по уровню образованию и форме обучения (рис. 5).

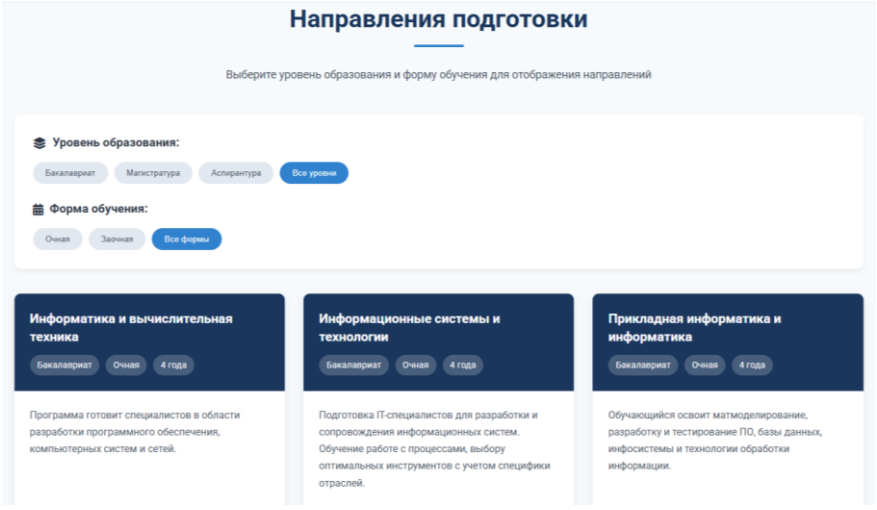


Рис. 5. Раздел «Направления подготовки»

На основной странице располагаются раздел преподавателей кафедры (рис. 6), в нем присутствует их ФИО, фотография, должность и преподаваемые дисциплины, что помогает абитуриентам познакомиться с будущими наставниками, а студенты могут уточнить к какому из преподавателей обращаться по конкретному вопросу. После идет информация для студентов (рис. 7). Также в раздел «Программы» добавлены официальные документы образовательных программ (рис. 8). Раздел «Часто задаваемые вопросы» содержит популярные вопросы о кафедре (рис. 9).

Рис. 6. Раздел преподавателей

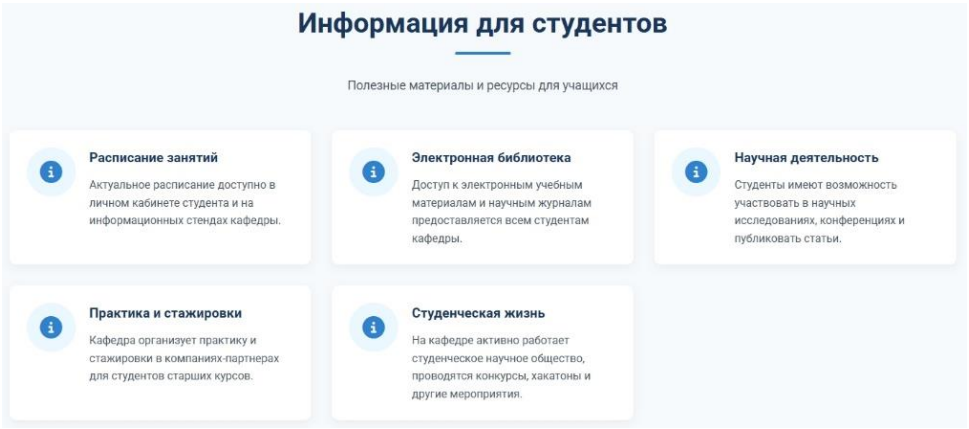


Рис. 7. Раздел «Информация для студентов»

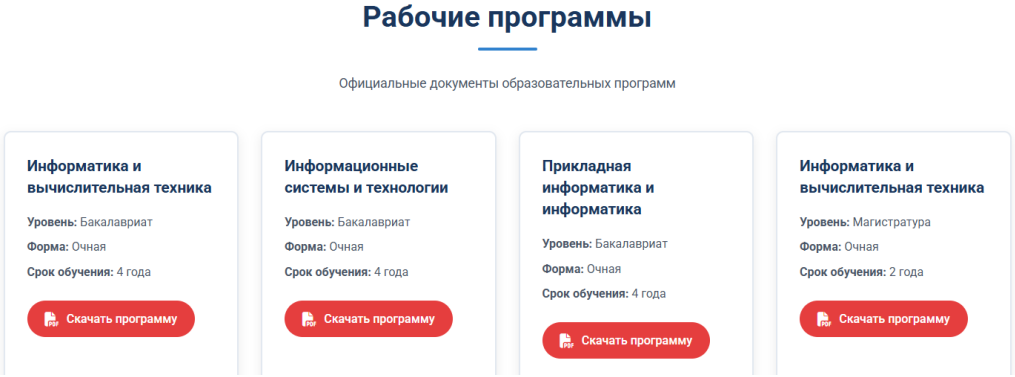


Рис. 8. Раздел «Рабочие программы»

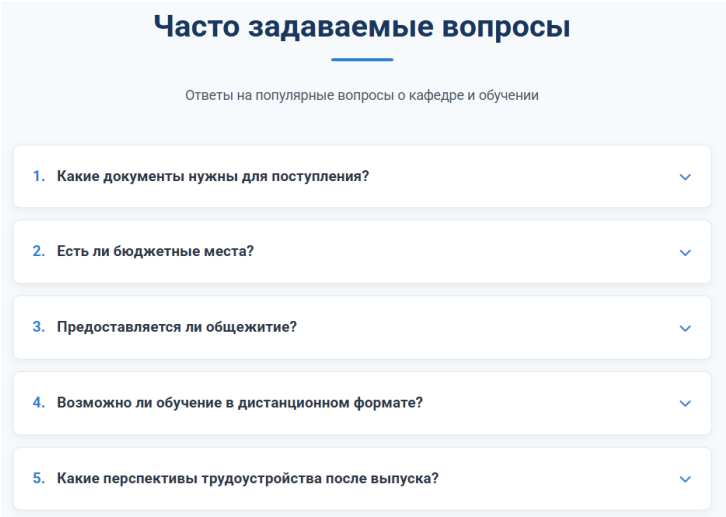


Рис. 9. Раздел «Часто задаваемые вопросы»

Реализована возможность администрирования сайта. Войдя, как администратор, можно добавить или изменить новости, с добавлением изображения и краткого описания, в каждой новости есть список блоков с возможностью добавления контента изображений, текста, списка (рис. 10). Есть возможность добавления и изменения педагогов, их фото, ФИО, должность и описание (рис. 11).

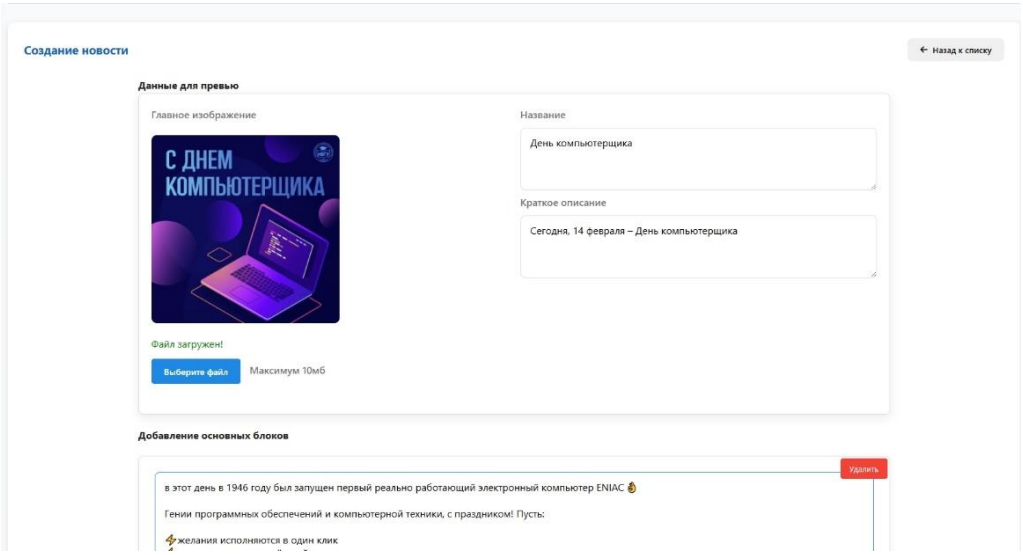


Рис. 10. Создание новости

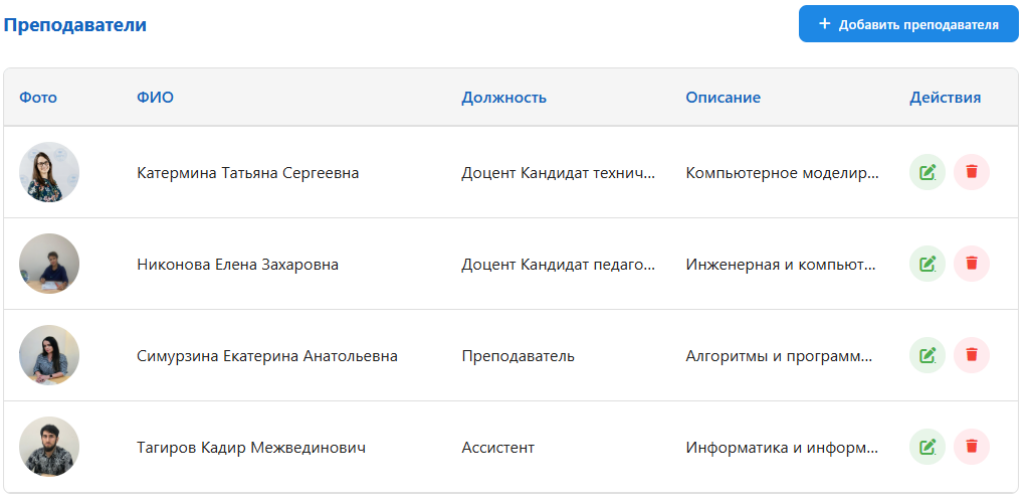


Рис. 11. Преподаватели

Таким образом, получилось реализовать гостевой доступ и доступ администратора для сайта кафедры информатики и МПИ что обеспечивает быстрый доступ к ключевой информации для абитуриентов и студентов. Структурно главная страница разделена на тематические блоки, что обеспечивает быстрый поиск необходимой информации.

Педагоги могут создавать свои предметы, выкладывать лекции и практики, а также проводить тестирования. У кураторов имеется возможность выкладывать актуальную информацию и работать с группой. А студенты могут это все изучать.

Литература

1. Калинников Д.И. Анализ и исследование инструментов backend-разработки web-приложений // Тенденции развития науки и образования. 2023. № 101-4. С. 84-88. https://doi.org/10.18411/trnio-09-2023-177.

2. Мусорина О.А. Использование потенциала сайта кафедры для оптимизации самостоятельной работы студентов // Организация самостоятельной работы студентов по иностранным языкам. 2023. № 6. С. 176-180.

3. Ткачёв Е.В., Белаш В.Ю. К вопросу о структуре образовательного веб-сайта // Проблемы современного педагогического образования. 2023. № 79-4. С. 230-232.

4. Турдубаева Ж.А., Исманов О.М. Разработка современного веб-сайта для кафедры информационных технологий и управления // Бюллетень науки и практики. 2023. Т. 9. № 11. С. 305-309. https://doi.org/10.33619/2414-2948/96/39

5. Шарова А.Ю., Матрохин А.Ю. Опыт разработки информационной системы вуза // Технологии и качество. 2022. № 4 (58). С. 25-31. https://doi.org/10.34216/2587-6147-2022-4-58-25-31.

Епифанов Д.Е.
Нижневартовский государственный университет
г. Нижневартовск, Россия

СРАВНИТЕЛЬНЫЙ АНАЛИЗ АРХИТЕКТУРНЫХ ПОДХОДОВ В СОВРЕМЕННЫХ ВЕБ-ПРИЛОЖЕНИЯХ: МОНОЛИТ, МИКРОСЕРВИСЫ И МОДЕЛИ РЕНДЕРИНГА (SSR, CSR, ГИБРИДНЫЕ СТРАТЕГИИ)

Современные веб-приложения занимают ключевое место в цифровой инфраструктуре общества. Они используются в сфере электронной коммерции, банковских сервисов, образования, промышленности и государственного управления. Рост объёма обрабатываемых данных, увеличение числа пользователей и усложнение бизнес-логики приводят к необходимости выбора эффективных архитектурных решений при разработке веб-приложений. От корректности выбранной архитектуры напрямую зависят производительность системы, её масштабируемость и устойчивость к отказам [1].

На ранних этапах развития веб-технологий широко применялась монолитная архитектура, которая объединяла все компоненты приложения в единую систему. Однако с ростом требований к гибкости и масштабированию всё больше популярны микросервисные решения [1; 3], основанные на разделении функциональности на независимые сервисы. Параллельно развивается подход к организации пользовательского интерфейса, включая клиентскую и серверную модели рендеринга, а также их гибридные варианты.

Выбор архитектурного подхода определяется спецификой проекта, предполагаемой нагрузкой, требованиями к безопасности и скорости разработки. Ошибки на этапе проектирования архитектуры могут привести к увеличению затрат на сопровождение и модернизацию системы. В связи с этим актуальным является рассмотрение современных архитектурных моделей веб-приложений, их особенностей и областей применения.

Целью данной статьи является анализ основных архитектурных подходов, применяемых при разработке современных веб-приложений, а также рассмотрение их преимуществ и ограничений в практической деятельности.

Монолитная архитектура веб-приложений

Монолитная архитектура является одним из первых и наиболее распространённых подходов к разработке веб-приложений. В рамках данной модели все компоненты системы объединяются в единое приложение и развёртываются как целостная система.

Рис. 1. Иллюстрация структуры монолитной архитектуры

Как правило, монолитное приложение представляет собой единую кодовую базу, в которой различные функциональные модули взаимодействуют напрямую. Развертывание осуществляется в виде одного исполняемого сервиса или набора тесно связанных компонентов. Такой подход упрощает начальный этап разработки, поскольку структура проекта остаётся централизованной, а взаимодействие между модулями реализуется без сетевых вызовов.

К основным особенностям монолитной архитектуры относятся:

- единая структура проекта и общая кодовая база;
- централизованное управление бизнес-логикой;
- использование общей базы данных;
- развертывание в виде одного приложения.

Преимуществами монолитного подхода являются относительная простота разработки, удобство отладки и тестирования, а также снижение требований к инфраструктуре на начальном этапе проекта. Для небольших и средних систем монолитная архитектура позволяет быстро реализовать функциональность без необходимости сложной настройки взаимодействия между сервисами.

Однако с ростом объёма кода и увеличением числа пользователей могут возникать сложности.

К числу ограничений монолитной архитектуры относятся:

- затруднённая масштабируемость отдельных компонентов;
- высокая зависимость модулей друг от друга;
- увеличение времени сборки и развертывания;
- усложнение сопровождения при росте проекта [1; 3].

Таким образом, монолитная архитектура остаётся актуальной для проектов с ограниченной сложностью, однако при увеличении нагрузки и расширении функциональности могут потребоваться более гибкие решения.

Микросервисная архитектура веб-приложений

Микросервисная архитектура представляет собой подход к разработке веб-приложений, при котором система разделяется на набор независимых сервисов, каждый из которых

отвечает за выполнение отдельной бизнес-функции. В отличие от монолитной архитектуры, компоненты приложения развёртываются и функционируют автономно, взаимодействуя друг с другом через сетевые протоколы, например HTTP/REST, gRPC.

Рис. 2. Иллюстрация структуры микросервисной архитектуры

Каждый микросервис имеет собственную область ответственности, может использовать отдельную базу данных и разрабатываться независимо от других компонентов системы. Взаимодействие между сервисами осуществляется посредством API, чаще всего с использованием протоколов HTTP/HTTPS или систем обмена сообщениями.

К основным особенностям микросервисной архитектуры относятся:

- разделение приложения на независимые сервисы;
- автономное развертывание компонентов;
- использование API для взаимодействия;
- возможность применения различных технологий в рамках одного проекта;
- распределённое хранение данных.

Преимуществами данного подхода являются высокая масштабируемость, гибкость разработки и устойчивость к отказам отдельных компонентов [1]. При увеличении нагрузки возможно масштабирование только тех сервисов, которые испытывают повышенную нагрузку, что позволяет эффективнее использовать вычислительные ресурсы. Кроме того, обновление одного сервиса не требует остановки всей системы.

Вместе с тем микросервисная архитектура предъявляет повышенные требования к инфраструктуре и организации разработки.

К её ограничениям можно отнести:

- усложнение настройки сетевого взаимодействия;
- необходимость мониторинга и управления большим количеством сервисов;
- возможные проблемы согласованности данных;
- увеличение требований к DevOps-практикам [3].

Таким образом, микросервисная архитектура целесообразна при разработке крупных и высоконагруженных систем, где важны масштабируемость и гибкость, однако её применение должно быть обосновано сложностью проекта.

Таблица

Основные архитектурные отличия монолита и микросервисов

Параметры	Монолитная архитектура	Микросервисная архитектура
Размер и сложность	Монолитная архитектура представляет собой одно большое приложение, в котором все компоненты связаны друг с другом напрямую	В микросервисной архитектуре приложение разбивается на множество микросервисов, каждый из которых отвечает за конкретную функциональность.
Гибкость и масштабируемость	В монолитной архитектуре, изменения и масштабирование требуют работы со всем приложением, что может быть неэффективным и сложным	Микросервисная архитектура даёт большую гибкость при разработке и поддержке приложения. Каждый микросервис может быть разработан, протестирован и развёрнут независимо от других.
Надежность	В монолитной архитектуре, если одна часть приложения отказывает, это может привести к отказу всего приложения	В микросервисной архитектуре, если один микросервис перестает работать, остальные микросервисы продолжают функционировать независимо
Сложность развертывания и управления	Монолитная архитектура может быть более простой в развертывании и управлении, поскольку ее элементы не связаны сложной системой зависимостей	Управление микросервисной архитектурой может быть сложнее, так как требуется управлять несколькими независимыми сервисами и их взаимодействием

Подобные архитектурные различия и их влияние на масштабируемость подтверждаются результатами современных исследований [1; 3].

Подходы к рендерингу веб-приложений

Помимо выбора серверной архитектуры, при разработке современных веб-приложений важную роль играет способ формирования пользовательского интерфейса. От модели рендеринга зависит скорость загрузки страницы [2; 5], нагрузка на сервер и клиентское устройство, а также возможности поисковой оптимизации. В настоящее время широко применяются клиентская и серверная модели рендеринга, а также их комбинированные варианты.

Client-Side Rendering (CSR)

Client-Side Rendering (клиентский рендеринг) представляет собой подход, при котором основная часть логики формирования интерфейса выполняется в браузере пользователя. Сервер передаёт клиенту минимальный HTML-документ и набор JavaScript-файлов, после чего построение страницы осуществляется на стороне пользователя.

При использовании данного подхода браузер загружает скрипты, которые динамически формируют структуру страницы, выполняют запросы к серверу и обновляют содержимое без полной перезагрузки. Такая модель широко применяется при создании одностраничных приложений (SPA).

К основным особенностям CSR относятся:

– формирование интерфейса на стороне клиента;

– активное использование JavaScript;

– динамическое обновление контента без перезагрузки страницы;

– снижение нагрузки на сервер при повторных действиях пользователя.

Преимуществами клиентского рендеринга являются высокая интерактивность интерфейса и удобство реализации сложных пользовательских сценариев. После первоначальной загрузки приложения взаимодействие с сервером ограничивается передачей данных, что может повысить отзывчивость интерфейса.

К ограничениям CSR относятся увеличение времени первоначальной загрузки, зависимость от производительности устройства пользователя, а также возможные сложности с поисковой оптимизацией [4; 5].

Server-Side Rendering (SSR)

Server-Side Rendering (серверный рендеринг) предполагает формирование готовой HTML-страницы на сервере до её передачи клиенту. В этом случае браузер получает уже сгенерированный документ, содержащий основные элементы интерфейса.

При каждом обращении пользователя сервер обрабатывает запрос, формирует страницу с учётом данных и отправляет её клиенту. Такой подход позволяет обеспечить более быструю первичную отрисовку страницы и повысить доступность контента для поисковых систем [2; 5].

К особенностям SSR относятся:

– генерация HTML на стороне сервера;

– улучшенная индексация поисковыми системами;

– снижение требований к вычислительным ресурсам клиента;

– увеличение нагрузки на сервер.

Преимуществами серверного рендеринга являются более быстрый первый показ страницы и лучшая совместимость с устройствами различной производительности. Это особенно важно для информационных порталов и сервисов с большим объёмом контента.

Вместе с тем данный подход требует более мощной серверной инфраструктуры и может усложнять масштабирование при высокой нагрузке.

Гибридные модели рендеринга

В современных веб-приложениях всё чаще применяются комбинированные подходы, объединяющие преимущества CSR и SSR. К ним относятся статическая генерация страниц, инкрементальное обновление и частичный серверный рендеринг [5].

Гибридные модели позволяют заранее формировать часть страниц, а динамические данные обновлять по мере необходимости. Это снижает нагрузку на сервер и одновременно сохраняет высокую скорость отображения контента.

В рамках данной статьи были рассмотрены современные архитектурные подходы, применяемые при разработке веб-приложений. Проанализированы особенности монолитной и микросервисной архитектур, а также основные модели рендеринга пользовательского интерфейса.

Таким образом, выбор архитектурной модели и способа рендеринга должен осуществляться с учётом целей проекта, предполагаемой нагрузки и требований к масштабируемости. Грамотное проектирование архитектуры на начальном этапе разработки позволяет снизить нагрузку на сопровождение и повысить устойчивость веб-приложения в его работе.

Литература

1. Бакаикина В.Г., Богатина В.А., Федоров В.М. Микросервисная архитектура vs monolith: сравнительный анализ на примере веб-приложения // Парадигма. 2025. № 11-4. С. 65-68.

2. Бондаренко О.С., Смирнов А.А., Вдовин В.С. Сравнительный анализ современных методов рендеринга веб-приложений и их влияния на производительность // Международный журнал информационных технологий и энергоэффективности. 2024. Т. 9. № 6(44). С. 113-121.

3. Василенко С.В. Переход от монолитной архитектуры к микросервисной архитектуре веб-приложения // Наука и инновации XXI века: сборник статей по материалам IX Всероссийской конференции молодых ученых (г. Сургут, 02 ноября 2022 года). В 4-х т. Т. I. Сургут: Сургутский государственный университет, 2023. С. 205-209.

4. Евдокимов А.О. Анализ и сравнение клиентского рендеринга (CSR) и серверного рендеринга (SSR) в современных веб-приложениях // Студенческая наука: созидая будущее: сборник научных статей I внутривузовской студенческой научно-практической конференции (г. Курск, 08–09 июня 2023 года). Курск: Курский государственный университет, 2023. С. 143-148.

5. Стефанова И.А., Сергеев М.С. Современные подходы к ускорению загрузки веб-приложений: сравнительный анализ методов рендеринга и их влияние на ключевые метрики // Развитие науки и практики в глобально меняющемся мире в условиях рисков: материалы XXXVII Международной научно-практической конференции (г. Москва, 23 мая 2025 года). М.: Университет ИТБО, 2025. С. 230-238.

Ибрагимов В.Ш.
Нижневартовский государственный университет
г. Нижневартовск, Россия
Подбережский Д.В.
Российский технологический университет
г. Москва, Россия

АРХИТЕКТУРНЫЕ ПАТТЕРНЫ ИНТЕГРАЦИИ ГЕТЕРОГЕННОГО ОБОРУДОВАНИЯ В MES-СИСТЕМАХ АДДИТИВНОГО ПРОИЗВОДСТВА

В условиях стремительного развития парадигмы «Индустрия 4.0» и масштабирования аддитивного производства (3D-печати) критическое значение приобретает автоматизация цеховых процессов. Ключевым элементом цифровизации предприятия выступают системы управления производственными процессами (Manufacturing Execution System, MES). MES-система представляет собой информационное связующее звено между уровнем планирования (ERP) и физическим оборудованием. Современные стратегии развития предприятий все чаще опираются на концепцию ценностно-ориентированного менеджмента (Management by Value, MBV) [1]. В рамках подхода MBV любые управленческие решения оцениваются через призму их влияния на итоговую ценность продукта и эффективность бизнес-процессов. Однако успешная реализация данной методологии невозможна без непрерывного потока достоверных, объективных данных непосредственно с производственных участков. Именно поэтому глубокая интеграция программного обеспечения с оборудованием (Industrial IoT) становится фундаментальным требованием.

На практике процесс подключения гетерогенного (разнородного) оборудования к единой информационной среде сопряжен с серьезными архитектурными трудностями [3]. В традиционной тяжелой промышленности проблема «зоопарка» устройств и проприетарных протоколов зачастую решается за счет внедрения унифицированных стандартов промышленной связи, таких как архитектура OPC UA (Open Platform Communications Unified Architecture). Применение OPC UA позволяет системе управления абстрагироваться от специфики аппаратного обеспечения и обмениваться данными по единому стандартизированному контракту.

Тем не менее, специфика сферы аддитивных технологий и вспомогательного периферийного оборудования (электронных весов, термопринтеров этикеток) такова, что производители далеко не всегда обеспечивают встроенную поддержку промышленных M2M-стандартов уровня OPC UA. На рынке 3D-печати доминируют узкоспециализированные прошивки (Klipper, Marlin, закрытые экосистемы наподобие BambuLab), использующие собственные, несовместимые между собой протоколы передачи данных (WebSocket, MQTT, Serial/USB). В ситуациях, когда производитель оборудования не предусмотрел унифицированный интерфейс интеграции, применение классических подходов становится невозможным.

Решением проблемы прозрачной интеграции нестандартизированного оборудования может стать MES-система, в основе которой лежит микросервисная парадигма с четким разделением зон ответственности между управлением производственными процессами и

низкоуровневым взаимодействием с оборудованием. Данный подход концептуально соотносится с международным стандартом автоматизации предприятий ISA-95, а также отвечает современным архитектурным подходам к интеграции данных с гетерогенных уровней управления в парадигме Индустрии 4.0 [4]. Согласно иерархической модели ISA-95, центральный модуль разрабатываемой системы (Core) функционирует на Уровне 3 (Управление производственными операциями – MES), тогда как физическое оборудование и его прошивки (Klipper, BambuLab, драйверы весов) находятся на Уровне 2 (Диспетчерское управление и контроль). Разработанные модули интеграции выступают в роли интеллектуального связующего звена (middleware), обеспечивающего прозрачный обмен данными между этими уровнями.

Для обеспечения максимальной гибкости и независимости центрального ядра от гетерогенной аппаратной среды был применен архитектурный паттерн «Порты и адаптеры». Главный принцип данного паттерна заключается в том, что ядро приложения (Core-модуль), инкапсулирующее бизнес-логику и хранящее состояния в базе данных PostgreSQL (Stateful), полностью изолировано от внешних агентов. Ядро не содержит специфичного кода для работы с конкретными 3D-принтерами или электронными весами. Вместо этого оно декларирует унифицированные интерфейсы (порты), а взаимодействие с реальным миром делегируется внешним модулям (адаптерам).

В контексте описываемой системы роль таких адаптеров выполняют изолированные сервисы интеграции (Equipment Service и Printer Service), функционирующие как автономные модули. Ключевой особенностью данных сервисов является динамическое конфигурирование: они не хранят персистентные данные локально, признавая Core-модуль единым источником истины (Single Source of Truth). При запуске интеграционный модуль по HTTP-протоколу запрашивает у ядра параметры, необходимые для подключения к назначенному пулу оборудования (IP-адреса, порты, ключи авторизации). Получив конфигурацию, сервис самостоятельно устанавливает постоянное соединение с физическим устройством (например, по протоколу WebSocket для прошивки Klipper) и берет на себя управление сессией, ретрансляцию состояний в In-Memory кэш (Redis) и генерацию бизнес-событий в очередь (RabbitMQ). Такая декомпозиция предоставляет ряд существенных инженерных и эксплуатационных преимуществ таких как: горизонтальное масштабирование (Scale-Out), топологическая гибкость и бесшовная модернизация, что соответствует современным концепциям построения распределенных MES-систем [5].

Фундаментом надежности такой распределенной системы выступает контрактное взаимодействие (Contract-First API). Между ядром и интеграционными сервисами устанавливается жесткий программный контракт, регламентирующий структуру JSON-сообщений. Ядру системы не важно, с какой прошивкой оно взаимодействует: оно отправляет стандартизированную абстрактную команду (например, START_PRINT) с набором стандартизированных параметров. В свою очередь, трансляция телеметрии в каналы Redis и публикация событий (начало/конец печати, расход филамента) в RabbitMQ осуществляется интеграционным сервисом строго в рамках согласованных структур данных.

Подобная архитектура сводит к минимуму связность (coupling) компонентов, обеспечивая высокую отказоустойчивость системы управления аддитивным производством.

Для обеспечения работы вспомогательных производственных процессов, таких как взвешивание расходных материалов и печать идентификационных QR-этикеток (для катушек с филаментом, лотков и ячеек хранения), был разработан специализированный микросервис – Equipment Service. Данный компонент транслирует высокоуровневые HTTP-запросы от ядра системы (Core) в низкоуровневые аппаратные команды (Serial/USB).

Программная реализация модуля выполнена на базе асинхронного веб-фреймворка FastAPI. Архитектурно сервис построен на принципах доменно-ориентированного проектирования (Domain-Driven Design, DDD) и строгой изоляции слоев (Clean Architecture). Логика приложения декомпозирована на два независимых предметных домена: управление принтерами этикеток (printers) и управление электронными весами (scales).

Ключевой особенностью разработанного сервиса является его полностью stateless-архитектура (отсутствие сохранения состояния). Equipment Service не имеет собственной базы данных и не хранит персистентные конфигурации подключенных устройств. Реализован подход динамического внедрения параметров: при каждом обращении центральная MES-система (Core) передает в теле HTTP-запроса не только полезную нагрузку (например, текст для печати), но и унифицированную схему DeviceConnection, содержащую тип интерфейса (USB или Ethernet) и адрес для подключения (например, /dev/ttyUSB0). Это делает каждый запрос полностью самодостаточным и позволяет разворачивать любое количество экземпляров сервиса без необходимости их взаимной синхронизации.

Для решения проблемы аппаратной гетерогенности (поддержки различных производителей и прошивок) на уровне взаимодействия с оборудованием был применен структурный паттерн «Абстрактная фабрика» (Abstract Factory). Внутри каждого домена определены базовые абстрактные классы-контракты (PrinterDriver, ScaleDriver). Процесс обработки запроса выглядит следующим образом:

- Сервис получает HTTP-запрос от Core-модуля, проходит проверку авторизации (межсервисная авторизация при помощи токена) и валидацию структур данных.
- На основе переданного типа устройства фабричный метод динамически инстанцирует конкретную реализацию драйвера. Например, для термопринтеров инициализируется TSPLDriver (TSPL – язык команд производителя термопринтеров TSC), а для весов – MassaKDriver или MeRDriver.
- Драйвер выполняет специфичную для устройства логику: TSPLDriver предварительно генерирует растровое изображение (bitmap) этикетки с QR-кодом, формирует пакет TSPL-команд и отправляет их в последовательный порт, тогда как MassaKDriver считывает байтовый поток из порта, парсит проприетарный протокол весов и возвращает нормализованное значение веса в граммах.
- Сформированный результат возвращается в ядро системы в виде стандартизированного JSON-ответа.

Подобный подход обеспечивает соответствие принципу открытости/закрытости (Open-Closed Principle из SOLID). Добавление поддержки весов от нового производителя не требует изменения бизнес-логики сервиса или ядра системы. Разработчику достаточно создать новый класс драйвера, наследующий базовый контракт ScaleDriver, и зарегистрировать его в фабричном методе маршрутизатора.

Дополнительным уровнем надежности выступает централизованная система обработки аппаратных исключений (Middleware). Поскольку взаимодействие с физическими портами (Serial/USB) подвержено нестабильности, сервис изолирует аппаратные ошибки (обрывы связи, таймауты портов) от бизнес-логики, автоматически конвертируя их в соответствующие HTTP-статусы с детализированным JSON-описанием проблемы. Это позволяет центральному ядру MES-системы корректно обрабатывать отказы периферии, не нарушая общий цикл управления производством.

С точки зрения интеграции в единую производственную среду, 3D-принтер представляет собой сложный киберфизический узел, требующий двунаправленного информационного обмена [2]. С одной стороны, он выступает конечным исполнителем управляющих директив. С другой стороны, в процессе работы он генерирует непрерывный высокочастотный поток статусов, отражающих его фактическое функционирование (телеметрия). Однако при построении MES-системы и попытке автоматизировать этот двунаправленный обмен разработчики сталкиваются с двумя фундаментальными архитектурными проблемами, требующими специфических инженерных решений.

Проблема 1: Отсутствие единого стандарта межмашинного взаимодействия

Попытка реализовать универсальный интеграционный сервис для всех типов прошивок приводит к концентрации разнородной логики внутри одного программного модуля. В таком сервисе неизбежно формируется множественная ветвящаяся обработка, зависящая от типа прошивки, что нарушает принцип единственной ответственности (Single Responsibility Principle), увеличивает связанность (coupling) компонентов системы, усложняет тестирование и дальнейшее сопровождение кодовой базы, повышает риск возникновения регрессионных ошибок при добавлении поддержки оборудования от нового производителя.

Для обхода этих ограничений в разработанной архитектуре был принят строгий принцип: «одна прошивка – один сервис интеграции». Каждый такой сервис реализует единый, стандартизированный API-контракт взаимодействия с центральным Core-модулем, но при этом полностью инкапсулирует специфичную логику работы с конкретной прошивкой, скрывая низкоуровневые детали от остальной системы

Проблема 2: Семантический разрыв между аппаратными данными и бизнес-логикой

3D-принтер генерирует плотный поток «сырых» аппаратных данных, которые без логической интерпретации лишены управленческой ценности. Поскольку ядро MES-системы (Core) функционирует в рамках событийно-ориентированной архитектуры (Event-Driven Architecture), ему не требуется пуллинг ежесекундных координат экструдера. Вместо этого бизнес-логика опирается на высокоуровневые события (начало и конец печати или

использование материалов), что требует конвертации низкоуровневых статусов в осмысленные контракты сообщений.

Для преодоления этого семантического разрыва разработанный микросервис интеграции выполняет роль интеллектуального шлюза (Smart Gateway). Он не просто транслирует данные, а производит их первичную агрегацию, фильтрацию и строгую маршрутизацию. Чтобы обеспечить баланс между надежностью бизнес-процессов и производительностью системы, информационный обмен был декомпозирован на три независимых канала связи. Каждый канал использует оптимальный транспортный протокол, строго соответствующий характеру передаваемых данных:

1. **Директивное управление (REST API)**

Данный канал предназначен для передачи управляющих директив от Core-модуля к оборудованию.

2. **Гарантированная доставка бизнес-событий (RabbitMQ)**

Этот канал выступает фундаментом событийно-ориентированного взаимодействия. Внутри интеграционного сервиса реализован программный конечный автомат (FSM – Finite State Machine), который непрерывно анализирует асинхронный поток аппаратных статусов принтера.

3. **Высокочастотная потоковая телеметрия (Redis Pub/Sub)**

Канал выделен исключительно для эфемерных данных, которые имеют высокую частоту обновления, но быстро теряют свою актуальность (текущая температура экструдера и стола, процент завершения печати, номер текущего слоя). Сервис интеграции применяет стратегию «Last-write-wins» и публикует нормализованные метрики в in-memory шину Redis Pub/Sub. Это позволяет подписываться, на топик телеметрии, неограниченному количеству клиентов и сервисов.

Управление пулом оборудования и изоляция отказов

Интеграционный сервис не хранит персистентную конфигурацию оборудования. При запуске, а также с заданной периодичностью в фоновом режиме, микросервис запрашивает у центрального ядра MES актуальный список назначенных ему 3D-принтеров. На основе полученных данных внутри сервиса динамически создаются изолированные программные контексты для каждого физического устройства. Каждый такой контекст обладает независимым пулом соединений (WebSocket для телеметрии и HTTP для команд), собственным экземпляром конечного автомата (FSM) и автономным циклом обработки ошибок. Подобная аппаратная абстракция гарантирует, что сбой в сети, зависание прошивки или экстренная перезагрузка одного 3D-принтера приведет лишь к локальному циклу переподключения (с использованием механизма Exponential Backoff) внутри его контекста, абсолютно не влияя на стабильность обмена данными с остальным производственным парком.

Декомпозиция интеграционного слоя на узкоспециализированные микросервисы в связке с разделением информационных потоков успешно решает проблему аппаратной гетерогенности аддитивного производства. Проблема несовместимости прошивок

инкапсулируется на уровне сервисов-адаптеров, а непрерывный поток низкоуровневых данных на лету трансформируется в строгие бизнес-контракты.

В результате центральное ядро MES-системы получает надежный, очищенный от аппаратного «шума» информационный фундамент для реализации управленческой логики. При этом архитектура остается полностью готовой к кратному росту: горизонтальное масштабирование (Scale-Out) при расширении парка станков сводится к тривиальному развертыванию дополнительных экземпляров (контейнеров) интеграционного сервиса, не требуя рефакторинга центральной системы.

Применение автономных модулей интеграции, взаимодействующих с центральным ядром (Core) через строгие API-контракты и брокеры сообщений (RabbitMQ, Redis), позволило успешно решить проблему интеграции гетерогенного парка оборудования.

Главное преимущество такого подхода – масштабируемость и отказоустойчивость. Добавление новых 3D-принтеров с нестандартными прошивками или периферийных устройств (весов, сканеров) происходит путем создания новых легковесных модулей или модификации существующих, без вмешательства в исходный код центральной бизнес-логики. Кроме того, физическое разделение компонентов позволяет развертывать интеграционные сервисы непосредственно в производственных цехах, снижая задержки при опросе оборудования.

Эксплуатационные выгоды от внедрения предложенного решения многократно превосходят инфраструктурные издержки. Разработанная система создает надежный цифровой фундамент для реализации концепции ценностно-ориентированного менеджмента (MBV), обеспечивая руководство объективными производственными данными в режиме реального времени.

Литература

1. Решетников И.С. MES. Стратегическая инициатива. Москва, 2019. 31 с.
2. Beregi R., Pedone G., Háy B., Váncza J. Manufacturing Execution System Integration through the Standardization of a Common Service Model for Cyber-Physical Production Systems // Applied Sciences. 2021. Vol. 11. No. 16. P. 7581. https://doi.org/10.3390/app11167581
3. Chowdhury A., Nuruzzaman M. Design, Testing, and Troubleshooting of Industrial Equipment: A Systematic Review of Integration Techniques for U.S. Manufacturing Plants // Review of Applied Science and Technology. 2023. Vol. 2. No. 1. P. 53-84. https://doi.org/10.63125/893et038
4. Horak T., Strelec P., Kebisek M., Tanuska P., Vaclavova A. Data Integration from Heterogeneous Control Levels for the Purposes of Analysis within Industry 4.0 Concept // Sensors. 2022. Vol. 22. No. 24. P. 9860. https://doi.org/10.3390/s22249860
5. Oñate W., Sanz R. Integration of Fog Computing in a Distributed Manufacturing Execution System Under the RAMI 4.0 Framework // Applied Sciences. 2024. Vol. 14. No. 22. P. 10539. https://doi.org/10.3390/app142210539

Ибраимов Н.Н., Эмильев А.Э.
Кыргызский государственный технический университет им И.Раззакова
Политехнический колледж КГТУ
г. Бишкек, Кыргызстан

ИНТЕГРИРОВАННАЯ СИСТЕМА АУДИОИНФОРМИРОВАНИЯ УЧЕБНОГО КОРПУСА

Введение. В современных образовательных учреждениях система централизованного звукового оповещения является важным элементом инженерной инфраструктуры здания. Радиоузел обеспечивает оперативную передачу информации, координацию учебного процесса и своевременное оповещение людей в экстренных ситуациях [1]. Особенно актуально применение таких систем в многоэтажных учебных корпусах с большим количеством помещений и высокой плотностью людей.

Объектом проектирования является учебный корпус Политехнического колледжа Кыргызского государственного технического университета имени И. Раззакова. Здание состоит из четырёх этажей и включает 44 учебных кабинета, а также четыре коридора, по одному на каждом этаже. Планировка здания требует применения распределённой системы радиотрансляции, обеспечивающей равномерное распространение звукового сигнала во всех помещениях. Основной целью организации радиоузла является

- обеспечение общей трансляции служебной информации;
- передачи сигналов начала и окончания занятий;
- экстренного оповещения при возникновении чрезвычайных ситуаций;
- трансляции фоновой музыки и учебных звонков.

Наличие централизованного управления позволяет одновременно охватывать все зоны здания и оперативно изменять режимы работы системы. При проектировании радиоузла учитывались требования к надёжности, стабильности работы и возможности дальнейшего расширения системы.

Методы и проектные решения. Основное требование – равномерное распределение звука во всех помещениях и коридорах учебного корпуса. Для учебных кабинетов выбран вариант с одним потолочным динамиком мощностью 5 Вт на кабинет. Всего в корпусе 44 кабинета, что соответствует установке 44 динамиков. Такой подход обеспечивает локальную слышимость и стабильное качество звука без увеличения мощности.

Коридоры требуют повышенного внимания, так как в них наблюдается максимальный поток студентов во время перемен. Для равномерного озвучивания в каждом коридоре установлены три динамика, что в сумме даёт 12 динамиков на четыре этажа. Размещение выполнено равномерно по длине коридоров для устранения «мертвых зон» и достижения одинакового уровня громкости вдоль всей линии. На рисунке показан расположение радиоузла на четвертом этаже колледжа.

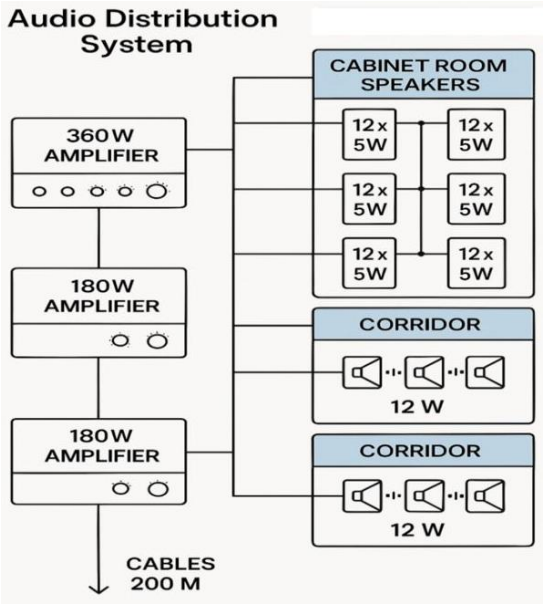


Рис. Схема расположение радиоузла

Такое распределение динамиков делает систему универсальной, позволяет легко добавлять новые зоны или корректировать громкость без замены оборудования, а также обеспечивает комфортную акустическую среду для студентов и персонала.

Для обеспечения работы акустической системы учебного корпуса выбран трансляционный усилитель мощностью 360 Вт с выходным напряжением 100 В. Он рассчитан на покрытие всей акустической нагрузки, которая составляет 280 Вт. Запас мощности около 28% обеспечивает стабильную работу системы даже при кратковременных пиковых нагрузках и предотвращает перегрузку оборудования [2].

Основной усилитель подключается к линии всех 56 динамиков и обеспечивает равномерное распределение сигнала по всей системе. Применение линии напряжением 100 В позволяет минимизировать потери мощности на длинных кабельных участках и обеспечивает простоту подключения динамиков в последовательную или смешанную схему.

Для повышения надёжности предусмотрен дополнительный усилитель мощностью 180 Вт. Его использование может быть организовано в двух режимах:

• в качестве резервного усилителя на случай выхода из строя основного;

• для отдельного зонирования системы, например, только на первый этаж или коридоры, что позволяет частично управлять системой и экономить энергию при необходимости [4].

Такое резервирование обеспечивает бесперебойную работу радиоузла и позволяет гибко управлять системой в зависимости от текущих задач. Все компоненты усилительной линии рассчитаны на длительный срок службы и соответствуют стандартам трансляционного оборудования.

Использование усилителей с запасом мощности и возможностью резервирования является ключевым элементом проектирования радиоузла. Оно обеспечивает надёжность передачи сообщений, стабильность работы акустической системы и позволяет адаптировать её под изменения в структуре здания или требований учебного процесса.

Выбор сечения кабеля 2×1,5 мм² обеспечивает достаточную пропускную способность линии для работы всей акустической системы, а также совместимость с используемыми трансляционными усилителями. Система рассчитана на длительную эксплуатацию в условиях ежедневного использования учебного корпуса, включая нагрузку в часы пик. При проектировании радиоузла особое внимание уделялось надёжности работы системы и возможности бесперебойного оповещения в любых ситуациях. Основной усилитель рассчитан с запасом мощности около 28%, что позволяет выдерживать кратковременные пиковые нагрузки и предотвращает перегрузку акустической линии.

Для повышения отказоустойчивости предусмотрен резервный усилитель мощностью 180 Вт, который может включаться автоматически или вручную в случае выхода из строя основного. Дополнительно резервирование позволяет выделять отдельные зоны, например, только коридоры или первый этаж, обеспечивая минимальный уровень оповещения при ограниченных ресурсах.

Все элементы системы, включая динамики, кабельную линию и усилители, рассчитаны на длительный срок службы. Кабели проложены с учётом механической защиты, что снижает риск повреждения и облегчает обслуживание. Установка динамиков в потолочных и коридорных зонах выполнена с соблюдением правил безопасности и удобства доступа [3].

Таблица

Данные оборудования

№ п/п	Наименование оборудование	Технические характеристики	Кол-во		Цена	Сумма
1.	Усилитель	360w	1	шт	18 000	18 000
2.	Усилитель	180 w	1	шт	14 000	14 000
3.	Динамик	5w	56	шт	700	39 000
4.	Кабель	2–1,5	300	Метр	30	9000
5.	Микрофон	Конденсаторный	1	шт	2000	2000
					Итого	**82 000**

Как видно из таблицы 1 суммарное количество динамиков в системе составляет 56 штук. Мощность каждого динамика 5 Вт, что обеспечивает общую нагрузку 280 Вт. Это значение учитывается при выборе трансляционного усилителя и гарантирует стабильную работу системы.

Результаты.

В результате проектирования радиоузла учебного корпуса Политехнического колледжа КГТУ им. И. Раззакова была разработана система, обеспечивающая:

- общую трансляцию сообщений и музыки;
- экстренные оповещения по всему зданию;
- равномерное распределение звука в кабинетах и коридорах;
- надежность работы благодаря резервированию усилителей и запасу мощности;
- возможность дальнейшего расширения и модернизации системы.

Таким образом, проект радиоузла соответствует современным требованиям к системам оповещения в учебных заведениях и может быть рекомендован к внедрению в аналогичных образовательных учреждениях.

Литература

1. Завьялов С.А., Мурасов К.В. Схемотехника усилителей мощности низких частот: учебное пособие. Омск: ОмГТУ, 2010. 93 с.

2. Наундорф У. Аналоговая электроника. Основы, расчет, моделирование (+CD): учебник / пер. с нем. М.М. Ташлицкого. М.: Техносфера, 2008. 472 с.

3. Селиванова З.М. Схемотехника электронных средств: учебное пособие. Тамбов: изд-во Тамб. гос. техн. ун-та, 2008. 80 с.

4. Миловзаров О.В. Электроника: учебник. М.: Юрайт, 2015. 407 с.

5. Каримов Б.Т. Основы электроники: Учебное пособие. Б.: Текник, 2016. 134 с.

Канцер К.И., Гаврилова Ю.С.
Кузбасский гуманитарно-педагогический институт
Кемеровского государственного университета
г. Новокузнецк, Россия

ПРОЕКТИРОВАНИЕ ИГРЫ «KITTENCLICK»

В современном мире компьютерные игры представляют собой не просто способ досуга, а масштабную индустрию, оказывающую влияние на миллионы людей по всему миру. Разнообразие игровых жанров чрезвычайно широко: от простых казуальных игр, не требующих от пользователя серьезных временных затрат, до масштабных сюжетных проектов. Популярность игровых проектов кроется в их уникальной способности давать игроку то, чего ему порой не хватает в реальной жизни: ощущение контроля над ситуацией, возможность добиваться целей и погружаться в иные миры [8, с. 85]. Постоянный поиск новых форм вовлечения пользователя в игровой процесс при одновременном стремлении к упрощению жанров в итоге положил начало возникновению игрового жанра, как кликер, который, на первый взгляд, идет вразрез с устоявшимися представлениями об увлекательной игре [5, с. 398].

Игровая индустрия демонстрирует, что даже самый простой жанр может порождать различные увлекательные и сложные проекты [7, с. 166]. Ярким примером таких проектов являются кликеры, в которых основным действием является клик мышью.

Условно кликеры можно разделить на две большие категории: игры с коротким, почти отсутствующим сюжетом (например, Cookie Clicker или AdVenture Capitalist), и игры с длинной, проработанной нарративной линией (например, Idling to Rule the Gods или SPACEPLAN). Цель первой категории кликеров заключается в накапливании чисел (печенек, денег) ради самого процесса. Сюжет здесь минимален и служит лишь атмосферой для бесконечной прогрессии. Популярность таких игр обусловлена их простотой и доступностью: они не требуют погружения в историю, позволяя игроку занять руки и мысли в фоновом режиме, получая радость от открытия нового достижения. Цель второй категории кликеров заключается в предоставлении длинного и постепенно разворачивающегося сюжета, мотивируя игрока кликать не ради рекорда, а чтобы узнать, что произойдет дальше. Такие кликеры привлекают более широкую аудиторию, которая в первую очередь ищет в играх историю и эмоции.

Популярность кликеров кроется не только в простоте, но и в продуманном прогрессе, а также в уникальных механизмах, удерживающих внимание пользователя [9, с. 139]. Именно поэтому разработчики кликеров вынуждены искать новые подходы, внедрять в устоявшиеся стандартные кликеры необычные механики и элементы, превращающие простой кликер в сложную игровую систему [6, с. 135]. Поиск таких подходов делает этап сбора требований ключевым в работе над игрой.

На предпроектной стадии разработки программного приложения были собраны и проанализированы функциональные и нефункциональные требования [1, с. 128; 3, с. 105; 4, с. 150]. Функциональные требования представляют собой основные функции игры, которые

обязательно должны быть реализованы, а нефункциональные – являются характеристиками качества разрабатываемого приложения [2, с. 339; 10, с. 148].

Проектирование игры «KittenClick» осуществлялось на основе ранее собранных и проанализированных требований [4 с. 148]. В результате была определена диаграмма переходов состояний, представленная на рисунке 1. Рассмотрим экраны игры подробнее.

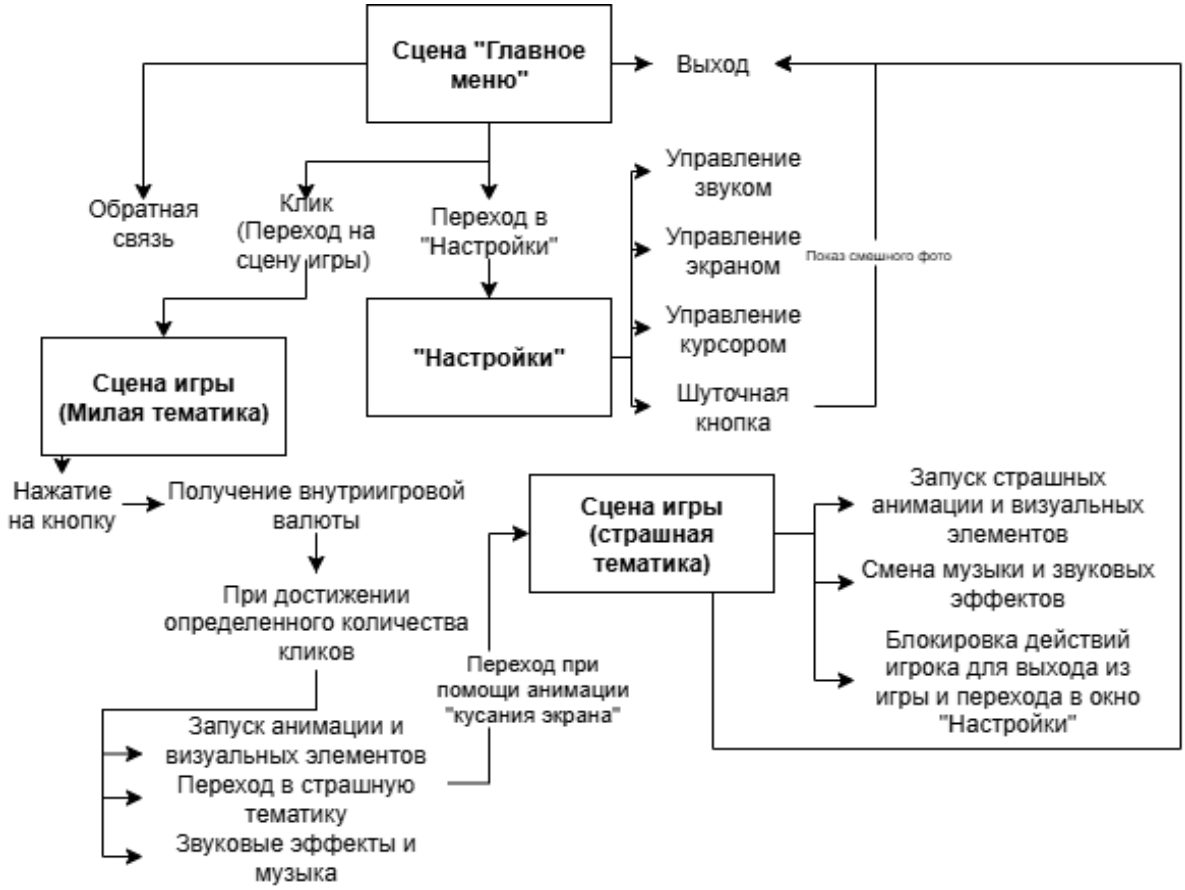


Рис. 1. Диаграмма переходов состояний [составлено автором]

Сцена «Главное меню» является центральным элементом навигации. Данное состояние имеет следующие действия: «Обратная связь» (при выборе данного пункта происходит переход на сайт разработчика); переход на сцену игры; переход на сцену «Настройки»; выход из игры.

Экран «Настройки» позволяет пользователю регулировать громкость музыки и звуковых эффектов, настраивать размер экрана, включать или отключать кастомный курсор и дает возможность нажать на шуточную кнопку, показывающую смешную картинку со звуком «Мяу» и закрывающую игру.

Сцена игры представляет наиболее сложный компонент с разветвленной логикой. Внутри данного экрана можно выделить несколько ключевых состояний интерфейса и переходов между ними:

1. Основное состояние (милая тематика). Находясь в этом состоянии, интерфейс принимает основное пользовательское взаимодействие – нажатие кнопки для получения внутриигровой валюты.

2. Срабатывание триггера. При достижении определенного количества кликов запускается следующий этап (анимации, визуальные элементы и подготовка к смене тематики).

3. Смена тематики. Наступает при соблюдении условии количества кликов, запуская следом анимацию «кусания экрана», после чего интерфейс мгновенно сменяется на страшную тематику, одновременно изменяются музыка и звуковые эффекты.

4. Состояние блокировки (страшная тематика). Финальное состояние, в которое переходит игра, блокируя возможности игрока выйти из игры или перейти в настройки, создающее эффект неожиданности и полного погружения в изменившуюся атмосферу.

Макеты приложения были разработаны при помощи редактора Figma. Когда пользователь открывает игру, он видит главное меню с названием игры и подзаголовком «meow :3», задающим игривый тон (рис. 2).

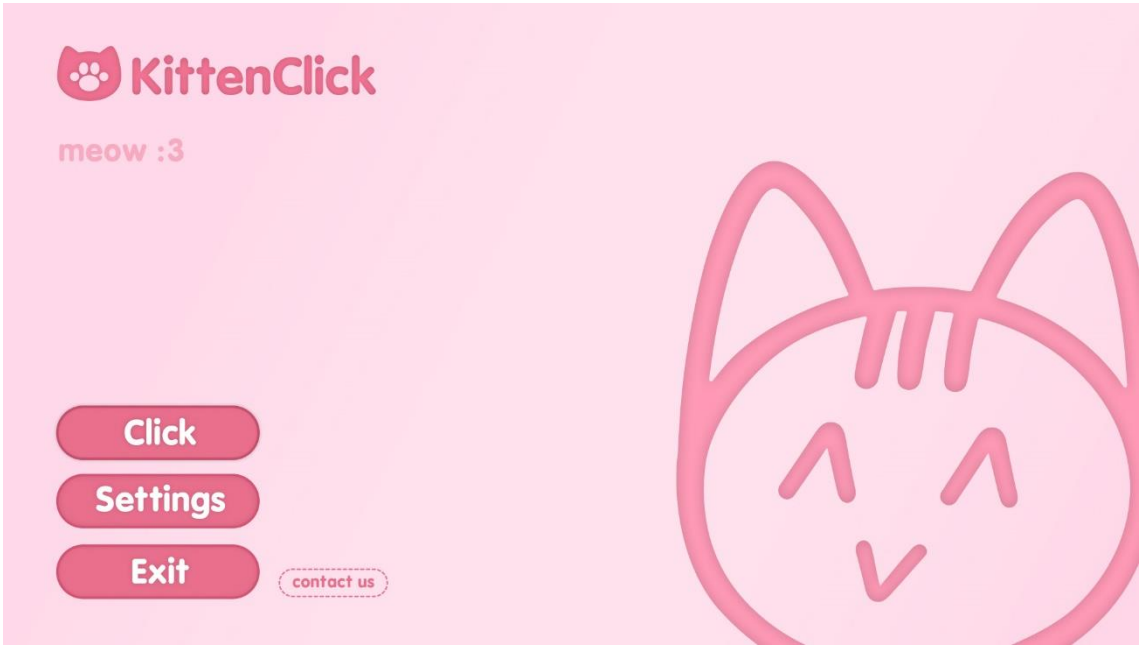


Рис. 2. Окно «Главное меню» [составлено автором]

В главном меню расположены четыре кнопки: click (начать игру), setting (настройки), exit (выход из игры) и contact us (обратная связь). Рядом с панелью кнопок расположено анимированное изображение кота, что усиливает привлекательность милой тематики.

Когда пользователь нажимает кнопку click, начинается игра. Он видит интерфейс в милой тематике: кнопка сделана в виде кошачьей лапки, а за основу цветовой палитры взят розовый цвет, дизайн минималистичный. Над кнопкой представлен счетчик, который будет фиксировать количество кликов (внутриигровая валюта). В верхней части экрана расположены две кнопки управления с интуитивно-понятными иконками: выход из игры и переход в окно «Настройки» (рис. 3).

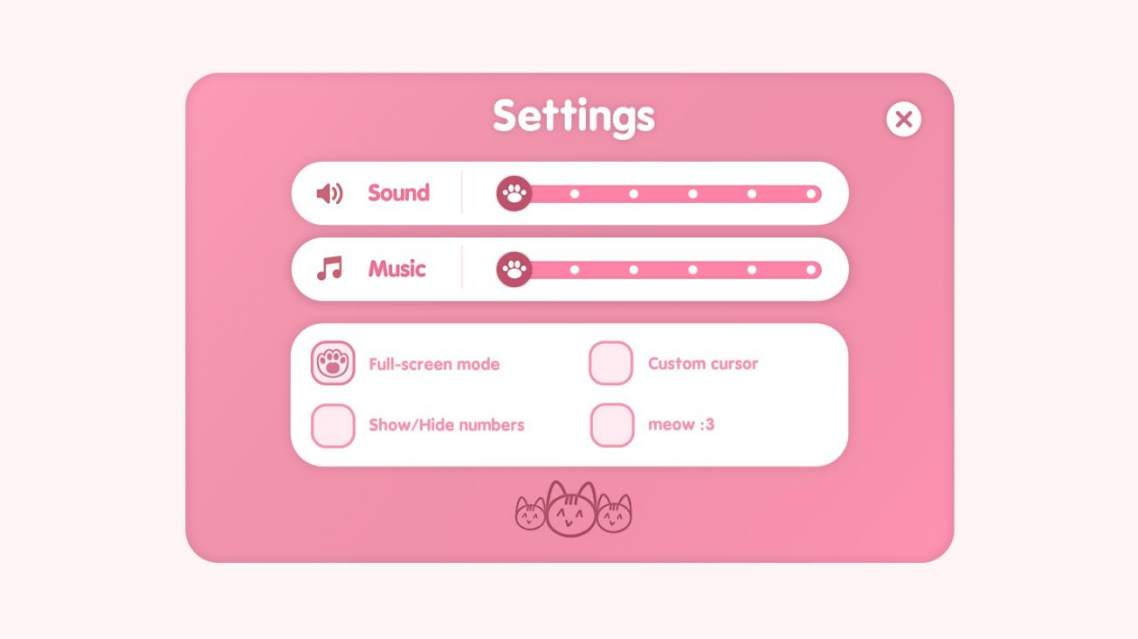


Рис. 3. Окно «Настройки» [составлено автором]

Данное окно содержит следующие элементы управления: два ползунка, выполненные в розовых тонах с метками звуковых значений, для настройки звуков (Sound) и музыки (Music), четыре чекбокса – Full-screen mode (полноэкранный режим), Custom cursor (отображение кастомного курсора), Show/Hide numbers (показ и скрытие цифр), meow :3 (шуточная кнопка). Визуальное оформление активации опций при нажатии на чекбокс реализовано в виде изображения лапки. В интерфейсе окна присутствуют три кота, которые олицетворяют команду разработчиков.

В определенный момент игры атмосфера меняется на пугающую: кнопка для получения внутриигровой валюты теперь имеет вид рыбьей кости, а кнопка в верхней панели принимает форму черепа кота – кнопка для перехода в окно «Настройки». Для усиления напряженности кнопка для выхода из игры пропадает, а кнопка для перехода в окно «Настройки» становится заблокированной, ограничивая действия игрока. Цветовая палитра фона переходит на оттенки красного, близкие к бордовому, что создает мрачную и агрессивную атмосферу. Также были разработаны кастомные курсоры для милой и страшной тематик, соответственно, розового и черного цветов.

Таким образом, были спроектированы диаграмма переходов состояний интерфейса и сам интерфейс для игры-кликера «KittenClick». Разработаны механики привлечения интереса пользователя, такие как: анимации, визуальные эффекты, звуки, картинки с котиками, а также переход с милой темы на жуткую. На основе разработанного проекта можно перейти к реализации игры.

Литература

1. Белая В.С. Сбор требований и проектирование 2D-игры «Попробуй выйти!» // Фундаментальные и прикладные исследования молодых ученых: материалы XV Всероссийской научно-практической конференции студентов, аспирантов и молодых учёных

(Новокузнецк, 07–25 апреля 2025 года). Новокузнецк: Кемеровский государственный университет, 2025. С. 128-131.

2. Виноградская М.Ю., Салдаева А.А. Определение требований при разработке компьютерной игры // Дневник науки. 2025. № 7(103).

3. Гришин М.Ю., Гудкова Е.А. Анализ и документирование требований для компьютерной игры в жанре платформер // Современные информационные технологии. 2024. № 40 (40). С. 105-109.

4. Канцер К.И., Чернодаров А.А. Сбор требований для чат-бота приемной комиссии вуза // Фундаментальные и прикладные исследования молодых ученых: материалы XV Всероссийской научно-практической конференции студентов, аспирантов и молодых ученых (г. Новокузнецк, 07–25 апреля 2025 года). Новокузнецк: изд-во Кемеровского государственного университета, 2025. С. 148-151.

5. Козырев А.С. Реализация модели планетарной системы в Unity // Современные тенденции развития науки и мирового сообщества в эпоху цифровизации: сборник материалов X Международной научно-практической конференции (Москва, 15 декабря 2022 года). М.: Алеф, 2022. С. 397-400.

6. Попов А.М. Проектирование 2D игры «Hero Defender» в Unity // Пространство современного региона: вызовы, трансформации, барьеры: материалы II Всероссийской научно-практической конференции (г. Новокузнецк, 25–27 ноября 2024 года). М.: АКТУАЛЬНОСТЬ.РФ, 2024. С. 135-138.

7. Подберезкин А.В. Разработка концепции новой компьютерной игры «Out Underground» // Регионы. Города. Ракурсы и параллели: сборник научных статей IX Всероссийской научно-практической конференции с конкурсом докладов молодых ученых, магистрантов (г. Омск, 29–30 ноября 2024 года). Омск: Омский государственный технический университет, 2025. С. 166-168.

8. Скворцов М.В. Проектирование игры spellzard // Научно-инновационный вектор современного развития: материалы I Всероссийской научно-практической конференции с международным участием (Новокузнецк, 20 апреля 2023 года) / отв. редактор Т.А. Евсина; редколлегия: Ю.А. Кузнецова [и др.]. Кемерово: Кузбасский государственный технический университет имени Т.Ф. Горбачева, 2023. С. 85-87.

9. Смородин В.Р., Гаврилова Ю.С. Разработка изометрических игр на движке Unity // Пространство современного региона: вызовы, трансформации, барьеры: материалы II Всероссийской научно-практической конференции (г. Новокузнецк, 25–27 ноября 2024 года). М.: АКТУАЛЬНОСТЬ.РФ, 2024. С. 139-141.

10. Ямщиков Ю.С., Никитин В.Е. Сбор и анализ требований для проекта «Сказки народов России» // Пространство современного региона: вызовы, трансформации, барьеры: материалы II Всероссийской научно-практической конференции (Новокузнецк, 25–27 ноября 2024 года). М.: АКТУАЛЬНОСТЬ.РФ, 2024. С. 148-151.

Котенев П.П.
Нижневартовский государственный университет
г. Нижневартовск, Россия

СРАВНИТЕЛЬНЫЙ АНАЛИЗ АРХИТЕКТУРНЫХ ПОДХОДОВ К РАЗРАБОТКЕ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ НА KOTLIN С ИСПОЛЬЗОВАНИЕМ MVVM И ДЕКЛАРАТИВНОЙ МОДЕЛИ JETPACK COMPOSE

Введение

Актуальность исследования обусловлена стремительным ростом сложности мобильных информационных систем и повышением требований к их надежности, производительности и безопасности. Современный рынок мобильной разработки характеризуется высокой конкуренцией, где ключевыми факторами успеха становятся скорость внесения изменений, стабильность работы приложения и удобство сопровождения кодовой базы. В условиях постоянных обновлений операционной системы Android и инструментов разработки выбор корректной архитектуры становится критически важным этапом проектирования программных систем.

Развитие цифровой трансформации образовательных, корпоративных и государственных сервисов приводит к росту роли мобильных приложений как основного канала взаимодействия пользователей с информационными системами. Повышение требований к доступности сервисов и скорости обработки данных обусловливает необходимость использования архитектурных решений, обеспечивающих устойчивость и масштабируемость мобильных систем.

Традиционные подходы к разработке часто приводят к формированию монолитных структур, в которых бизнес-логика тесно связана с пользовательским интерфейсом, что затрудняет тестирование и масштабирование. В рамках анализа установлено, что переход к современным архитектурным паттернам и языковым средствам позволяет снизить связность компонентов и повысить предсказуемость поведения программных модулей [3, с. 85]. Целью данной работы является теоретическое обоснование комплексного подхода к разработке Android-приложений, базирующегося на связке Kotlin, MVVM и Jetpack Compose.

Современные мобильные приложения функционируют в условиях асинхронного обмена данными и высокой динамики пользовательских взаимодействий, что требует применения реактивных архитектурных моделей и эффективного управления состоянием интерфейса.

Представляется целесообразным рассмотреть не только преимущества выбранных технологий, но и методологические аспекты их взаимодействия, включая управление состоянием, разделение ответственности и обеспечение безопасности данных. Полученные результаты позволяют сделать вывод о высокой эффективности предложенного стека технологий для создания корпоративных и потребительских мобильных решений.

Эволюция архитектур мобильных приложений

История развития архитектурных паттернов в мобильной разработке отражает тенденцию к усложнению программных систем и необходимости строгого разделения ответственности. На ранних этапах доминировал паттерн Model-View-Controller (MVC),

предполагающий разделение данных, интерфейса и логики управления. Однако в контексте Android-разработки классическая реализация MVC часто приводила к возникновению эффекта «Massive Controller», при котором Activity или Fragment брали на себя избыточное количество обязанностей, включая сетевые операции и работу с базой данных [3, с. 86].

Следующим этапом эволюции стало внедрение паттерна Model-View-Presenter (MVP), позволившего вынести логику представления в отдельный класс-посредник. Это повысило тестируемость компонентов, однако сохранило проблему жесткой привязки интерфейса к жизненному циклу Android-компонентов. При этом внедрение MVP увеличивает количество связующих интерфейсов и объём шаблонного кода, что усложняет сопровождение крупных проектов. Частые пересоздания активностей при изменении конфигурации устройства приводили к потере состояния и необходимости написания значительного объёма шаблонного кода.

В рамках анализа установлено, что наиболее эффективным решением на текущий момент является паттерн Model-View-ViewModel (MVVM). Это особенно важно для приложений с динамически обновляемым интерфейсом и интенсивной асинхронной загрузкой данных. Ключевое отличие MVVM заключается в наличии слоя ViewModel, выступающего в роли абстракции данных для представления. ViewModel не имеет прямых ссылок на View, что обеспечивает независимость бизнес-логики от реализации интерфейса.

Взаимодействие осуществляется через наблюдаемые потоки данных, что позволяет интерфейсу реактивно обновляться при изменении состояния модели. Проведённое сравнение демонстрирует, что MVVM снижает связность компонентов и упрощает поддержку кода в долгосрочной перспективе.

Помимо MVVM, в мобильной разработке применяется слоистая архитектура и подход Clean Architecture, ориентированный на разделение системы на независимые уровни. Данный подход обеспечивает изоляцию бизнес-логики от инфраструктурных зависимостей и позволяет адаптировать систему к изменениям технологической среды.

Проведённый анализ показывает, что эволюция архитектурных подходов отражает переход от тесно связанных структур к модульным и реактивным системам, ориентированным на масштабируемость и сопровождаемость.

Архитектура MVVM и её методологическое обоснование

Архитектурный подход MVVM в экосистеме Android реализуется с использованием компонентов Android Jetpack. Центральным элементом является класс ViewModel, предназначенный для хранения и управления данными, связанными с пользовательским интерфейсом. Важной особенностью является устойчивость ViewModel к конфигурационным изменениям, что обеспечивает сохранность состояния без дополнительной сериализации.

Взаимодействие между слоями архитектуры строится на принципах реактивного программирования. Для передачи данных от ViewModel к View используются потоки StateFlow или LiveData. StateFlow предоставляет поток состояний с гарантией доставки последнего значения новому подписчику, что обеспечивает актуальность отображаемых данных и эффективность использования ресурсов.

Слой модели инкапсулирует работу с источниками данных. Для абстрагирования доступа применяется паттерн Repository, предоставляющий единый интерфейс получения данных независимо от источника (локальное хранилище, сеть или кэш). Использование репозиториев упрощает замену источников данных и облегчает модульное тестирование бизнес-логики без запуска сетевого стека [3, с. 91].

Язык Kotlin играет ключевую роль в реализации данной архитектуры. Наличие корутин позволяет описывать асинхронные операции в линейном стиле, избегая сложных цепочек обратных вызовов. Data-классы и расширения языка способствуют сокращению шаблонного кода и повышению читаемости. Методологическая значимость подхода заключается в создании типобезопасного и лаконичного кода, что снижает вероятность ошибок выполнения. Использование корутин снижает накладные расходы потокового управления и повышает эффективность обработки асинхронных операций.

Роль языка Kotlin в современной Android-разработке

Kotlin является основным языком разработки Android-приложений, официально поддерживаемым Google. Его ключевые особенности включают строгую типизацию, безопасность работы с null-значениями, лаконичный синтаксис и встроенную поддержку функционального программирования [1, с. 221].

Поддержка корутин обеспечивает эффективное управление асинхронными задачами без блокировки потоков, что особенно важно для мобильных систем с ограниченными ресурсами.

Расширяемость языка, механизм extension-функций и DSL-подход упрощают разработку декларативных интерфейсов и повышают выразительность кода [2, с. 273].

Jetpack Compose и декларативная модель UI

Переход от императивной модели построения интерфейсов к декларативной является одним из наиболее значимых изменений в современной Android-разработке. Jetpack Compose позволяет описывать интерфейс как функцию от состояния, устраняя необходимость ручного управления жизненным циклом View-компонентов и синхронизации данных [4, с. 413].

Принцип работы Compose основан на механизме рекомпозиции: при изменении состояния соответствующие элементы интерфейса автоматически перерисовываются. Это требует соблюдения принципа единственного источника истины, при котором состояние хранится в ViewModel и передаётся в UI в виде неизменяемых моделей данных.

Важным аспектом является управление состоянием. Подход State Hoisting предполагает вынесение состояния в родительский компонент или ViewModel, а дочерние компоненты получают данные и события через параметры. Это повышает переиспользуемость компонентов и упрощает сопровождение интерфейса. Исследования показывают, что декларативный подход снижает объём UI-кода и повышает производительность отрисовки интерфейса [4, с. 413].

Интеграция Compose с MVVM обеспечивает непрерывный поток данных от бизнес-логики до пользовательского интерфейса. ViewModel экспонирует состояние через StateFlow, которое собирается в composable-функциях, минимизируя риск рассинхронизации интерфейса и данных.

Декларативный подход к построению интерфейсов способствует уменьшению количества ошибок синхронизации состояния и визуального представления. Compose устраняет необходимость ручного обновления UI, позволяя системе автоматически отслеживать зависимости состояния.

Кроме того, Compose обеспечивает лучшую интеграцию с реактивными потоками данных и упрощает создание адаптивных интерфейсов для различных форм-факторов устройств.

Масштабируемость и тестируемость Android-приложений

Масштабируемость мобильных приложений обеспечивается внедрением принципов Clean Architecture. Данный подход предполагает разделение системы на слои Domain, Data и Presentation. Слой Domain содержит бизнес-сущности и правила использования, не зависящие от внешних фреймворков. Слой Data реализует интерфейсы репозиториев, а слой Presentation взаимодействует исключительно с Domain [3, с. 92].

Такая модульность позволяет разработчикам работать над различными частями системы параллельно и облегчает внедрение новых функций. Замена источника данных требует изменений только в слое Data и не затрагивает бизнес-логику или интерфейс.

Тестируемость является ключевым показателем качества архитектуры. Использование MVVM и Clean Architecture позволяет тестировать бизнес-логику без зависимости от Android-фреймворка. ViewModel могут тестироваться изолированно с использованием подменных репозиториев, а интерфейс – с помощью инструментов Compose UI Testing. Высокий уровень тестового покрытия снижает стоимость сопровождения приложения и риск регрессионных ошибок.

Дополнительным фактором масштабируемости является модульная структура проекта. Разделение системы на Gradle-модули по функциональному признаку позволяет ускорить сборку проекта, упростить повторное использование компонентов и обеспечить независимую разработку функциональных модулей.

Модульность также способствует внедрению динамической доставки функциональности (Dynamic Feature Modules), что позволяет оптимизировать размер приложения и загружать функциональность по требованию.

Вопросы безопасности мобильной разработки

Обеспечение информационной безопасности мобильных приложений требует комплексного подхода и соблюдения фундаментальных свойств защиты данных [5, с. 273].

Аутентичность

Система должна гарантировать подлинность пользователя и источника запросов. Использование защищённых протоколов аутентификации (OAuth 2.0, многофакторная аутентификация) и механизмов проверки подлинности приложений предотвращает несанкционированный доступ.

Конфиденциальность

Передача данных должна осуществляться по защищённым каналам (HTTPS/TLS). Чувствительная информация не должна храниться в открытом виде на устройстве.

Шифрование и безопасное хранение токенов доступа обеспечивают защиту персональных данных.

Целостность данных

Серверные правила безопасности должны предотвращать несанкционированное изменение данных. Firebase Security Rules позволяют контролировать операции чтения и записи, обеспечивая соответствие действий пользователя его правам.

Контроль доступа

Правила безопасности должны дублировать логику авторизации и проверять роль пользователя перед выполнением операций. Например, изменение документа должно быть разрешено только владельцу или администратору ресурса.

Устойчивость к атакам

Применение принципов архитектуры нулевого доверия (Zero Trust) предполагает проверку каждого запроса независимо от источника. Дополнительные механизмы, такие как Firebase App Check и проверка целостности приложения, снижают риск автоматизированных атак и злоупотреблений.

Тем самым фундаментальные свойства защиты информации сохраняются при использовании облачной инфраструктуры, однако их обеспечение переносится на серверный уровень политики безопасности и требует строгой настройки правил доступа.

Сетевая и вычислительная эффективность

Использование облачных сервисов позволяет перенести вычислительную нагрузку на серверную сторону, снижая требования к ресурсам мобильного устройства. При этом применение кэширования и локального хранения данных уменьшает сетевую нагрузку и повышает отзывчивость интерфейса.

Итоговое сопоставление

Проведённый анализ показывает, что современная архитектура мобильных приложений должна сочетать механизмы аутентификации, серверный контроль доступа и защищённые каналы передачи данных. Такой подход обеспечивает баланс между безопасностью, производительностью и удобством эксплуатации.

Сравнительный анализ архитектурных подходов представлен в таблице.

Таблица

Сравнение архитектурных подходов

Критерий	MVC	MVP	MVVM	MVVM + Clean Architecture
Разделение ответственности	низкое	среднее	высокое	максимальное
Связность компонентов	высокая	средняя	низкая	минимальная
Тестируемость	низкая	высокая	высокая	максимальная
Управление состоянием	слабое	частично	эффективное	централизованное
Устойчивость к изменениям UI	низкая	средняя	высокая	высокая
Масштабируемость	низкая	средняя	высокая	максимальная
Поддержка реактивности	отсутствует	ограничена	встроенная	встроенная
Сложность сопровождения	высокая	средняя	низкая	минимальная
Архитектурная гибкость	низкая	средняя	высокая	максимальная
Поддержка модульности	ограничена	ограничена	возможна	полноценно реализуема

Заключение

Проведённый анализ архитектурных подходов к разработке мобильных приложений показал, что сочетание языка Kotlin, паттерна MVVM и декларативного фреймворка Jetpack Compose формирует эффективную архитектурную основу современных Android-приложений. Данный подход обеспечивает высокую степень разделения ответственности, повышает сопровождаемость кода и упрощает тестирование.

Использование декларативной модели интерфейсов и реактивного управления состоянием устраняет проблемы рассинхронизации данных и UI. Применение принципов Clean Architecture обеспечивает масштабируемость и технологическую независимость системы. Комплексный подход к безопасности, основанный на серверных правилах доступа и современных механизмах аутентификации, обеспечивает защиту пользовательских данных.

Методологическая значимость работы заключается в систематизации архитектурных практик, позволяющих создавать надежные мобильные системы с предсказуемым поведением. Перспективы развития направления связаны с дальнейшим развитием декларативных интерфейсов, автоматизацией тестирования и внедрением интеллектуальных инструментов разработки.

Полученные результаты подтверждают, что применение современных архитектурных подходов позволяет существенно повысить устойчивость мобильных систем к изменениям требований и технологической среды. Выбор архитектуры должен рассматриваться как стратегическое решение, определяющее жизненный цикл программного продукта.

Литература

1. Маринин А.К. Выбор архитектуры для мобильных приложений // Известия КБНЦ РАН. 2024. № 26 (5). С. 84-93.

2. Богданова В.С. Java vs Kotlin для Android-разработки // Актуальные вопросы современной науки и практики: Сборник научных статей по материалам XII Международной научно-практической конференции (Уфа, 16 мая 2023 г.). Ч. 2. Уфа: Научно-издательский центр «Вестник науки», 2023. С. 272-275. EDN EEPDTE.

3. Богданова В.С. Kotlin и его применении в мобильной разработке // Fundamental science and technology: Сборник научных статей по материалам XII Международной научно-практической конференции (Уфа, 14 апреля 2023 г.). Ч. 3. Уфа: Научно-издательский центр «Вестник науки», 2023. С. 220-223. EDN IVYDSD.

4. Гуляева С.Т., Тотков К.О., Канюков А.А. Особенности проектирования и разработки мобильных приложений под AndroidOS в рамках реализации цифровой трансформации СГУ им. Питирима Сорокина // Севергеоэкотех-2024: Материалы XXV Международной молодёжной научной конференции. В 2-х частях (Ухта, 28–29 марта 2024 г.). Ухта: Ухтинский государственный технический университет, 2024. С. 271-277. EDN AKTFPB.

5. Дындин А.В., Новиков П. Исследование применения Kotlin Multiplatform и Jetpack Compose Multiplatform в мобильной разработке // Научный журнал «Информационные технологии и вычислительные системы». 2024. № 4. С. 410-421.

Кучерявый С.В., Афендикова М.Е., Афендиков А.Т.
Нижневартовский государственный университет
г. Нижневартовск, Россия

ИССЛЕДОВАНИЕ ВЛИЯНИЯ ГЕОГРАФИЧЕСКОГО РАСПОЛОЖЕНИЯ СЕРВЕРОВ И ТИПА СЕТЕВОГО ПОДКЛЮЧЕНИЯ НА ВЕЛИЧИНУ СЕТЕВЫХ ЗАДЕРЖЕК (PING)

В современном цифровом мире скорость и стабильность сетевого соединения являются критически важными параметрами для эффективной работы, учёбы и проведения досуга. В условиях, когда цифровая среда стала полноценной средой обитания современного человека, вопросы её качества выходят на первый план. Пользователи чаще всего оперируют понятием ping в разговорах об онлайн-играх или видеозвонках, что делает исследование причин его колебаний не просто технической задачей, а социально значимым вопросом цифровой грамотности.

Углубляясь в данную тему, мы выяснили, что задержка в сети обозначается как Latency, это измеряемая величина от пользователя до сервера, а ping в свою очередь инструмент для измерения задержки в сети, только он измеряет время от пользователя до сервера и обратно. Понимание причин различий в задержке сигнала к разным ресурсам, позволяет не только удовлетворить техническое любопытство, но и по-новому взглянуть на архитектуру глобальной сети [1].

Несмотря на развитие сетевых технологий, пользователи регулярно сталкиваются с ситуацией, когда задержки сети (ping) до различных интернет-ресурсов значительно отличается, что вызывает закономерный вопрос о причинах такой разницы. На ping влияет множество факторов: географическое расположение серверов, пропускная способность соединения, загруженность сети, тип подключения к сети. Ping напрямую влияет на качество видеоконференций, онлайн-игр, потокового вещания и облачных сервисов [2; 3].

Актуальность исследования обусловлена тем, что в настоящий момент всё больше сфер нашей жизни охватывает цифровизация, все больше задач решаются напрямую, через сеть Интернет. Например, запись на приём к врачу или онлайн-обучение. Поэтому для комфортной деятельности цифровом мире очень важно стабильное подключение к сервисам.

Целью нашего исследования является анализ задержек к разным ресурсам в зависимости от их расположения с помощью кода-программы. Мы исследуем в систематическом сравнении сетевые параметры трех Интернет-ресурсов.

Для начала нам необходимо было разработать программное средство для измерения сетевых задержек. Мы выбрали 6 сайтов, которые поделили на три категории по географическому признаку.

Таблица 1

Используемые сайты и их возможное расположение

Категория	Сайты	Кол-во	Предполагаемая локация серверов
Локальные (города Нижневартовск)	nvsu.ru; www.n-vartovsk.ru	2	Нижневартовск
Российские	vk.com; dzen.ru	2	Россия/Москва
Зарубежные	github.com; wikipedia.org	2	США, Европа

Основным инструментом измерения выбран протокол ICMP и утилита ping, как наиболее распространенный и стандартизированный метод оценки сетевых задержек. Для каждого сайта выполняется серия из 25 последовательных запросов с интервалом 200 мс, что позволяет получить статистически значимую выборку.

Эксперимент проводился в три этапа:

1. *Предварительный этап:* проверка доступности всех сайтов, настройка программного обеспечения;
2. *Основной этап:* последовательное измерение ping для всех 6 сайтов в одинаковых условиях (под одинаковыми условиями подразумевается использование одного ноутбука, провайдера, роутера и единого времени проведения измерений);
3. *Контрольный этап:* повторное измерение через 6-8 часов для оценки временной стабильности результатов.

Для автоматизации измерений разработана программа на языке Python, использующая стандартную библиотеку subprocess для вызова системной команды ping (рис. 1). Данная программа также выводит необходимые для исследования метрики.

```python
import subprocess
import re
import time
from datetime import datetime

def simple_ping_test():
    sites = [
        ("Локальный", "nvsu.ru"),
        ("Локальный", "www.n-vartovsk.ru"),
        ("Российский", "vk.com"),
        ("Российский", "dzen.ru"),
        ("Зарубежный", "github.com"),
        ("Зарубежный", "wikipedia.org")
    ]

    all_results = []

    for category, site in sites:
        print(f"\n{category}: {site}")
        print("-" * 40)

        ping_times = []

        for i in range(1, 21):
            try:
                print(f"  Запрос {i}...", end=" ", flush=True)

                cmd = ['ping', '-n', '1', site]
                output = subprocess.run(cmd,
                                        capture_output=True,
                                        text=True,
                                        encoding='cp866',
                                        timeout=8)

                time_match = re.search(r'время[=<](\d+)мс|time[=<](\d+)ms',
                                       output.stdout,
                                       re.IGNORECASE)

                if time_match:
                    for num in time_match.groups():
                        if num:
                            ping_time = int(num)
                            ping_times.append(ping_time)
                            print(f"✓ {ping_time} мс")
                            break
                else:
                    print("нет ответа")

            except subprocess.TimeoutExpired:
                print("таймаут")
            except Exception as e:
                print(f"ошибка")

            if i < 25:
                time.sleep(2)

        if ping_times:
            avg = sum(ping_times) / len(ping_times)
            min_time = min(ping_times)
            max_time = max(ping_times)
            spread = max_time - min_time

            print(f"\n  Статистика для {site}:")
            print(f"    Все значения: {ping_times}")
            print(f"    Среднее: {avg:.1f} мс")
            print(f"    Мин/Макс: {min_time}/{max_time} мс")
            print(f"    Разброс: {spread} мс")

            all_results.append({
                'category': category,
                'site': site,
                'avg': avg,
                'min': min_time,
                'max': max_time,
                'times': ping_times
            })
        else:
            print(f"\n  Не удалось получить данные для {site}")

if __name__ == "__main__":
    simple_ping_test()
```

Рис. 1. Код-программы

Измерения проводились в одном населенном пункте, но в разных точках города Нижневартовска. Первое и второе измерение происходило в северо-восточной части города, через проводное соединение и wi-fi соединение. Третье и четвертое – в юго-западной части города, также через wi-fi соединение и проводное соединение.

Измерения проводились два раза в день на протяжении двух дней, примерно в середине дня и вечером, всего четыре измерения для каждого типа соединения. На рисунках 2-4 представлены результаты, полученные в ходе эксперимента.

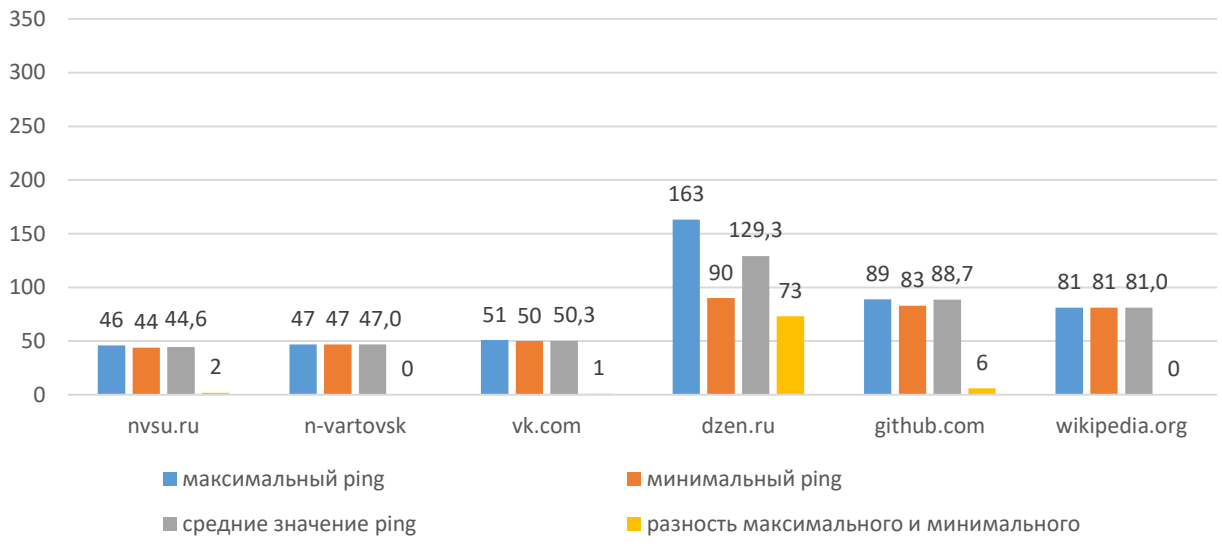


Рис. 2. Измерение ping в северо-восточной части с помощью проводного соединения

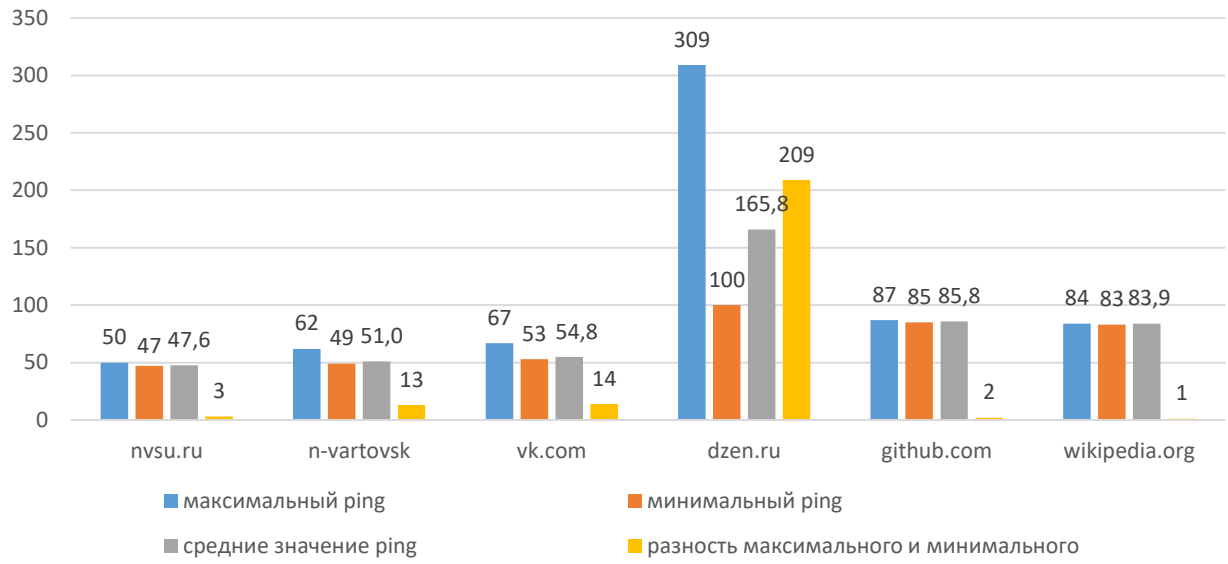


Рис. 3. Измерение ping северо-восточной части с помощью беспроводного соединения

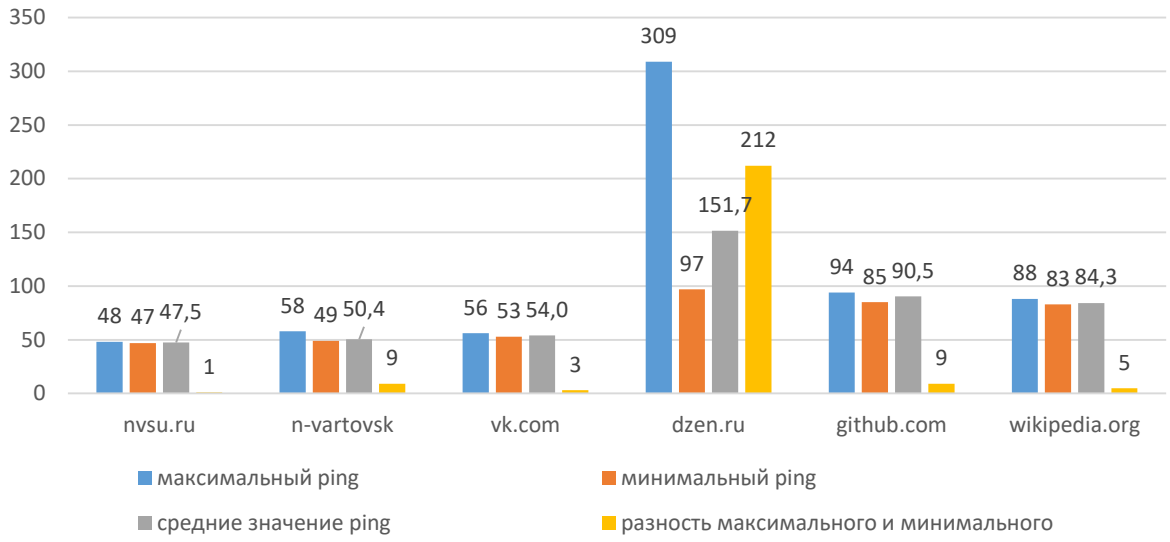


Рис. 4. Измерение ping в юго-западной части с помощью проводного соединения

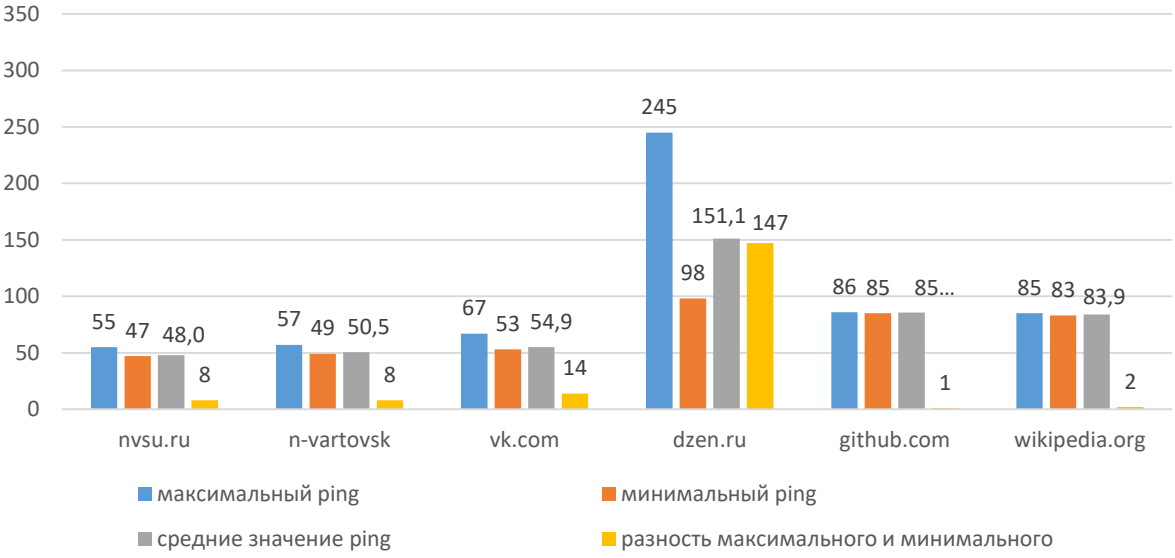


Рис. 5. Измерение ping в юго-западной части с помощью беспроводного соединения

В таблице 2 представлены результаты в упрощенной форме по критериям:

- Стабильный – минимальный и максимальный ping различаются не более чем на 10 мс.
- Быстрый отклик – ping не превышает 50 мс.
- Средний отклик – ping больше 50 мс, но меньше 80 мс
- Медленный отклик – ping не менее 80 мс.

Для удобного представления результатов были введены следующие обозначения:

1. северо-восточная часть с проводным соединением.
2. северо-восточная часть беспроводным соединением.
3. юго-западная часть с проводным соединением.
4. юго-западная часть с беспроводным соединением.

Таблица 2

Результаты эксперимента

Сайт	Категория	Стабильность				Быстрый отклик				Средний отклик				Медленный отклик			
		1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
nvsu.ru	Локальный (Нижневартовск)	+	+	+	+	+	+	+	+								
n-vartovsk.ru		+		+	+	+		+	+		+						
vk.com	Страна (Россия/Москва)	+		+		+					+	+	+				
dzen.ru														+	+	+	+
github.com	Зарубежные (США, Европа)	+	+	+	+									+	+	+	+
wikipedia.org		+	+	+	+									+	+	+	+

Проведенный анализ показал, что на скорость отклика сайтов влияет не только географическое расположение серверов, но и качество подключения, а также загруженность сети. Локальные ресурсы Нижневартовска ожидаемо показали наименьший ping, а зарубежные – наибольший. Однако самым нестабильным оказался *dzen.ru*, чей ping иногда превышал показатели зарубежных сайтов независимо от условий замера. Выяснилось, что проводной интернет работает быстрее wi-fi [4]. Подводя итог можно сказать, что комфортная работа в сети зависит от совокупности факторов: расположения сервера, типа подключения и числа активных пользователей.

В дальнейшем мы планируем: расширить исследования на мобильные и общественные сети, провести сравнительный анализ разных интернет-провайдеров и операторов связи, исследовать влияния VPN и прокси-серверов на сетевые задержки, разработать рекомендации по оптимизации сетевой инфраструктуры для образовательных учреждений и обычных пользователей.

Литература

1. Бескровный А.С., Конасов В.Р. Тестирование производительности сетевого стека // Новая наука: от идеи к результату. 2025. № 1. С. 133-141.

2. Гончаров В.В., Грищенко В.И. Исследование методов снижения задержки в протоколах передачи данных в компьютерных сетях // Информатика, управляющие системы, математическое и компьютерное моделирование (ИУСМКМ-2020): Сборник материалов XI Международной научно-технической конференции в рамках VI Международного Научного форума Донецкой Народной Республики (Донецк, 27–28 мая 2020 г.). Донецк: Донецкий национальный технический университет, 2020. С. 132-134.

3. Ланских Е.В., Шкарбан А.В., Скрипка Я.С. Исследование методов оптимизации производительности сетей // Современные научные исследования и инновации. 2014. № 5. Ч. 1.

4. Network Latency: Definition, Causes and Best Practices. URL: https://phoenixnap.com/blog/network-latency (дата обращения: 10.02.2026).

Ларин Д.Е.
Нижневартовский государственный университет
г. Нижневартовск, Россия

СОВРЕМЕННЫЕ ПЛАТФОРМЫ И ФРЕЙМВОРКИ ВЕБ-РАЗРАБОТКИ

В современном мире любая компания стремится иметь свой собственный веб-сайт или веб-приложение. Вместе с интересом бизнеса растет количество фреймворков, которые помогают создавать продукты быстро и эффективно. Инструменты кардинально изменили веб-разработку, став ее неотъемлемой частью веб-сайта, поскольку эти требования к веб-приложениям постоянно растут, а технологии усложняются. В статье будут рассмотрены фреймворки веб-разработки по двум направлением как фронтенд, так и бэкенд.

Цель исследования заключается в анализе современных платформ и фреймворков веб-разработки, выявлении основных и ключевых инструментов их развития разработки и сравнительной характеристике наиболее востребованных направлений frontend- и backend разработки веб-сайтов.

Объектом исследования являются платформы и фреймворки, которые используют для разработки веб-сайтов. Предметом исследования являются функциональные, архитектурные и эксплуатационные характеристики инструментов, которые определяют применимость в различные веб-проекты.

Фреймворк (Framework – «каркас», «структура») – это программная платформа, которая определяет структуру веб-приложения и предоставляет набор инструментов в виде библиотек для того, чтобы упростить и ускорить процесс создания веб-сайтов или веб-приложений.

Клиентская часть веб-сайта (Frontend – «фронтенд») – это совокупность веб-технологий и кода, выполняющие на устройстве пользователя в качестве клиента в браузере. Это то, что все отображается в веб-браузере и обеспечивает удобство и функциональность интерфейса веб-сайта или веб-приложений.

Северная часть веб-сайта (Backend – «бэкенд») – это часть программного обеспечения (ПО), которая работает на сервере. Бэкенд управляет данные для обработки, выполняет бизнес-логики и взаимодействует с базами данных (БД). В отличие от фронтенда бэкенд скрывает от глаз пользователя, обеспечивая функциональность, оставаясь в тени интерфейса веб-сайта или веб-приложений [4].

Ниже представлена схема клиент-серверной архитектуры с компонентами Frontend, Backend, SQL и файловым сервером (рис.).

Схема клиент-серверной архитектуры включает в себя следующие компоненты:

- Клиент;
- Сервер;
- Протоколы обмена данными.

Мы рассмотрим современные фронтенд- и бэкенд-фреймворки, которые самые актуальные для веб-разработчиков в 2026 году:

1. Фронтенд-фреймворки – React, Vue.js, Angular, Svelte;
2. Бэкенд-фреймворки – Django, FastAPI, Flask, Express.js.

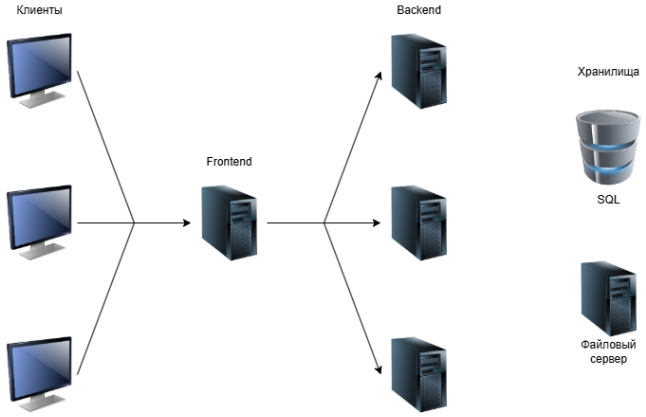


Рис. Схема клиент-серверной архитектуры с компонентами Frontend, Backend, SQL и файловым сервером

React.js – это JavaScript-библиотека от «Facebook» для удобной разработки интерфейсов, то есть внешней части сайтов и приложений, с которой взаимодействует пользователь. Главной фишкой React.js являются компоненты и состояния. В основе библиотека React использует декларативный подход к веб-разработке. Если в императивном программировании необходимо задать каждое действие, то в декларативном достаточно описания конечного результата и реакции на действия пользователя. Веб-разработчик должен определить, как веб-приложение должно выглядеть при определенных данных, а библиотека React.js управляет процессом обновления элементов, оптимизируя этот процесс. Использование декларативного подхода позволяет создавать веб-приложения на библиотеке React.js быстро [7].

Vue.js – это JavaScript-фреймворк, предназначенный для создания пользовательского интерфейса веб-приложений. Ключевая особенность фреймворка Vue.js – высокая степень адаптивности. В отличие от других фреймворков, Vue.js ориентирован на уровень представления. Он позволяет легко интегрировать его в существующие проекты и подключать к нему дополнительные библиотеки.

Vue.js использует декларативный подход и расширяет стандартный HTML с помощью директив – это специальные атрибуты, связывающие разметку с логикой приложения. Фронтенд-разработчик использует как встроенными директивами, так и создавать собственные. Фреймворк Vue.js подходит для создания сложных одностраничных приложений [6].

Angular – это популярный фреймворк для разработки веб-приложений, созданный и поддерживаемый Google. Он позволяет создавать динамичные, одностраничные приложения (SPA) с использованием TypeScript. Angular предоставляет мощные инструменты для работы с данными, компонентами и сервисами, что делает его идеальным выбором для крупных и сложных проектов.

Angular решает множество проблем, с которыми сталкиваются разработчики при создании веб-приложений, таких как управление состоянием, маршрутизация, а также взаимодействие с сервером. Благодаря своей модульной архитектуре и богатому набору

встроенных функций, Angular позволяет создавать масштабируемые и поддерживаемые приложения.

Одной из ключевых особенностей Angular является его двухстороннее связывание данных (two-way data binding), что позволяет автоматически синхронизировать данные между моделью и представлением. Это упрощает разработку и уменьшает количество кода, который необходимо писать вручную. Кроме того, Angular использует декларативный подход к созданию пользовательских интерфейсов, что делает код более читаемым и поддерживаемым [1].

Svelte – это передовой фреймворк для создания веб-приложений, который отличается своей уникальной концепцией и подходом к разработке. В отличие от традиционных фреймворков, Svelte компилирует приложения в оптимизированный JavaScript-код, обеспечивая высокую производительность и быструю загрузку веб-страниц.

Svelte предлагает ряд значимых преимуществ перед другими популярными фреймворками, включая React, Angular и Vue.

Высокая производительность: благодаря компиляции приложений в оптимизированный JavaScript код, Svelte значительно ускоряет выполнение программ и время их загрузки, что критически важно для пользовательского опыта.

Проще для разработчиков: Отсутствие виртуального DOM и реактивное программирование через простой и понятный синтаксис делают Svelte легким в освоении и использовании. Это экономит время на обучение и разработку.

Меньший размер финальных бандлов: так как Svelte компилируется в исходный код, который напрямую выполняется браузером, финальные бандлы оказываются меньше по размеру по сравнению с другими фреймворками, что также способствует более быстрой загрузке страниц.

Легкая интеграция: Svelte позволяет легко интегрировать созданные компоненты в существующие проекты, что делает его удобным инструментом для расширения функциональности без необходимости переработки всего приложения.

Эти преимущества делают Svelte весомым конкурентом на рынке фреймворков для разработки веб-приложений, предоставляя разработчикам мощный инструмент создания производительных и легковесных приложений. Однако важно отметить, что его экосистема пока меньше, чем у более зрелых фреймворков [5].

Django – это бесплатный высокоуровневый бэкенд-фреймворк с открытым исходным кодом для создания веб-приложений на языке Python. Главная задача Django – помочь бэкенд-разработчикам быстро и безопасно выстраивать серверную честь сайтов.

Django содержит все необходимое для решения типичных задач веб-разработки, и он позиционируется как полностью укомплектованный фреймворк.

Django предлагает готовые инструменты для:

- Создания и структурирования новых приложений;
- Удобной работы с базами данных;
- Реализации бизнес-логики и CRUD-операций;

- Управления пользовательскими аккаунтами;
- Обработки форм и многих других повседневных задач.

Благодаря подходу бэкенд-разработчики могут сосредоточиться на уникальной логике своего проекта, не технические детали [8].

FastAPI – это современный фреймворк для создания веб-приложений и API на Python, который стал популярным благодаря простоте и скорости работы. С его помощью можно буквально за несколько строк кода создать сервер, который принимает запросы, обрабатывает их и возвращает результаты – например, данные из базы, ответ нейросети или результат вычислений.

FastAPI создали в 2018 году, чтобы исправить слабые места старых Python-фреймворков вроде Flask и Django. Эти инструменты отлично подходили для создания классических сайтов, но начинали тормозить с ростом числа запросов и переходом на асинхронный Python.

FastAPI применяют там, где нужно оперативно обрабатывать запросы, – от небольших сервисов до крупных платформ. Python-разработчики создают на нём бэкенд веб-приложений, DevOps-инженеры и инженеры по интеграции используют его для микросервисов и соединения разных систем между собой, а ML-специалисты разворачивают с его помощью модели и делают их доступными через API.

Фреймворк используют для создания чат-ботов и даже IoT-устройств, где нужно мгновенно реагировать на поток данных [2].

Flask – это легковесный веб-фреймворк для языка Python, который предоставляет минимальный набор инструментов для создания веб-приложений. На нём можно сделать и лендинг, и многостраничный сайт с кучей плагинов и сервисов.

У Flask много преимуществ, которые выделяют его среди других фреймворков:

- простой синтаксис – это всё-таки Python;
- удобные шаблоны – можно быстро создавать прототипы веб-приложений;
- куча инструментов для гибкой настройки сайтов под любые нужды.

Ещё под Flask написаны сотни расширений и плагинов, которые добавляют дополнительные возможности – разные виды аутентификации, управление базами данных и работу с формами. И всё это – бесплатно и с открытым кодом [3].

Express – самый популярный веб-фреймворк для Node.js, который служит базовой библиотекой для других фреймворков и предоставляет веб-разработчикам ключевые механизмы для создания веб-приложений.

Express остается минималистичным и его главная сила – это экосистема. Веб-разработчики создали огромное количество совместимых middleware-пакетов для решения любых задач: работа с cookie-файлами, сессиями, авторизацией пользователей и многим другим [9].

Для сравнения представлена таблица анализов фреймворков фронтенд- и бэкенд-разработки веб-сайтов.

Таблица

Сравнительный анализ современных фреймворков веб-разработки

Фреймворк	Тип	Язык	Ключевая особенность	Лучшее применение
React	Фронтенд-библиотека UI	JavaScript	Виртуальный DOM, богатая экосистема	SPA, крупные фронтенды
Vue.js	Прогрессивный фронтенд-фреймворк	JavaScript	Постепенное внедрение, простота синтаксиса	Стартапы, гибридные проекты
Angular	Полноценный фреймворк	TypeScript	Многофункциональность, DI	Корпоративные приложения
Svelte	Компилятор	JavaScript	Компиляция, отсутствие Virtual DOM	Высокопроизводительные сайты
Django	Бэкенд-фреймворк	Python	«Батарейки в комплекте», безопасность	Быстрая разработка, MVP
FastAPI	API-фреймворк	Python	Асинхронность, автодокументация	Высоконагруженные API
Flask	Микрофреймворк	Python	Минимализм, гибкость	Микросервисы, простые сайты
Express.js	Микрофреймворк	JavaScript	Минимализм, NPM-экосистема	REST API, микросервисы

Вывод:

Проведенный анализ фронтенд- и бэкенд-фреймворков позволяет сделать вывод, что данные фреймворки лидируют в наше время – это React, Vue.js, Django, FastAPI. Поскольку они набирают популярность среди веб-разработчиков для создания веб-проектов как для клиентской части, так и северной. Остальные вышеперечисленные фреймворки теряют позиции по рейтингам лучших фреймворков в 2026 году.

Литература

1. Андрухив Л.В., Павлюченко П.С. Сравнительная характеристика web-фреймворков: Angular, React и Vue // Высокопроизводительные вычисления для решения прикладных задач. Сборник материалов X-й (67-й) ежегодной научно-практической конференции студентов, преподавателей и молодых ученых Северо-Кавказского федерального университета (Ставрополь, 17-23 апреля 2023 г.). Ставрополь, 2023. С. 5-10.

2. Богачёв Р.Е., Зариковская Н.В. Реализация серверной части веб систем с использованием фреймворка FastAPI // Актуальные научные исследования в современном мире. 2021. № 11-15 (79). С. 368-373.

3. Галкин А.С., Мирошниченко М.В., Галибин С.В. Разработка приложения-мессенджера на языке Python с использованием технологий Flask, PyQt5 // Информатика и вычислительная техника. сборник научных трудов. Чебоксары, 2021. С. 61-64.

4. Гринчар Н.Н., Бахтиарова А.Ж., Масленникова Д.А. Обзор возможностей веб-фреймворка // Научный аспект. 2023. Т. 5. № 1. С. 510-515.

5. Летон Г., Глазько П.Е. Анализ причин отказа от использования Virtual DOM на примере Svelte // XXIV Всероссийская студенческая научно-практическая конференция

Нижневартовского государственного университета. Материалы конференции (г. Нижневартовск, 5-6 апреля 2022 г.). Нижневартовск, 2022. С. 119-123.

6. Морщинина Л.В., Зубарева А.Л. Сравнительный анализ реактивности Vue.js 2 И Vue.js 3 // Научно-технические инновации и веб-технологии. 2023. № 2. С. 48-52.

7. Мусаев М.Ш. Сравнительный анализ фреймворков Flutter, React и React Native для создания десктопных, веб- и мобильных приложений // ИИАСУ'24 – Искусственный интеллект в автоматизированных системах управления и обработки данных. Сборник статей III Всероссийской научной конференции. В трех томах. Москва, 2025. С. 105-111.

8. Пятигорец А.Ю., Бобов В.А. Сравнительный обзор в использовании фреймворков Django и Django Rest Framework в разработке веб-приложений // Инновационные процессы в науке и технике XXI века. материалы XXI Международной научно-практической конференции студентов, аспирантов, ученых, педагогических работников и специалистов-практиков: в 2-х томах (Нижневартовск, 26 апреля 2024 года). Тюмень, 2024. С. 160-166.

9. Чернышев Я.Р. Разработка и самостоятельный хостинг веб-приложения на основе фреймворка Express.js // Студенческая наука – взгляд в будущее. Материалы XVIII Всероссийской студенческой научной конференции (Красноярск, 15–17 марта 2023 года). Красноярск, 2023. С. 291-293.

Лисина Т.Б.
Казанский национальный исследовательский технический университет им. А.Н. Туполева – КАИ
г. Казань, Россия

ПРОГРАММНО-АППАРАТНЫЙ КОМПЛЕКС ДЛЯ МОНИТОРИНГА МИКРОКЛИМАТА В ПОМЕЩЕНИИ НА БАЗЕ ESP32

Разработка программного обеспечения для микроконтроллерного устройства мониторинга микроклимата представляет сложную задачу. Она требует глубокого понимания как аппаратной части, так и принципов построения надежных встраиваемых систем. Это исследование развивает подход, представленный в более ранней работе, где был описан базовый алгоритм программы [1]. В качестве центрального процессора выбран модуль ESP32. Он облагает высокой производительностью, наличием встроенного Wi-Fi 2,4 ГГц и Bluetooth. ESP32 оснащён 34 программируемыми портами ввода-вывода общего назначения (GPIO), 12-битным аналого-цифровым преобразователем (АЦП) и 2 цифро-аналоговыми преобразователями (ЦАП), 10 сенсорными датчиками и различными интерфейсами (4 SPI, 2 I2C, 2I2S, 3 UART, ETH), а также энкодерами для ШИМ-управления двигателем и ШИМ-управления светодиодом [6].

Выбор данной платформы позволяет заложить основу для будущего расширения системы, например, путем интеграции с облачными сервисами или сетями Интернета вещей (IoT). IoT – это сеть устройств, пользователей и сервисов; их взаимодействие позволяет осуществлять сбор данных, принимать решения и реагировать на них [3]. Современные исследования часто используют ESP32 как основу для создания сложных IoT-устройств, предназначенных для мониторинга окружающей среды, управления умными зданиями и проведения научных экспериментов [5].

Программная реализация осуществлялась в среде разработки Visual Studio Code, которая является стандартом де-факто для многих разработчиков, работающих с микроконтроллерами. Это связано с ее легковесностью, гибкостью и возможностью интеграции с мощными плагинами, такими как PlatformIO или Arduino. Фреймворк Arduino значительно упрощает взаимодействие с периферийными устройствами благодаря абстракции низкоуровневых деталей работы контроллера. Такой подход позволяет сосредоточиться на алгоритмической логике и функциональности приложения. Широкое распространение Arduino-фреймворка подтверждается многочисленными образовательными и исследовательскими проектами, где он используется для быстрой разработки прототипов и создания промышленных IoT-решений [3].

Основой программной архитектуры является структура, основанная на двух ключевых функциях: setup() и loop(). В отличие от базового алгоритма инициализации [1], в данной реализации особое внимание уделено обработке ошибок инициализации шин связи и отключению автокалибровки датчика CO_2 для работы в специфичных условиях.

Для повышения читаемости и надежности в разработке используются сторонние библиотеки:

Arduino.h – содержит объявления базовых команд.

Wire.h – для работы со шиной I2C.

GyverOLED– для удобной работы с OLED-дисплеями на базе контроллера SSD1306.

GyverBME280 – упрощает работу с датчиком BME280, скрывая сложную последовательность команд.

MHZ19.h – для работы с датчиком CO_2 MH-Z19 через UART.

Использование таких библиотек позволяет сфокусироваться на решении основной задачи – алгоритмической обработке данных.

Функция setup() выполняется один раз при включении устройства или после перезагрузки. Этот этап является критически важным, поскольку любая ошибка инициализации может привести к некорректной работе всей системы.

1. Инициализация последовательного порта осуществляется вызовом функции Serial.begin(9600). Она необходима для отправки отладочной информации и диагностических сообщений на компьютер.

2. Настройка шины I2C выполняется посредством вызова Wire.begin(), который конфигурирует микроконтроллер в режиме мастера для связи с датчиком BME280 и OLED-дисплеем.

3. Инициализация OLED-дисплея реализуется через создание объекта GyverOLED<SSD1306_128x64> oled и последующий вызов метода oled.init(), который отправляет команды инициализации контроллеру SSD1306. После этого экран очищается (oled.clear()) и обновляется (oled.update()).

4. Инициализация датчика BME280 включает создание объекта GyverBME280 bme. Метод bme.begin() проверяет наличие устройства на шине, считывает его идентификатор и загружает заводские калибровочные коэффициенты, необходимые для точного преобразования сырых данных в физические величины.

5. Инициализация датчика MH-Z19 предполагает конфигурацию UART-интерфейса: задание пинов RX/TX и скорости передачи данных. Создаётся экземпляр MHZ19 myMHZ19, и вызывается myMHZ19.begin(mySerial). Ключевым моментом является вызов myMHZ19.autoCalibration(false), который отключает автоматическую калибровку. Это необходимо в условиях, где уровень углекислый газ постоянно повышен, поскольку автокалибровка может привести к систематическим ошибкам измерений.

Основная логика работы реализована в бесконечном цикле loop(): периодический опрос датчиков, обработка полученных данных, их усреднение и вывод на экран. Для организации временных интервалов без использования блокирующей функции delay() применяется функция millis() , которая возвращает количество миллисекунд, прошедших с момента включения устройства. Это позволяет эффективно управлять таймерами, не блокируя выполнение других процессов [4].

Алгоритмическая логика сбора и обработки данных включает три ключевых элемента:

1. Управление временем. Сравнивая текущее значение с сохраненным временем последнего измерения, программа определяет, настал ли момент для нового опроса датчиков.

2. Последовательный опрос датчиков. Если интервал истек, вызываются функции для чтения данных.

3. Подготовка данных для усреднения. Полученные значения добавляются в специальные суммирующие переменные, а счетчик выборок увеличивается на единицу. Этот метод скользящего среднего широко применяется для сглаживания случайных шумов и кратковременных выбросов, что позволяет получить более стабильное и достоверное представление о состоянии микроклимата.

Заключительный этап обработки данных происходит после накопления достаточного количества выборок.

1. Математический расчет средних значений. Выполняются операции деления сумм на количество выборок. Эти вычисления представляют собой арифметическое среднее и являются классическим примером фильтрации сигнала, целью которой является подавление случайного шума.

2. Визуализация. На OLED-дисплее и последовательный порт: алгоритм включает очистку экрана, установку курсора, вывод текстовых меток и значений и обновление содержимого. Это позволяет отлаживать программу, анализировать динамику изменений и контролировать работоспособность системы в реальном времени.

3. Подготовка к следующему циклу. После отображения данных все суммирующие переменные и счетчик выборок обнуляются, готовя систему к сбору новых данных.

Экспериментальная часть исследования направлена на практическое подтверждение корректности работы разработанной системы мониторинга микроклимата. Эксперимент состоит из двух этапов.

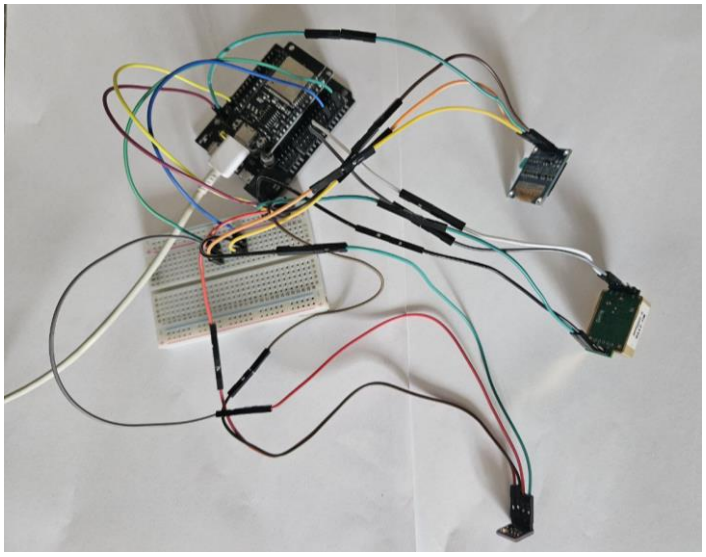


Рис. 1. Макет устройства

Первый этап эксперимента заключался в проверке точности измерения температуры путем сравнения показаний датчика BME280, отображаемых на OLED-дисплее, с эталонными значениями, полученными от бытового комнатного термометра. Измерения проводились в нескольких точках помещения при различных условиях.

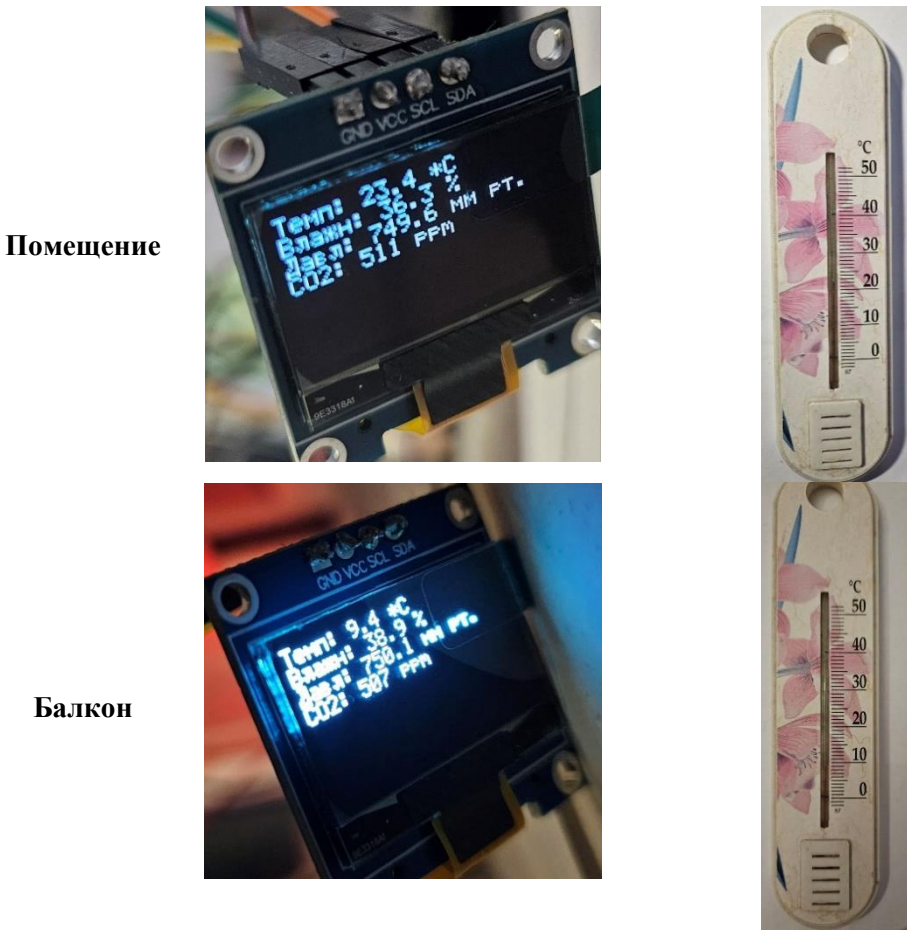
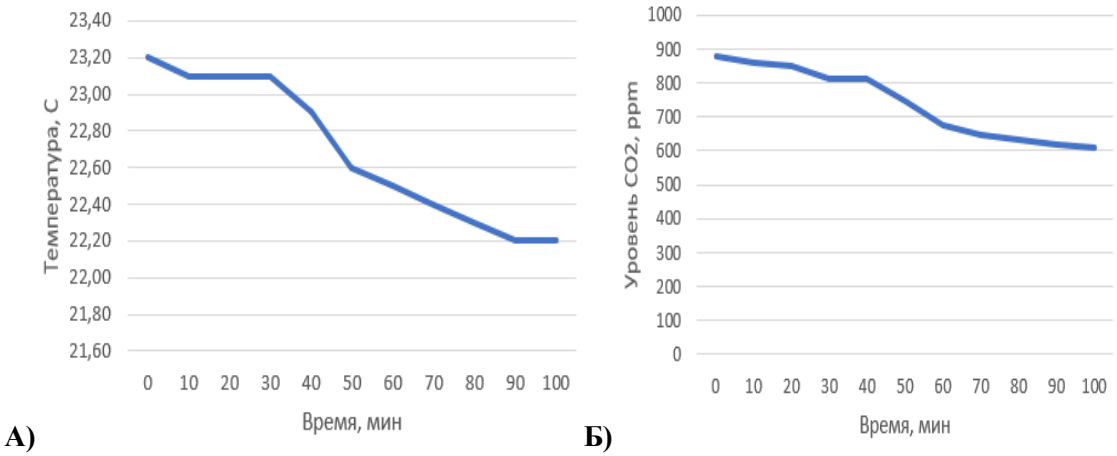


Рис. 2. Сравнение показаний датчика BME280 и эталонного термометра

В жилом помещении температура по BME280 составляла 22,5 °C, у термометра 22,0°C что дает абсолютную погрешность 0,5 градуса. На закрытом балконе значения составили 15,2°C и 15,0 °C соответственно, с погрешностью 0,2 градуса. Эти результаты подтверждают приемлемость точности датчика BME280 для задачи мониторинга микроклимата, учитывая его собственную погрешность и влияние нагрева от микроконтроллера.

Второй этап эксперимента был направлен на исследование влияния проветривания на уровень углекислого газа и температуры в помещении. Этот эксперимент наглядно демонстрирует функциональность системы и ее способность отслеживать динамические изменения параметров воздушной среды. Изначально, при закрытом окне, уровень CO_2 составлял 1200 ppm. После открытия окна наблюдался экспоненциальный спад концентрации: через 30 минут – 650 ppm, через 60 минут – 500 ppm, и через 100 минут – 450 ppm. Считается, что в помещении уровень диоксида углерода не должен превышать атмосферные значения примерно в 1,5 раза, то есть до 450–800ppm [2].



А) Б)

Рис. 3. Изменение А) температуры, Б) углекислого газа

Полученный график зависимости уровня CO_2 от времени имеет характерный вид экспоненциального спада, наблюдающийся спустя 30 минут после начала эксперимента, типичный для процессов смешивания газов в помещениях. Открытие окна создает поток свежего воздуха, который быстро смешивается со старым воздухом в комнате, приводя к его обогащению кислородом и обеднению углекислым газом. Система мониторинга, используя датчик MH-Z19, успешно фиксирует этот процесс, демонстрируя его практическую применимость для оценки качества воздуха и эффективности вентиляции.

Таким образом, полученные результаты подтверждают, что разработанное устройство корректно работает и способно надежно измерять ключевые параметры микроклимата, отслеживая динамические изменения в окружающей среде в реальном времени.

Литература

1. Лисина Т. Б. Алгоритм программы для мониторинга микроклимата в помещении // Информационные технологии в современном мире – 2025: материалы XXI Всероссийской студенческой конференции (г. Екатеринбург, 16 мая 2025 года). Екатеринбург: Гуманитарный университет, 2025. С. 108-110.

2. Савченко О.А., Чуенко Н.Ф., Рева М.В., Луговская А.Ю. Проблемы нормирования содержания диоксида углерода в окружающей среде, и его влияние на человека // Интерэкспо Гео-Сибирь. 2025. Т. 3. С. 220-226. https://doi.org/10.33764/2618-981X-2025-3-220-226

3. Hercog D., Lerher T., Truntič M., Težak O. Design and Implementation of ESP32-Based IoT Devices // Sensors. 2023. Vol. 23, No. 15. Art. 6739. https://doi.org/10.3390/s23156739

4. Morchid A., Taibi Bouali E., Oughannou Z., Alami R.E., Qjidaa H., Ouazzani Jamil M., Boufounas E.-M., Abid M.R. IoT-Enabled Smart Agriculture for Improving Water Management: A Smart Irrigation Control Using Embedded Systems and Server-Sent Events // Scientific African. 2025. Art. e02527. https://doi.org/10.1016/j.sciaf.2024.e02527

5. Szczurek A., Gonstał D., Maciejewska M. A Multisensor Device Intended as an IoT Element for Indoor Environment Monitoring // Sensors. 2024. Vol. 24, No. 5. Art. 1461. https://doi.org/10.3390/s24051461

6. Vitali G., Arru M., Magnanini E. A Scalable Device for Undisturbed Measurement of Water and CO_2 Fluxes through Natural Surfaces // Sensors. 2023. Vol. 23, № 5. Art. 2647. https://doi.org/10.3390/s23052647

Локай Д.С., Мотченко Л.А.
Донбасский государственный технический университет
г. Алчевск, ЛНР, Россия

ЦИФРОВАЯ ПОДДЕРЖКА КАРЬЕРНОЙ ТРАЕКТОРИИ СОИСКАТЕЛЯ НА ОСНОВЕ ВЕБ-СЕРВИСА ФОРМАЛИЗАЦИИ ПРОФЕССИОНАЛЬНОГО РЕЗЮМЕ

Резюме остаётся базовым инструментом коммуникации между соискателем и работодателем, однако переход к цифровому рынку труда радикально меняет требования к структуре и содержанию этого документа. От рекрутера и информационных систем ожидается работа не с произвольным текстом, а с относительно формализованным набором сведений о профессиональной биографии кандидата, пригодным для машинной обработки. В условиях массового онлайн-рекрутинга стихийное «ручное» составление резюме по случайным образцам приводит к потере информации, снижению сопоставимости профилей и росту нагрузки на HR-специалистов [2].

Актуальность разработки специализированных веб-сервисов создания резюме обусловлена тем, что ключевые бизнес-процессы в сфере подбора персонала стремительно интегрируются в глобальную сеть. Типичный жизненный цикл взаимодействия соискателя и работодателя включает: публикацию вакансии, формирование резюме, подачу отклика через электронные каналы, предварительный автоматический скрининг, экспертное рассмотрение и последующие этапы отбора. На каждом шаге задействованы информационные системы: корпоративные порталы, система отслеживания кандидатов (ATS), HRM-платформы. Если исходное резюме плохо структурировано, содержит типичные ошибки (отсутствие ключевых разделов, нарушения хронологии, некорректные контакты), это искажает работу всей цепочки: алгоритмы ранжирования дают некорректные результаты, рекрутеры тратят время на ручное «дочитывание», а соискатель теряет шансы пройти первичный фильтр [3; 4]. Поэтому сервис, который изначально формирует «технически корректное» резюме, становится важным элементом цифровой инфраструктуры рынка труда.

Лингвистический анализ массивов электронных резюме показывает, что тексты соискателей подчиняются устойчивым жанровым и структурным нормам: воспроизводятся блоки «образование», «опыт», «навыки», используются типовые формулы описания обязанностей и достижений [2]. Одновременно выявляются систематические дефекты: пропуски ключевых разделов, смешение периодов, чрезмерная общность формулировок. Эти проблемы напрямую влияют на эффективность бизнес-процессов отбора: снижается качество входящего потока кандидатов, возрастают затраты времени на оценку резюме. Веб-сервис, который задаёт пользователю структуру данных и контролирует их корректность, можно рассматривать как инструмент снижения этих потерь.

Сравнительный анализ существующих программных решений позволяет выделить несколько классов ПО. К первому классу относятся простые онлайн-генераторы, предлагающие минимальный набор полей и один шаблон оформления. Они удобны, но слабо учитывают требования ATS-систем, почти не поддерживают повторное использование данных и плохо вписаны в цепочку бизнес-процессов работодателя. Второй класс формируют

комплексные карьерные платформы, где модуль резюме встроен в более широкий контекст: поиск вакансий, рекомендации, календарь интервью. Эти решения глубоко интегрированы с корпоративными HR-процессами, но часто являются «закрытыми»: профили пользователей привязаны к одной платформе и слабо переносимы в другие системы [5]. Третий класс составляют специализированные веб-сервисы конструирования резюме, ориентированные на формализацию данных: профиль соискателя хранится как набор сущностей (образование, опыт, навыки и др.), а визуальное представление резюме автоматически генерируется из этой модели [5]. Предлагаемый нами подход относится именно к этому третьему классу и объединяет простоту использования с открытостью модели данных.

Ключевым проектным решением является использование профильной модели представления данных о соискателе. Профессиональный профиль описывается как набор взаимосвязанных сущностей: личные и контактные сведения, формальное образование, дополнительное обучение, трудовой опыт, профессиональные и мягкие навыки, участие в проектах, владение иностранными языками, портфолио и цифровые идентичности. Каждая сущность имеет чётко заданный набор атрибутов и ограничения целостности (обязательность поля, формат даты, формат e-mail и т. п.). Для бизнес-процессов подбора персонала это означает, что данные о кандидате могут быть агрегированы, отфильтрованы и сопоставлены с требованиями вакансий, а для информационных систем профиль соискателя легко транслируется в структуры, ожидаемые системами электронного рекрутинга и автоматизированного анализа резюме [1; 3].

Практика электронного рекрутинга показывает, что для корректной работы ATS и модулей аналитики важно не только наличие разделов резюме, но и единообразие описания сведений. Даже при заполненных полях качество данных может оставаться низким: одинаковые должности фиксируются разными формулировками, смешиваются уровни квалификации, не указываются технологии, критичные для автоматического ранжирования. В результате снижается сопоставимость профилей, а первичный скрининг становится менее точным, что увеличивает нагрузку на HR-специалистов и удлиняет цикл найма [3]. Поэтому веб-сервису целесообразно не ограничиваться набором полей, а поддерживать нормализацию данных: предлагать контролируемые справочники (должности, области деятельности), подсказывать типовые значения и предотвращать семантические дубликаты. Такой подход сохраняет удобство для пользователя, но при этом повышает технологическую «пригодность» резюме для дальнейшей машинной обработки и интеграции с карьерными платформами [5].

Отдельного внимания требует описание навыков и компетенций. В работах, посвящённых интеллектуальному анализу текстов резюме и вакансий, подчёркивается, что качество сопоставления кандидата с требованиями работодателя зависит от полноты и точности терминов, которыми описан опыт [3]. В рамках профильной модели навыки целесообразно фиксировать не только как свободный текст, но и как структурированный элемент: название навыка, уровень владения, контекст применения (учёба, проект, работа), подтверждающий факт (результат, роль, используемые инструменты), а при необходимости – ссылку на портфолио или артефакт. Если сервис предлагает пользователю выбор из подсказок

и одновременно сохраняет исходную формулировку, это позволяет, с одной стороны, не «зажимать» индивидуальный стиль резюме, а с другой – обеспечить более устойчивую основу для поиска и фильтрации по компетенциям. В результате снижается барьер для автоматизированного анализа и упрощается интеграция с методами обработки резюме, упоминаемыми в исследованиях и практических разработках [1; 3].

Контроль корректности вводимых сведений должен учитывать типичные ошибки, выявленные при анализе электронных резюме: пропуски ключевых разделов, несогласованность дат, нарушения хронологии, некорректные контакты и чрезмерно общие описания обязанностей [2]. Веб-сервис может реализовать многоуровневую валидацию: проверку форматов (телефон, e-mail), логические ограничения (дата начала опыта не позже даты окончания; «по настоящее время» допускается только для одной позиции), контроль обязательных полей и предупреждения о пропущенных элементах. Полезной является и «мягкая» валидация – например, подсказка о необходимости конкретизации результатов, если пользователь вводит слишком короткое описание. Подобные механизмы переводят часть рутины рекрутера в интерфейс сервиса и снижают вероятность дефектов, описанных в исследованиях жанра электронного резюме [2].

Поскольку резюме остаётся жанровым документом, важной функцией сервиса становится поддержка пользователя в формулировании содержательных описаний. Лингвистические наблюдения показывают, что соискатели часто воспроизводят шаблонные формулы без указания результата и масштаба работы, из-за чего работодателю сложно оценить вклад кандидата [2]. Поэтому интерфейс может включать контекстные подсказки: примеры глагольных конструкций для описания задач, напоминания о выделении достижений и рекомендации по добавлению измеримых показателей (сроки выполнения, объём работ, доля автоматизации, оптимизация затрат). При этом сервис не должен «писать за пользователя», а должен направлять: задавать структуру описания и стимулировать конкретику. Такой подход повышает информативность резюме и делает его более пригодным для алгоритмов анализа, которые опираются на ключевые сущности и маркеры результата при обработке резюме и вакансий [3].

С точки зрения интеграции с цифровой инфраструктурой рынка труда полезно предусмотреть несколько режимов экспорта. Помимо визуально оформленного PDF, востребована выгрузка структурированного представления профиля, пригодного для загрузки в корпоративные системы и последующей аналитики: например, как набор полей для импорта в HRM/ATS либо как единый файл профиля для переноса между платформами. В специализированных конструкторах резюме повторное использование данных профиля рассматривается как ключевое условие удобства для пользователя и технологической совместимости с внешними сервисами [5]. Кроме того, наличие формализованного профиля облегчает применение инструментов автоматизированного сбора и анализа резюме, включая парсеры и аналитические модули, описываемые в профильной литературе [1].

При проектировании сервиса необходимо учитывать требования информационной безопасности и защиты персональных данных, поскольку профиль соискателя включает

контактные сведения и элементы профессиональной биографии. На уровне архитектуры оправданы разграничение прав доступа (соискатель, администратор, HR-пользователь), хранение паролей в виде криптографических хешей, защита сессий, журналирование критичных действий, а также безопасная работа с базой данных посредством параметризованных запросов. Важно предусмотреть управление согласиями: какие поля доступны при экспорте, какие – при передаче в сторонние системы, а также возможность удаления или деактивации профиля. Для пользователя такая политика доступа повышает доверие и делает модель хранения профиля более устойчивой к рискам утечки данных [5].

Дополнительно целесообразно выделить требования к пользовательскому сценарию заполнения резюме, поскольку качество итогового профиля во многом определяется тем, насколько последовательно сервис ведёт пользователя и минимизирует когнитивную нагрузку. Для этого можно применять пошаговый мастер (wizard) с прозрачной структурой разделов, индикатором прогресса и возможностью возвращаться к ранее заполненным блокам без потери данных. В контексте жанровых особенностей резюме важно поддерживать баланс между стандартизацией и индивидуализацией: сервис может предлагать несколько шаблонов оформления и набора разделов под разные роли (например, разработчик, аналитик, менеджер), при этом не ограничивая пользователя жёстким «каркасом» [2]. Удобным решением становится контекстная адаптация формы: если пользователь указывает опыт управления командой, система предлагает поля о размере команды, метриках эффективности и ответственности; если указан учебный проект – поля о роли, инструментах и результате. Такие механизмы повышают полноту и сопоставимость данных, что особенно важно для дальнейшего поиска, фильтрации и автоматизированного анализа профилей, описываемых в работах по обработке резюме и поддержке отбора [1; 3]. Наконец, при использовании конструктора резюме как долгоживущего инструмента полезно предусмотреть сохранение нескольких версий профиля (например, «под продуктовую компанию» и «под консалтинг») и историю изменений, что облегчает адаптацию резюме под разные вакансии без повторного ввода информации [5].

Наконец, формализованная модель данных открывает возможность построения прикладной аналитики, ориентированной на поддержку карьерной траектории. На основе структурированных полей можно рассчитывать показатели полноты профиля, выявлять «слабые места» (недостаточно описанные проекты, отсутствие подтверждённых навыков), формировать рекомендации по улучшению резюме под требования конкретных вакансий и отслеживать динамику изменений. Поскольку исследования подчёркивают перспективность интеллектуального сопоставления резюме и вакансий, наличие исходно структурированных данных снижает порог внедрения таких методов и повышает качество результирующих рекомендаций [3]. Это даёт основу для развития экосистемы инструментов, описываемых в работах об автоматизации отбора и анализе резюме [1; 3].

Архитектура веб-сервиса реализуется как многоуровневое клиент-серверное приложение. Уровень представления отвечает за интерактивный пользовательский интерфейс: формы ввода и область предпросмотра реализованы на HTML5, CSS3 и JavaScript.

Клиентская часть обеспечивает «живое» обновление вида резюме при изменении полей, подсветку ошибок и переключение шаблонов. Серверный уровень реализует функции аутентификации, управления сессиями, валидации данных и работы с реляционной базой. Доступ к СУБД выполняется через параметризованные запросы, что снижает вероятность SQL-инъекций и нарушений целостности. Такая архитектура согласуется с тенденциями разработки веб-систем в области электронного рекрутинга и карьерных сервисов [5].

Отделение слоя данных от слоя представления реализуется с помощью системы шаблонов резюме. Шаблон задаёт порядок блоков, визуальные акценты и правила отображения полей. Один и тот же профиль может быть автоматически развёрнут в несколько шаблонов: классический (акцент на последовательности опыта), компетентностный (акцент на навыках и результатах), академический (расширенный блок образования и научной деятельности). Это даёт возможность адаптировать резюме под разные бизнес-сценарии: массовый рекрутинг, стажировки, академическое трудоустройство [5].

Для соискателя базовый процесс включает регистрацию, поэтапное заполнение структурированных блоков, интерактивную корректировку формулировок и экспорт резюме в формат, удобный для подачи откликов (PDF или структурированный HTML). Встроенные механизмы валидации отслеживают заполнение ключевых разделов, корректность формата контактных данных и непротиворечивость временных интервалов, тем самым устраняя типичные ошибки, зафиксированные в исследованиях электронных резюме [2]. Для работодателя и HR-службы основными процессами являются первичный скрининг резюме, поиск по ключевым навыкам и фильтрация по опыту и образованию. Структурированное представление данных облегчает автоматизацию этих операций. Для систем автоматизированного подбора и управления персоналом особенно важно, чтобы данные о кандидате можно было легко загружать и обрабатывать. Формализованное описание профиля позволяет напрямую использовать их в таких системах и применять специализированные методы автоматического разбора резюме [1; 3].

В ряде работ подчёркивается, что дальнейшее повышение эффективности подбора персонала связано с внедрением автоматизированных средств сбора и анализа резюме, включая роботов-парсеров, способных извлекать ключевую информацию из неструктурированных документов и подготавливать её для последующей аналитики [1]. Наличие изначально структурированного профиля в веб-сервисе существенно упрощает интеграцию с подобными инструментами: вместо сложного синтаксического разбора текста достаточно использовать уже нормализованные поля. Это снижает технические риски, упрощает развитие цифровых HR-процессов и открывает возможность построения комплексных аналитических панелей для оценки качества потока кандидатов.

Таким образом, веб-сервис создания резюме может рассматриваться как важный элемент цифровой поддержки карьерной траектории соискателя. Во-первых, формализованная модель данных резюме и архитектура сервиса повышают качество и структурированность информации о кандидате, что напрямую влияет на эффективность бизнес-процессов подбора персонала. Во-вторых, встроенные механизмы валидации и интерактивного предпросмотра

снижают порог входа для пользователя и уменьшают количество типичных ошибок, описанных в исследованиях электронных резюме. В-третьих, использование реляционной базы, чётко определённых сущностей и ориентация на интеграцию с системами электронного рекрутинга создают основу для построения аналитических модулей и внедрения методов интеллектуального анализа текстов резюме и вакансий. Всё это позволяет рассматривать данный класс веб-сервисов как перспективное направление развития цифровых инструментов сопровождения карьеры.

Литература

1. Коробова О. О. Эффективные методы подбора персонала // Современные наукоёмкие технологии. Региональное приложение. 2025. № 3(83). С. 24-36. https://doi.org/10.6060/snt.20258303.0003

2. Мисюро А. Я. Содержательное наполнение электронных резюме соискателей на вакансию из Беларуси // The Scientific Heritage. 2021. № 74-3(74). С. 44-47. https://doi.org/10.24412/9215-0365-2021-74-3-44-47

3. Сизова И. Л. Особенности подбора персонала: интеллектуальный анализ текстов резюме и вакансий // Регионология. 2025. Т. 33, № 2(131). С. 271-293. https://doi.org/10.15507/2413-1407.129.033.202502.271-293

4. Черняков М. К., Чернякова И. А. Цифровая трансформация HR-сферы: анализ технологий, эффективности и перспектив развития // Международный научно-исследовательский журнал. 2025. № 9 (159). https://doi.org/10.60797/IRJ.2025.159.27. URL: https://research-journal.org/media/articles/20027.pdf (дата обращения: 18.02.2026).

5. Arora A. Rezume: A Web-based Resume Builder System for University Students: Master’s Project // Rochester Institute of Technology, 2021. 63 p. URL: https://clck.ru/3TjPVo (дата обращения: 12.01.2026)

Маслова А.К.
Дальневосточный государственный университет путей сообщений
г. Хабаровск, Россия

ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА СИСТЕМЫ ПЕРСОНАЛЬНОГО ПЛАНИРОВАНИЯ НА ОСНОВЕ МОДЕЛИ ЦИФРОВОГО ДВОЙНИКА СОТРУДНИКА

Современные компании уже давно начали оптимизировать и механизировать тяжкие и трудоёмкие процессы. Однако большая часть компании забывает о планировании процессов и проектов, что в дальнейшем приводит к эмоциональному выгоранию сотрудников, перезагрузок рабочего времени и переноса сроков сдачи проекта. Цель данной научной работы: разработать цифрового двойника, способного распределять задачи на неделю, оценивая загруженность сотрудника, сложность задачи и уровень концентрации для задачи. Данная разработка может использоваться не только компаниями, но и в повседневной жизни для оптимизации рутинных дел.

Цифровой двойник – это виртуальная модель, которая точно воспроизводит характеристики, поведение и состояние физического объекта, процесса или системы [3]. На основе введённых пользователем данных запрограммированная программа делает необходимые расчёты и предоставляет пользователю оптимальный план загрузки, подходящий под его расписание.

Концепция цифрового двойника появилась в 2003 г., когда Майкл Гривс, вдохновлённый работой с Джоном Викерсом, предложил новую концепцию управления жизненным циклом продукта. Идея заключалась в том, чтобы перейти от бумажной документации о продукте к цифровой модели, которая была бы единственной точкой отсчёта для управления всеми этапами жизненного цикла продукта [1].

Технология цифровых двойников выходит далеко за пределы простой визуализации – она создаёт динамическую копию, способную:

- собирать и обрабатывать данные с физических датчиков;
- прогнозировать поведение реального объекта при разных сценариях;
- выявлять скрытые закономерности и аномалии в работе оригинала;
- тестировать изменения без риска для реального объекта;
- оптимизировать процессы, повышая эффективность на 15–30%.

При создании цифрового двойника используют объектно-ориентированный подход к представлению сложных систем. Этот подход позволяет представить сложную структуру простым и понятным для человека образом [2]. Программа выполняет сложные вычисления по формулам, используя не одну функцию. А пользователю только необходимо ввести запрашиваемые данные и в результате он получит структурированные данные в удобной форме.

По данным аналитиков Gartner, к 2027 году более 85% крупных промышленных компаний будут использовать цифровые двойники, что приведёт к снижению производственных затрат на 10–20% и сокращению времени вывода продуктов на рынок до

50% [4; 5]. Переход от прогноза к цифровой трансформации предприятий требует решения фундаментальной задачи – интеграции данных. В качестве решения данной задачи предлагается система, основанная на работе с тремя типами данных:

– данные о задачах включают в себя: название задачи, тип (встреча, трудоёмкая работа, рутина и обучение), сложность, ожидаемая длительность в часах, приоритет, срок сдачи (если имеется);

– данные о сотруднике включают в себя: предпочтительное время работы, время обеда, часы максимальной продуктивности, текущая загрузка по дням недели;

– расписание включает в себя: запланированные задачи по дням, общая нагрузка на каждый день.

Для оценки нагрузки от задачи была использована формула (1):

$$S = m \times R \times k \quad (1)$$

где S –сложность задачи, m – вес типа задачи, R – уровень сложности задачи, k – коэффициент длительности.

Вес типа задачи зависит от её вида. Самым большим весом обладает трудоёмкая работа, так как является самой тяжёлой среди остальных типов. Следующими по сложности является встреча и обучения. Последним по сложности будут рутинные задачи. В табл. Таблица представлены значения веса типа задачи, используемые в программе.

Таблица 1

Значения веса типа задачи

Тип задачи	Вес
Трудоёмкая работа	1.5
Встреча	1.2
Обучение	1.0
Рутинные дела	0.8

В программе предусмотрено 4 уровня сложности от 1 до 4, которые пользователь вводит сам. Введённый пользователем уровень сложности умножается на 0.8, позволяя сохранить ощутимую нагрузку и при этом не допустить чрезмерного доминирования.

Коэффициент длительности считается по формуле (2):

$$k = \frac{1 + t^{0,7}}{3} \quad (2)$$

где t – длительность задачи в часах.

Данная формула гарантирует что даже у короткой по времени задачи коэффициент не будет меньше 1. Возведение в степени создаёт реалистичную сложность, потому что не всегда задачи длиною в 3 часа сильно сложнее задач длинной в 1 час. Деление на 3 позволяет избежать сильно больших значений для коэффициента.

При распределении задач используется следующий алгоритм:

1. Сортировка задач. Задачи сначала сортируются по приоритету, затем по сроку сдачи и в конце по сложности. Данная модель позволяет правильно распределить задачи.

2. Выбор дня. Для каждой задачи программа ищет подходящий день по некоторым правилам:

– день должен быть рабочим (понедельник – пятница);
– общая нагрузка дня и сложность задачи в сумме ≤ 8;
– учитывается срок сдачи;
– сложные задачи программа старается ставить в середине недели.

3. Выбор времени. Для каждого типа задачи есть рекомендуемые промежутки времени:

– трудоёмкая работа: с 9:00 до 15:00;
– встречи: с 11:00 до 16:00;
– обучение: с 10:00 до 16:00;
– рутинные дела: с 12:00 до 17:00.

Система построена как класс (*DigitalTwinPlanner*) на языке *Python* с основными методами:

– *add_task()* – добавление новой задачи;
– *auto_schedule_tasks()* – автоматическое распределение;
– *show_schedule()* – показ расписания;
– *save_to_file()* – сохранение данных;
– *load_from_file()* – загрузка данных.

Данные хранятся в формате json-файла и сохраняются по указанному в коде пути. Также предоставлена возможность загрузить исходные данные, которые должны быть представлены в формате json-файла. Данная возможность позволяет добавлять новые задачи к уже спланированной рабочей неделе.

Для пользователя был разработан консольный интерфейс с меню. При запуске программы пользователь видит список со всеми возможными задачами, ему предлагается ввести номер необходимой задачи. После выполнения выбранной задачи, осуществляется возврат в основное меню с повторным выбором. Таким образом реализуется цикл задач. Консольное меню пользователя представлено на рисунок 1.

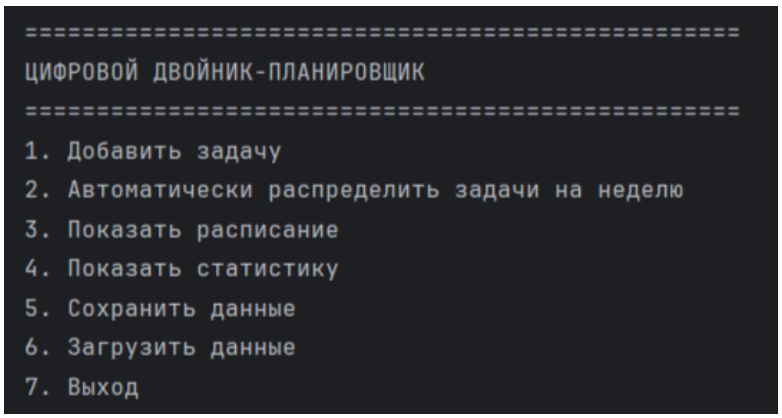


Рис. 1. Консольное меню пользователя

Для примера работы системы было подготовлено несколько сценариев работы программы:

– Сценарий 1: Добавление задачи. Пользователь из предложенных задач выбираем задачу под номером 1 (Добавить задачу). После программа запрашивает от пользователя название задачи, предлагает выбрать тип, указать сложность и длительность, установить приоритет и задать срок сдачи (если имеется). Далее по формуле (1) рассчитывается сложность задачи и выбирается время и день. Результат отработки сценария 1 представлен на рисунке 2.

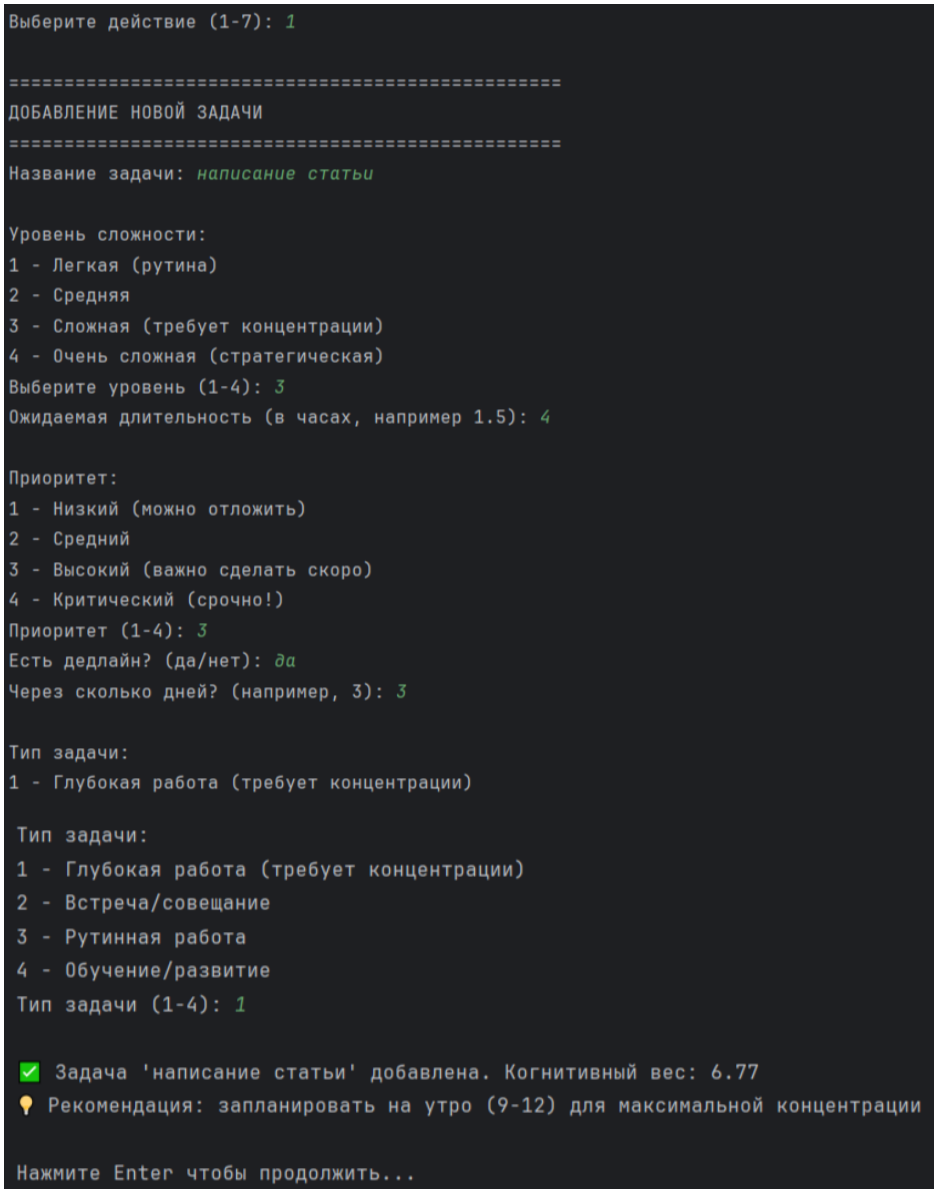


Рис. 2. Добавление задачи

– Сценарий 2: Распределение задач. Пользователь из предложенных задач выбираем задачу под номером 2 (Распределить задачи на неделю). Программа по описанному ранее алгоритму выбирает наиболее подходящие день время для каждой задачи. Результат отработки сценария 1 представлен на рисунке 3.

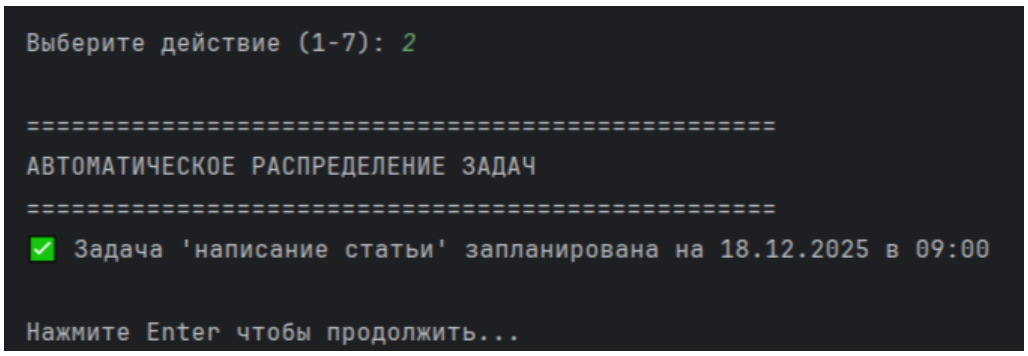


Рис. 3. Автоматическое распределение задачи

– Сценарий 3: Просмотр расписание. Пользователь из предложенных задач выбираем задачу под номером 3 (Показать расписание). Программа выводит список с распределённым по дням и времени задачами. Результат отработки сценария 1 представлен на рис. 4**Рис.** .

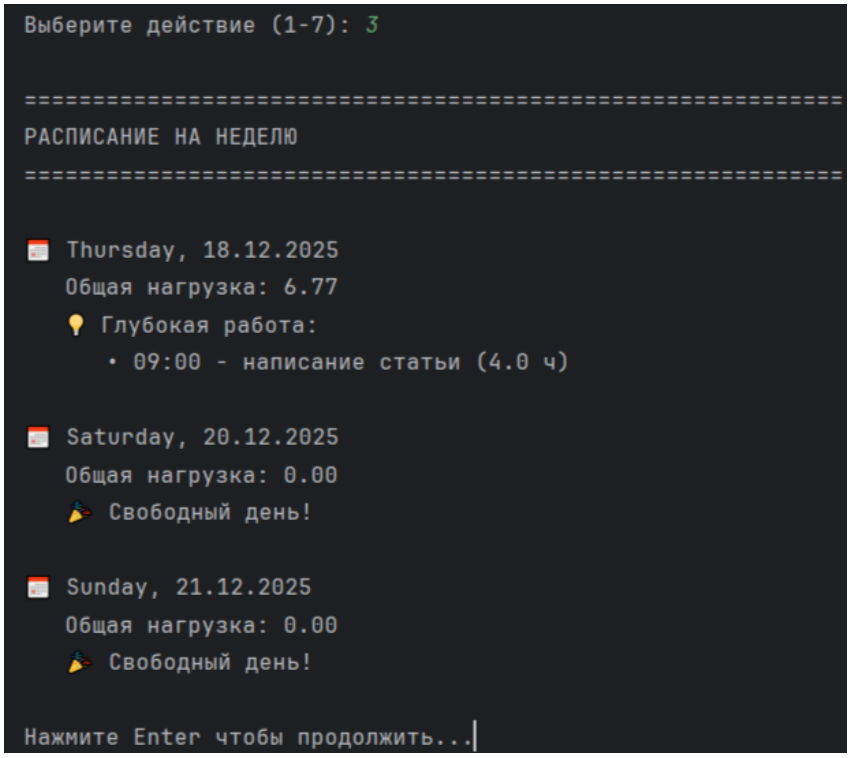


Рис. 4. Просмотр расписания

Преимуществами разработанной система для сотрудника является: снижения стресса, баланс нагрузки, наглядность и учёт особенностей. Все эти характеристики позволяют увеличить продуктивность сотрудника не в ущерб его эмоциональному состоянию.

Цифровая трансформация организаций с использованием цифровых двойников приведёт к повышению продуктивности и создаст условия для выполнения работ в указанные сроки, а эмоциональная составляющая коллектива предотвратит утечку кадров.

В дальнейшем планируется усовершенствование тестовой модели под незапланированные ранее задачи, чтобы обеспечить быстрое перераспределение всех задач. Также планируется добавить оценку эмоционального состояния сотрудника в текущий день для создания индивидуального рабочего плана для каждого.

Практическая значимость разработки состоит в том, что уже на данном этапе она может быть использована компаниями, учебными заведениями и в личных целях. Цифровая трансформация позволить структурировать задачи и улучшить планирование, учитывая индивидуальные особенности человека.

Литература

1. Лапина М. А., Багаутдинова А. Р., Лапин В. Г., Лапин В. В. Цифровые двойники: обзор решений и перспективы развития // Auditorium. 2024. №3 (43). URL: https://clck.ru/3TJgav (дата обращения: 17.12.2025).

2. Лихтциндер Б.Я. Связь и цифровые двойники // T-Comm: Телекоммуникация и транспорт. 2023. Т. 17. № 8. С. 30-37.

3. Румянцева И.А., Кротенко Т.Ю., Жернакова М.Б. Оценка мотивации и демотивации сотрудников в качестве параметров модели «цифровой двойник организации» // Вестник университета. 2020. № 6.

4. Benoit Leleux digital twins entering the mainstream // Gartner. 2019.

5. Gartner Identifies the Top 10 Strategic Technology Trends for 2023 // Gartner URL: https://clck.ru/3TJgbz (дата обращения: 16.12.2025).

Орлов Д.Ю., Слива М.В.
Нижневартовский государственный университет
г. Нижневартовск, Россия

ПРИМЕНЕНИЕ МИКРОСЕРВИСНОЙ АРХИТЕКТУРЫ ДЛЯ СОЗДАНИЯ ИНФОРМАЦИОННОЙ СИСТЕМЫ УПРАВЛЕНИЯ ПРОЕКТАМИ

В условиях текущей цифровой экономики, где проекты становятся ключевым инструментом реализации бизнес-стратегий и создания инновационных продуктов, требования к инструментам управления проектами кардинально возрастают. Компании сталкиваются с необходимостью координировать распределенные команды, управлять сложными ресурсными зависимостями и адаптироваться к динамично меняющимся рыночным условиям. В таком контексте информационные системы управления проектами (ИСУП) перестают быть просто вспомогательными утилитами и превращаются в стратегические платформы, от эффективности которых напрямую зависит конкурентоспособность бизнеса [6]. Систематические обзоры в этой области подтверждают, что эффективное использование ИСУП напрямую влияет на успешность завершения проектов в срок и в рамках бюджета [4].

Традиционный подход к построению подобных систем, основанный на монолитной архитектуре, демонстрирует все большее несоответствие современным вызовам [2, с. 25]. Единая, неразрывная кодовая база, несмотря на первоначальную простоту разработки, создает значительные препятствия для гибкости и масштабирования. Любое, даже незначительное, изменение в одном функциональном модуле требует полной пересборки и тестирования всего приложения, что замедляет циклы разработки. Кроме того, монолитная структура не позволяет избирательно наращивать мощность отдельных компонентов, что приводит к неэффективному использованию ресурсов и снижению общей производительности системы.

В качестве ответа на возникшие вызовы выступает микросервисная архитектура – современный архитектурный стиль, предполагающий декомпозицию приложения на набор небольших, автономно развертываемых и слабосвязанных сервисов. Каждый такой сервис концентрируется на выполнении одной конкретной бизнес-задачи: управление пользователями, ведение задач, формирование отчетов. С этой реализацией, сервис может иметь собственную технологическую базу, базу данных и команду разработки. Такой подход не только обеспечивает технологическую независимость и ускоряет разработку за счет параллельной работы команд, но и фундаментально меняет принципы масштабирования и обеспечения отказоустойчивости системы.

Актуальность данного исследования продиктована растущим спросом со стороны заказчиков на высокопроизводительные и гибкие ИСУП, способные обрабатывать большие объемы данных в режиме реального времени и гарантировать непрерывность бизнес-процессов. Переход к микросервисам становится не просто техническим трендом, а насущной необходимостью для компаний, стремящихся оптимизировать управление проектами и получить аналитическое преимущество. Как показывают современные исследования, грамотно спроектированная микросервисная система способна значительно повысить

производительность и сократить время отклика по сравнению с классическими монолитными решениями [6].

Целью настоящей работы является формулирование и теоретическое обоснование архитектуры системы управления проектами, построенной на микросервисном подходе. Ключевыми критериями проектируемой архитектуры являются высокая масштабируемость, позволяющая динамически подстраиваться под нагрузку, и отказоустойчивость, обеспечивающая сохранение работоспособности системы при сбоях отдельных компонентов. Для достижения поставленной цели необходимо решить следующие задачи: провести сравнительный анализ архитектурных подходов, разработать модель декомпозиции системы на микросервисы, предложить оптимальный технологический стек для реализации и оценить потенциальные характеристики производительности и надежности предложенной архитектуры.

Теоретические основы микросервисной архитектуры. Микросервисная архитектура – не просто технологическое решение, а целостная парадигма построения программных систем, базирующаяся на наборе фундаментальных принципов и концепций. Понимание этих основ необходимо для грамотного проектирования масштабируемой и отказоустойчивой системы управления проектами. Далее рассматриваются ключевые теоретические аспекты, определяющие суть микросервисного подхода.

В основе микросервисной архитектуры лежат несколько принципов, которые коллективно обеспечивают её гибкость и масштабируемость [2].

- **Принцип единственной ответственности.** Каждый микросервис фокусируется на выполнении одной четко определенной бизнес-функции или управляет определенной предметной областью (Domain-Driven Design). Например, в системе управления проектами могут существовать отдельные сервисы для управления пользователями, проектами, задачами и уведомлениями. Это позволяет изолировать логику и упрощает понимание и модификацию каждого компонента;
- **Независимость и автономность сервисов.** Микросервисы являются независимыми единицами развертывания. Это означает, что команда разработчиков может изменять, собирать и выпускать новую версию одного сервиса без необходимости пересборки и развертывания всей системы. Такая автономия значительно ускоряет непрерывную интеграцию и непрерывную доставку (CI/CD);
- **Децентрализованное управление данными.** В отличие от монолитного подхода, где все данные обычно хранятся в одной большой базе данных, в микросервисной архитектуре каждый сервис управляет собственной базой данных. Это позволяет выбрать наиболее подходящую СУБД (реляционную, графовую, NoSQL и т. д.) для конкретной задачи и избежать жесткой связанности на уровне данных;
- **Отказоустойчивость и изоляция.** Поскольку сервисы изолированы друг от друга, сбой в одном из них не должен приводить к каскадному отказу всей системы. Достигается за счет применения паттернов, таких как Circuit Breaker и Bulkhead, которые ограничивают распространение сбоев и обеспечивают работу системы в критических ситуациях;

- **Масштабируемость по требованию.** Одним из главных преимуществ является возможность независимого масштабирования. Если в системе управления проектами самым нагруженным оказался сервис push-уведомлений, можно увеличить количество его экземпляров, не затрагивая другие сервисы. Это обеспечивает лучшее использование вычислительных ресурсов по сравнению с масштабированием всей монолитной структуры приложения.

Для более глубокого понимания преимуществ микросервисов можно сопоставить их с традиционной монолитной архитектурой (табл.).

Таблица

Сравнительный анализ архитектурных подходов

Аспект	Монолитная архитектура	Микросервисная архитектура
Масштабируемость	Масштабируется приложение целиком, что может быть неэффективно.	Масштабируются только нагруженные сервисы, что экономит ресурсы.
Гибкость технологий	Обычно используется единый технологический стек для всего приложения.	Каждый сервис может использовать свой технологический стек и язык программирования.
Темпы разработки	Изменения в любой части требуют полного тестирования и развертывания системы.	Благодаря независимой работе команд, разработка и внедрение функций происходит быстрее.
Надежность	Одна ошибка в каком-либо модуле может привести к сбою всей системы.	Отказ одного сервиса изолирован, система в целом остается работоспособной.
Сложность развертывания	Изначально проще, но усложняется с ростом приложения.	Изначально сложнее из-за распределенной природы, но упрощает управление в долгосрочной перспективе.

Технологический стек и паттерны. Реализация микросервисной архитектуры опирается на определенный набор технологий и проектных решений.

Контейнеризация и оркестрация. Технологии контейнеризации, такие как Docker, стали стандартом для упаковки микросервисов. Они обеспечивают консистентность окружений на всех этапах (разработка, тестирование, производство). Для управления большими кластерами контейнеров используются системы оркестрации, например, **Kubernetes**, которые автоматизируют развертывание, масштабирование и восстановление сервисов.

API-шлюз. Так как у системы появляется множество независимых сервисов, возникает потребность в единой точке входа для клиентских приложений. **API-шлюз** выполняет роль фасада, направляя запросы к соответствующим сервисам, а также берет на себя сквозные задачи: аутентификацию, логирование, балансировку нагрузки.

Взаимодействие между сервисами. Сервисы взаимодействуют друг с другом по сети. Для этого используются два основных подхода:

- **Синхронное взаимодействие** через RESTful API или более эффективный gRPC.
- **Асинхронное взаимодействие** с помощью брокеров сообщений (RabbitMQ или Apache *Kafka* и др.). Асинхронный подход способствует дальнейшему ослаблению

связанности и повышению отказоустойчивости, так как сервисы не ждут непосредственного ответа друг от друга.

Управление данными. Как уже упоминалось, каждый сервис владеет своей базой данных. Для поддержания согласованности данных между сервисами вместо распределенных транзакций (крайне неэффективны в распределенных системах) часто используются паттерны Event Sourcing (хранение последовательности событий, изменивших состояние) и CQRS (Command Query Responsibility Segregation – разделение моделей для записи и чтения данных) [5].

Изложенные выше паттерны и принципы образуют методологическую базу для создания современных распределенных систем. Применяя эту методологию на практике, можно перейти к непосредственному проектированию архитектуры информационной системы управления проектами, где каждый теоретический аспект находит свое конкретное применение.

Проектирование архитектуры системы управления проектами. На основе теоретических принципов, изложенных ранее, была спроектирована архитектура ИСУП, ориентированная на высокую масштабируемость и отказоустойчивость. Ключевым решением является декомпозиция монолитной функциональности на набор слабосвязанных микросервисов, каждый из которых отвечает за свою предметную область.

Предлагаемая архитектура включает следующие основные микросервисы, взаимодействующие через API-шлюз, который выступает в роли единой точки входа для клиентских приложений:

- **User Service**: Ответственен за аутентификацию, авторизацию, управление профилями пользователей и ролями доступа (администратор, менеджер, пользователь). Этот сервис имеет критически важное значение, поскольку изолирует логику безопасности.
- **Project-Service**: Обеспечивает полный контроль над проектами: от запуска и внесения изменений до завершения и архивирования. Содержит данные о команде, времени выполнения и финансовых ресурсах.
- **Task-Service**: Служит центральным элементом системы, обеспечивая инструменты для формирования задач, их распределения и мониторинга текущего состояния. Обеспечивает возможность организации задач в иерархическую структуру (включая подзадачи) и установление взаимосвязей между ними.
- **Notification-Service**: Асинхронно обрабатывает и отправляет уведомления пользователям (e-mail, push-уведомления) о ключевых событиях, таких как назначение новой задачи или приближение дедлайна;
- **Analytics-Service**: Собирает и агрегирует данные из других сервисов для формирования отчетов и дашбордов. Использует отдельное хранилище данных для оптимизации аналитических запросов.

Обмен данными между микросервисами. Коммуникация основывается на двух ключевых моделях: синхронной и асинхронной. Синхронный обмен данными реализуется посредством RESTful API или gRPC для задач, нуждающихся в оперативном подтверждении.

Например, клиент запрашивает детали задачи, Task Service может синхронно обратиться к User Service для получения данных об исполнителе. Асинхронный обмен реализуется с помощью брокера сообщений RabbitMQ или Apache Kafka. Данный метод оптимален для событий, характеризующихся низкой степенью взаимозависимости. Или другой пример, когда Task Service меняет статус на «*Выполнено*», он публикует событие *TaskCompleted*. Notification Service и Analytics Service, подписанные на это событие, получают его и выполняют свою логику независимо друг от друга. Таким образом снижается связанность и повышается отказоустойчивость.

Стратегия управления данными. Каждый микросервис владеет собственной базой данных, что обеспечивает максимальную автономию. Сервисы пользователей, проектов и задач используют реляционную базу данных PostgreSQL, где важна целостность данных и поддержка транзакций. Аналитический сервис использует колоночную базу данных ClickHouse, оптимизированную для быстрых агрегаций и чтения больших объемов данных [3, с. 13]. Сервис уведомлений может обходиться без постоянного хранилища, используя очередь сообщений. Такой подход, известный как Polyglot Persistence, позволяет выбрать оптимальный инструмент для каждой конкретной задачи.

Предложенная архитектура, с ее декомпозицией на независимые сервисы, гибкими паттернами взаимодействия и стратегией управления данными, представляет собой целостное решение, ориентированное на достижение высокой масштабируемости и отказоустойчивости. Рассмотрев ключевые аспекты проектирования, можно перейти к обобщению полученных результатов и формулированию итоговых выводов о применимости и эффективности микросервисного подхода для систем управления проектами.

В рамках настоящей работы была решена задача по декомпозиции и теоретическому обоснованию архитектуры системы управления проектами, построенной на микросервисном подходе. Поставленная цель – планирование масштабируемой и отказоустойчивой системы – была достигнута через последовательное решение ряда взаимосвязанных задач.

Сравнительное исследование монолитной и микросервисной архитектур выявило превосходство второй по показателям гибкости и эффективности при решении задач, соответствующих современным требованиям к информационным системам. Предложена и подробно раскрыта схема разбиения системы на набор основных микросервисов, обеспечивающих их изоляцию и независимое масштабирование. Реализацию архитектуры можно выполнить с использованием продуманного технологического стека – Docker для контейнеризации, Kubernetes для управления оркестровкой, а также RabbitMQ для организации асинхронного взаимодействия между компонентами, что соответствует актуальным стандартам индустрии.

На основе проведённого анализа и обоснованного проектирования можно утверждать, что предложенная архитектура демонстрирует высокую производительность и способна адаптивно реагировать на изменение нагрузки за счёт динамической масштабируемости. Это полностью соответствует первоначальным гипотезам исследования и подтверждает актуальность выбранного направления. Разработанная архитектура является не просто

теоретической моделью, а практически ориентированным инструментом, отвечающим на насущные потребности бизнеса в гибких и масштабируемых решениях для управления проектами.

Следует отметить, что переход на микросервисную архитектуру сопряжен с возрастанием сложности разработки и эксплуатации, требуя высокого уровня автоматизации и квалифицированных специалистов в области DevOps. Однако для сложных и критически важных систем, таких как ИСУП, эти компромиссы оправданы.

Перспективными направлениями для дальнейших исследований являются интеграция элементов искусственного интеллекта для предиктивной аналитики рисков проектов, а также изучение возможности применения бессерверных вычислений для отдельных сервисов с целью дальнейшей оптимизации затрат и инфраструктуры [1, с. 50].

В заключение, можно констатировать, что микросервисный подход является наиболее перспективной архитектурной парадигмой для создания современных информационных систем управления проектами, способных удовлетворить растущие требования рынка и обеспечить долгосрочную конкурентоспособность бизнеса.

Литература

1. Дворецкий А.Г. Оптимизация процессов управления it-проектами с использованием методов машинного обучения // Программные системы: теория и приложения. 2024. Т. 15, № 2. С. 45-58.

2. Кабарухин А.П. Выгоды перехода от монолитной к микросервисной архитектуре приложения // Информационные технологии и системы. 2022. №. 3. С. 25-32.

3. Степанов М.А., Сластников С.А., Климин Н.А. Проектирование архитектуры программного обеспечения для анализа цифрового следа в образовательных системах // Программные системы: теория и приложения. 2025. Т. 16, №. 4. С. 3-21.

4. Massimo R. Project Management Information Systems: a Systematic Review // Procedia Computer Science. 2025. Т. 235. С. 1259-1268.

5. Newman S. Building Microservices: Designing Fine-Grained Systems. O'Reilly Media, 2021.

6. Taresh N.S.S. et al. Information Systems Impact on Project Management Success // Buildings. 2025. Т. 15, №. 8. С. 1260.

Приходько М.И., Баиров А.Б., Кочкин И.И., Шайдо Ю.А.
Томский государственный университет систем управления и радиоэлектроники
г. Томск, Россия

КАК БАЗОВЫЕ МАТЕМАТИЧЕСКИЕ КОНЦЕПЦИИ ВВОДНОГО КУРСА МАТЕМАТИКИ ЛЕЖАТ В ОСНОВЕ ПРОГРАММИРОВАНИЯ И АЛГОРИТМОВ

Современная информатика и программирование исторически выросли из математической логики и вычислительной математики. Многие начинающие разработчики ошибочно полагают, что для написания кода достаточно знать синтаксис языка, однако понимание глубинных математических принципов позволяет создавать эффективные, оптимизированные и безопасные программные продукты. Цель данной работы – проследить связь между базовыми разделами вводного курса математики (теория множеств, векторная алгебра, теория чисел) и их непосредственным применением в алгоритмизации и разработке программного обеспечения.

Теория множеств является одним из краеугольных камней современной математики. Она изучает совокупности элементов и отношения между ними. Множество можно определить как неструктурированную коллекцию уникальных объектов. Основные операции – объединение ($A \cup B$), пересечение ($A \cap B$) и разность ($A \setminus B$) – обладают свойствами коммутативности и ассоциативности, что позволяет строить на их основе сложные логические конструкции.

В программировании эти абстрактные идеи находят прямое воплощение в структурах данных. Тип данных set (множество), доступный в большинстве современных языков (Python, Java, C++), реализует именно математическую модель множества: он хранит только уникальные элементы и поддерживает операции объединения, пересечения и разности с гарантированной временной сложностью $O(1)$в среднем для поиска благодаря хешированию.

Приведём пример программной реализации операций над множествами на языке Python, который наглядно демонстрирует идентичность математической нотации и синтаксиса языка:

```
def set_operations_demo():
    # Исходные списки (преобразованные в множества)
    A = set([1, 2, 3, 4, 5])
    B = set([4, 5, 6, 7, 8])

    print(f"Множество A = {A}")
    print(f"Множество B = {B}")

    # 1. Объединение (Union)
    union_result = A | B  # Оператор | перегружен для union
    print(f"Объединение A ∪ B = {union_result}")

    # 2. Пересечение (Intersection)
    intersection_result = A & B
    print(f"Пересечение A ∩ B = {intersection_result}")
```

```
    # 3. Разность (Difference)
    difference_AB = A – B
    print(f"Разность A \\ B = {difference_AB}")

    # 4. Симметрическая разность (Symmetric Difference)
    sym_diff = A ^ B
    print(f'Симметрическая разность A △ B = {sym_diff}")
if __name__ == "__main__":
    set_operations_demo()
```

Как видно из примера, операции над множествами в Python (|, &, -, ^) являются прямыми аналогами математических символов. Это не просто синтаксическое совпадение, а глубокая реализация теоретико-множественной парадигмы, которая лежит в основе реляционных баз данных (SQL-операторы UNION, INTERSECT, EXCEPT), систем контроля версий и многих алгоритмов графовой обработки [2, с. 45].

Помимо явных структур данных типа set, теоретико-множественная парадигма лежит в основе системы типов в языках программирования. Например, понятие перечисления (enum) в языках C#, Java или Python (модуль Enum) является классическим примером множества с фиксированным набором допустимых значений. Операции проверки принадлежности элемента множеству (in в Python или contains в Java) напрямую соответствуют математическому отношению принадлежности ($\in$) [5].

Кроме того, понимание теории множеств критически важно при работе с системами управления базами данных (СУБД). Реляционная алгебра, лежащая в основе SQL, целиком построена на операциях над множествами. Рассмотрим пример SQL-запроса, который является прямым аналогом кода из нашего примера:

```
Таблица (отношение) A и B – это множества записей
CREATE TABLE A (id INT PRIMARY KEY);
INSERT INTO A VALUES (1), (2), (3), (4), (5);
CREATE TABLE B (id INT PRIMARY KEY);
INSERT INTO B VALUES (4), (5), (6), (7), (8);
1. Объединение (UNION) – аналог A | B
SELECT id FROM A
UNION
SELECT id FROM B;
Результат: 1,2,3,4,5,6,7,8
2. Пересечение (INTERSECT) – аналог A & B
SELECT id FROM A
INTERSECT
SELECT id FROM B;
Результат: 4,5
```

3. Разность (EXCEPT) – аналог A – B

```
SELECT id FROM A
EXCEPT
SELECT id FROM B;
```

Результат: 1,2,3

Вектор – это математический объект, характеризующийся величиной (длиной) и направлением. В отличие от скаляра, вектор требует для описания нескольких координат. В двумерном пространстве это пара (x, y), в трёхмерном – тройка (x, y, z). Векторная алгебра включает операции сложения, вычитания, умножения на скаляр и скалярного произведения.

В разработке компьютерных игр и систем моделирования векторы являются основным инструментом описания физического мира. Любой движущийся объект описывается тремя ключевыми векторами:

1. **Положение** ($\vec{r}$) – определяет текущие координаты объекта в пространстве.
2. **Скорость** ($\vec{v}$) – определяет быстроту и направление изменения положения.
3. **Ускорение** ($\vec{a}$) – определяет быстроту изменения скорости.

Второй закон Ньютона в векторной форме $\vec{F} = m\vec{a}$и уравнения кинематики напрямую переносятся в код. Например, модель движения тела под действием постоянного ускорения (силы тяжести) описывается системой уравнений:

$$\vec{v}(t) = \vec{v}_0 + \vec{a} \cdot t$$

$$\vec{r}(t) = \vec{r}_0 + \vec{v}_0 \cdot t + \frac{\vec{a} \cdot t^2}{2},$$

где $\vec{a} = (0, -g, 0)$– вектор ускорения свободного падения, направленный вертикально вниз.

В игровых движках (Unity, Unreal Engine) эти вычисления производятся на каждом кадре (обычно 60 раз в секунду) для тысяч объектов одновременно. Понимание векторной природы этих величин позволяет программисту корректно реализовывать инерцию, столкновения и реакции тел. Без векторной математики невозможно создание реалистичной анимации, расчёта траекторий полёта снарядов или симуляции движения автомобиля [3, с. 50].

Алгоритм Евклида (https://clck.ru/3SHnh4), описанный ещё в III веке до н. э. в труде «Начала», является одним из старейших и наиболее элегантных алгоритмов [1]. Он предназначен для нахождения наибольшего общего делителя (НОД) двух целых чисел. Алгоритм основан на простом наблюдении: $\text{НОД}(a, b) = \text{НОД}(b, a \bmod b)$. Последовательное применение этого правила позволяет быстро найти НОД без необходимости факторизации чисел. Псевдокод алгоритма выглядит следующим образом:

```
функция НОД(a, b)
  пока b ≠ 0
    остаток = a mod b
    a = b
    b = остаток
  вернуть a
```

Несмотря на свою простоту, алгоритм Евклида играет критическую роль в современной криптографии. Рассмотрим два основных применения.

1. Генерация ключей в системе RSA – для создания асимметричного ключа выбираются два больших простых числа pи q, вычисляется их произведение $n = p \cdot q$и функция Эйлера $\phi(n) = (p-1)(q-1)$. Затем выбирается открытая экспонента e (обычно 65537), которая должна удовлетворять условию $\text{НОД}(e, \phi(n)) = 1$. Проверка этого условия выполняется именно с помощью алгоритма Евклида. Если НОД не равен единице, выбранное eне подходит, так как не будет существовать обратного элемента – закрытого ключа [4].

2. Протокол Диффи-Хеллмана – при установке общего секретного ключа по открытому каналу стороны выбирают большое простое число pи генератор g. Для обеспечения безопасности необходимо, чтобы gбыл первообразным корнем по модулю p, что также требует проверки взаимной простоты на определённых этапах. Алгоритм Евклида используется здесь для проверки условий и для вычисления обратных элементов в расширенной версии (https://clck.ru/3SHoR7).

Таким образом, алгоритм Евклида, изучаемый в рамках школьной и вводной вузовской математики, является фундаментом, на котором строятся протоколы безопасности интернета – HTTPS, SSH и другие.

Проведённый анализ показывает, что базовые математические концепции не являются абстрактными и оторванными от реальности дисциплинами. Теория множеств напрямую транслируется в структуры данных и запросы к базам данных; векторная алгебра составляет математическую основу любой физической симуляции и компьютерной графики; а древний алгоритм Евклида оказался востребованным в самых современных криптографических системах. Осознание этих связей позволяет будущим программистам не просто заучивать синтаксис, а понимать глубинную природу происходящих в компьютере процессов, что является залогом создания качественного и надёжного программного обеспечения.

Литература

1. Кормен Т.Х., Лейзерсон Ч.И., Ривест Р.Л., Штайн К. Алгоритмы: построение и анализ. 3-е изд. М.: Вильямс, 2023. 1328 с.

2. Семенова И.В. Теория множеств и её применение в информатике // Электронные учебные материалы СГАУ. 2023. С. 40-55.

3. Соколов А.А. Физические симуляторы в компьютерной графике // UNIGINE Open Air. 2024. С. 45-62.

4. Шнайер Б. Прикладная криптография. Протоколы, алгоритмы и исходные тексты на языке Си. М.: Триумф, 2022. 816 с.

5. Чернышев С.А. Алгоритмы и структуры данных на языке GO: учебник. М.: КноРус, 2024. 353 с.

Пшеничников А.А.
Нижневартовский государственный университет
г. Нижневартовск, Россия

АРХИТЕКТУРА УНИВЕРСАЛЬНОГО МЕХАНИЗМА ОТЧЁТОВ ДЛЯ СИСТЕМЫ ОТОБРАЖЕНИЯ ПРОГРАММНЫХ ОБЪЕКТОВ В РЕЛЯЦИОННОЙ СУБД

Современные информационные системы, построенные на объектно-ориентированном подходе, сталкиваются с проблемой формирования отчетов. Программные объекты (транзакции, пользователи, категории) имеют сложную структуру и взаимосвязи, тогда как отчеты требуют плоского, табличного представления данных. Разработка отдельных механизмов отчетности под каждую сущность приводит к дублированию кода и усложнению поддержки [2, с. 47].

Проблема заключается в отсутствии универсального подхода к трансформации иерархической объектной модели в реляционное представление отчета. Существующие решения (ORM, конструкторы отчетов) либо жестко привязаны к схеме базы данных, либо требуют написания сложных запросов для каждой формы отчетности [1, с. 156; 3, с. 118].

Целью работы является разработка архитектуры универсального механизма отчетов, позволяющего описывать структуру отчета на уровне метаданных и автоматически формировать данные из программных объектов, хранящихся в реляционной СУБД. Для достижения цели решались такие задачи, как анализ подходов к генерации отчетов, проектирование модульной архитектуры механизма, разработка системы мета-описаний, реализация прототипа на примере финансового приложения.

Для выбора оптимального подхода проведен сравнительный анализ существующих механизмов отчетности по критериям: гибкость настройки, производительность, сложность интеграции и универсальность [4, p. 12].

Таблица

Сравнительный анализ подходов к генерации отчетов

Подход	Гибкость настройки	Производительность (отн. ед.)	Сложность реализации	Универсальность
Жесткие SQL-запросы	Ограниченная	0.95	Низкая	Низкая
Шаблонизаторы (JasperReports)	Умеренная	0.65	Высокая	Средняя
ORM + построители запросов	Умеренная	0.70	Средняя	Средняя
Мета-описания + Visitor	Высокая	0.60	Высокая	Высокая

Производительность измерялась относительно эталонного значения (оптимизированный SQL-запрос принят за 1.0) на тестовом наборе данных. Как видно из таблицы, подход на основе мета-описаний и паттерна Visitor демонстрирует наилучшие показатели универсальности и гибкости при незначительном снижении производительности, что согласуется с выводами исследования [4, p. 18].

На основе проведенного анализа выбран следующий стек технологий:

- SQLite – реляционная СУБД;
- Knex.js – построитель запросов;
- Vue.js 3 – для визуализации отчетов (интерфейс);
- Meta-описания в формате JSON – для конфигурации отчетов [1, с. 189].

Архитектура приложения построена по модульному принципу и включает три основных слоя (рис. 1) [2, с. 49; 3, с. 120]:

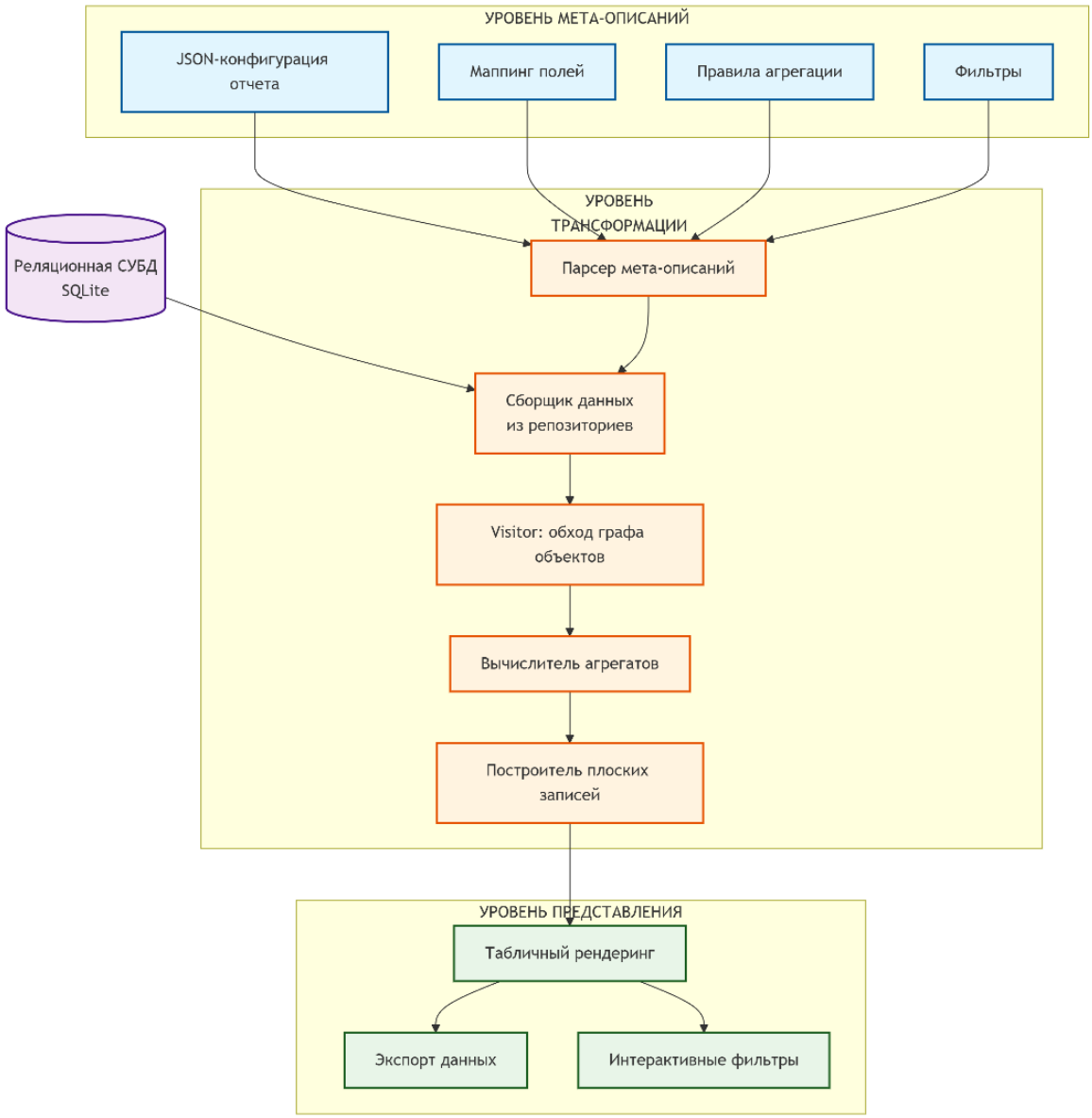


Рис. 1. Архитектура универсального механизма отчетов по модульному принципу

Уровень мета-описаний определяет структуру отчетов через конфигурационные файлы, содержащие [1, с. 192]:

- маппинг полей объекта на колонки отчета;
- правила агрегации и группировки;
- фильтры и условия отбора;
- форматы представления данных (дата, валюта, процент).

Такой подход позволяет отделить логику формирования отчета от кода приложения, что соответствует принципу разделения ответственности [1, с. 194; 2, с. 50].

Уровень трансформации данных реализует паттерн Visitor для обхода графа объектов [2, с. 51; 3, с. 121] и включает:

- парсер мета-описаний;
- сборщик данных из репозиториев, реализованный с использованием Knex.js [7];
- вычислитель агрегатных функций (суммы, средние, остатки);
- построитель плоских записей отчета.

Применение паттерна Visitor обеспечивает гибкость обхода сложных объектных структур без изменения классов самих объектов [3, с. 122; 4, p. 22]. Использование подхода на основе репозиториев позволяет абстрагироваться от конкретной СУБД и формировать сложные запросы [5, p. 456].

Уровень представления обеспечивает визуализацию полученных данных:

- табличный рендеринг с поддержкой группировок;
- экспорт в CSV, PDF, JSON;
- интерактивная фильтрация и сортировка.

Ключевые модули механизма:

1. **Модуль мета-описаний –** позволяет декларативно описать любой отчет на основе подхода, предложенного в [1, с. 195; 3, с. 119]:

```
{
  "name": "Расходы по категориям",
  "source": "Transaction",
  "fields": [
    { "source": "category.name", "label": "Категория" },
    { "source": "amount", "label": "Сумма", "aggregate": "sum" },
    { "source": "date", "label": "Период", "format": "MM.yyyy" }
  ],
  "filters": [
    { "field": "type", "operator": "=", "value": "expense" }
  ],
  "groupBy": ["category.name", "date"]
}
```

Рис. 2. Модуль мета-описаний

2. **Модуль обхода объектов (Visitor)** – рекурсивно обходит граф связанных объектов (транзакция → категория → теги) и собирает данные согласно мета-описанию [3, с. 122; 4, p. 25].

3. **Модуль построения отчетов** – формирует результирующий набор записей, применяя группировки и агрегатные функции.

Разработан прототип механизма отчетов в составе кроссплатформенного финансового приложения. Функционал:

- формирование отчета «Расходы по категориям за месяц»;
- формирование отчета «Динамика доходов/расходов»;
- формирование отчета «Остатки по бюджетам»;
- экспорт в CSV и PDF [4].

Тестирование проводилось на конфигурации:

- CPU: Intel Core i5-1035G1;
- RAM: 8 ГБ;
- ОС: Windows 11;
- Объем данных: 10 000 транзакций, 50 категорий.

Преимущества предложенной архитектуры:

- **Универсальность** – один механизм для любых объектов и отчетов [2, с. 52];
- **Расширяемость** – новые отчеты добавляются через JSON без программирования [1, с. 198];
- **Слабая связанность** – изменения в объектной модели требуют только корректировки мета-описаний [3, с. 123];
- **Производительность** – оптимизированные запросы с предварительной выборкой связанных данных (eager loading) [5, p. 512].

Ограничения:

- сложность первоначальной настройки мета-модели [1, с. 200];
- необходимость оптимизации при глубоких графах объектов (проблема N+1 запроса решается через eager loading) [5, p. 514].

В работе предложена и реализована архитектура универсального механизма отчетов для систем, работающих с программными объектами в реляционной СУБД. Доказана эффективность использования подхода на основе мета-описаний и паттерна Visitor [3, с. 124; 4, p. 28]. Основные преимущества решения – гибкость, универсальность и простота добавления новых форм отчетности. Перспективы работы включают оптимизацию производительности, реализацию визуального конструктора и расширение форматов экспорта [1, с. 202; 2, с. 53].

Литератураа

1. Мамедли Р.Э. Системы управления базами данных. Нижневартовск: Изд-во Нижневарт. гос. ун-та, 2021. 214 с.

2. Петров А.С. Современные веб-технологии в разработке desktop-приложений // Программирование. 2022. № 4. С. 45-52.

3. Селиверстова А.В., Виноградова М.В., Лосева С.С., Гапанюк Ю.Е. Интеграционный подход к документированию на основе метаграфов // Computational Nanotechnology. 2024. Т. 11. № 5. С. 116-123.

4. Bree D.C., Ó Cinnéide M. Energy efficiency of the Visitor Pattern: contrasting Java and C++ implementations // Empirical Software Engineering. 2023. Vol. 28. № 6.

5. Halpin T.A., Morgan A.J. Information Modeling and Relational Databases. 3rd ed. Morgan Kaufmann, 2024. 1032 p.

Розендаль Т.А., Розендаль К.А, Карфидов И.М.
Уральский государственный экономический университет
г. Екатеринбург, Россия

АНАЛИЗ ФУНКЦИОНАЛЬНЫХ ВОЗМОЖНОСТЕЙ NO-CODE И LOW-CODE ПЛАТФОРМ В КОНТЕКСТЕ КОРПОРАТИВНЫХ ИНФОРМАЦИОННЫХ СИСТЕМ

В условиях цифровой трансформации организаций существенно возрастает потребность в оперативной разработке и адаптации информационных систем, поддерживающих ключевые бизнес-процессы. Согласно исследованиям в области цифровой экономики и управления ИТ-проектами, традиционные подходы к разработке корпоративных информационных систем характеризуются высокой трудоёмкостью, длительными сроками реализации и значительными затратами на сопровождение, что ограничивает гибкость организаций и замедляет внедрение цифровых изменений.

В последние годы в научных и прикладных публикациях всё большее внимание уделяется платформам визуальной разработки: no-code и low-code. Данные платформы выступают как альтернатива традиционным подходам к разработке программного обеспечения, ориентированным на ручное кодирование и требующим высокой квалификации ИТ-специалистов. Исследования показывают, что использование визуальных средств моделирования и конфигурирования позволяет существенно сократить сроки разработки, снизить порог входа для бизнес-пользователей и повысить вовлечённость предметных специалистов в процессы цифровизации. Вместе с тем в литературе отмечается недостаточная проработанность вопросов применимости подобных платформ в корпоративном контексте, особенно с точки зрения архитектуры, безопасности, управления данными и масштабируемости, что обуславливает необходимость дальнейшего анализа конкретных платформ и практик их использования.

Научная новизна работы заключается в систематизированном анализе функциональных возможностей no-code и low-code платформ с акцентом на её применение в условиях повышенных требований к безопасности, управлению данными и автоматизации бизнес-процессов, что позволяет расширить представления о возможностях данного подхода к разработке в корпоративных информационных системах.

Целью настоящей работы является анализ функциональных возможностей платформам визуальной разработки.

Для достижения поставленной цели в работе решаются следующие задачи:

– обосновать актуальность применения платформам визуальной разработки в корпоративных информационных системах на основе анализа научных и прикладных источников;

– проанализировать функциональные особенности существующих no-code и low-code платформ.

Объектом исследования является корпоративные low-code и no-code платформы, предназначенная для разработки бизнес-приложений и информационных систем.

Предметом исследования являются функциональные возможности no-code и low-code платформ, обеспечивающие разработку, интеграцию и эксплуатацию корпоративных информационных систем, включая средства визуального моделирования, управления данными, обеспечения безопасности и автоматизации бизнес-процессов.

В рамках настоящего исследования использовались методы анализа и обобщения научных и прикладных источников, включая статьи в рецензируемых журналах и публикации, посвящённые вопросам цифровой трансформации. Систематизация результатов данных работ позволила выявить ключевые проблемы традиционной разработки информационных систем и определить место данных подходов в современных исследованиях.

В ряде публикаций подчёркивается проблема разрыва между ИТ-специалистами и бизнес-пользователями, выступающая одним из ключевых организационных барьеров цифровой трансформации. Согласно исследованию авторов Prinz et al. традиционные процессы разработки программного обеспечения затрудняют участие представителей бизнеса в формировании и реализации требований, что снижает скорость цифровых изменений и эффективность внедряемых решений. Аналогичные выводы приводятся в работе Rokis и Kirikova, где отмечается, что высокая зависимость от дефицитных ИТ-кадров ограничивает адаптивность организаций и замедляет процессы цифровой трансформации [7; 8].

В более широком контексте исследований цифровых инноваций подчёркивается, что успешная цифровая трансформация требует соблюдения баланса между технологическими и организационными изменениями. Так, Н.В. Долбня и Д.С. Карпачев анализируя работы других авторов пришли к выводу о том, что чрезмерная концентрация на технологической составляющей при недостаточном учёте организационных аспектов и человеческого фактора приводит к снижению эффективности и неполной реализации потенциала внедряемых инноваций. Успешное внедрение цифровых решений, по мнению авторов, предполагает комплексный подход, включающий адаптацию организационной структуры, развитие цифровых компетенций персонала и формирование культуры, поддерживающей изменения [1].

В этом контексте no-code и low-code платформы в современной научной литературе рассматриваются как один из инструментов преодоления как технологических, так и организационных ограничений цифровой трансформации. Ряд авторов (Prinz et al.; Chandran и Abdulla) отмечают, что использование визуального моделирования, предварительно настроенных компонентов и декларативных средств разработки позволяет существенно сократить сроки создания программных решений, снизить издержки и расширить круг участников разработки за счёт вовлечения сотрудников, обладающих знаниями предметной области и бизнес-процессов, но не имеющих профессиональных навыков программирования [6; 7].

Исследование Bernsteiner et al. подчёркивает, что данный подход способствует демократизации информационных технологий и повышению организационной гибкости за счёт более тесного взаимодействия между бизнес-подразделениями и ИТ-службами, а также перераспределения ответственности за разработку и сопровождение цифровых решений [5].

Таким образом, no-code и low-code платформы могут рассматриваться не только как технологическое средство ускорения разработки, но и как инструмент интеграции технологических и организационных изменений, что соответствует требованиям комплексной цифровой трансформации.

Несмотря на значительное число преимуществ, в научных исследованиях и прикладных кейсах подчёркивается наличие ряда барьеров и ограничений, связанных с внедрением платформ визуальной разработки в корпоративной среде. Отмечается, что в отличие от традиционного программирования, для которого накоплен обширный эмпирический опыт и сформированы устойчивые профессиональные сообщества, low-code и no-code платформы пока не обладают сопоставимым уровнем методологической и инструментальной проработки. Хотя концепции разработки, управляемой моделями, существуют давно, их широкое применение в универсальных корпоративных платформах остаётся предметом активных исследований [3; 4].

Одним из ключевых барьеров внедрения платформ визуальной разработки является необходимость изменения организационной культуры. Использование данных решений предполагает перераспределение ролей между ИТ-подразделениями и бизнес-пользователями, а также признание возможности участия так называемых «гражданских разработчиков» в создании и модификации приложений. В ряде работ подчёркивается, что подобные изменения требуют стратегического видения и поддержки со стороны высшего руководства, без чего внедрение платформ может носить фрагментарный и ограниченный характер [2].

Существенным фактором также является сложность освоения самих платформ. Несмотря на декларируемую простоту, low-code и no-code решения не являются тривиальными инструментами и требуют времени для формирования устойчивых навыков проектирования данных, логики и пользовательских интерфейсов. Особенно это проявляется при реализации сложных сценариев автоматизации, включающих вложенные условия, циклы и обработку исключений, которые остаются нетривиальными вне зависимости от выбранной технологической парадигмы [2-4].

Отдельно в литературе отмечается ограниченность ресурсов и поддержки со стороны профессиональных сообществ. В отличие от традиционных языков программирования, таких как Java или C#, для которых доступны многочисленные курсы, учебные материалы и экспертные сообщества, многие low-code и no-code платформы обладают существенно меньшей базой знаний, особенно в случае относительно новых или нишевых решений. Это может осложнять процесс внедрения и сопровождения систем, а также повышать зависимость организаций от конкретных вендоров [2].

Практические кейсы внедрения платформ визуальной разработки демонстрируют, что данные платформы действительно ускоряют цифровизацию бизнес-процессов за счёт упрощения внесения изменений и обеспечения кроссплатформенной доступности приложений. Однако в ряде случаев выявляются сложности интеграции low-code и no-code решений с индивидуальным программным обеспечением, разработанным с использованием

традиционных технологий. Различия в используемых языках программирования, протоколах и архитектурных подходах могут требовать дополнительных усилий по согласованию и адаптации систем [2; 3].

Вместе с тем анализ публикаций показывает, что большинство работ носит обзорный характер и не фокусируется на сравнении отдельных решений. Как отмечают Prinz et al. и Bernsteiner et al., вопросы масштабируемости, механизмов обеспечения безопасности и управления данными в корпоративных no-code платформах остаются недостаточно изученными, что создаёт методологический и практико-ориентированный барьер для обоснованного выбора и внедрения данных платформ в корпоративных информационных системах, а также ограничивает возможность их объективной оценки с точки зрения соответствия требованиям промышленной эксплуатации. Дополнительно в работах Arica и Oliveira подчёркивается, что при внедрении цифровых платформ одной из ключевых угроз является обеспечение безопасности персональных и конфиденциальных данных, что особенно актуально для крупных организаций [6; 7; 9].

Таким образом, no-code и low-code платформы целесообразно рассматривать не как универсальную замену традиционной разработки, а как элемент гибридной ИТ-стратегии, дополняющий существующие подходы. Выявленные в литературе пробелы в изучении архитектурных, функциональных и эксплуатационных характеристик корпоративных no-code платформ обосновывают актуальность дальнейших исследований, направленных на анализ конкретных решений и оценку их применимости в условиях корпоративных информационных систем.

Современные no-code и low-code платформы развиваются в направлении унификации архитектурных подходов, ориентированных на ускорение разработки корпоративных информационных систем при сохранении требований к безопасности, масштабируемости и интеграции. Платформы CORTA (https://protolab.systems), Mendix (https://docs.mendix.com) и OutSystems (https://www.outsystems.com), несмотря на различия в степени абстракции и целевой аудитории, демонстрируют схожую концептуальную модель, основанную на визуальном проектировании, модульности и API-ориентированной архитектуре.

Для всех подобных платформ характерно наличие встроенной среды разработки, объединяющей моделирование данных, пользовательских интерфейсов и прикладной логики. В Mendix и OutSystems ключевым элементом является визуальное описание логики приложения с использованием графов исполнения, что позволяет формализовать бизнес-логику без прямого написания кода. Аналогичный подход реализован и в CORTA, где автоматизация процессов и логика поведения системы задаются через визуальные сценарии и правила, дополняемые возможностью обращения к API и скриптам при необходимости реализации более сложных алгоритмов.

С точки зрения архитектуры управления доступом, CORTA, Mendix и OutSystems используют ролевую модель разграничения прав (RBAC), обеспечивающую контроль доступа к данным, страницам и функциям приложения. При этом CORTA выделяется расширенной поддержкой контекстных ролей и явных разрешений и запретов, что позволяет учитывать не

только принадлежность пользователя к роли, но и условия выполнения операций. В Mendix и OutSystems разграничение доступа также тесно связано с моделью данных и логикой приложения, однако в большей степени ориентировано на стандартные сценарии корпоративных систем.

Общей функциональной чертой рассматриваемых платформ является развитый визуальный конструктор пользовательских интерфейсов. В Mendix, OutSystems и CORTA интерфейс формируется на основе готовых компонентов: формы, таблицы, отчёты, диаграммы. При этом каждая из платформ поддерживает адаптацию интерфейса под различные роли пользователей и типы устройств, что снижает затраты на разработку многоканальных решений. Наличие маркетплейсов готовых модулей и компонентов в Mendix и OutSystems расширяет функциональность платформ за счёт повторного использования типовых решений; схожая концепция модульности и повторного использования реализована и в CORTA на уровне прикладных компонентов и процессов.

Таким образом, сравнительный анализ CORTA, Mendix, OutSystems показывает, что современные no-code и low-code платформы развиваются в сторону универсальных корпоративных инструментов, объединяющих визуальное моделирование данных, интерфейсов и логики, а также развитые механизмы интеграции и безопасности. При этом различия между платформами проявляются в степени ориентации на процессное управление и целевых сценариях применения: от быстрого создания внутренних бизнес-приложений до построения масштабируемых корпоративных информационных систем.

В результате проведённого анализа научных публикаций и сравнительного исследования современных платформ визуальной разработки установлено, что данные решения эволюционировали от инструментов быстрой автоматизации отдельных бизнес-задач к полноценным корпоративным платформам, ориентированным на использование в критической информационной инфраструктуре. Существенным результатом исследования является выявление тенденции к сближению архитектурных и функциональных принципов ведущих платформ при сохранении различий в целевых сценариях применения.

Ключевым наблюдением, полученным в ходе работы, является значительный прогресс современных no-code и low-code платформ в области обеспечения информационной безопасности. Если ранее вопросы защиты данных, разграничения доступа и соответствия корпоративным требованиям рассматривались как одно из основных ограничений визуальной разработки, то современные платформы в значительной степени нивелировали данные риски. Анализ показал, что такие решения, как CORTA, Mendix и OutSystems, реализуют комплексные механизмы безопасности, включающие ролевые модели управления доступом, централизованную аутентификацию и авторизацию, поддержку интеграции с корпоративными каталогами пользователей, а также контроль доступа на уровне данных, интерфейсов и бизнес-логики. Встроенные средства журналирования, контроля действий пользователей и изоляции приложений в данных решёных позволяют рассматривать их не только как инструменты ускорения разработки, но и как зрелые платформы, способные обеспечить управляемость и защищённость жизненного цикла приложений.

Результаты сравнительного анализа также показали, что унификация архитектурных подходов – визуальное моделирование, модульность, API-ориентированность и интеграция с внешними системами – способствует снижению порога входа и ускорению цифровизации бизнес-процессов без критической потери управляемости ИТ-ландшафта. При этом выявлено, что повышение уровня безопасности и встроенных механизмов контроля сопровождается ростом сложности самих платформ, что требует развития цифровых компетенций пользователей и сохранения активной роли ИТ-подразделений.

Таким образом, полученные результаты свидетельствуют о том, что современные no-code и low-code платформы достигли уровня зрелости, позволяющего существенно снизить ранее характерные для них ограничения, связанные с обеспечением информационной безопасности и интеграцией с существующими корпоративными информационными системами. Это позволяет рассматривать данные платформы как обоснованный и практически применимый элемент корпоративной ИТ-архитектуры. В то же время их эффективное использование в корпоративной среде требует системного организационного подхода, а также согласованного сочетания с традиционными методами разработки и управления информационными системами.

В ходе выполнения данной работы был проведён анализ научных и прикладных источников, а также сравнительное исследование архитектурных и функциональных особенностей современных платформ визуальной разработки. Полученные результаты подтверждают, что данные решения прошли эволюционный путь от инструментов локальной автоматизации отдельных бизнес-задач к зрелым корпоративным платформам, ориентированным на использование в сложных информационных средах.

Сравнительный анализ показал наличие устойчивой тенденции к сближению архитектурных и функциональных принципов ведущих платформ визуальной разработки при сохранении различий в целевых сценариях их применения. Использование визуального моделирования, модульных компонентов, API-ориентированных архитектур и механизмов интеграции с внешними системами способствует ускорению цифровизации бизнес-процессов и снижению порога входа в разработку без существенного снижения управляемости корпоративного ИТ-ландшафта.

Существенным выводом исследования является установление значительного прогресса современных no-code и low-code платформ в области обеспечения информационной безопасности. Реализация ролевых моделей управления доступом, централизованных механизмов аутентификации и авторизации, интеграции с корпоративными каталогами пользователей, а также контроля доступа на уровне данных, пользовательских интерфейсов и бизнес-логики позволяет рассматривать такие платформы не только как средства ускорения разработки, но и как инструменты, обеспечивающие управляемость и защищённость жизненного цикла корпоративных приложений. Наличие встроенных средств журналирования и контроля действий пользователей расширяет возможности их применения в корпоративной и потенциально критической информационной инфраструктуре.

В то же время результаты исследования показывают, что рост уровня безопасности и встроенных механизмов контроля сопровождается увеличением сложности самих платформ, что требует развития цифровых компетенций пользователей и сохранения активной роли ИТ-подразделений в процессах проектирования, внедрения и эксплуатации решений. В данном контексте no-code и low-code платформы не могут рассматриваться как универсальная замена традиционной программной разработки, а их наибольшая эффективность достигается в рамках гибридных ИТ-стратегий, предполагающих согласованное сочетание визуальной разработки и классических инженерных подходов.

Таким образом, современные no-code и low-code платформы представляют собой обоснованный и практически применимый элемент корпоративной ИТ-архитектуры, использование которого требует системного организационного подхода и адаптации управленческих практик. Дальнейшие исследования в данной области представляются актуальными в части углублённого анализа архитектурных решений конкретных платформ, оценки их долгосрочной масштабируемости и выработки методических подходов к внедрению no-code и low-code технологий в корпоративной среде с учётом требований безопасности и управляемости.

Литература

1. Долбня Н.В., Карпачев Д.С. Цифровая трансформация предприятий: проблемы и перспективы внедрения электронной отчетности // Цифровые модели и решения. 2025. Т. 4, № 2. С. 44-58. EDN NGRZBE. https://doi.org/10.29141/2949-477X-2025-4-2-4

2. Магомадов В.С. Платформы low-code и no-code как способ сделать программирование более доступным для широкой общественности // Международный научно-исследовательский журнал. 2021. № 6-1 (108). С. 100-103.

3. Онокой Л.С., Лаптев К.А. Цифровые платформы с применением решений No Code / Low Code как инструмент повышения эффективности бизнес-процессов // Инновации и инвестиции. 2023. № 10. С. 307-310.

4. Радзиевская А.А. Принципы разработки многофункциональных веб-приложений, используя no-code подход // Инновации и инвестиции. 2023. № 1. С. 158-161.

5. Bernsteiner R. et al. Citizen vs. professional developers: differences and similarities of skills and training requirements for low-code development platforms // ICERI2022 Proceedings. Valencia: IATED, 2022. P. 4257-4264.

6. Chandran L.C., Abdulla M.S. A survey of Low-Code / No-Code software development tools with an application. Working Paper. IIMK/WPS/524/ITS/2022/08. 2022.

7. Prinz N., Rentrop C., Huber M. Low-Code Development Platforms: A Literature Review // Proceedings of the AMCIS. 2021.

8. Rokis K., Kirikova M. Challenges of low-code / no-code software development: A literature review // Business Informatics Research: Proceedings of the International Conference. Cham: Springer, 2022. P. 3-17.

Савченко С.А., Ерохин В.В.
Иркутский государственный университет
г. Иркутск, Россия

РАЗРАБОТКА ТЕСТОВОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ ИССЛЕДОВАНИЯ ПОБОЧНЫХ ЭЛЕКТРОМАГНИТНЫХ ИЗЛУЧЕНИЙ И НАВОДОК ПРИ ПОДКЛЮЧЕНИИ ПЕРИФЕРИЙНЫХ УСТРОЙСТВ К ПЕРСОНАЛЬНОМУ КОМПЬЮТЕРУ

С развитием компьютерных технологий и ростом скоростей передачи данных увеличиваются риски утечки информации через ПЭМИН. Интерфейс PCI-Express, являясь высокоскоростной шиной для подключения периферийных устройств, представляет собой потенциальный канал утечки информации. Актуальность исследования ПЭМИН, генерируемых при работе PCI-Express, обусловлена необходимостью оценки защищённости информационных систем и разработки мер по противодействию перехвату данных.

Целью данной работы является разработка тестового программного обеспечения для генерации нагрузки на интерфейс PCI-Express и организация экспериментальных исследований ПЭМИН с использованием специализированной измерительной аппаратуры.

Побочные электромагнитные излучения

ПЭМИ называют электромагнитные поля, возникающие в окружающем пространстве в результате работы технических устройств и не предназначенные для целенаправленной передачи сигнала. В отличие от целевых передатчиков (радиостанции, мобильные телефоны), ПЭМИ образуются, при функционировании мониторов, принтеров, элементов хранения данных и интерфейсов компьютерных систем. По частотному диапазону ПЭМИ могут охватывать диапазон от единиц кГц до единиц ГГц (включая гармонические составляющие спектра излучения), причём рост скоростей цифровой передачи в современных системах сдвигает спектр излучений в сторону более высоких частот.

К основным источникам ПЭМИ в вычислительных системах относятся интерфейсы и шины передачи данных: видеосигналы, интерфейсы матриц дисплея, SATA, PCI-Express, USB, Ethernet, аудиоканалы, а также элементы плат (чипсет, процессор, контроллеры, шина ОЗУ) [3, с. 972]. Следует отметить, что высокоразрядные шины (широкие параллельные шины) генерируют потенциально более сложные для декодирования сигналы, а низкие по току линии дают малую амплитуду излучения; оба фактора влияют на практическую возможность перехвата информации.

С практической точки зрения ПЭМИ делят на информативные (несущие полезную для разведки информацию) и неинформативные (не содержащие значимой информации о передаваемых данных) сигналы [1, с. 82]. Оценка защищённости от утечек по ПЭМИ включает выявление потенциально информативных источников и количественную характеристику уровня поля в зависимости от расстояния – что и служит основанием для расчёта контрольных зон А [5, с. 270].

В прикладной методике выделяют две взаимодополняющие процедуры оценки: (1) специальные инструментально-расчётные исследования, дающие радиусы зон r1, r1′ и R2 (соответственно для зон, где возможна съёмка в сосредоточенных и распределённых антеннах,

и для границы контролируемой зоны); (2) измерение реального затухания сигнала до границы контролируемой зоны и оценка отношения сигнал/шум. Контролируемая зона – территория, где исключено бесконтрольное нахождение посторонних лиц и технических средств разведки; граница этой зоны служит практической чертой для расчётов и организационных мер защиты.

Пространство вокруг ОТСС, в пределах которого напряженность ЭМ-поля превышает допустимое (нормированное) значение, называется зоной 2 (R2). Фактически зона R2 – это зона, в пределах которой возможен перехват средством разведки ПЭМИН с требуемым качеством [2, с. 3]. Зона 2 для каждого ОТСС определяется инструментально-расчетным методом и, как правило, указывается в эксплуатационной документации.

Пространство вокруг ОТСС, в пределах которого уровень наведенного от ОТСС информативного сигнала в сосредоточенных антеннах превышает допустимое (нормированное) значение называется зоной 1 (r1), а в распределенных антеннах – зоной 1’ (r1’). В отличие от зоны R2, размер зоны r1 (r1’) зависит не только от уровня побочных электромагнитных излучений ТСПИ, но и от длины случайной антенны (от помещения, в котором установлено ТСПИ до места возможного подключения к ней средства разведки). Зоны r1 (r1') для каждого ОТСС определяется инструментально-расчетным методом при проведении специальных исследований технических средств на ПЭМИН и указывается в предписании на их эксплуатацию или сертификате соответствия.

Для определения реальных затуханий в канале необходимо ввести в него тест-сигнал большого уровня, позволяющий надежно его идентифицировать и измерить на границе КЗ.

Излучающая антенна должна быть установлена на месте ТС или очень близко от него. При этом излучающая антенна должна быть не направленной (хотя бы в горизонтальной плоскости), так как ее сигнал должен имитировать ПЭМИН. Излучающую антенну рекомендуется размещать на том же расстоянии от стены окна, что и исследуемый объект. Это связано с тем, что в современных зданиях используются железобетонные конструкции и основной путь ЭМ-волны к границе КЗ – это окно и переизлучение металлоконструкциями стен. Общая схема измерений ПЭМИН приведена на рисунке 1.

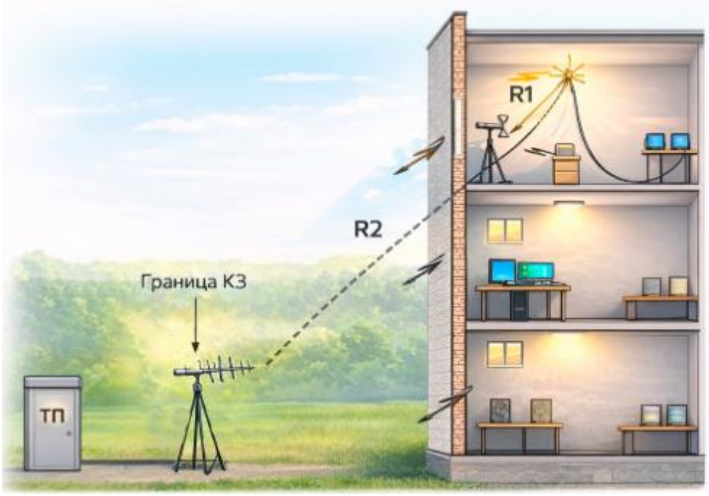


Рис. 1. Общая схема измерений ПЭМИН (составлено автором)

Оборудование, которое непосредственно используется для обработки конфиденциальной информации называется «основное техническое средство и система» (ОТСС).

Разработанное программное обеспечение

Для генерации контролируемой нагрузки на интерфейс PCI-Express было разработано тестовое приложение на языке программирования C++. Программа реализует инициализацию низкоуровневых библиотек WinIo и WinRing0 для получения прав доступа к портам и конфигурационному пространству PCIe и проверяет успешность установки драйверов перед началом выполнения. На этапе инициализации приложение сканирует шину, определяет целевое устройство по параметрам bus, device, function, считывает его конфигурационные регистры и формирует карту доступных регистров/адресов для тестирования [4, с. 45]. Основной режим работы – цикл записи псевдослучайных или программируемых тестовых блоков в выбранные регистры/буферы с последующей проверкой чтением и контролем целостности через простую контрольную сумму.

Программа поддерживает несколько типов нагрузки: непрерывный поток, серии коротких «пакетов» и скриптовые последовательности с паузами, что позволяет моделировать разные сценарии активности шины и подбирать оптимальные условия для снятия ПЭМИ. Для управления предусмотрены параметры командной строки и интерактивный режим: задаются размер блока записи, период между итерациями, число повторов, режим теста и уровень логирования.

Для воспроизводимости используется инициализация генератора псевдослучайных чисел по времени или по фиксированной последовательности, реализованы проверки корректности доступа и обработка ошибок. В программу встроены защитные меры: опциональный «сухой прогон» без фактической записи, проверка границ адресов и предупреждения при попытке записи в критические области памяти. Цель ПО – создать устойчивую, измеримую и легко воспроизводимую нагрузку на PCI-Express, необходимую для снятия ПЭМИ и корректного расчёта зон R2/r1. Фрагмент кода основной функции (main) представлен на рисунке 2.

```
int main(int argc, char* argv[]) {
    int bus, device, function;

    if (argc != 4) {
        std::cout << "Please enter the PCI configuration (bus device function):" << std::endl;

        std::cout << "Enter bus: ";
        std::cin >> bus;

        std::cout << "Enter device: ";
        std::cin >> device;

        std::cout << "Enter function: ";
        std::cin >> function;
    }
    else {
        std::istringstream(argv[1]) >> bus;
        std::istringstream(argv[2]) >> device;
        std::istringstream(argv[3]) >> function;
    }

    readPciConfigSpace(static_cast<BYTE>(bus), static_cast<BYTE>(device), static_cast<BYTE>(function));

    return 0;
}
```

Рис. 2. Фрагмент кода основной функции разработанного ПО (составлено автором)

Особенности сопряжения ПЭВМ и аппаратуры для исследования ПЭМИНя

Для проведения исследований ПЭМИН необходимо обеспечить корректное взаимодействие между тестовой ПЭВМ, на которой выполняется разработанное ПО, и измерительной аппаратурой. В качестве измерительного комплекса использовались:

- рупорная антенна П6-223 (диапазон 0,8–18 ГГц);
- анализатор спектра Anritsu MS2723C (диапазон 9 кГц – 13,3 ГГц).

Особенностью сопряжения является необходимость минимизации влияния внешних помех и обеспечение направленного приёма излучения от исследуемого интерфейса. Антенна размещалась на расстоянии 1 м от тестового компьютера и направлялась на системный блок, где установлен накопитель с интерфейсом PCI-Express (см. схему проведения измерений ПЭМИН, рисунок 3). Кабельное соединение антенны с анализатором спектра было экранировано для снижения наводок. Измерения проводились в двух смежных комнатах для исключения влияния оператора на результаты.

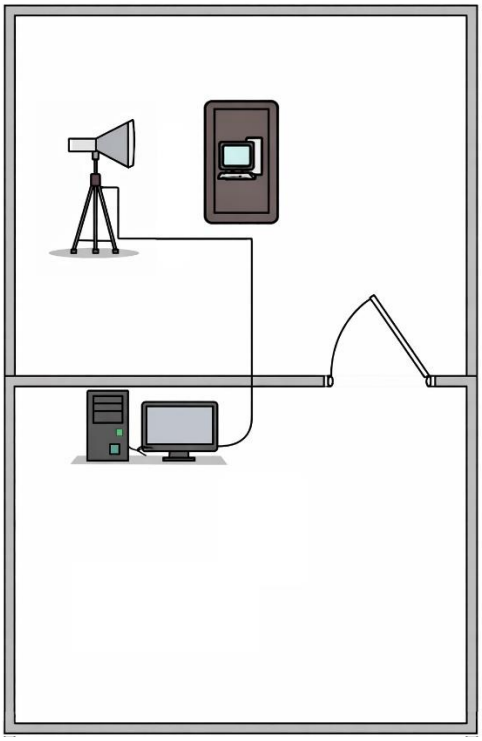


Рис. 3. Схема проведения измерений ПЭМИН (составлено автором)

Математический аппарат для оценки зоны R2

Для оценки уровня защищённости от утечки информации по каналу ПЭМИН был выполнен расчёт зоны R2. Использовалась методика, основанная на измерении уровней электрической составляющей электромагнитного поля.

Измеренные значения напряжённости поля:

- без нагрузки ($E_{ш}$): 29,1 dBμV (рис. 4);

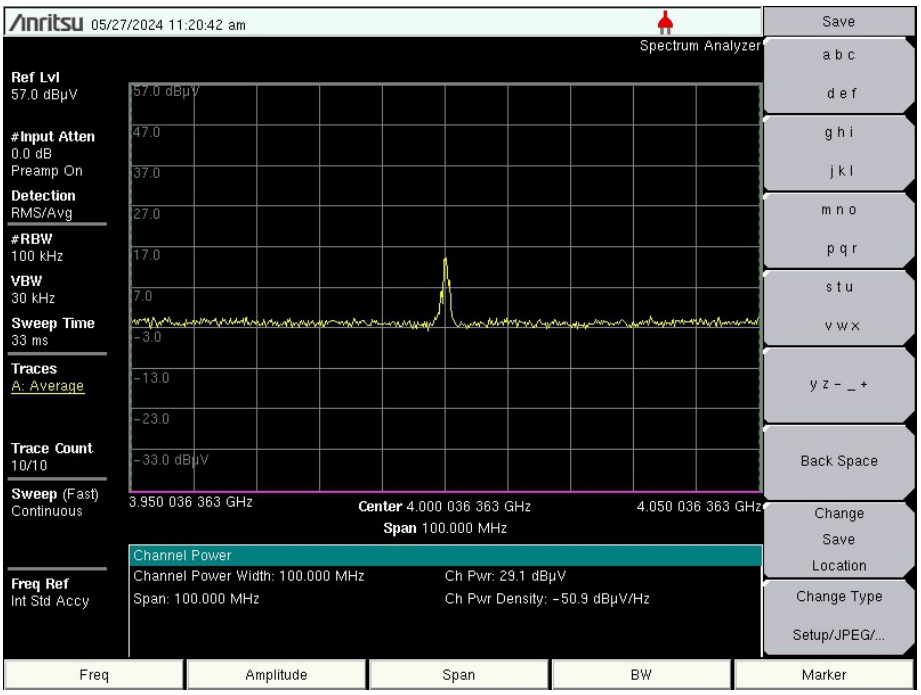


Рис. 4. Без нагрузки (составлено автором)

- с нагрузкой (E_o): 29,8 dBµV (рис. 5).

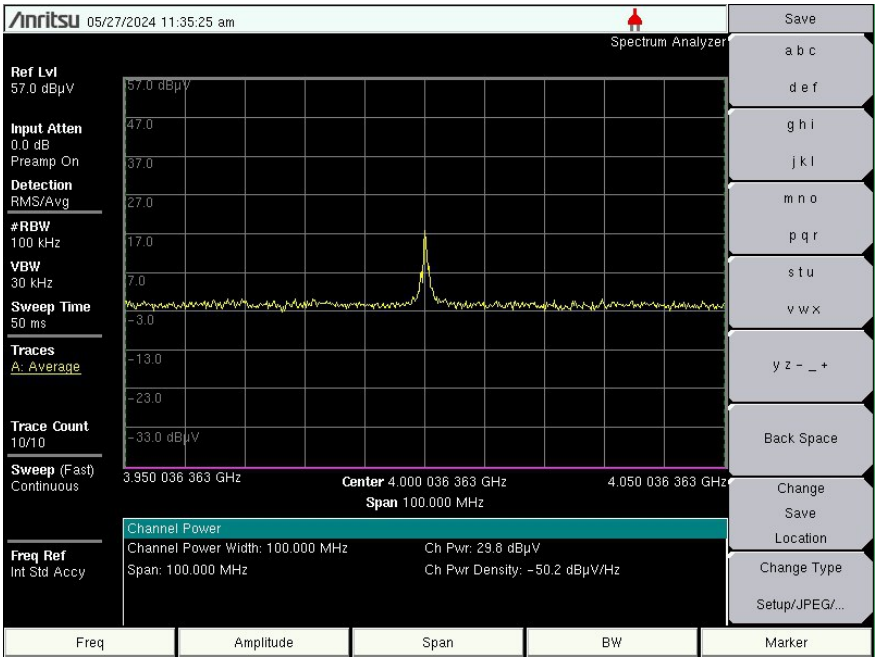


Рис. 5. Уровень при работе ПО (составлено автором)

Данные значения анализатора спектра необходимо перевести в микровольты по формуле:

$$E = 10^{\frac{E}{20(\text{дБ})}} \text{ мкВ/м},$$

где E – электрическая составляющая электромагнитного поля.

Далее идет уровень информационного сигнала для электрической составляющей электромагнитного поля, излучаемого исследуемым интерфейсом, вычисляются по формуле:

$$E_{ci} = \sqrt{Eoi^2 - Eши^2} = \sqrt{(30{,}91)^2 - (28{,}85)^2} = 11{,}1 \text{ мкВ/м},$$

Расчёт радиусов зон для частоты 4 ГГц ($\lambda = 0{,}075$ м) по степенному закону затухания поля при введении эмпирического коэффициента k:

$$R_{\text{ближ}} = \frac{1}{\sqrt[3]{\frac{k \cdot E_{\text{ш}}}{E_c}}} \approx 0{,}86 \text{ м},$$

$$R_{\text{пром}} = \frac{1}{\sqrt{\frac{k \cdot E_{\text{ш}}}{E_c}}} \approx 0{,}8 \text{ м},$$

$$R_{\text{дал}} = \frac{1}{\frac{k \cdot E_{\text{ш}}}{E_c}} \approx 0{,}64 \text{ м}.$$

где $k = 0{,}6$ – коэффициент, учитывающий наличие видеомонитора

Затем нужно рассчитать ближнюю и дальнюю зону что выполняется по формулам:

$$L_{1i} = \frac{\lambda i}{6} = \frac{0.075}{6} = 0{,}0125 \text{ м},$$

$$L_{2i} = 6 * \lambda_i = 6 * 0{,}075 = 0{,}45 \text{ м},$$

где L_{1i} – ближняя зона; L_{2i} – дальняя зона.

Радиус зоны Ri определяется из данного условия

$$R_i = \begin{cases} R_{i\ \text{ближ}}, \text{если } R_{i\ \text{ближ}} < L_{1i} \\ R_{i\ \text{пром}}, \text{если } L_{1i} < R_{i\ \text{пром}} < L_{2i} \\ R_{i\ \text{дал}}, \text{если } R_{i\ \text{дал}} > L_{2i} \end{cases}$$

В соответствии с методикой выбора, для данных условий распространения поля итоговый радиус зоны R2 соответствует радиусу дальней зоны и составляет 0,64 м.

Результаты экспериментальных исследований

В ходе тестирования разработанного ПО зафиксировано увеличение уровня мощности канала (Ch Pwr) с 29,1 дБ до 29,8 дБ при включении генерации нагрузки на интерфейс PCI-Express. Это свидетельствует о росте уровня побочных электромагнитных излучений, что подтверждает эффективность ПО для создания контролируемой нагрузки.

Проведённые измерения и расчёты показали, что зона R2 для исследуемого технического средства составляет 0,64 м, что указывает на необходимость учёта данного параметра при организации защищённых помещений.

Заключение

Разработано тестовое программное обеспечение для генерации нагрузки на интерфейс PCI-Express, позволяющее создавать условия для исследования ПЭМИН. Описаны особенности сопряжения ПЭВМ с измерительной аппаратурой, представлен математический аппарат для оценки зоны R2. Экспериментально подтверждено влияние работы ПО на уровень электромагнитных излучений. Полученные результаты могут быть использованы для оценки защищённости компьютерных систем и разработки мер по предотвращению утечки информации по каналу ПЭМИН.

Литература

1. Паршуткин А.В., Неаскина М.Р. Повышение защищенности информации от утечки через побочные электромагнитные излучения // Вопросы кибербезопасности. 2022. № 3 (49). С. 82-89. https://doi.org/10.21681/2311-3456-2022-3-82-89

2. Du C., Xia D., Huang Q. et al. Research on electromagnetic susceptibility of electronic modules in component-level HEMP PCI test // Energies (MDPI). 2022. Vol. 15(4):1409. https://doi.org/10.3390/en15041409

3. Dutta S.B., Naghibijouybari H., Abu-Ghazaleh N., Márquez A., Barker K. Leaky Buddies: Cross-Component Covert Channels on Integrated CPU-GPU Systems // Proc. of the ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA). 2021. P. 972-984. https://doi.org/10.1109/ISCA52012.2021

4. Kormanyos C. Real-Time C++: Efficient Object-Oriented and Template Microcontroller Programming. 4th ed. Cham: Springer, 2022. 389 p.

5. Side M., Yao F., Zhang Z. LockedDown: Exploiting Contention on Host-GPU PCIe Bus for Fun and Profit // Proc. IEEE European Symposium on Security and Privacy (EuroS&P). 2022. P. 270-285.

Салаватов А.А., Казиахмедов Т.Б.
Нижневартовский государственный университет
г. Нижневартовск, Россия

СРАВНИТЕЛЬНЫЙ АНАЛИЗ ВЕРСИЙ ТЕХНОЛОГИЧЕСКИЙ ПЛАТФОРМЫ 1С:ПРЕДПРИЯТИЕ 8.3 И 8.5

Сегодня практически каждый второй продукт человеческой деятельности, предлагаемый покупателю, имеет отношение к высоким технологиям. А это значит, что он не только удовлетворяет конкретные потребности человека, но и является воплощением научной и инженерной мысли. Очевидно, что в некоторых случаях заложенные в продукт идеи и достижения, полностью определяют его сущность и потребительскую ценность.

Отечественным лидером по автоматизации, обмену и обработке информации является фирма «1С», которая регулярно выводит на рынок программные продукты для самых разных сфер деятельности [3, с. 283]. Система программ «1С: Предприятие» состоит из технологической платформы и прикладных решений, разработанных на ее основе, позволяющих автоматизировать ту или иную деятельность организации или частного лица. Сама технологическая платформа не является программным продуктом, в котором может работать конечный пользователь [6, с. 242].

Стоит отметить, что разработчики программных продуктов актуализируют свои разработки всоответствии с российским законодательством. Особенно это касается бухгалтерских платформ, таккак бухгалтерам и руководителям компаний важна правильность формирования отчетности для налоговой службы и социального фонда России [2, с. 37].

Коронавирусные ограничения внесли изменения в привычный образ жизни людей. Отразилось это и на сфере образования. Облачный сервис «1С: Предприятие 8 через Интернет» позволяет удаленно получать доступ к продуктам компании [7, с. 164].

Конфигурации платформы «1С: Предприятие 8.3» могут служить источником актуальных данных для анализа деятельности той или иной компании [1, с. 28].

1С: Предприятие – это одна из самых популярных систем автоматизации бизнеса в России и странах СНГ. Каждая новая версия программы приносит с собой множество улучшений и новых возможностей.

Основные отличия 1С: Предприятие 8.5 от 8.3.

1. Архитектурные особенности

Одним из ключевых изменений в версии 8.5 является переработка архитектуры платформы. В новой версии значительно улучшена производительность и стабильность работы системы. Это достигается за счет оптимизации работы с базами данных, что особенно важно для крупных организаций с большими объемами данных.

2. Пользовательский интерфейс

1С: Предприятие 8.5 предлагает более современный и удобный интерфейс по сравнению с версией 8.3:

– Обновленный дизайн: интерфейс стал более интуитивным, с улучшенной навигацией и возможностью кастомизации под нужды пользователя.

– Поддержка различных разрешений экранов: теперь система лучше адаптируется под различные устройства, включая планшеты и смартфоны.

В релизе 8.5 значительно переработан пользовательский интерфейс. Визуальная часть теперь адаптируется под предпочтения пользователей, а скорость реакции системы выросла:

- Интерфейс стал чище и современнее: внедрены векторные иконки и карточный формат элементов
- Реализована поддержка светлого и тёмного режима
- Оптимизирована обработка больших объёмов данных в веб-клиенте
- Улучшена многопоточность и взаимодействие с СУБД
- В конфигураторе добавлены инструменты для упрощённой миграции старых формИнтерфейс 8.5 спроектирован так, чтобы снизить нагрузку на пользователя и повысить скорость выполнения операций.

3. Новые возможности для разработчиков

Версия 8.5 предоставляет разработчикам ряд новых инструментов и возможностей:

– Расширенные возможности языка 1С; включение новых функций и операторов, что позволяет создавать более сложные и гибкие решения.

– Улучшенная работа с веб-технологиями: Поддержка REST API и интеграция с внешними веб-сервисами значительно расширяет функциональность программ.

4. Облачные технологии

Программы «1С: Предприятие» 8.5 и 8.3 поддерживают облачные технологии. Это позволяет организациям использовать систему без необходимости установки на локальные серверы.Версия 8.5 активно поддерживает облачные технологии, что позволяет организациям использовать систему без необходимости установки на локальные серверы. Это обеспечивает:

– Гибкость: Легкий доступ к данным и функционалу из любого места.

– Экономию: Снижение затрат на IT-инфраструктуру и обслуживание.

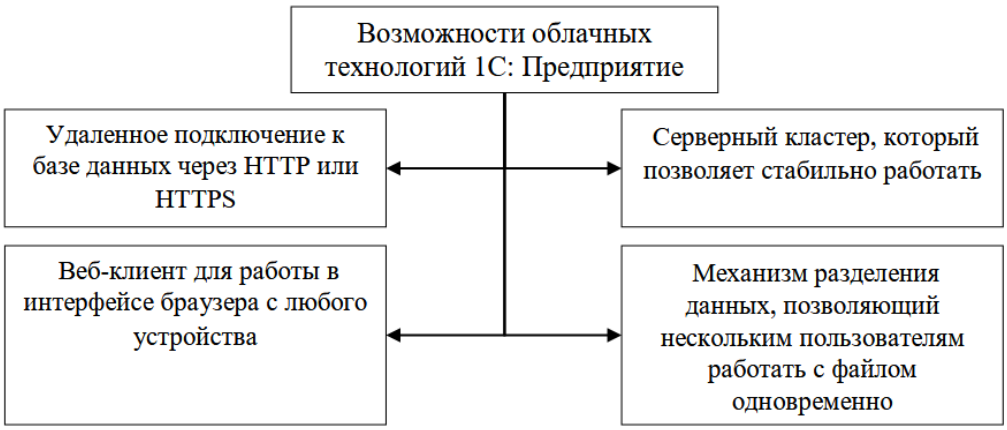


Рис. 1. Потребности пользователей 1С: Предприятие 8.3 в ходе использования облачных технологий [4, с. 212]

5. Интеграция с другими системами

Версия 8.5 значительно улучшила возможности интеграции с внешними системами:

– Поддержка обмена данными с другими ERP-системами: Это позволяет компаниям легко интегрировать 1С с существующими системами.

– Интеграция с CRM-системами: упрощает управление клиентскими данными и взаимодействие с клиентами.

6. Новые возможности отчетности

В версии 8.5 была значительно улучшена система отчетности:

– Гибкие настройки отчетов: Пользователи могут легко настраивать отчеты под свои нужды без необходимости программирования.

– Интерактивные отчеты: Возможность визуализации данных в виде графиков и диаграмм прямо в отчетах.

В 8.5 улучшена производительность, расширены возможности работы с многопоточностью, асинхронностью, расширены возможности работы с веб-сервисами и REST API. Можно постепенно внедрять эти возможности в обработку.

7. Режим совместимости

В платформе «1С:Предприятие» есть режимы совместимости, связанные с версиями 8.5 и 8.3. Эти режимы позволяют запускать конфигурации, разработанные в более младших версиях платформы, на новой версии без внесения изменений в конфигурацию и реструктуризации данных.

Самый практичный подход сейчас – поставить 8.5 рядом с 8.3 и не удалять старую версию.

Важно:

- удалять 8.3 не нужно;
- платформы 8.3 и 8.5 могут быть установлены одновременно;
- можно запускать разные конфигурации под разными платформами и спокойно сравнивать.

8. Взаимодействие с искусственным интеллектом

Существуют в природе сообщества насекомых и животных (муравьи, пчелы и др), которые существуют устойчиво и выполняют определенные, может с рождения алгоритмы (инстинкты), которые обеспечивают устойчивое существование колонии, хотя за каждым видом закреплены вполне элементарныефункции при всей сложности функций всей колонии в целом. Ведь и скорость человеческой мысли усиливается, если так можно выразиться, за счет «многопроцессорности» нашего мозга. продукта [5, с.5]

В платформах «1С: Предприятие» 8.5 и 8.3 есть возможности взаимодействия с искусственным интеллектом (ИИ). Интеграция ИИ в 1С происходит несколькими способами:

– Использование готовых облачных сервисов от компании 1С. Например, сервис «1С:Распознавание первичных документов» обрабатывает сканы и фото документов (счета, накладные, УПД, акты и пр.), извлекая контрагентов, номенклатуру и реквизиты и автоматически подготавливая их для ввода в базу 1С.

– Подключение внешних ИИ-сервисов через API. Например, можно интегрировать возможности распознавания речи от Yandex SpeechKit или Microsoft Azure AI.

– Создание специализированных модулей под конкретные задачи бизнеса. Здесь потребуются знания в области машинного обучения и использование библиотек вроде TensorFlow или PyTorch.

Версия 8.3

Данная версия вышла в 2012 году и претерпела ряд изменений:

1. Толстый клиент для Linux и Mac OS.
2. Мобильные платформы для Android и iOS.
3. Улучшение работы веб-клиента.
4. Возможность создания сложных аналитических отчетов.
5. Автоматизированное тестирование.
6. Новые инструменты для разработчика.
7. Работа фоновых заданий в файловом варианте.

Новое во внешнем виде:

- Дизайн приближен к веб-документу (кнопки «Избранное», переход к главной странице);
- Эффект «прозрачности»;
- Крупный шрифт;
- Панель разделов переместилась в левую часть;
- Возможность настройки собственного внешнего вида.

Подробное описание нововведений для 8.3 на сайте фирмы 1С.

Версия 8.5

Дата выхода – конец 2025 года. Является продолжением версии 8.3.

Новое:

- Упрощенный современный дизайн. Заявляется, что для работы нужно будет меньше лишних действий.
  - Светлая и темная темы.
  - Более красивые иконки.
  - Больше отступов для лучшего восприятия.
  - Группировка элементов и кнопок в виде карточек.
- Адаптивный интерфейс:
  - Дизайн для мобильных устройств максимально приближен к внешнему виду десктопов.
  - Векторные иконки. Избавляет от размытости при просмотре на экранах с разным разрешением.
- Улучшение производительности:
  - Ускорен запуск приложения.
  - Требуется меньше оперативной памяти.
  - Повышена производительность при работе с базами данных.
  - При работе с большими таблицами, браузерная версия более отзывчива.

Переход на новую версию происходит плавно при обновлении с версии 8.3.

Таблица

Сравнительные характеристики1С:Предприятие 8.5 и 8.3

Характеристика	1С:Предприятие 8.5	1С:Предприятие 8.3
Архитектура	Улучшенная, модульная	Более старая, менее гибкая
Поддержка технологий	Современные стандарты, веб, облако	Ограниченная, преимущественно клиент-сервер
Пользовательский интерфейс	Современный, адаптивный	Старый, менее дружественный
Возможности разработки	Развитые, расширяемые	Ограниченные
Поддержка и обновление	Постоянное развитие	Устаревшая, прекращение поддержки
Мультиплатформенность	Да	Ограниченная

Таким образом, переход от версии 8.3 к версии 8.5 в 1С: Предприятие стал значительным шагом вперед в развитии системы автоматизации организации. Новые функции, улучшенная производительность, безопасность и удобство работы делают эту версию более привлекательной для пользователей всех уровней.

Литература

1. Аграновский А.В., Турнецкая Е.Л. Совместное применение результатов различных видов анализа данных 1С для совершенствования деятельности малого предприятия // Актуальные проблемы экономики и управления. 2021. № 4 (32). С. 25-29. EDN FYNENJ.29.

2. Барабаш А.О., Городецкая О.Ю. Совершенствование бизнес-процессов дилерского центра на основе двусторонней интеграции «1С: Альфа-Авто» с рекламными интернет-площадками // Новые информационные технологии в образовании: Сборник научных трудов 18-й международной научно-практической конференции, (Москва, 30–31 января 2018 года) / Ч. 1. М.: 1С-Паблишинг, 2018. С. 36-39. EDN YMYCFC.

3. Володин С.М. Маринич А.Л. Опыт использования прикладных решений 1С для формирования профессиональных компетенций программиста для рынка труда в условиях цифровой экономики // Новые информационные технологии в образовании: Сборник научных трудов XXV Международной научно-практической конференции, (Москва, 04–05 февраля 2025 года). М.: 1С-Паблишинг, 2025. С. 282-285. EDN JLARHV.

4. Введенская Д.Ю. Проблематика и особенности облачных технологий в системе 1С: Предприятие 8.3 // Экономика и социум. 2023. № 1-1(104). С. 210-214. EDN RCJVPR.

5. Казиахмедов Т.Б. Искусственный интеллект как актуальное направление научных исследований // Информационные ресурсы в образовании: Материалы Международной научно-практической конференции, (Нижневартовск, 17–19 апреля 2013 года) / научный редактор: Т.Б. Казиахмедов. Нижневартовск: Нижневартовский государственный университет, 2013. С. 5-6. EDN SWHXRH.

6. Кислицын Е.В., Попова А.А. Проблемы автоматизации бизнес-процессов ведения эффективных контрактов сотрудников // Информационные системы и технологии в моделировании и управлении: Сборник трудов VII Международной научно-практической конференции. Отв. редактор К.А. Маковейчук, (Ялта, 24–25 мая 2023 года). Симферополь:

Общество с ограниченной ответственностью «Издательство Типография «Ариал», 2023. С. 241-244. EDN QELDIZ.

7. Романов М.Н., Фомина И.К. Роль технологической платформы «1С: Предприятие 8.3» в формировании цифровых компетенций современного специалиста // Новые информационные технологии в образовании: Сборник научных трудов XXIII Международной научно-практической конференции (Москва, 31 января – 01 февраля 2023 года). Т. 1. М.: 1С-Паблишинг, 2023. С. 164-166. EDN BGELKB

Степанов Н.А.
Нижневартовский государственный университет
г. Нижневартовск, Россия

ОБУЧЕНИЕ С ПОДКРЕПЛЕНИЕМ НЕЙРОАГЕНТА НА ОСНОВЕ АНАЛИЗА ВИДЕОИНФОРМАЦИИ В ДИНАМИЧЕСКОЙ СРЕДЕ

В последние годы наблюдается стремительное развитие технологий искусственного интеллекта, особенно в области обучения с подкреплением (Reinforcement Learning, RL) и компьютерного зрения. Одним из перспективных направлений является создание автономных систем – нейроагентов, способных воспринимать окружающую среду и принимать решения на основе визуальной информации без явного программирования. Такие системы находят применение в робототехнике, игровых средах, системах мониторинга и других областях, где недоступны традиционные сенсоры или команды оператора [4].

Целью данной работы является разработка и реализация модели нейроагента, способного обучаться в динамической среде на основе анализа видеопоследовательностей без предварительной разметки [2].

Обучение с подкреплением представляет собой парадигму машинного обучения, в которой агент учится принимать оптимальные решения через взаимодействие со средой [6]. Основными элементами RL являются:

- Агент – сущность, принимающая решения.
- Среда – контекст, в котором действует агент.
- Награда (Reward) – скалярный сигнал, указывающий, насколько действие было успешным.
- Политика (Policy) – стратегия, по которой агент выбирает действия.
- Функция ценности (Value function) – оценка ожидаемой награды при выполнении определённого действия.

Рассмотрим концептуальную модель нейроагента. Для построения эффективной модели нейроагента, способного обучаться в динамической среде на основе видеоданных, предлагается использовать следующую концептуальную структуру, которая описывает последовательность обработки информации (рис. 1):

- Видео – входной сигнал от окружающей среды (например, поток кадров с камеры).
- Encoder – модуль, извлекающий признаки из видеопоследовательности (CNN, Vision Transformer и др.).
- Latent State – внутреннее представление состояния среды в числовом пространстве.
- Policy Network – модель, формирующая действие на основе текущего состояния.
- Action – выходное действие агента, влияющее на среду.
- Такой подход позволяет моделировать поведение агента без явных показаний сенсоров или текстовых команд, используя только визуальные данные [7].

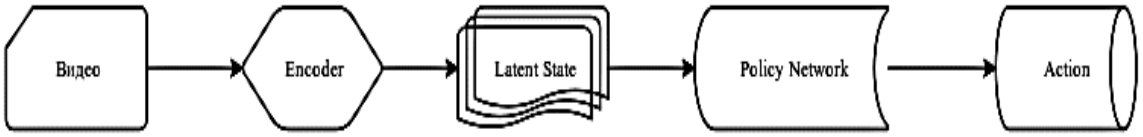


Рис. 1. Концептуальная структура модели нейроагента

Для преобразования видеоданных в пригодное для RL состояние можно использовать несколько типов энкодеров:

- CNN (ResNet, EfficientNet) – извлечение локальных признаков из отдельных кадров.
- RNN / LSTM / GRU – моделирование временных зависимостей между кадрами.
- 3D-CNN – одновременная обработка пространственной и временной информации.
- Vision Transformers (ViT, TimeSformer, ViViT) – современные архитектуры, эффективно обрабатывающие видеопоследовательности.

После получения векторного представления состояния среды (latent state) необходимо определить, каким образом будет формироваться стратегия действий. Для этого могут использоваться следующие методы:

- Q-learning / DQN – если действия дискретны.
- PPO / SAC / DDPG – для непрерывных действий.
- Imitation learning – обучение по образцам, если доступны демонстрации.
- Reward shaping – коррекция функции награды на основе изменений в кадре.

Представим практическую реализацию проекта в workflow n8n, схема которого на рисунке 2.

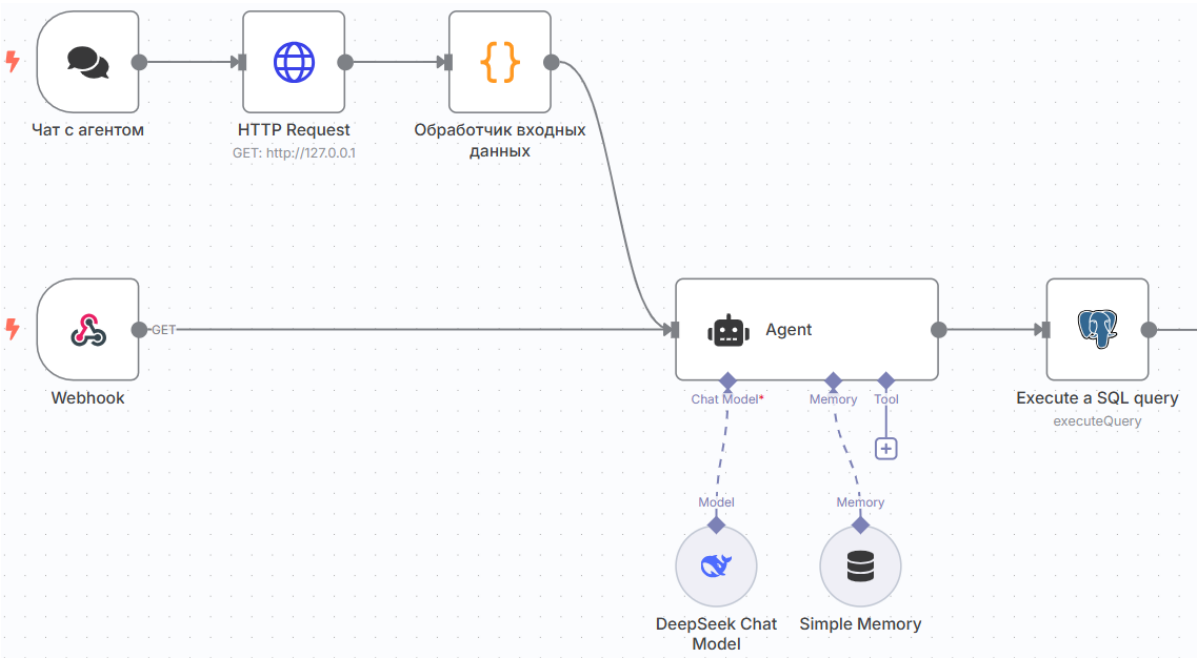


Рис. 2. Схема проекта в workflow в n8n

Узлы workflow:

1. Чат с агентом – Получение сообщения от пользователя через чат.

2. HTTP Request – Запрос к серверу для получения текущего состояния или действия.
3. Обработчик входных данных – Обработка входных данных и преобразование их в формат, понятный для агента.
4. Agent – Центральный компонент, принимающий решения на основе видеоданных и предыдущей истории.

- DeepSeek Chat Model – Модель для анализа текстовых команд пользователя.
- Simple Memory – Хранение истории действий и состояний.

5. Execute a SQL query – Запись результатов в базу данных

Рассмотрим пример кода для работы с задачей: «Найди на изображении людей в синей шляпе».

```
import cv2
import numpy as np
# 1. Загрузка изображения
def load_image(image_path):
    img = cv2.imread(image_path)
    return img
# 2. Детектирование людей на изображении
def detect_people(img):
    # Используем предварительно обученную модель для детектирования людей
    net = cv2.dnn.readNetFromCaffe("deploy.prototxt",
"res10_300x300_ssd_iter_140000.caffemodel")
    blob = cv2.dnn.blobFromImage(cv2.resize(img, (300, 300)), 1.0, (300, 300), (104.0, 177.0, 123.0))
    net.setInput(blob)
    detections = net.forward()
    people_boxes = []
    for i in range(detections.shape[2]):
        confidence = detections[0, 0, i, 2]
        if confidence > 0.5:  # Порог уверенности
            box = detections[0, 0, i, 3:7] * np.array([img.shape[1], img.shape[0], img.shape[1],
img.shape[0]])
            (startX, startY, endX, endY) = box.astype("int")
            people_boxes.append((startX, startY, endX, endY))
    return people_boxes
# 3. Анализ атрибутов (например, наличие синей шляпы)
def analyze_attributes(people_boxes, img):
    # Здесь можно использовать дополнительные модели для анализа атрибутов
    # Например, классификатор для определения цвета шляпы
    blue_hat_people = []
    for box in people_boxes:
        startX, startY, endX, endY = box
        person_roi = img[startY:endY, startX:endX]
        # Простой анализ цвета (синий – RGB [255, 0, 0])
        hsv = cv2.cvtColor(person_roi, cv2.COLOR_BGR2HSV)
```

```
            lower_blue = np.array([100, 150, 0])  # Нижняя граница синего цвета в HSV
            upper_blue = np.array([140, 255, 255])  # Верхняя граница синего цвета в HSV
            mask = cv2.inRange(hsv, lower_blue, upper_blue)
            # Если есть значительная площадь синего цвета, считаем, что это шляпа
            if np.sum(mask) > 1000:  # Пороговое значение для площади синего цвета
                blue_hat_people.append(box)
        return blue_hat_people
    # 4. Отображение результатов
    def draw_results(img, boxes, color=(0, 255, 0)):
        for box in boxes:
            startX, startY, endX, endY = box
            cv2.rectangle(img, (startX, startY), (endX, endY), color, 2)
        cv2.imshow("Result", img)
        cv2.waitKey(0)
        cv2.destroyAllWindows()
    # Главная функция
    def main():
        # Шаг 1: Загрузка изображения
        image_path = "path_to_your_image.jpg"
        img = load_image(image_path)
        # Шаг 2: Детектирование людей
        people_boxes = detect_people(img)
        # Шаг 3: Анализ атрибутов (найти людей в синей шляпе)
        blue_hat_people = analyze_attributes(people_boxes, img)
    if __name__ == "__main__":
        main()
```

Было загружено тестовое изображение, представленное на рисунке 3.

Рис. 3. Тестовое изображение

После обработки изображения с использованием Vision Transformer и модели обучения с подкреплением (PPO), система формирует следующие выходные данные:

[Анализ кадра]

Текущее состояние среды:

– В поле зрения находятся 3 человека.

– Один человек находится справа от центра кадра.

– У одного из людей на голове – синяя шляпа.

[Решение агента]

Действие: "Переместиться вправо"

Причина: "Обнаружен объект интереса (человек в синей шляпе)."

[Награда]

Reward: +10

(за успешное обнаружение и корректное движение в сторону объекта)

[Состояние системы]

Latent State: [0.72, 0.34, -0.12, 0.56, ...] // Вектор признаков

Использованная модель: Vision Transformer + PPO

[Время реакции]

Processing time: 220 ms

Эти выходные данные могут быть представлены в различных форматах, включая текстовый вывод, JSON-файл, графическое отображение или SQL-запись в базе данных [2].

Такой подход позволяет не только анализировать поведение агента, но и сохранять результаты для последующего обучения и оптимизации стратегии [5].

Формат JSON (для интеграции с системой или БД):

```
 "timestamp": "2025-04-05T12:34:56Z",
 "frame_analysis": {
  "detected_objects": [
   {"class": "human", "position": [120, 150, 280, 360], "attributes": {"hat_color": "blue"}},
   {"class": "human", "position": [400, 100, 150, 300], "attributes": {"hat_color": "none"}},
   {"class": "human", "position": [700, 200, 180, 380], "attributes": {"hat_color": "red"}}
  ]
 },
 "agent_decision": {
  "action": "move_right",
  "reason": "Цель обнаружена справа (человек в синей шляпе)"
 },
 "reward": 10,
 "state_vector": [0.72, 0.34, -0.12, 0.56, 0.11, -0.45, ...],
 "processing_time_ms": 220
}
```

Логирование в БД (SQL):

```
INSERT INTO agent_actions (
   timestamp,
   frame_data,
   detected_objects,
   action,
   reward,
   latent_state
) VALUES (
```

```
        NOW(),
        'frame_20250405_123456.jpg',
        '[{"class": "human", "position": [120, 150, 280, 360], "hat_color": "blue"}, {"class": "human", "position": [400, 100, 150, 300], "hat_color": null}]',
        'move_right',
        10,
        '[0.72, 0.34, -0.12, 0.56, 0.11, -0.45, ...]'
    );
```

В ходе проведённых исследований была разработана и реализована система нейроагента, способного обучаться в динамической среде на основе анализа видеоданных. Было показано, что использование платформы n8n позволяет легко интегрировать различные компоненты (чат, видеоанализ, RL-агент, база данных) в единый рабочий процесс. Практическая реализация системы подтвердила её работоспособность и потенциал для применения в реальных условиях [8].

А также были были выявлены следующие преимущества, ограничения:

1. Преимущества:

- Интерактивность: Возможность управления через чат.
- Масштабируемость: Легко расширяемая архитектура.
- Автономность: Система работает только на основе визуальных данных.

2. Ограничения:

- Высокая размерность данных.
- Сложность интерпретации.
- Недостаток обучающих данных.
- Проблема обобщаемости.

Несмотря на существующие ограничения, такие как высокая вычислительная сложность и проблема обобщаемости, развитие методов self-supervised learning, transfer learning и гибридных архитектур даёт основание полагать, что эти проблемы будут успешно решены в ближайшем будущем [3].

Практическая значимость работы состоит в возможности применения разработанной модели в автономных системах, адаптированных к изменяющимся условиям [1].

Литература

1. Городецкий В.А., Васильев В.Н., Кузнецов О.П. и др. Искусственный интеллект: современные технологии и перспективы развития. М.: Физматлит, 2021. 432 с.

2. Лебедев Б.К., Лебедева О.Б., Лебедева Е.О. Методы оптимизации и искусственный интеллект. Таганрог: Изд-во ТРТУ, 2020. 398 с.

3. Манюкова Н.В. Компьютерное зрение как средство извлечения информации из видеоряда // Математические структуры и моделирование. 2015. № 4 (36). С. 123-128.

4. Носов В.А. Основы теории игр и её приложения в информационных технологиях. М.: МФТИ, 2019. 240 с.

5. Руткас А.Г., Руткас Н.А. Нейросетевые модели и алгоритмы обучения. СПб.: Политехника, 2022. 304 с.

6. Саттон Р.С., Барто Э.Г. Обучение с подкреплением. М., 2014. 402 с.

7. Тарасова Г.Г. Компьютерное зрение: от распознавания образов к пониманию смысла // Искусственный интеллект и принятие решений. 2021. № 3. С. 12-23.

8. Черногоров А.В. Обучение с подкреплением в системах управления роботами // Вестник новых информационных технологий. 2020. № 4. С. 35-42.

Столярова А.Г.
Нижневартовский государственный университет
г. Нижневартовск, Россия

РАЗРАБОТКА МОДУЛЯ УЧЕТА РАБОЧЕГО ВРЕМЕНИ СОТРУДНИКОВ В 1С:ПРЕДПРИЯТИИ

Руководителю подразделения необходимо следить за явкой своих подчинённых, при малом количестве сотрудников подразделения и очном нахождении на работе это сделать довольно просто, но при удаленном режиме работы и большом количестве сотрудников это может быть достаточно проблематично. Многие работодатели для учета рабочего времени используют электронные пропуска для кадров, работающих очно, а для дистанционных сотрудников используют тайм-трекеры [2].

В данной статье представлена разработка модуля для учета рабочего времени сотрудников.

На территории Российской Федерации долгое время одним из популярных решений для автоматизации бизнес-процессов, учета, управления и аналитики остаются решения на платформе 1С:Предприятие [4], поэтому для создания модуля была выбрана именно эта платформа.

Разработка модуля началась с определения возможных ролей пользователей и их функционала:

1. «Администрирование» – администратор с полными правами.
2. «Пользователь» – роль для обычного сотрудника, у которого есть возможность отметить начало и конец рабочего дня, возможность создать отклонение рабочего времени на работу в выходной день или на отпуск.
3. «Отчеты и аналитика» – роль для руководителей, включающая возможность посмотреть отчеты по отделу за день, общую статистику за месяц, а также статистику по конкретному сотруднику.
4. «Табелирование» – роль для табельщика, включающая возможность создания графиков работы, создание для определенного пользователя табеля с учетом его болезней и отпусков.

Важные моменты, которые стоит отметить:

- Пользователь может отметить причину опоздания, выбрав одну из предложенных причин или описав свою ситуацию.
- При болезни сотрудника начальник должен создать соответствующий документ на имя сотрудника, который впоследствии будет учтен в табеле, в этом документе также указывается заместитель.

Все функции данного модуля размещены в 3 подсистемах для удобного доступа:

1) Отчеты и аналитика;
2) Учет рабочего времени;
3) Табелирование.

Перейдем к основному функционалу.

На рабочей области начальной страницы размещается обработка «Форма отметки рабочего времени», на которой можно отметиться о начале (рис. 1) или о завершении (рис. 2) работы. Добавлены соответствующие картинки, которые символизируют остановку и работу. Данная обработка доступна всем пользователям.

Время начала и завершения работы фиксируются в документах «Фиксация начала работы» и «Фиксация завершения работы». В них также идет метка и причина опоздании. Данные документов доступны для редактирования ролям «Администрирование» и «Табелирование».

Был создан справочник «Пользователи», который сам обновляется при добавлении новых пользователей через Конфигуратор. В нем содержится основная информация о пользователях: ФИО, адрес проживания, день рождения, а также в табличной части содержатся роли доступа.

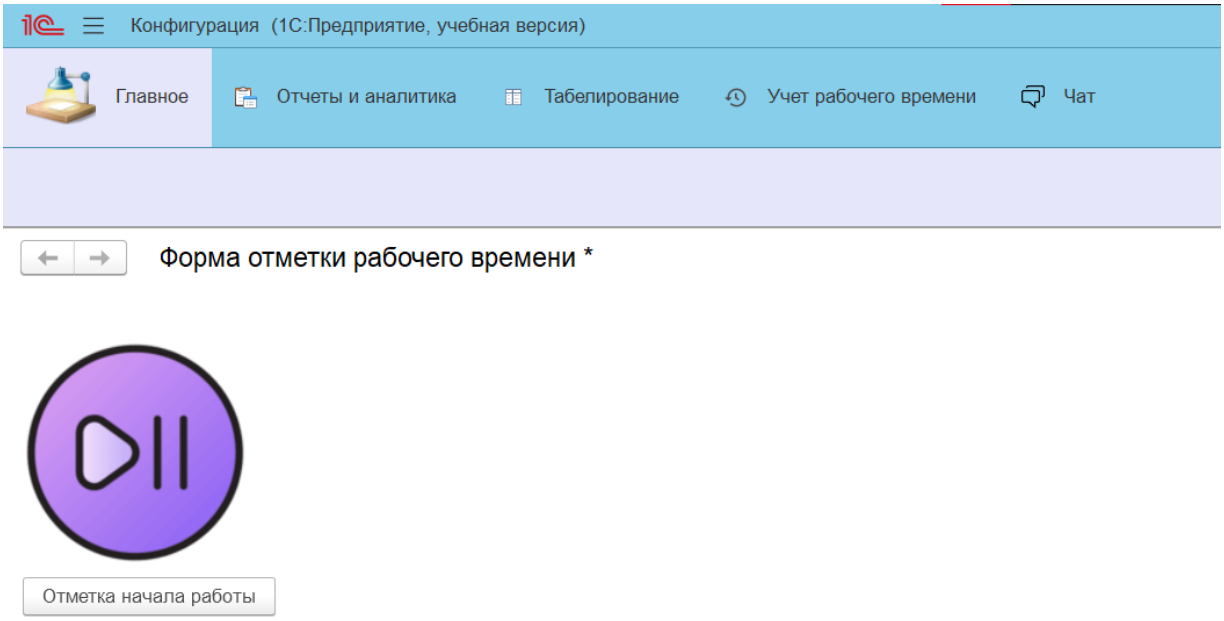


Рис. 1. Обработка “Форма отметки рабочего времени”, функция отметки начала работы

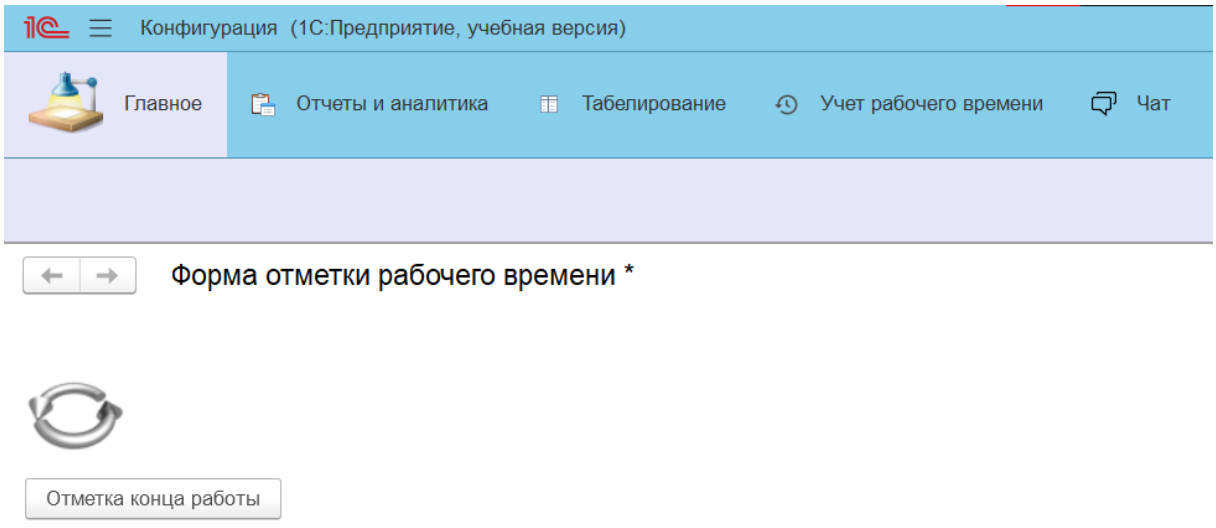


Рис. 2. Обработка «Форма отметки рабочего времени», функция отметки конца работы

Так как часто люди на работе могут испытывать стресс, было решено изменить стандартное желтое оформление дизайна. В качестве основных цветов были выбраны: небесно-голубой, аквамарин, бледно-лиловый и бледно-бирюзовый (рис. 3). Данное решение обусловлено тем, что оттенки голубого цвета в психологии символизируют спокойствие и создают атмосферу безопасности [5].

Каждый пользователь может создать документ «Отклонение рабочего времени» (рис. 4) на отпуск или на работу, в нем указываются даты, вид отклонения, длительность, при отпуске выбирается заместитель. Оно учитывается при создании табеля.

Табельщик может создать новый график работы (рис. 5), а также указать его для конкретного пользователя и скорректировать определенные дни работы и выходных. Для этого были созданы два справочника «Графики работы» и «Графики пользователей».

Для создания графика работы выбирается периодичность работы: «2 дня работы / 2 дня выходных» или «5 дней работы / 2 дня выходных», количество часов в день, перерыв и другие параметры. Данные табличной части обновляются при изменении параметров или через кнопку «Заполнить график», она доступна для редактирования. Есть возможность для добавления большего количества периодичностей.

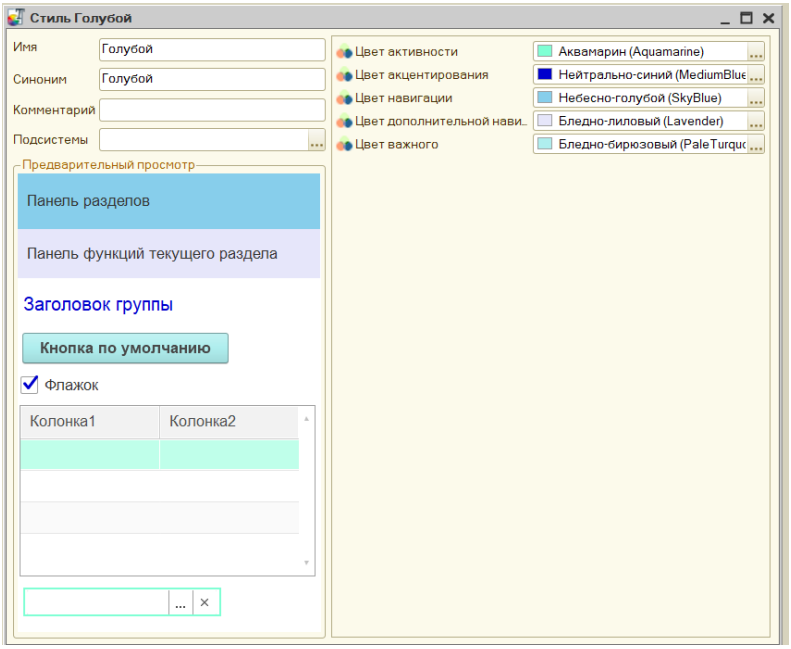


Рис. 3. Настройки стиля

Рис. 4. Отклонение рабочего времени

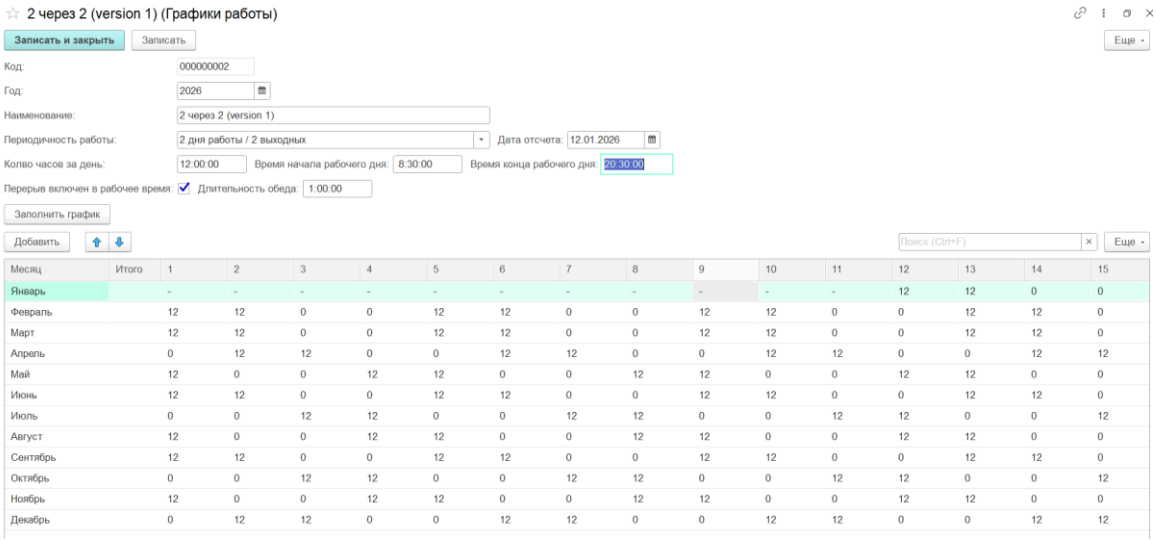


Рис. 5. Создание графика работы

Для обмена информации по рабочим вопросам часто используют различные мессенджеры. Но при взломе аккаунтов работников [3] могут произойти серьезные утечки данных, а также финансовые потери для бизнеса.

По этим причинам удобно иметь внутренний корпоративный чат для решения рабочих вопросов. Так как платформа 1С дает широкие возможности для исполнения прикладных решений [1] и достаточно распространена, что дает возможность для более быстрой интеграции в работу предприятия. Было решено реализовать чат.

Для удобства чат размещен, как подсистема с одной командой, то есть при нажатии сразу открывается форма чата. В ней необходимо выбрать адресата, после этого появляются данные в поле «Сообщения» с типом «Поле HTML документа». Реализовано отображение статуса сообщения, «//» означает прочтено, а «/» – не прочтено (рис. 6).

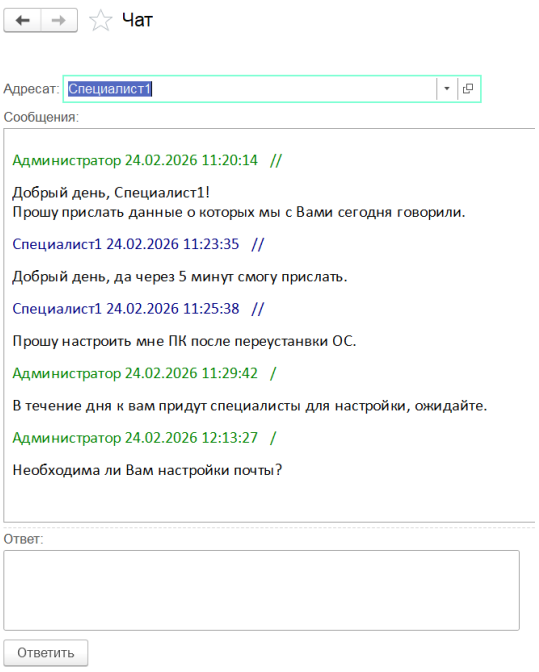


Рис. 6. Обработка чата

Для сохранения сообщений был создан регистр сведений, в котором содержится информация: отправитель, адресат, текст сообщения и булевская переменная «прочитано».

Таким образом, был реализован модуль на платформе 1С:Предпирятие для учета рабочего времени сотрудников, со встроенным чатом для общения.

У этого модуля есть возможность добавить некоторые полезные функции, например, подробная и детальная аналитика опозданий работников, групповые чаты, отправление фотографий и файлов.

Литература

1. Афоничкин А.А. Инновационные технологии в программном продукте 1С // Индустрия 1С: Сборник статей II региональной научно-практической конференции (Брянск, 28 ноября 2023 года). Брянск: Брянский государственный инженерно-технологический университет, 2023. С. 46-50.

2. Бегичева С.В. Автоматизированные системы учета рабочего времени сотрудников // Экономика и бизнес: теория и практика. 2023. № 9 (103). С. 12-16.

3. Бедняков Н.Д., Хахина А.М. Сетевая безопасность в повседневной жизни // Информационные технологии: прошлое, настоящее, будущее: сборник статей по материалам межинститутской студенческой научно-практической конференции (г. Санкт-Петербург, 20 мая 2021 года). Санкт-Петербург: Санкт-Петербургский государственный лесотехнический университет им. С.М. Кирова, 2021. С. 18-23.

4. Борлакова Б.Ф. Программирование 1С: развитие и преимущества // Молодой учёный: сборник статей V Международной научно-практической конференции. В 2 частях. Том 5 (г. Пенза, 23 января 2024 года). Пенза: Наука и Просвещение, 2024. С. 85-87.

5. Краснянская Т.М., Тылец В.Г., Иохвидов В.В. Цветовой дизайн персональных концепций безопасности // Эргодизайн. 2023. № 3 (21). С. 267-275.

Томшин Н.А.
Нижневартовский государственный университет
г. Нижневартовск, Россия

РАЗРАБОТКА ФРЕЙМВОРКА НА ОСНОВЕ JAVA SWING ДЛЯ СОЗДАНИЯ ПРОГРАММ С ИНТЕРАКТИВНЫМИ ГРАФАМИ

Данная статья посвящена разработке модуля на основе Java Swing, упрощающего создание графических интерфейсов для представления и моделирования плоских графов с интерактивными компонентами.

Граф – математическая модель, определяющая множество вершин и множество их взаимоотношений, причём в общем случае нет никаких ограничений по их структуре, что позволяет использовать графы для моделирования любых других объектов: иерархий, сложных систем, алгоритмов.

Плоский граф – граф, отображённый на плоскость. В контексте типичных двумерных графических интерфейсов, граф можно представить множеством графических элементов на плоскости экрана, изображающих узлы и связи, причём как правило узлы имеют постоянную форму, а связи процедурно видоизменяются, чтобы изобразить путь между узлами и сопутствующую информацию.

Одна из подзадач отображения графа на плоскости – это укладка. Под этим подразумевается размещение узлов на плоскости таким образом, чтобы граф был как можно легче и предельно однозначно читаем. Есть много рекомендаций [5, с. 31] и алгоритмов [3, с. 14; 4, с. 124] для укладки графов, но если граф топологически невозможно разместить на плоскости без взаимопересечения дуг (рёбер), это всегда ведёт к повышению трудности чтения из-за возросшей трудности различия двух дуг или косвенно из-за дополнительных компонентов и деталей, которые создаются, чтобы от пересечения избавиться (дополнительный графический элемент на пересечении или повторное появления одного и того же узла по обе стороны).

Разрабатываемые инструменты должны быть пригодны для создания любых читаемых человеком интерактивных графов. Рассматривая интерактивный граф как процедурно генерируемый графический интерфейс, граф будет являться иерархией графических компонентов, содержащей на втором уровне отображение узлов и взаимосвязей. Касательно интерактивности, у программиста-пользователя должна быть возможность реализовать логику любого поведения независимо от предустановленных по умолчанию настроек взаимодействия с графами. Первичная декомпозиция интерактивного графа приведена на рисунке 1.

Согласно принципу разделения ответственности, контроллер графа будет отдельным модулем от панели, это позволяет независимо программировать особенности размещения компонентов и высокоуровневую логику интерактивности графа.

Рассмотрим спектр моделируемых графов. В зависимости от моделируемых объектов и предметной области в графах варьируется сложность и количество используемых компонентов (рис. 2).

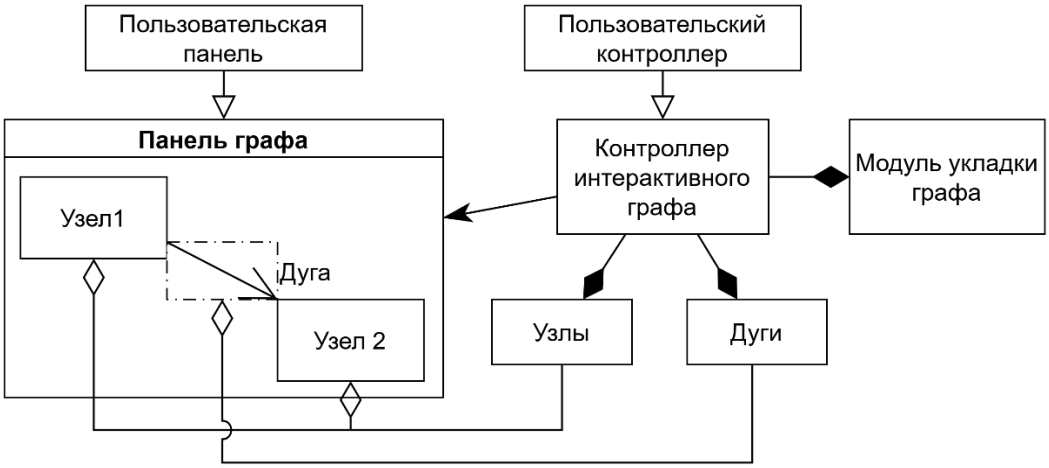


Рис. 1. Декомпозиция графового интерфейса в общем случае

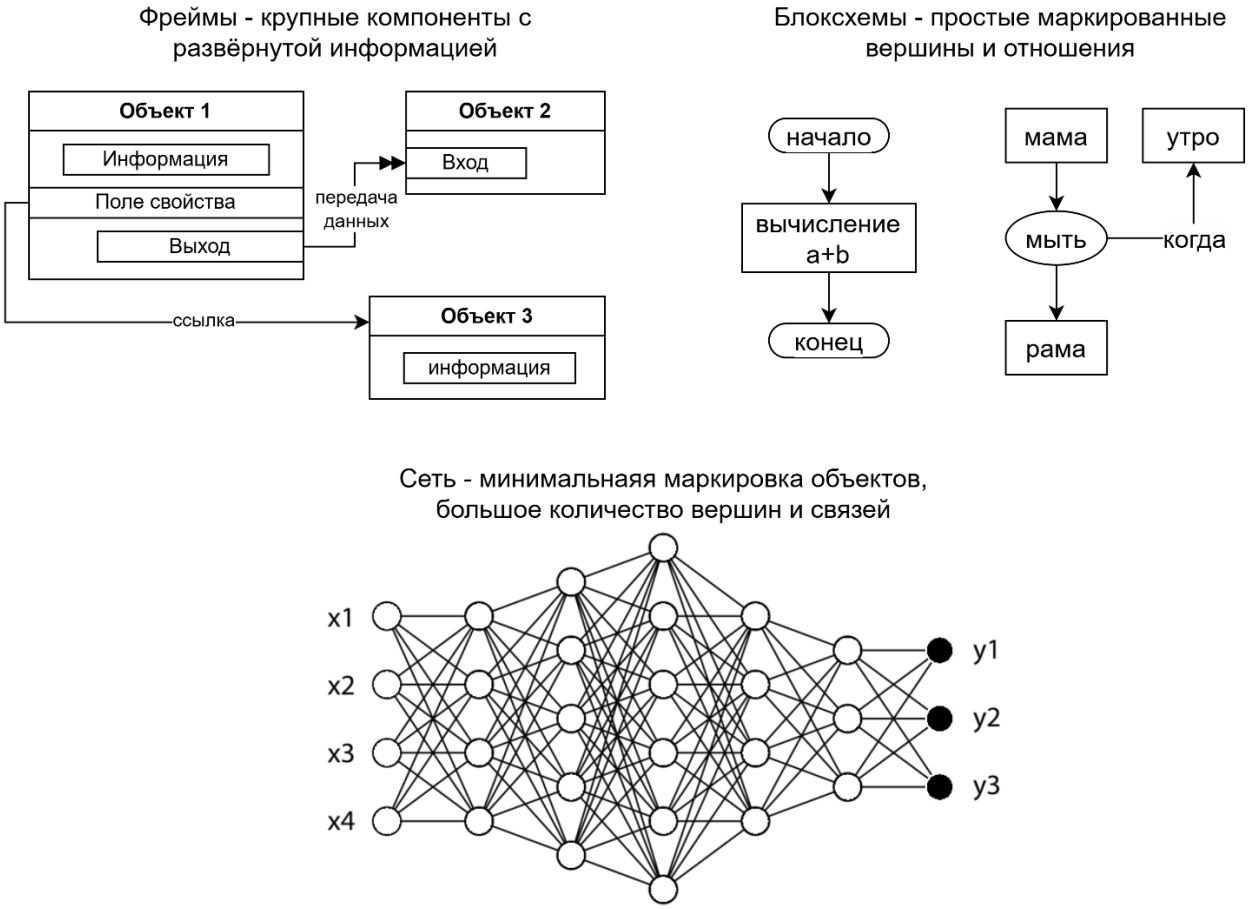


Рис. 2. Спектр детализации графов

Узлы разной сложности используются для представления разных объектов в разных контекстах. Разберём прецеденты их использования в таблице.

Нетрудно заметить, что количество деталей (информации) на площадь изображения остаётся примерно постоянной. Это обуславливается прежде всего ограничением зрительного восприятия, что также согласуется с возрастающей сложностью отрисовки возрастающего количества объектов и ограниченным разрешением дисплеев.

Таблица

Спектр детализации графов

Сложность узлов	Детализация	Области применения
Большая	Заголовки, поля свойств, вложенные компоненты графического интерфейса	– визуальное программирование; – работа с фреймами (форма представления знаний); – представление иерархических графов.
Средняя	Название, единственный графический компонент.	визуализация структур, иерархий, алгоритмов
Малая	Цветовая, сокращённые индивидуальные идентификаторы, маркировка групп узлов	визуализация структуры графов с большим количеством элементов

Масштабирование в Swing можно делать множеством способов. Можно не изменяя компонентов, показывать их уменьшенное изображение через панель просмотра (рис. 3).

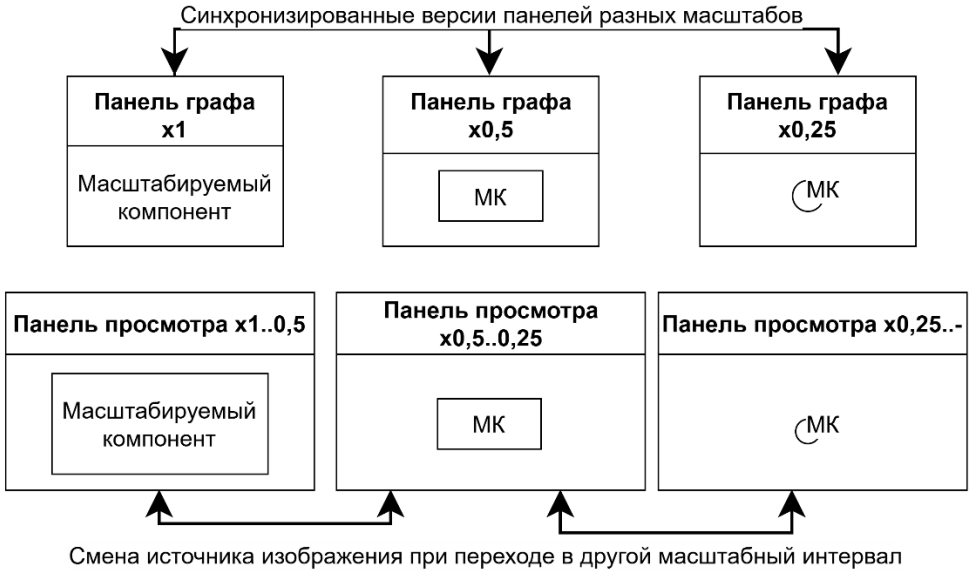


Рис. 3. Решение уменьшения масштаба

Разные детализации узлов (и отношений) могут использоваться для их отрисовки при достижении определённого отдаления (масштаба).

Увеличение изображения возможно, но оно не будет эквивалентно изображению увеличенного компонента, т.к. это будет масштабирования растра, а не векторных форм. Для «реального» масштабирования придётся полностью увеличить геометрические размеры всей иерархии, что будет становиться всё более дорогой операцией по мере увеличения количества элементов графа. Из этого следует что, если требуется высокое качество укрупнённых объектов без задержек на их перестройку, нужно чтобы все компоненты изначально были увеличены до предельного масштаба, а стандартный просмотр показывал их уменьшенную картинку. Разумеется, все эти варианты имеют свои недостатки и преимущества и должны быть доступны программисту и пользователю готовой программы, чтобы получить оптимальную производительность в конкретном случае.

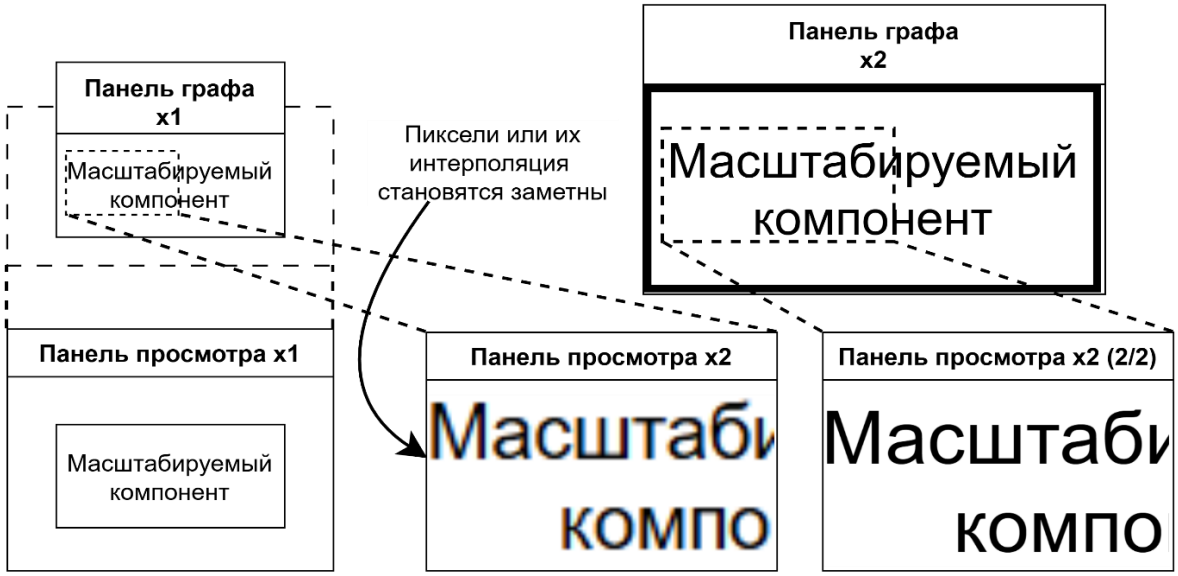


Рис. 4. Решение увеличивающего масштабирования

Можно сделать вывод, что Swing не лучшая платформа для отрисовки динамически масштабируемых компонентов, но это набор относительно простых инструментов, который идёт вместе с Java и потому имеет свою область применения. Представленные выше решения должны сделать работу с графами на Swing приемлемыми.

Следует отметить, что на Java существует совместимая со Swing библиотека GraphStream (https://graphstream-project.org/), предназначенная для визуализации и укладки мелко-узловых графов, но она не предназначена для непосредственной работы с крупными компонентами. Автор видит возможность в использовании инструментов из GraphStream для создания производительных модулей укладки для графовых интерфейсов (рис. 1).

Несмотря на широкое применение графов в науке, технологиях и многих программных продуктах [1; 2, с. 38], до сих пор не появилось общепринятого формата представления графов общего назначения и графических интерфейсов для работы с ними (в отличии от текста, таблиц, документов, изображений и т. д.). При разработке фреймворка, была предпринята попытка адресовать эту проблему. Модели графов, созданные при помощи разрабатываемых инструментов, должны быть легко переводимы в файловый формат, который может быть интерпретирован и воссоздан как менее специализированная интерактивная модель без потерь значимых свойств элементов. Разумеется, в результате работы будет получена лишь ещё одна пара формата кодирования и интерпретатора, но так как их предметная область не ограничена, а программные продукты их использующие образуют семейство совместимых редакторов – это по крайней мере будет прецедент, который может привести к дальнейшему появлению стандартов для записи и чтения графовых данных между приложениями.

Анализируя популярные прецеденты использования графов был определён расширяемый формат записи данных о строении графов, аналогичный концепции Labeled Property Graphs (графы меток и атрибутов) [5, с. 26]. На рисунке 5 показана структура описания графа.

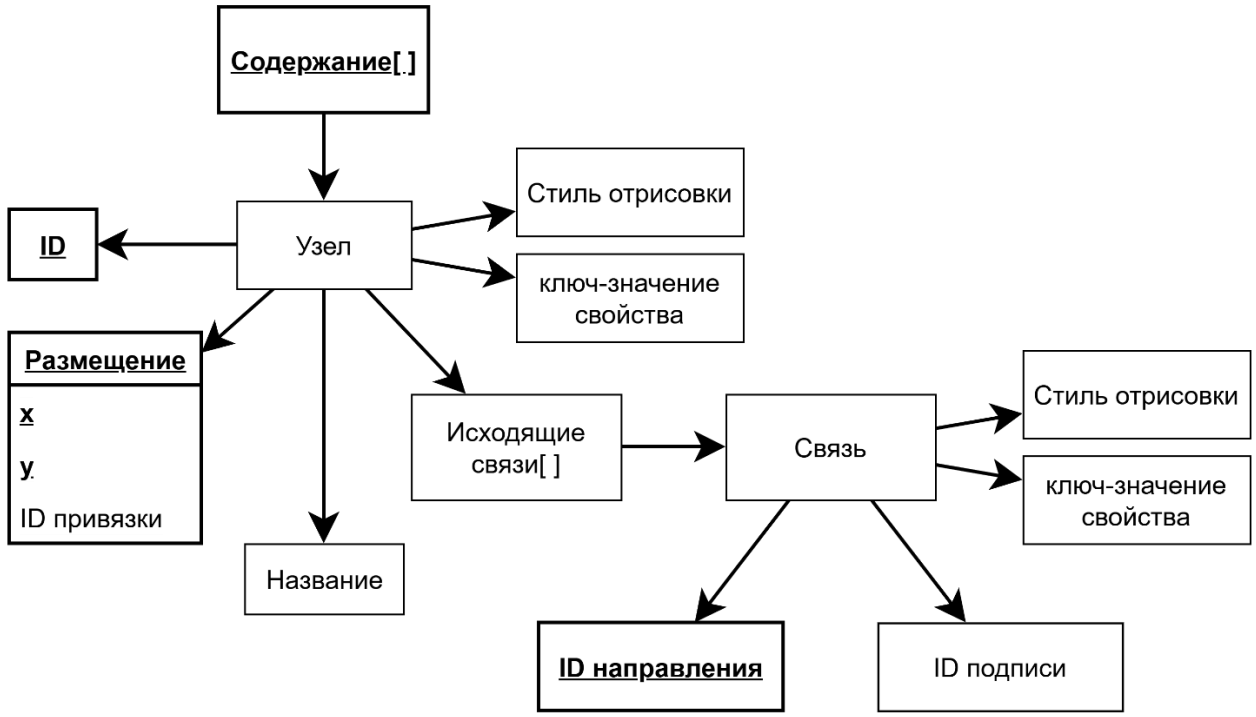


Рис. 5. Структура записи графа

Жирный текст и линии обозначают обязательные свойства. Содержание графа представляет массив узлов, который может быть пустым. У узлов обязательно есть ID и опциональное отличное от ID название подписи. У узла должно быть размещение на плоскости, которое может обладать дополнительными свойствами, например привязкой к другому узлу (но только координатами, без образования отношения), и главное – массив исходящих связей. Каждая связь (дуга) представлена в файле единожды, в массиве исходного узла. У каждой связи одно обязательное свойство ID конечного узла, также может быть добавлен ID узла, являющегося подписью для связи.

У узлов и связей могут быть, структуры включающие специальные параметры, описывающие стиль изображения и параметры специализированных свойств (предметных областей), впрочем, можно размещать эти свойства и на одном уровне с обязательными, главное, чтобы не возникало конфликтов. Простая версия интерпретатора графа считывает цвета текста, фона и стиль границ узлов, а также стили линии и стрелок для отношений, включая сдвиг вбок, чтобы встречные отношения не накладывались друг на друга.

Для сохранения и чтения графов используется нотация и файлы JSON, однако приведённая выше структура не требует какой-то специальной нотации, при условии, что интерпретатор графа способен без потерь сохранить и загрузить граф.

По итогам всех перечисленных изысканий была разработана структура типичного модуля программы для создания обеспечения интерактивной модели графа (рис. 6).

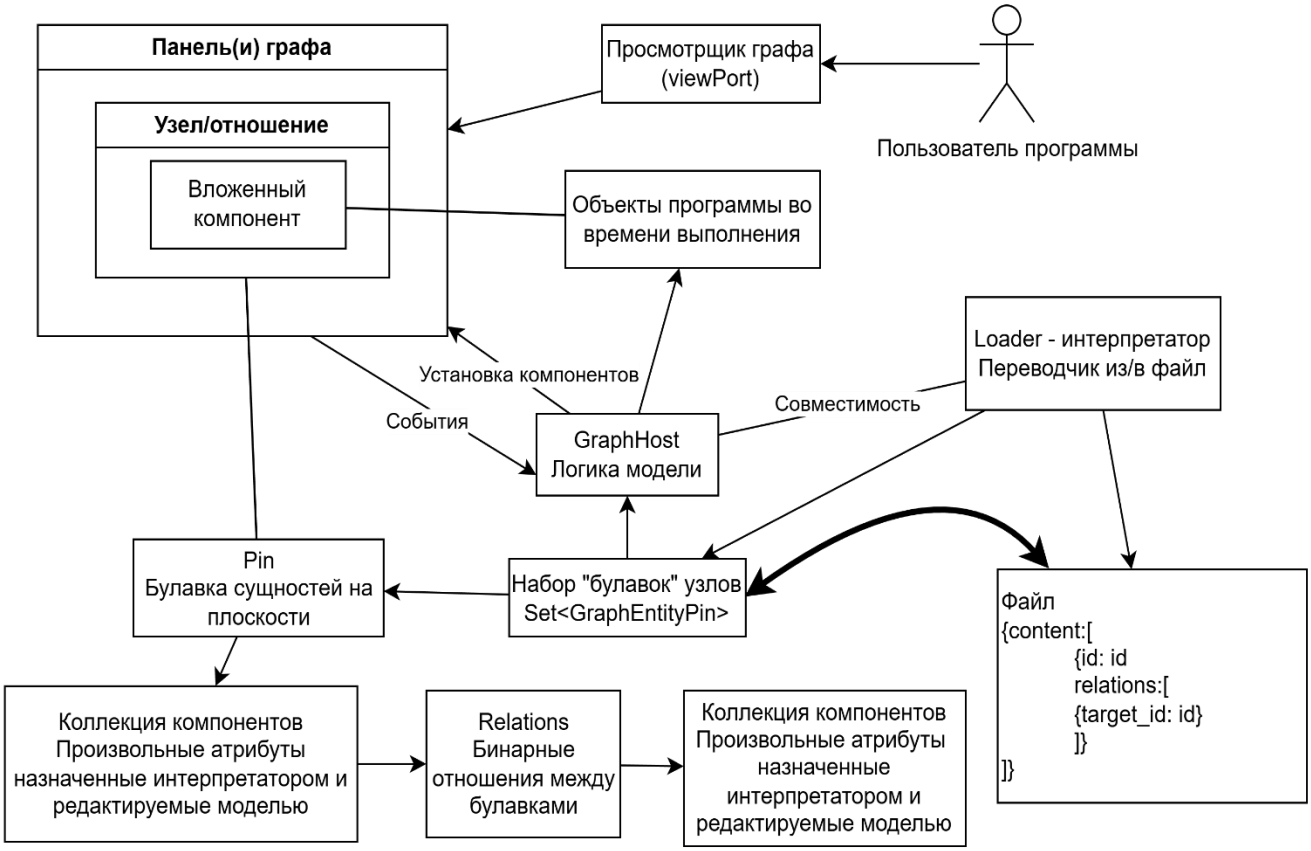


Рис. 6. Структура системы поддержки интерактивного графа

GraphHost (название может быть изменено) – класс, отвечающий за логику и поведение модели. Он работает с совместимым интерпретатором файлов, чтобы правильно сохранять и загружать модели графов. Графы создаются на панелях графов, которых, может быть, несколько штук для осуществления описанных ранее возможностей масштабирования. Логика модели содержит методы для синхронизации всех экземпляров панелей графа между собой через «булавочную» модель и собственные уникальные преобразования элементов, которые повторяются на каждом экземпляре. Просмотрщик графа – производный класс от JViewer, который поддерживает автоматическое переключение между экземплярами панелей графа при масштабировании или если того потребует специальная логика модели.

На момент написания статьи разработка фреймворка ещё не была завершена. Но до конца апреля публичная версия будет выложена на странице автора статьи на сайте github.com (https://github.com/NikolayTomshin).

Литература

1. Дворецкий Я.В., Покуса Т.В. Применение плоской укладки графа в создании печатных плат // Международный студенческий научный вестник. 2026. № 1. С. 2-2.

2. Исаев Р.А., Подвесовский А.Г. Когнитивная ясность графовых моделей: подход к пониманию идеи и способ выявления влияющих факторов с использованием визуального анализа // Научная визуализация. 2022. Т. 14, № 4. С. 38-51. https://doi.org/10.26583/sv.14.4.04. EDN PCGSEH

3. Касьянов В.Н. Методы и средства визуализации информации на основе атрибутированных иерархических графов с портами // Сибирский аэрокосмический журнал. 2023. Т. 24, № 1. С. 8-17. https://doi.org/10.31772/2712-8970-2023-24-1-8-17

4. Кулакова И.М. Лебедева О.А. Визуализация графа транспортной сети с небольшим количеством узлов // Современные технологии и научно-технический прогресс. 2022. № 9. С. 123-124. EDN PYORFV

5. Лисин В.А., Серый А.С., Сидорова Е.А. Модель представления онтологии предметных областей на основе графовых баз данных // Вестник НГУ. Серия: Информационные технологии. 2022. Т. 20, № 4. С. 24-38. https://doi.org/10.25205/1818-7900-2022-20-4-24-38

Фатеев К.А.
Нижневартовский государственный университет
г. Нижневартовск, Россия

АРХИТЕКТУРНЫЕ ПОДХОДЫ К РАЗРАБОТКЕ DESKTOP-ПРИЛОЖЕНИЙ ДЛЯ УПРАВЛЕНИЯ МУЛЬТИМЕДИЙНЫМ КОНТЕНТОМ

Введение.

Современный этап развития информационных технологий характеризуется ростом объёмов мультимедийных данных, видео, аудио и графической информации. Активное распространение цифровых медиа в профессиональной и повседневной деятельности требует создания специализированных приложений для хранения, обработки и систематизации мультимедийных данных. Для эффективного управления таким контентом необходима разработка desktop-приложений, обеспечивающих высокую производительность, стабильность работы и имеющих прямой доступ к аппаратным ресурсам вычислительной системы.

Основным фактором эффективности таких систем является выбранный архитектурный подход. Архитектура программного обеспечения определяет взаимодействия между компонентами, влияет на масштабируемость, расширяемость, поддерживаемость и производительность этого приложения. При разработке мультимедийного контент-менеджера нужно учитывать особенности обработки большого объёма данных, требования к отзывчивости пользовательского интерфейса, возможность подключения внешних модулей и поддержки различных форматов файлов. Также необходимо обеспечить кроссплатформенную совместимость и устойчивую работу на разных операционных системах.

В настоящее время существует ряд архитектурных подходов к организации desktop-приложений, шаблон MVC(Model-View-Controller) Model отвечает за данные, методы работы с этими данными и структуру программы, View отвечает за визуализацию информации на уровне пользовательского интерфейса, Controller работает связующим между моделью и представлением, обрабатывает полученные действия, проверяет информацию и передает в модель, шаблон MVP(Model-View-Presenter) облегчает тестирование и изолирует View, отображает данные пользователю и отображает ввод, представление пассивно и не содержит бизнес-логики, лишь отображает данные полученные от Presenter и передает пользовательский ввод. Шаблон MVVM(Model-View-ViewModel) Model содержит основную бизнес-логику и данные. Model отвечает за хранение и валидацию данных, вычисления и операции обработки информации. View представляет отображение данных и обрабатывает взаимодействие с пользователем, но не содержит никакой логики обработки данных. В рамках архитектуры ViewModel отвечает за интеграцию модели и интерфейса, обеспечивая подготовку данных к отображению и передачу команд управления в View, и многослойные решения. Каждый из подходов обладает определёнными плюсами и ограничениями, которые по-разному проявляются при работе с мультимедийными данными. Выбор архитектуры напрямую влияет на качество и перспективу развития продукта.

Целью работы является сравнительный анализ архитектурных подходов к разработке desktop-приложения мультимедийного контент-менеджера с требованиями, производительности, масштабируемости, расширяемость. Для достижения цели необходимо решить задачи:

1. проанализировать требования к программным средствам управления мультимедийным контентом;
2. рассмотреть основные архитектурные подходы;
3. выявить плюсы и ограничения каждого подхода, при работе с мультимедийными данными;
4. сопоставить архитектурные решения по критериям производительности, расширяемости, кроссплатформенности;
5. определить наиболее рациональный архитектурный подход для разработки мультимедийного контент-менеджера.

Данный анализ направлен на обоснование выбора архитектурного решения при проектировании специализированного desktop-приложения для работы с мультимедийным контентом.

Требования к программным средствам управления мультимедийным контентом.

Среди основных категорий требований выделяются функциональные, интерфейсные, совместимости и безопасности.

Функциональные требования:

- возможность создавать и редактировать мультимедийный контент (аудио, видео, анимации и изображения);
- поддержка разных форматов файлов и кодексов чтобы обеспечить совместимость с разными источниками медиаданных;
- инструменты комбинирования различных видов медиа;
- функции воспроизведения мультимедийного контента;
- возможность экспортировать или импортировать проект.

Требования к интерфейсу:

- минимальный набор элементов управления;
- различимость элементов управления;
- доступность элементов управления.

Требования безопасности:

- механизм аутентификации пользователя;
- обеспечение целостности данных.

Требования совместимости:

- совместимость элементов управления;
- совместимость с инструментальной средой.

Основные архитектурные подходы.

Подход Model-View-Controller структурирует приложение на три взаимосвязанных модуля: модель, представление, контроллер (рис. 1). Их совместная работа позволяет разграничить обязанности компонентов и обеспечить модульность архитектуры.

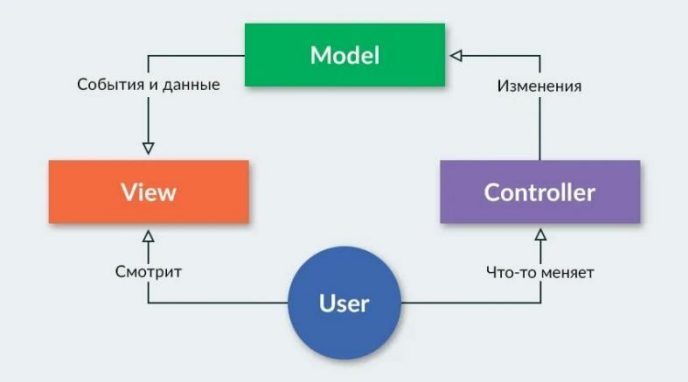


Рис. 1. Шаблон MVC

Модель самый независимый компонент, любые изменения контроллера или представления не влияют на неё. Представление визуализирует данные модели, адаптируя их для восприятия пользователем. Контроллер принимает запросы от пользователя и вносит изменения в модель [1].

Архитектурный подход Model-View-Presenter приложение разделено на три основных модуля: модель (Model) – представление (View) – презентатор (Presenter) (рис. 2). Работа этих модулей позволяет разделить логику приложения, сделать понятным и повысить тестируемость.

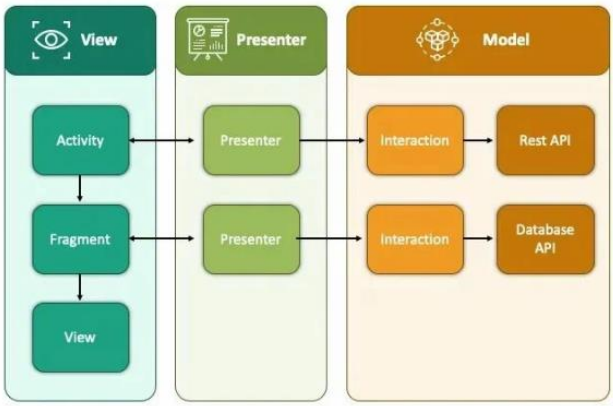


Рис. 2. Шаблон MVP

Модули Model и View соответствуют аналогичным компонентам архитектуры MVC. View отвечает за обновление интерфейса на основе информации, полученной от Presenter. Presenter обрабатывает действия пользователя полученные от View, и взаимодействует с моделью для корректировки данных [2].

Подход Model-View-ViewModel предполагает организацию приложения в виде трёх компонентов: модель (Model), представление (View), и компонента ViewModel (рис. 3). Такое разделение позволяет изолировать логику приложения от его визуальной части [3].

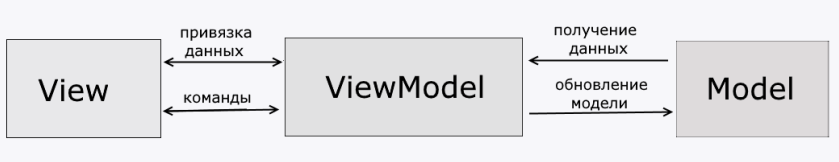


Рис. 3. Шаблон MVVM

Центральный компонент шаблона, выполняет роль посредника между Model и View. Он отвечает за преобразование данных модели в форму которую можно отобразить, а также представляет набор команд, доступных представлению. Шаблон содержит логику, специфичную для представления, но не содержит визуальных элементов.

Слоистая архитектура подход при котором функциональность разделяется на несколько уровней, чтобы обеспечить независимость и модульность компонентов, также организация кода для оптимизации разработки, масштабирования и поддержки приложения.

Структура многослойной архитектуры (рис. 4):

presentation layer представляет собой пользовательский интерфейс, отображающий данные и обработку действий пользователя;

application layer отвечает за координация действий, управление сенсами и организацию рабочих процессов;

business layer – реализация бизнес-правил, обработка данных и бизнес-процессов;

data access layer – взаимодействие с базами данных и другими источниками информации;

data layer – физическое хранение данных в базе данных, файловых системах.

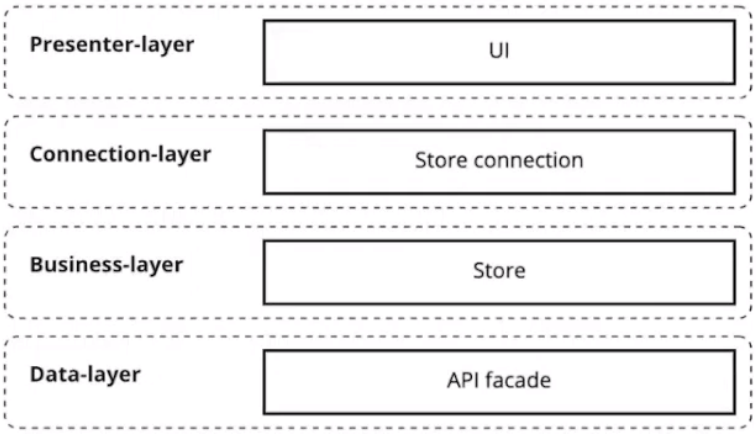


Рис. 4. Слоистая архитектура

Ключевая идея архитектуры состоит в организации обмена данными только между соседними слоями. Такой подход обеспечивает высокую степень независимости компонентов и упрощает их замену без нарушения работы системы в целом [4].

Плюсы и ограничения каждого подхода, при работе с мультимедийными данными.

Все плюсы и ограничения для каждого архитектурного подхода при работе с мультимедийными данными представлены в таблице 1.

Таблица 1

Архитектура	Плюсы	Ограничения
MVC	Разделение ответственности, параллельная разработка	Тесная связь между View и Controller, View напрямую образается к Model, Тослтый Controller
MVP	Полная развязка, тестируемость	Перегруженный Presenter, высокая частота вызова интерфейса
MVVM	Сбалансированная производительность, двухсторонняя связь данных	Трудности отладки, связывание не подходит для больших потоков данных
Слоистая	Четкое разделение логики, легкая замена технологий внизу	Копирование данных между слоями снижает производительность

Сопоставить архитектурные решения по критериям производительности, расширяемости, кроссплатформенности.

При разработке desktop-приложений можно использовать кроссплатформенные средства, так каждый разработчик сможет работать в удобной для себя операционной системе. Это обеспечивает определённую независимость от конерктной операционной системы, включая её жизненный цикл и потенциальные изменения условий использования [5].

Сравнительный анализ архитектурных подходов по производительности, расширяемости и кроссплатформенности представлен в таблице 2.

Таблица 2

Архитектура	Пояснения
MVC	Производительность: прямая связь меджду View и Controller дает низкую задержку, но View дергает Model, блокируя отрисовку. Расширяемость: добавление нового функционала часто ведет к разрастанию. Controller Кроссплатформенность: логика часто завязана на конкретный фреймворк.
MVP	Производительность: Presenter может управлять потоками данных. Расширяемость: легко добавлять новые View, переиспользуя Presenter. Кроссплатформенность: Бизнес-логика полностью оторвана от кода кнопок, её легко переносить.
MVVM	Производительность: механизм data binding съедает ресурсы процессора. Привязка больших данных противопоказана. Расширяемость: очень легко расширять интерфейсы и добавлять новые функции за счёт декларативной привязки. Кроссплатформенность: сильно зависит от фреймворка биндинга который есть везде.
Слоистая	Производительность: копирование данных между слоями создаёт оверхед. Для высоконагруженных систем требуется разрыв слоёв для скорости. Расширяемость: можно менять базы данных, добавлять новые модули обработки, не затрагивая интерфейс. Кроссплатформенность: нижние слои можно скомпилировать куда угодно, верхние переписать под платформу.

Определить наиболее рациональный архитектурный подход для разработки мультимедийного контент-менеджера.

Анализ показал, что ни одна из архитектур не является универсальной для задач управления мультимедийным контентом. Наиболее рациональным представляется комбинированный подход на основе MVVM и элементов слоистой архитектуры.

MVVM позволяет эффективно отделить пользовательский интерфейс от логики обработки данных, что особенно важно при работе с мультимедийными потоками и интерфейсами редактирования. Внедрение слоистой архитектуры внутри модели и бизнес-логики обеспечивает независимость модулей обработки медиафайлов, упрощает замену технологий хранения.

Оптимальной стратегией является проектирование мультимедийного контент-менеджера на основе MVVM для уровня представления и многослойной архитектуры для реализации бизнес-логики и доступа к данным.

Заключение.

По итогам работы были проанализированы требования к программным средствам управления мультимедийным контентом, рассмотрены архитектурные подходы MVC, MVP, MVVM и слоистая архитектура. Осуществлён анализ сравнения с критериями по производительности, работе расширяемости и кроссплатформенности разработки.

Самый простой в реализации шаблон MVC, но при масштабировании происходит усложнение логики контроллеров. MVP обеспечивает высокую тестируемость, однако может создавать избыточную нагрузку на Presenter. MVVM предоставляет гибкий механизм разделения логики и интерфейса, требует аккуратного управления механизмами привязки данных. Слоистая архитектура обеспечивает модульность и технологическую независимость однако в высоконагруженных системах, может снижать производительность из-за дополнительного уровня абстракции.

Наиболее рациональным для разработки desktop-приложения мультимедийного контент-менеджера является комбинированный подход архитектур, объединяя MVVM и многослойную организацию системы. Обеспечивая баланс между производительностью, устойчивостью программы и расширяемость.

Литература

1. Борисов И.Д., Бычкова Я.А., Гоголев А.М., Шубенкин Д.А. MVVM паттерн для пользовательского приложения. Калининград, 2022. С. 93-95.

2. Гончаров Е.И. Средства разработки современных кроссплатформенных десктоп приложений. Смоленск, 2023. С. 98-102.

3. Кушнир Н.В., Аравопуло Е.И. Обработка потока информации при использовании архитектуры веб-приложения с делением на backend и frontend. Краснодар, 2023. С. 118-122.

4. Новосельцева. Е.И., Юркова О.Н. MVC: Основы зарождения, модули и архитектура. Брянск, 2023. С. 33-35.

5. Шаповалова Э.В., Свищёва И.В. Сравнительный анализ MVC, MVP, MVVM: Раскрытие архитектурных закономерностей. М., 2023. С. 222-231.

Шишкин Д.И.
Нижневартовский государственный университет
г. Нижневартовск, Россия

СОЗДАНИЕ ФАЙЛОВОГО СЕРВЕРА С СОБСТВЕННЫМ API

Начнем с того, что первая реализация файловых серверов появилась в начале 1980-х годов [3]. Причиной этому было то, что в те времена объем памяти был сильно ограничен. Конечно, была возможность увеличить объем с помощью дискет, но этого все равно было недостаточно, чтобы иметь необходимый перечень программ, которые использовались в повседневной жизни [5]. Именно поэтому стали разворачивать файловые серверы, на которых хранились готовые программы, которые можно было установить, предварительно освободив место.

Однако, в современном мире, в ходе роста объема данных [1], возникла необходимость в оптимизации передачи данных в локальной сети, так как посредник в виде переносного накопителя данных стал излишним [4].

Именно поэтому необходим такой легковесный инструмент, который позволит передавать данные, как в локальной, так и глобальной сети.

Рис. 1. Схема последовательности действий при передаче до и после [составлено автором]

Описание идеи. Перед описанием реализации стоит отметить некоторые важные моменты, которые лягут в основу нашей реализации. Начнем с того, что наш инструмент должен иметь шифрование, чтобы у злоумышленников не было возможности украсть наши данные. Также, необходимо сделать так, чтобы наш итоговый проект был кроссплатформенным – это избавит нас от проблем с написанием реализаций под разные операционные системы. Помимо этого, необходимо позаботиться о том, чтобы было небольшое потребление ресурсов, что позволит использовать его даже на достаточно старых устройствах. И наконец, самый главный пункт – нужно сделать инструмент, который будет представлять из себя один бинарный файл, и в зависимости от аргументов терминала/командной строки будет менять свое поведение, а именно – запуск сервера или запуск клиента.

Стек технологий. Теперь нужно разобраться в стеке технологий для написания нашего инструмента. Начнем с языка программирования. Мы выбрали Rust из-за его положительной черты в виде безопасной работы с памятью и скорости работы на уровне языка Си. Это позволяет нам не беспокоиться об утечках памяти [2].

Далее разберемся с библиотеками, которые используются в нашей реализации. Первая библиотека – это Tokio, она является одной из самых популярных Rust библиотек. Данная библиотека реализует асинхронную неблокирующую среду ввода/вывода [7], что необходимо для серверов, так как обработка запроса клиента не должна блокировать выполнение. В нашем случае, она используется для создания асинхронной задачи для каждого отдельного клиента, где происходит дальнейшая его обработка, и для неблокирующего чтения/записи файла.

Теперь переходим к самой важной библиотеке у сервера, а именно – s2n-quic. Она реализует транспортный протокол QUIC. Вкратце, это умная обертка над UDP сокетами, которая исправляет проблемы UDP, а именно – потеря, дупликация и неправильный порядок пакетов, а также добавляет шифрование TLS 1.3 [6]. Данный протокол стал новым стандартом передачи данных и является предпочтительным в нашем случае.

Протокол взаимодействия. Для нашей задачи нам было необходимо придумать простое бинарное взаимодействие, чтобы наш инструмент был простым и компактным. Мы пришли к такой структуре пакета.

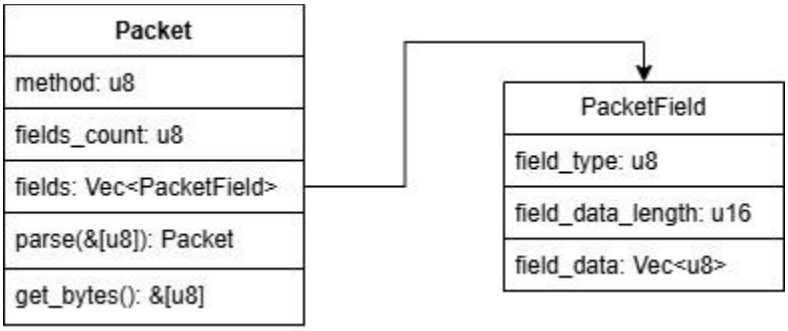


Рис. 2. Структура пакета [составлено автором]

Как вы могли заметить, пакет состоит из полей: method, fields_count и fields. Давайте подробнее пройдемся по полям. Поле method хранит метод, который запрашивает клиент, а в свою очередь сервер обрабатывает, а также это поле имеет три значения: Standard (только для сервера), Download и Upload. Поле fields_count хранит количество полей пакетов, полезно при десериализации. Поле fields хранит вектор структуры, хранящей такие поля: field_type, field_data_length, field_data.

Давайте также подробно пройдемся по этим полям. Поле field_type хранит тип поля и имеет четыре значения: Path, Status, FileSize, ErrorMsg. Поле field_data_length хранит количество байт, хранимых в поле field_data. Поле field_data хранит последовательность байт в зависимости от значения в поле field_type. Если у нас Path, то в поле будет храниться строка пути к файлу в байтовом представлении. Если у нас Status, то мы будем хранить одно из трех возможных значений: Ready, Error и Ok. Если у нас FileSize, то мы будем хранить 64-битное беззнаковое число в Big Endian (например, имеется число в шестнадцатеричной системе исчисления 0x1122334455667788, то в памяти это число будет записано так [0x11, 0x22, 0x33, 0x44, 0x55, 0x66, 0x77, 0x88]. Если у нас ErrorMsg, то в поле будет храниться текст ошибки в байтовом представлении.

Разобравшись со структурой пакета, можем перейти к описанию самого взаимодействия. Взаимодействие строится так: клиент отправляет запрос на скачивание/загрузку. В свою очередь сервер отправляет ответное сообщение со статусом Ready. После этого, в зависимости от метода, сервер или клиент начинает отправлять сырые куски файла, пока не закончится файл. После этого сервер шлет завершающее сообщение со статусом Ok. В случае, если где-то возникла ошибка, сервер отправляет сообщение со статусом Error и еще одним полем с типом ErrorMsg и текстом ошибки.

Давайте наглядно посмотрим, как выглядят сообщения у клиента и сервера (мы не будем показывать, как это выглядит в байтах, чтобы лишний раз вас не запутать). Начнем с сервера:

- Сообщение о готовности для загрузки (если файл найден): [Download, 2, [[Status, 1, Ready], [FileSize, 8, 12800326]]];
- Сообщение о готовности для выгрузки: [Upload, 1, [[Status, 1, Ready]]];
- Сообщение об успешном завершении: [Standard, 1, [[Status, 1, Ok]]];
- Сообщение об ошибке: [Standard, 2, [[Status, 1, Error], [ErrorMsg, 14, “File not found”]]].

Теперь продемонстрируем тоже самое, но для клиента:

- Запрос на загрузку: [Download, 1, [[Path, 17, “./files/video.mkv”]]];
- Запрос на выгрузку (если файл найден): [Upload, 2, [[Path, 16, “./files/game.exe”], [FileSize, 8, 1378500]]].

Реализация. Разобравшись со всеми тонкостями, можем перейти к главной части данной статьи. Чтобы у нас все заработало, сперва необходимо смотреть на аргументы терминала/командной строки и в зависимости от них запускать клиент или сервер. Вот примеры запуска инструмента с разными аргументами (для удобства назовем наш исполняемый файл – qfile):

- qfile server – запускает сервер со стандартным портом 2222;
- qfile server --port <PORT> – запускает сервер с портом PORT;
- qfile download <ADDR> <SOURCE> <DEST> – запускает клиент, который скачивает SOURCE с сервера ADDR и записывает его в DEST;
- qfile upload <ADDR> <SOURCE> <DEST> – запускает клиент, который загружает SOURCE на сервер ADDR в DEST.

Также укажем, что клиенты в нашей реализации не держат соединение, ибо они выполняют действие и завершают свое выполнение, а сервер напротив, будет выполняться до тех пор, пока пользователь не прервет его выполнение.

После этого начинаем выполнение клиента или сервера, в зависимости от аргументов.

Экспериментальная оценка. Для большей информативности данной статьи, мы провели эксперимент в условиях локальной сети, так как этот инструмент в первую очередь преподносится для этого. Сперва стоит описать, благодаря какой технике были проведены измерения.

Сервером выступил стационарный компьютер. Ознакомим только с теми компонентами, которые используются инструментом. Процессор – Intel Core i5-12400F, ОЗУ – 32ГБ DDR4

3200МГц, Диск – M.2 NVME SSD (скорость чтения/записи – 2500МБ/с), Сетевой контроллер – Realtek® RTL8111H Gigabit LAN controller.

В качестве клиента использовался Apple MacBook Air M2 2022 16GB ОЗУ.

Роутер – Mercusys AC1200G. Все, что нам нужно знать, так это то, что его скорость в локальной сети около 60-70 МБ/с.

Разобравшись с техникой, можем перейти к самим результатам. Начнем с того, что инструмент потребляет около 22 МБ ОЗУ, около 7% загруженности процессора, около 180МБ/с скорости записи/чтения и в среднем 45МБ/с скорости соединения.

Теперь перейдем непосредственно к скорости передачи файлов. Файл размером в 55МБ передается в среднем за 2 секунды. А файл в 5ГБ передается в среднем за 130 секунд.

Проверялись как скачивание, так и загрузка. Результаты практически равнозначны, за исключением ситуаций, когда погрешность может быть в районе от 1 до 5%.

Улучшения, которые можно сделать в будущем. Данная реализация не полная, так как является прототипом, которую в будущем можно улучшить, но в нее заложена архитектура, которая без особых сложностей может быть дополнена. Сейчас мы приведем несколько вещей, которые можно улучшить/дополнить:

- Добавить возможность скачивания/загрузки нескольких файлов, предварительно сжав в архив tar, rar или 7z;
- Добавить метод списка файлов, чтобы знать, какие файлы имеются на сервере;
- Сделать клиент, который будет держать соединение до тех пор, пока пользователь сам не вызовет завершение работы;
- Ограничить скачивание/загрузку до определенной директории;
- Добавить авторизацию и аутентификацию перед скачиванием/загрузкой защищенных файлов, чтобы их нельзя было скачать посторонним;
- Добавить графический интерфейс;
- Добавить возможность подключаться в качестве клиента через браузер.

Вы можете дополнительно придумать еще кучу разных улучшений. Это лично те, которые мы посчитали уместными.

Заключение. Разработанный прототип файлового сервера подтвердил принципиальную возможность создания компактного решения для прямого обмена данными, призванного заменить традиционные переносные носители данных. В отличие от зрелых и многократно оптимизированных FTP-серверов, представленная реализация ожидаемо уступает в скорости передачи, что объясняется её прототипным характером и отсутствием тонкой настройки на данном этапе. Тем не менее, полученная кодовая база закладывает основу для дальнейшего развития.

Ключевое преимущество разработанного сервера заключается в его модульности и компактности, достигаемой за счёт использования современного стека технологий. Это позволяет рассматривать его не как готовый коммерческий продукт, а как основу для построения специализированных решений. В частности, направлением будущей оптимизации является рефакторинг используемых библиотек: замена универсальных,

многофункциональных библиотек на более узкоспециализированные или самописные аналоги, что позволит снизить накладные расходы и приблизить производительность к эталонным показателям.

Таким образом, даже в текущем виде предложенная система демонстрирует жизнеспособность подхода и служит отправной точкой для создания высокопроизводительных, легковесных файловых сервисов нового поколения.

Литература

1. Abdalla H.B. A brief survey on big data: technologies, terminologies and data-intensive applications // Journal of Big Data. 2022. Vol. 9. Article 107. P. 1-36. https://doi.org/10.1186/s40537-022-00659-3

2. Bugden W., Alahmar A. Rust: The Programming Language for Safety and Performance // 2nd International Graduate Studies Congress (IGSCONG'22). 2022. P. 1-9. https://doi.org/10.48550/arXiv.2206.05503

3. Engineer S., Engineer A. The Structure and Interpretation of the SMB Protocol. Singapore: Springer Singapore, 2026. XXVI, 366 p. https://doi.org/10.1007/978-981-10-7805-7

4. Lin W., Xu M., He J., Zhang W. Privacy, security and resilience in mobile healthcare applications // Enterprise Information Systems. 2023. Vol. 17, No. 3. P. 345-359. https://doi.org/10.1080/17517575.2021.1939896

5. Raghunathan K.R. History of Microcontrollers: First 50 Years // IEEE Micro. 2021. Vol. 41, No. 6. P. 97-104. https://doi.org/10.1109/MM.2021.3114754

6. RFC 9000. QUIC: A UDP-Based Multiplexed and Secure Transport. Internet Engineering Task Force (IETF), 2021. https://datatracker.ietf.org/doc/html/rfc9000

7. Wang T.-Y., Wang S.-H., Tu C.-H., Liang W.-Y. CAT: Context Aware Tracing for Rust Asynchronous Programs // Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing (SAC '23). New York, NY, USA: Association for Computing Machinery, 2023. P. 483-492. https://doi.org/10.1145/3555776.3577669

ИСКУСТВЕННЫЙ ИНТЕЛЛЕКТ. РОБОТОТЕХНИКА. БЕСПИЛОТНЫЕ ЛЕТАТЕЛЬНЫЕ АППАРАТЫ. ЗАЩИТА ИНФОРМАЦИИ

Абрамова А.А.
Дальневосточный государственный университет путей сообщения
г. Хабаровск, Россия

КВАНТОВЫЕ ВЫЧИСЛЕНИЯ: СОВРЕМЕННОЕ СОСТОЯНИЕ И ПЕРСПЕКТИВЫ РАЗВИТИЯ

Классическая архитектура вычислительных систем, основанная на бинарной логике, по мере усложнения задач сталкивается с возрастающими технологическими и ресурсными трудностями, что затрудняет дальнейшее масштабирование вычислений, что стимулирует развитие альтернативных моделей, среди которых особое место занимают квантовые вычисления.

Квантовые вычисления - передовая область компьютерной науки, которая использует квантовую теорию для революционного решения сложных вычислительных задач, основанную на принципах квантовой механики, в особенности квантовую запутанность и интерференцию. В отличие от классического бита, принимающего значение 0 или 1, квантовый бит (кубит) может находиться в суперпозиции состояний, он использует субатомные частицы – такие как электроны или фотоны – для одновременного существования в нескольких состояниях, что значительно расширяет вычислительные возможности.

Теоретические исследования показывают, что квантовые алгоритмы способны значительно превосходить классические методы при факторизации больших чисел, поиске в неструктурированных массивах данных и моделировании квантовых систем [1]. Учитывая быстрый прогресс в разработке квантовых технологий и их потенциальное влияние на вычислительные методы, анализ уровня развития квантовых вычислений, существующих архитектурных решений и направлений дальнейший исследований приобретает особую значимость.

Целью работы является анализ текущего состояния квантовых вычислений и определение дальнейших перспектив их развития в контексте решения прикладных и теоретических задач.

Концепция квантовых вычислений сформировалась во второй половине XX века на пересечении квантовой механики и теории алгоритмов. Классическая модель вычислений, разработанная Аланом Тьюрингом, не учитывала физическую природу информационных процессов. Но в 1981 году Ричард Фейнман обосновал необходимость использования квантовых систем для эффективного моделирования, а в 1985 году Дэвид Дойч предложил формальную модель универсального квантового компьютера. Существенный импульс развитию направления придали алгоритм факторизации Питера Шора и алгоритм поиска Лова Гровера, продемонстрировавшие потенциальное преимущество квантовой модели над классической [2]. При этом ключевым элементом квантовой вычислительной системы стала новая единица информации – квантовый бит, или кубит.

Классический бит принимает одно из двух возможных значений: 0 или 1. Квантовый бит (кубит) описывает состояние двухуровневой квантовой системы и допускает существование в линейной суперпозиции базисных состояний |0⟩ и |1⟩. Состояние кубита выражает вектор в двумерном гильбертовом пространстве определяется по формуле:

$$|\psi\rangle=\alpha|0\rangle+\beta|1\rangle,$$

где α и β – коэффициенты α и β являются комплексными числами и удовлетворяют условию нормировки |α|^2+|β|^2=1 [3].

Данная особенность обусловливает вероятностный характер квантовых вычислений. Квантовый алгоритм не выдаёт детерминированный результат при однократном запуске, а формирует распределение вероятностей возможных исходов. Для получения корректного решения проводят серию повторных измерений и анализируют статистику результатов. При измерении кубита система переходит в одно из базисных состояний с вероятностью, определяемой квадратом модуля соответствующего амплитудного коэффициента.

Квантовая модель обработки информации опирается на ключевые положения квантовой механики, среди которых центральное место занимают суперпозиция, запутанность и интерференция.

Принцип суперпозиции – это фундаментальный принцип квантовой механики, позволяющий частицам существовать в нескольких состояниях одновременно. В контексте квантовых вычислений он относится к способности кубитов находиться в состоянии 0, 1 или суперпозиции обоих состояний. Это позволяет обрабатывать большее количество информации по сравнению с классическими битами.

Квантовая запутанность отражает наличие нелокальных корреляций между состояниями кубитов. Это явление, при котором две или более частицы связываются так, что состояние одной напрямую зависит от состояния другой вне зависимости от расстояния между ними. В запутанной системе состояние отдельных элементов нельзя описать независимо друг от друга, поэтому корректное описание требует рассмотрения всей совокупности кубитов как единого объекта, что позволяет квантовым компьютерам выполнять сложные вычисления более эффективно и точно.

Интерференция играет ключевую роль в реализации квантовых алгоритмов. При формировании коллективной суперпозиции кубиты описываются амплитудами вероятности, каждая из которых соответствует определённому вычислительному результату. Квантовое состояние приобретает волновую структуру, где амплитуды определяют вероятность регистрации того или иного исхода при измерении. Конструктивная интерференция увеличивает вероятность отдельных результатов, тогда как деструктивная подавляет другие. Управляя интерференционной картиной, квантовый алгоритм перераспределяет вероятности в пользу корректного решения задачи.

Квантовые алгоритмы

Для оценки преимуществ квантовых алгоритмов важно учитывать сложность задач в классической модели вычислений. Теория вычислительной сложности выделяет ключевые классы: P (полиномиально решаемые задачи), NP (решения которых проверяются за

полиномиальное время), NP-complete (наиболее сложные задачи NP) и BPP (вероятностные задачи с ограниченной ошибкой).

С появлением квантовых вычислений возникла необходимость появления нового класса. BQP–класс задач принятия решений, которые квантовый алгоритм решает за полиномиальное время с ограниченной вероятностью ошибки. По своей структуре BQP представляет квантовый аналог класса BPP, однако использует принципы квантовой механики для перераспределения вероятностей и достижения вычислительного ускорения. Этот класс расширяет возможности классической модели и демонстрирует квантовое преимущество. Например, алгоритм Шора решает факторизацию целых чисел с экспоненциальным ускорением по сравнению с классическими методами, а алгоритм Гровера обеспечивает квадратичное ускорение поиска в неструктурированной базе данных.

Алгоритм Шора

Значительный вклад в развитие квантовых вычислений внёс Питер Шор, предложивший в 1994 году алгоритм факторизации больших целых чисел за полиномиальное время. Алгоритм основан на сведении задачи факторизации к задаче нахождения периода функции с последующим применением квантового преобразования Фурье. Экспоненциальное ускорение по сравнению с известными классическими методами продемонстрировало принципиальное преимущество квантовой модели для задач, лежащих в основе криптографических систем с открытым ключом [4].

Алгоритм Гровера

В 1996 году Лов Гровер разработал алгоритм поиска элемента в неструктурированной базе данных. Классический перебор требует в среднем O(N) операций, тогда как квантовый алгоритм достигает решения за $O(\sqrt{N})$ шагов. Ускорение достигается посредством итеративного применения оператора отражения, усиливающего амплитуду целевого состояния. Алгоритм Гровера не обеспечивает экспоненциального выигрыша, однако демонстрирует универсальный механизм квадратичного ускорения для широкого класса поисковых задач [4].

Современное состояние технологий

По мере перехода квантовых технологий от теоретических моделей к экспериментально реализуемым устройствам их развитие стало ориентироваться на повышение практической эффективности и расширение возможностей применения.

Вопрос о практической значимости квантовых вычислений неизбежно сводится к сравнению их возможностей с ресурсами классических суперкомпьютеров. До определённого этапа в развитии квантовых устройств преобладал экспериментальный характер, их результаты оставались сопоставимыми с классическим моделированием. Однако с увеличением числа управляемых кубитов и усложнением квантовых схем возникла необходимость определить тот момент, когда квантовый компьютер начинает решать задачи, которые классические машины не могут эффективно выполнить. В рамках сопоставления вычислительных возможностей классических и квантовых моделей вводится понятие «квантового превосходства» [1]. Термин «квантовое превосходство» обозначает выполнение

квантовым компьютером задачи, которую невозможно эффективно решить на классическом суперкомпьютере за разумное время. В 2019 году компания Google заявила о достижении такого состояния, продемонстрировав решение специфической задачи моделирования случайной квантовой схемы быстрее, чем классические системы.

В настоящее время такие компании, как IBM, Google, Microsoft, D-Wave, Rigetti Computing и многие другие, создают реальное квантовое оборудование. Инженеры регулярно создают все более мощные сверхпроводящие квантовые процессоры, добиваясь важных успехов в разработке программного обеспечения.

Современные прототипы квантовых компьютеров создаются на нескольких физических принципах. Сверхпроводниковые кубиты, разрабатываемые компаниями IBM и Google, обеспечивают высокую скорость операций, которые выполняются на квантовых компьютерах. Кроме разработки квантовых процессоров, компания IBM активно развивает инфраструктуру удалённого доступа к ним. В результате квантовые вычисления стали доступны не только в лабораторных условиях, но и через облачные сервисы. Так, платформа IBM Quantum Experience позволяет исследователям и студентам запускать квантовые алгоритмы и проводить эксперименты без необходимости создания собственной экспериментальной базы, что значительно расширяет возможности образовательных и научных проектов.

Ионные ловушки, над которыми активно работают Росатом и ФИАН формируют кубиты на основе захваченных ионов, управляемых лазерными импульсами, что позволяет достичь длительного времени когерентности.

Проблемы и ограничения

Несмотря на значительный потенциал квантовых вычислений, их практическая реализация сталкивается с большим количеством ограничений. Например, для функционирования кубиты необходимо охладить до температуры, лишь на долю градуса выше абсолютного нуля, что холоднее, чем в открытом космосе. Высокомощное охлаждение, необходимое для создания состояния когерентности для функциональных кубитов, является серьезной проблемой для квантовых вычислений. Так, даже незначительное внешнее воздействие способно вызвать коллапс системы, функционирующей на основе принципов квантовой механики. В процессе выполнения вычислений такие устройства требуют практически полной изоляции от тепловых, электромагнитных и иных внешних шумов, способных нарушить когерентность состояний. Тем не менее были достигнуты определенные успехи в использовании кубитов в сильных магнитных полях, но короткий срок жизни кубитов остается одной из главных нерешенных проблем квантовых вычислений.

Кроме того, особенность кубитов как носителей информации усугубляет проблему надёжности вычислений. В отличие от битов, они не допускают прямого применения стандартных методов коррекции ошибок. Любая ошибка в ходе вычислений может нарушить когерентность системы и привести к полной потере корректности результата. Даже извлечение выходных данных после завершения вычислительного процесса связано с риском искажения информации, что делает разработку эффективных схем защиты и коррекции

ошибок критически важной задачей для масштабирования современных вычислительных систем.

При рассмотрении перспектив массового внедрения вычислительных систем, основанных на квантовых принципах, возникает ещё одна потенциальная трудность –дефицит специалистов, обладающих необходимой квалификацией для проектирования, эксплуатации и оптимизации таких устройств. Навыки в области квантовых исследований носят узкоспециализированный характер, и большинство специалистов, обладающих ими, получили свой опыт, работая в лабораториях и академических кругах. В результате, в растущем коммерческом секторе не хватает работников с квантовыми навыками.

Перспективы развития

Современные квантовые вычисления уже демонстрируют потенциал значительного повышения вычислительной эффективности в ряде прикладных областей, что подтверждает переход технологий из экспериментальной стадии к практическому использованию в будущем [5].

Хотя кубитные процессоры, используемые в квантовых вычислениях, потенциально могут значительно превзойти по производительности битовые процессоры, современные квантовые процессоры могут поддерживать лишь небольшое количество потенциальных кубитов. По мере развития исследований IBM планирует к 2029 году представить квантовую систему с 200 логическими кубитами, способную выполнять 100 миллионов квантовых вентилей. Долгосрочная цель компании – создать систему с 2000 логическими кубитами, способную выполнять 1 миллиард вентилей к 2033 году.

Квантовые компьютеры способны оптимизировать работу искусственного интеллекта и машинного обучения при обработке многомерных данных. Так, развиваются подходы квантового машинного обучения, основанные на квантовых нейронных сетях. Такие сети используют суперпозицию и запутанность, вместо традиционного бинарного представления, что потенциально ускоряет обучение моделей и повышает эффективность обработки данных.

Квантовые вычисления также активно применяются в оптимизации бизнес-процессов. Разработки позволяют совершенствовать процессы исследований и разработок, заметно повышая эффективность производственных цепочек и логистики. Компании начинают внедрять квантовые подходы для оптимизации маршрутов поставок, планирования загрузки грузов, распределения ресурсов и других критически важных процессов, в работе с которыми традиционные методы сталкиваются с ограничениями по времени и объёму.

Особое внимание в перспективах развития квантовых технологий уделяется квантовой криптографии, которая предлагает совершенно другие уровни безопасности. Квантовое распределение ключей обладает уникальными свойствами такими как невозможность копирования, так как квантовое состояние не может быть заранее известно, то это обеспечивает такую защиту информации, которая может противостоять различным кибератакам.

Квантовые вычисления представляют собой перспективное направление, объединяющее принципы квантовой механики с вычислительными технологиями. Системы, построенные на

кубитах в состоянии суперпозиции и квантовой запутанности могут решать задачи, которые недоступны классическим компьютерам, что подтверждается алгоритмами Шора и Гровера, а также экспериментами, демонстрирующими квантовое превосходство.

Внедрение квантовых вычислений на практике уже частично осуществляется через облачные платформы, такие как IBM Quantum Experience, что позволяет исследователям и студентам тестировать квантовые алгоритмы без необходимости в создании собственной лаборатории. Тем не менее широкое промышленное применение часто граничит с высокой стоимостью, общей технической нестабильностью, что требует дальнейшего более детального исследования и поиска новых решений.

В общем и целом, несмотря на существующие ограничения, квантовые вычисления продолжают активно развиваться и имеют потенциал для значительного влияния на будущее вычислительной техники и информационных технологий.

Литература

1. Андреев С.В., Килин С.Я. Квантовые алгоритмы и их физическая реализация // Успехи физических наук. 2022. Т. 192. № 4. С. 387-410.

2. Двуреченская В.Д. Квантовые алгоритмы: алгоритм Шора и алгоритм Гровера // Инновационная наука. 2025. № 6-1.С. 74-76.

3. Кузнецов Н.А. Квантовое превосходство: современное состояние исследований // Вестник Московского университета. Серия 3: Физика. Астрономия. 2021. № 5. С. 70-78.

4. Фёдоров А.К., Киктенко Е.О., Колачевский Н.Н. Вычислимое и невычислимое в квантовом мире: утверждения и гипотезы // Успехи физических наук. 2024. Т. 194. № 9. С. 960-966.

5. Шут А.А. От автоматизации к цифровизации: применение квантовых алгоритмов в задачах планирования и оптимизации // Программирование. 2025. № 3. С. 12-23.

Бармин А.Ю.
Нижневартовский государственный университет
г. Нижневартовск, Россия

СРАВНЕНИЕ БЕЗОПАСНОСТИ И ВЫЧИСЛИТЕЛЬНОЙ СЛОЖНОСТИ ИНТЕРАКТИВНЫХ ZKP И НЕИНТЕРАКТИВНЫХ ZKP, ПОЛУЧЕННЫХ С ИСПОЛЬЗОВАНИЕМ ПРЕОБРАЗОВАНИЯ ФИАТА-ШАМИРА

Современную информационную инфраструктуру можно охарактеризовать высокой степенью цифровой интеграции и постоянно растущими требованиями к защите данных. В связи с этим особую значимость приобретают криптографические механизмы защиты, которые обеспечивают доказуемую безопасность без раскрытия конфиденциальной информации. В статье рассматривается механизм доказательства с нулевым разглашением [1, с. 138] (Zero-Knowledge Proofs, ZKP), позволяющий одной стороне убедить другую в истинности утверждений без раскрытия дополнительной информации. Данный подход применяется в системах электронной идентификации, распределённых системах, электронном голосовании, а также в протоколах аутентификации и верифицируемых вычислений.

В рамках доказательств с нулевым разглашением различают интерактивную и неинтерактивную модели. Несмотря на широкое распространение обеих моделей, переход от интерактивных схем к неинтерактивным с использованием преобразования Фиата-Шамира сопровождается изменением модели безопасности, переходом к модели случайного оракула и пересмотром криптографических допущений. Целью данной работы является сравнительный анализ интерактивных и неинтерактивных доказательств с нулевым разглашением с точки зрения модели безопасности и вычислительной сложности. В работе особое внимание уделяется анализу влияния преобразования Фиата-Шамира на сохранение свойств полноты, корректности (soundness) и нулевого разглашения (zero-knowledge).

Для выполнения поставленной цели предполагается решение следующих задач:

1. Изучить теоретические основы ZKP, включая формальные свойства полноты, корректности и нулевого разглашения.
2. Рассмотреть структуру и особенности интерактивных доказательств, в частности Σ-протоколов, определить их модель безопасности и используемые криптографические предположения.
3. Исследовать механизм получения неинтерактивных доказательств с использованием преобразования Фиата-Шамира, а также особенности модели случайного оракула.
4. Выявить ключевые различия между интерактивными и неинтерактивными схемами с точки зрения формальной безопасности и доверительных предпосылок.
5. Сопоставить вычислительные и коммуникационные характеристики рассматриваемых моделей, включая количество раундов взаимодействия, размер доказательства и сложность проверки.

Zero-Knowledge Proof (ZKP) - это криптографический протокол, который позволяет одной стороне - доказывающему (Prover) убедить другую сторону - проверяющего (Verifier) в истинности некоторого утверждения без раскрытия каких-либо дополнительных данных, лежащих в основе этого утверждения. При этом проверяющий получает лишь уверенность в

том, что утверждение истинно, но не получает никакой дополнительной информации о самом секретном содержании доказательства или скрытой части данных [4, с. 1]. Формально ZKP определяется относительно языка $L \subseteq \{0,1\}*$ и бинарного отношения R(x,w), где x - публичное утверждение, а w - секрет(свидетельство), такое что R(x,w) = 1.

Протокол доказательства знания предполагает, что доказывающая сторона демонстрирует знание некоторого значения w, удовлетворяющего отношению R(x,w), не раскрывая самого значения. В зависимости от структуры взаимодействия различают интерактивные и неинтерактивные схемы, однако их формальные свойства определяются единым теоретическим аппаратом.

Классическое определение доказательства с нулевым разглашением базируется на трёх фундаментальных свойствах:

1. Полнота (completeness):

Если утверждение истинно и обе стороны следуют протоколу корректно, то проверяющий всегда принимает доказательство от честного доказывающего. Это обеспечивает доверие в случаях, когда доказывающая сторона действительно обладает нужной информацией.

2. Корректность (soundness):

Если утверждение ложно, то никакая нечестная стратегия доказывающего не может убедить честного проверяющего в его истинности, за исключением пренебрежимо малой вероятности. Это препятствует успешному обману со стороны доказывающего.

3. Нулевое разглашение (zero-knowledge):

Проверяющий не получает никакой дополнительной информации о скрытом секрете сверх факта истинности утверждения. Формально это означает, что должен существовать эффективный алгоритм-симулятор, который может воспроизвести разговор между доказывающим и проверяющим без знания самого секрета [4, с. 1].

Интерактивные ZKP реализуются в виде протоколов, в которых проверяющий формирует случайный вызов, а доказывающий отвечает на него с использованием секретного свидетельства. Наиболее распространённым типом интерактивных протоколов являются Σ-протоколы, относящиеся к классу публично-монетных (public-coin) интерактивных доказательств, в которых сообщение проверяющего представляет собой случайный вызов. Σ-протокол представляет собой схему взаимодействия из трёх шагов:

1. Доказывающий формирует сообщение-обязательство a (commitment) на основе случайного значения и отправляет его проверяющему.

2. Проверяющий выбирает случайный вызов e из заданного пространства вызовов.

3. Доказывающий вычисляет ответ z на основе вызова e, случайного значения и секретного свидетельства w.

Проверяющий принимает доказательство, если выполняется проверочное соотношение, определённое спецификацией протокола. При выполнении соответствующих формальных условий такая структура обеспечивает свойства полноты, корректности и нулевого разглашения.

Для Σ-протоколов характерны дополнительные свойства:

- Special soundness - существование эффективного алгоритма-извлекателя, способного восстановить секретное свидетельство w при наличии двух корректных ответов z_1, z_2 на различные вызовы $e_1 \neq e_2$ для одного и того же обязательства a;
- Honest-verifier zero-knowledge (HVZK) – существование эффективного алгоритма-симулятора, способного сгенерировать распределение сообщений, неотличимое от реального диалога, при условии честного поведения проверяющего.

Интерактивные Σ-протоколы, как правило, доказываются безопасными в стандартной модели без обращения к эвристическим предположениям. Однако необходимость многократного обмена сообщениями и присутствия проверяющего в процессе взаимодействия ограничивает их применение в распределённых и асинхронных системах.

Данное обстоятельство приводит к использованию методов устранения интерактивности, наиболее известным из которых является преобразование Фиата-Шамира, позволяющее заменить случайный вызов проверяющего значением криптографической хэш-функции. Анализ влияния данного преобразования на безопасность и вычислительную сложность является предметом дальнейшего рассмотрения.

Преобразование Фиата-Шамира представляет собой криптографический метод, предназначенный для устранения интерактивности из публично-монетных (public-coin) интерактивных доказательств, в том числе Σ-протоколов [3, с. 2]. Идея преобразования заключается в замене случайного вызова, генерируемого проверяющим в интерактивной модели, на значение, вычисляемое детерминированным образом с использованием криптографической хэш-функции.

Пусть интерактивный Σ-протокол определяется тройкой сообщений (a, e, z), где a – обязательство (commitment), e – случайный вызов проверяющего, а z – ответ доказывающего. В неинтерактивной версии вызов определяется следующим образом:

$$e = H(a, x)$$

где H – криптографическая хэш-функция, а x – публичное утверждение. Таким образом, вызов перестаёт формироваться проверяющим и вычисляется самим доказывающим на основе публичных данных и обязательства. Полученное доказательство имеет вид пары (a, z), а проверяющий воспроизводит значение вызова, вычисляя ту же хэш-функцию [3, с. 2].

Согласно современным исследованиям, преобразование Фиата-Шамира позволяет получать неинтерактивные доказательства из публично-монетных интерактивных схем с сохранением основных свойств безопасности в модели случайного оракула. В данной модели хэш-функция рассматривается как идеальный случайный оракул, возвращающий независимое случайное значение для каждого нового запроса. Это означает, что корректность анализа безопасности неинтерактивной схемы опирается на допущение о случайном характере хэш-функции.

Полнота протокола при применении преобразования сохраняется, поскольку корректно сформированный ответ z удовлетворяет проверочному соотношению так же, как и в интерактивной версии. Свойство корректности (soundness) обеспечивается благодаря тому,

что исходный Σ-протокол обладает свойством special soundness, а модель случайного оракула гарантирует непредсказуемость вызова e [3, с. 2]. Аналогично, свойство нулевого разглашения сохраняется в вычислительном смысле при наличии соответствующего симулятора, адаптированного к работе в модели случайного оракула [3, с. 2].

Отдельно подчёркивается, что преобразование Фиата–Шамира применимо не только к трёхшаговым Σ-протоколам, но и к более общим интерактивным доказательствам с несколькими раундами взаимодействия [3, с. 2], что расширяет область его применения в современных криптографических конструкциях.

В то же время переход от стандартной модели к модели случайного оракула означает изменение используемых предпосылок безопасности. Современные исследования, включая анализ конкретных SNARK-конструкций и схем подписи, указывают на необходимость аккуратного переноса доказательств безопасности из ROM в более реалистичные модели и учёта дополнительных факторов, таких как особенности реализации и возможные расширенные модели атак [2, с. 553-554]. Это подчёркивает, что практическое применение преобразования Фиата–Шамира требует анализа в контексте конкретной схемы и используемых криптографических примитивов.

Таким образом, преобразование Фиата-Шамира позволяет заменить интерактивный этап генерации случайного вызова вычислением хэш-функции, что устраняет необходимость многократного обмена сообщениями и делает возможной оффлайн-верификацию доказательства. Одновременно с этим происходит переход к иной модели безопасности, что оказывает влияние на формальные гарантии протокола и его вычислительные характеристики. В следующем разделе будет проведён сравнительный анализ интерактивной и неинтерактивной моделей с учётом указанных различий.

Сравнение модели безопасности

Интерактивные Σ-протоколы обычно анализируются в стандартной модели криптографии, где формальные гарантии soundness и zero-knowledge выводятся без обращения к идеализациям хэш-функций. Преобразование Фиата–Шамира позволяет получить неинтерактивную схему, но формальные доказательства безопасности такой схемы чаще всего строятся в ROM; это означает, что доказательство переносится в модель, где хэш-функция рассматривается как идеализованный оракул. Современные работы отмечают, что при рассмотрении практических схем (включая SNARK-конструкции) и при анализе устойчивости к новым видам атак (например, квантовым суперпозиционным запросам) возникают дополнительные вопросы, требующие отдельного исследования и осторожности при обобщении доказательств из ROM на «реальные» хэш-функции и стандартную модель. Практические исследования подтверждают полезность трансформации, но указывают на необходимость дополнительного анализа безопасности для конкретных применений [4, с. 3; 5, с. 1-2].

Сохранение фундаментальных свойств

Полнота.

В обеих моделях полнота сохраняется: честный доказывающий способен убедить честного проверяющего в истинности утверждения. Замена вызова на значение хэш-функции не влияет на корректность вычислений при честном поведении сторон.

Корректность (soundness).

В интерактивной модели корректность достигается за счёт непредсказуемости вызова проверяющего. После применения преобразования Фиата-Шамира корректность сохраняется в модели случайного оракула при условии корректного моделирования хэш-функции как случайного источника [3, с. 2].

Нулевое разглашение (zero-knowledge).

Интерактивные Σ-протоколы обладают свойством honest-verifier zero-knowledge в стандартной модели. В неинтерактивной версии нулевое разглашение сохраняется в вычислительном смысле при анализе в ROM, что подтверждается современными исследованиями безопасности преобразования Фиата-Шамира [3, с. 2].

Тем самым фундаментальные свойства доказательства формально сохраняются, однако их обоснование переносится в более сильную модель предположений.

Коммуникационная сложность

Интерактивный Σ-протокол требует обмена как минимум трёх сообщений: обязательство a, вызов e, ответ z. При повторении протокола для усиления корректности коммуникационная нагрузка возрастает.

В неинтерактивной версии доказательство представляется в виде одного сообщения (a, z), а проверяющий самостоятельно вычисляет вызов через хэш-функцию. Это устраняет необходимость многократного обмена сообщениями и делает схему пригодной для асинхронных и распределённых систем [2, с. 553-554].

Вычислительная сложность

В интерактивной модели вычислительная нагрузка распределена между сторонами и сопровождается необходимостью сетевого взаимодействия. В неинтерактивной версии добавляется операция вычисления хэш-функции, однако при этом исключается необходимость онлайн-взаимодействия.

Современные исследования отмечают, что устранение интерактивности повышает практическую применимость схем в распределённых средах при сохранении сопоставимых вычислительных затрат [2, с. 553-554].

Размер доказательства в обеих моделях остаётся сопоставимым, поскольку структура обязательства и ответа сохраняется.

Итоговое сопоставление

Проведённый анализ позволяет сделать следующие выводы:

1. Интерактивные Σ-протоколы обеспечивают строгие гарантии безопасности в стандартной модели.

2. Преобразование Фиата-Шамира устраняет интерактивность и снижает коммуникационную сложность, однако переносит доказательство безопасности в модель случайного оракула [3, с. 2].

3. Фундаментальные свойства (полнота, корректность, нулевое разглашение) сохраняются, но их доказательство требует дополнительных предположений ROM.

4. Неинтерактивные схемы обладают преимуществом в асинхронных и распределённых системах благодаря сокращению коммуникационных издержек.

Таким образом, выбор между интерактивной и неинтерактивной моделью определяется требованиями к строгости модели безопасности и условиями применения протокола.

Результаты проведённого анализа обобщены в таблице.

Таблица

Сравнительный анализ интерактивных ZKP и неинтерактивных ZKP, полученных с использованием преобразования Фиата-Шамира

Параметр	Σ-протокол (интерактивный)	Fiat-Shamir (неинтерактивный)
Модель безопасности	Стандартная модель	Модель случайного оракула (ROM)
Полнота	Сохраняется	Сохраняется
Soundness	В стандартной модели	В ROM
Zero-Knowledge	HVZK	Вычислительное ZK в ROM
Коммуникационная сложность	Высокая (многократный обмен)	Низкая (однократная передача)
Число раундов	≥ 3	1
Вычислительная сложность	Генерация случайного вызова проверяющим; требуется онлайн-взаимодействие	Добавляется вычисление хэш-функции; отсутствует онлайн-взаимодействие
Зависимость от хэш-функции	Нет	Да

В работе был проведён сравнительный анализ интерактивных Σ-протоколов и их неинтерактивных версий, полученных посредством преобразования Фиата-Шамира. Рассмотрены теоретические основы доказательств с нулевым разглашением, формальные свойства полноты, корректности и нулевого разглашения, а также особенности реализации данных свойств в интерактивной и неинтерактивной моделях.

Показано, что интерактивные Σ-протоколы обеспечивают строгие гарантии безопасности в стандартной модели криптографии, где корректность достигается за счёт непредсказуемости случайного вызова проверяющего. Преобразование Фиата-Шамира позволяет устранить интерактивность путём замены случайного вызова значением криптографической хэш-функции, что приводит к получению неинтерактивного доказательства, пригодного для асинхронных и распределённых систем.

Установлено, что фундаментальные свойства доказательства (полнота, корректность, нулевое разглашение) могут сохраняться при применении трансформации Фиата–Шамира, при этом формальные доказательства обычно даются в модели случайного оракула [3, с. 2]. Современные исследования показывают, что для ряда практических протоколов (включая некоторые SNARK-и) и в условиях расширенных моделей атак (включая квантовые сценарии) требуется отдельный, более тонкий анализ безопасности [4, с. 3; 5, с. 1-2]. Следовательно, при

выборе между интерактивной и неинтерактивной моделями следует учитывать не только абстрактные теоретические свойства, но и специфику реализации, угрозы платформы и актуальность доказательств для конкретной практической инстанции.

Сравнительный анализ показал, что неинтерактивные схемы обладают преимуществом с точки зрения коммуникационной эффективности и практической применимости, тогда как интерактивные Σ-протоколы обеспечивают более строгие формальные гарантии в рамках стандартной модели. Выбор конкретной модели должен определяться требованиями к уровню формальной строгости и условиями эксплуатации протокола.

Дальнейшие исследования могут быть направлены на анализ безопасности преобразования Фиата-Шамира вне модели случайного оракула, а также на разработку альтернативных подходов к построению неинтерактивных доказательств с нулевым разглашением в стандартной модели.

Литература

1. Юрцев А.Н. Доказательства с нулевым разглашением // Международный журнал гуманитарных и естественных наук. 2023. № 4-4(79). С. 138-141.

2. Arade M.S., Pise N.N. Comparative Study of Zero-Knowledge Proof Systems: Applications and Future Challenges // Journal of Harbin Engineering University. 2025. Vol. 46. № 05. https://clck.ru/3TvHdN

3. Attema T., Fehr S., Klooß M. Fiat-Shamir Transformation of Multi-Round Interactive Proofs (Extended Version) // Journal of Cryptology. 2023. Vol. 36. Art. 36. https://doi.org/10.1007/s00145-023-09478-y

4. Lavin R., Liu X., Mohanty H., Norman L., Zaarour G., Krishnamachari B. A Survey on the Applications of Zero-Knowledge Proofs // arXiv preprint arXiv:2408.00243. 2024.

5. Yuan Q., Sun C., Takagi T. Revisiting the Security of Fiat-Shamir Signature Schemes under Superposition Attacks // IACR ePrint Archive. 2024. С. 164-184.

Бимашова А.Б., Григорьева А.А., Мордвинова Е.А.
Нижневартовский государственный университет
г. Нижневартовск, Россия

МЕТОДЫ ИИ В ДРОНАХ

Развитие технологий искусственного интеллекта (ИИ) кардинально меняет возможности беспилотных летательных аппаратов (дронов). Современные дроны уже не просто дистанционно управляемые устройства – они превращаются в автономные интеллектуальные системы, способные анализировать окружающую среду, принимать решения и выполнять сложные задачи без постоянного контроля оператора.

Актуальность исследования методов ИИ в дронах обусловлена широким спектром их практического применения:

- Мониторинг и картографирование: создание точных 3D-карт, мониторинг инфраструктуры, лесов, полей.
- Поисково-спасательные операции: быстрый осмотр больших территорий, обнаружение людей в труднодоступных местах.
- Сельское хозяйство: анализ состояния посевов, точечное внесение удобрений и пестицидов.
- Безопасность и охрана: патрулирование территорий, обнаружение несанкционированного доступа.
- Строительство и инспекция: осмотр высотных зданий, мостов, линий электропередачи.

Интеграция методов ИИ позволяет дронам работать в реальном времени, адаптироваться к изменяющимся условиям и повышать общую эффективность операций.

Цель работы

Цель данной статьи – систематизировать и проанализировать современные методы искусственного интеллекта, применяемые в управлении и обработке данных с беспилотных летательных аппаратов, оценить их возможности и перспективы использования в различных сферах.

Задачи исследования

Для достижения поставленной цели необходимо решить следующие задачи:

1. Рассмотреть кластерный анализ как метод оптимизации групповых операций дронов (роевое управление, планирование траекторий, организация воздушного пространства).
2. Изучить применение нейронных сетей для классификации в задачах автономной навигации и анализа данных (обнаружение и идентификация объектов, предотвращение столкновений, динамическая корректировка маршрута).
3. Проанализировать методы сегментации изображений с помощью нейронных сетей для точного картографирования, мониторинга сельхозугодий и оценки состояния инфраструктуры.

4. Исследовать методы компьютерного зрения (CV), включая детекцию объектов на основе свёрточных нейронных сетей (CNN), трансформеров и YOLO, а также традиционные детекторы (SIFT, SURF, каскады Хаара).

5. Описать взаимодействие классификации и сегментации для повышения точности распознавания и выделения объектов (предварительная классификация перед сегментацией, использование общих признаков, многозадачные модели).

6. Кратко рассмотреть дополнительные методы машинного обучения (регрессия, случайный лес, градиентный бустинг) и их потенциальное применение в системах управления дронами.

7. Оценить преимущества и ограничения каждого метода в контексте реальных задач, решаемых дронами.

В таблице рассмотрены методы искусственного интеллекта, применяемых для работы дронов.

Таблица

Характеристика методов ИИ

Метод	Характеристика
Кластерный анализ	Определение: Метод машинного обучения без учителя для группировки объектов в кластеры на основе сходства признаков. Суть: Ищет скрытую структуру в неразмеченных данных, максимизируявнутрикластерное сходство. Примеры: K-means, DBSCAN, Affinity Propagation (APC)[4].
Нейронные сети для классификации изображений	Определение: Метод обучения с учителем, при котором сеть учится относить объекты к заранее заданным категориям. Суть: Автоматически выделяет признаки объектов с помощью сверточных слоев и строит границы между классами. Архитектуры: CNN (сверточные сети), AlexNet, ResNet, VGGNet[2, 3].
Нейронные сети для сегментации изображений	Определение: Разделение изображения на сегменты (попиксельная разметка) для точного выделения границ объектов. Суть: Каждому пикселю присваивается метка класса (семантическая) или отдельного экземпляра (инстанс-сегментация). Архитектуры: U-Net, SegNet, DeepLab, FCN[8, 9, 10, 12].
Детекция объектов (CV)	Определение: Метод компьютерного зрения для одновременного определения местоположения (координат рамки) и класса объекта. Суть: Поиск объектов на изображении и их классификация в режиме, близкому к реальному времени. Методы: • *Классические:* Каскады Хаара (быстрые, но менее точные), SIFT, SURF (поиск ключевых точек). • *Современные:* Семейство YOLO (одноэтапные детекторы на нейросетях, высокая скорость и точность)[6].
Вспомогательные методы (Регрессия, Бустинг и др.)	Определение: Статистические и ансамблевые методы машинного обучения. Суть: Используются для прогнозирования числовых значений или улучшения качества классификации на основе собранных данных. Примеры: Регрессионный анализ (оценка параметров), Случайный лес (RandomForest), Градиентный бустинг (XGBoost, CatBoost)[7].

Искусственный интеллект объединяет множество методов, поэтому для решения определенных задач с использованием ИИ необходимо использовать разные его методы, например:

1. Комбинация кластерного анализа и нейронных сетей эффективна для классификации и сегментации изображений с БПЛА. Это позволяет решать широкий спектр задач: обнаружение очагов пожаров, выявление стихийных свалок и незаконных вырубок, разметка карт, создание моделей рельефа, а также контроль за ходом строительства [4; 8]. Кластерный анализ группирует пиксели по схожим признакам, выделяя аномальные участки, требующие внимания (зоны возгорания, замусоривания или вырубки). Нейросети, в свою очередь, отвечают за распознавание специфических визуальных паттернов (например, дыма или пламени). Предварительная кластеризация может упростить задачу для нейросетей, беря на себя первичную группировку данных.

2. Для оценки ущерба от стихийных бедствий можно использовать комплекс методов. Кластерный анализ группирует пострадавшие зоны по общим признакам, регрессионные модели прогнозируют количественные показатели ущерба и объемы восстановительных работ, а алгоритмы случайного леса помогают систематизировать факторы риска, предпосылки и последствия разрушений [7]. Исходными данными могут быть спутниковые снимки высокого разрешения, данные дронов с мультиспектральными камерами, статистика прошлых стихийных бедствий, геоинформационные слои; совокупность данных с описанными методами дает возможность своевременно формировать детализированные карты ущерба и рассчитывать необходимые ресурсы для ликвидация последствий.

3. Методы компьютерного зрения применяются для поиска и идентификации объектов, поиска людей на открытых местностях и под завалами с использованием тепловизоров. В задачи CV также входит мониторинг динамики изменений: сравнение разновременных снимков для оценки масштабов бедствия, обнаружение разрывов трубопроводов или ЛЭП, а также автоматический подсчет пострадавших объектов и техники в зоне ЧС [5; 6; 11].

Рис. 1. БПЛА-мультикоптер

В современных условиях развития технологий, использование дронов выходит на новый уровень, характеризуемый активным внедрением использования искусственного интеллекта, а так же количеством и качеством методов ИИ [1; 3]. В ходе исследования были проанализированы современные методы ИИ, применяемые для обработки данных, получаемых БПЛА и установлены основные направления, активно использующие ИИ в контексте работы дронов. В том числе автономное управление и навигация, обработка изображений и видео с целью распознавания объектов, мониторинг больших территорий с автоматической классификацией состояний объектов. Сферы использования БПЛА расширяются и в перспективе могут включать в себя

- проведение комплексного анализа состояния отдельных агрокультур, включающего в себя автоматизированное реагирование на изменение отдельных показателей,
- проведение полной автоматизированной проверки качества объектов в периоды стройки и эксплуатации,
- прогнозирование возникновения чрезвычайных ситуаций и стихийных бедствий и своевременное оповещение необходимых инстанций [5; 7].

Дополнительного анализа требуют вопросы совершенствования безопасности использования БПЛА, в том числе кибербезопасности и этичности использования дронов в различных сферах современной жизнедеятельности общества.

Литература

1. Барский А.Б. Нейронные сети: распознавание, управление, принятие решений. М.: Финансы и статистика, 2004. С. 176.

2. Бычков А.Г., Киселева Т.В., Маслова Е.В. Использование сверточных нейросетей для классификации изображений // Вестник Сибирского государственного индустриального университета. 2023. № 1(43). С. 39-49.

3. Галушкин А.И. Нейронные сети: основы теории. Горячая линия–Телеком, 2010. С. 480.

4. Гитис Л.Х. Статистическая классификация и кластерный анализ. Горная книга, 2003. С. 157.

5. Евстраткин К.С., Ерпелев А.В., Султанова К.С. OPENCV: варианты использования компьютерного зрения // Цифровые технологии: наука, образование, инновации: Материалы III Международного научного Форума профессорско-преподавательского состава и молодых ученых (г. Москва 09 ноября 2020 года). 2020. С. 28-31.

6. Жарких В.А., Ляликова В.Г. Методы обнаружения транспортных средств на изображении с использованием YOLO и каскадов Хаара // Вестник науки. 2024. Т. 5, № 12-2(81). С. 329-335.

7. Кирсанова О.В., Кугаевских А.В., Муромцев Д.И. Классические методы машинного обучения: учебное пособие. Университет ИТМО, 2022. С. 53.

8. Лукашик Д.В. Анализ современных методов сегментации изображений // Экономика и качество систем связи. 2022. № 2(24). С. 57-65.

9. Мезенцев А.О., Минайчев А.А., Яндашевская Э.А. Разработка системы стегоанализа цифровых изображений на основе нейросетевого классификатора // Моделирование, оптимизация и информационные технологии. 2022. Т. 10, № 2(37).

10. Михайлов А.А. Автоматическая разметка данных для сегментации изображений документов с использованием глубоких нейронных сетей // Труды Института системного программирования РАН. 2022. Т. 34, № 6. С. 137-146.

11. Кокоулин А.Н., Тур А.И. Применение каскадов Хаара для распознавания объекта // Автоматизированные системы управления и информационные технологии: Материалы всероссийской научно-технической конференции: в 2х томах (г. Пермь, 5–7 июня 2025 г.). Пермь, 2018. Т. 1. С. 98-103.

12. Ушанкова В.Н., Широкова В.Н. Исследование применения сверточных нейронных сетей для решения задач семантической сегментации изображений // Системный анализ в науке и образовании. 2024. № 2. С. 21-29.

Бутяга Т.С.
Нижневартовский государственный университет
г. Нижневартовск, Россия

ПРОЕКТИРОВАНИЕ, МОДЕЛИРОВАНИЕ И 3D-ПЕЧАТЬ ИНТЕРАКТИВНОГО МАКЕТА УМНОГО ПОМЕЩЕНИЯ

3d-печать, как метод создания корпуса макета умного помещения, была выбрана из-за дешевизны и быстроты проектирования и исправлений неточностей деталей. Чтобы создавать детали с помощью 3d-принтера, нужно сначала смоделировать детали в САПР или в другом программном обеспечении, а после нарезать в программе-слайсере.

3d-печать – это аддитивная технология создания трехмерных изделий посредством послойного нанесения (наращивания) материала, что позволяет воссоздать сложные геометрические формы. Для создания корпуса макета умного помещения использовался 3d-принтер FlyingBear Ghost, который относится к технологии FDM печати [1; 2].

Система автоматического проектирования (САПР) представляет из себя программное обеспечение для проектирования, создания и анализа компьютерных 3d-моделей, определяемых геометрическими параметрами [4]. Моделирование происходило в программе КОМПАС-3D Учебная версия v23, распространяемая бесплатно в учебных целях российской компанией Аскон. Выбор обусловлен приоритетом использования отечественного программного обеспечения, что упрощает изучение и использование интерфейса программы, наличием бесплатной версии для обучающихся школ, колледжей и вузов, также гибкостью создания деталей и эскизов [3].

Слайсер переводит 3D-модель (обычно в формате STL или OBJ) в понятный для принтера файл G-code. Этот файл управляет движением экструдера, температурой и подачей материала, определяя, какой будет 3D печать деталей [1; 2]. Нарезка моделей под 3d-принтер осуществлялась в слайсере Ultimaker Cura 5.9.0-5.10.0.

Образом интерактивного макета послужила аудитория 206 корпуса № 4 НВГУ. Масштаб макета планировался 1 к 100, но для удобства 3d-печати масштаб был приведен 1 к 300, в следствии чего габариты макета составили 36,3 см, 24.4 см и 16,6 см.

Интерактивный макет состоит из множества составных деталей корпуса для упрощенной 3d-печати и сборки. В качестве основного вида скрепления используются цилиндрические пазы под шпонки и гребни деталей.

При моделировании деталей нужно учитывать допуски расширения пластика при печати, что составляет 0,1–0.2 мм для боковых стенок деталей, для верхней поверхности – 0,1 мм.

Начало создания модели макета начинается с проектирования механизма автоматического створчатого элемента, т. к. этот механизм задаст все последующие ограничения макета. Основная идея заключается в управляемом повороте створки, угол поворота которой считывается с помощью поворотного потенциометра, присоединенного к створке, и движения-вращения, передаваемого с помощью шестерней, dc-мотора. Вал мотора и ручка потенциометра находятся параллельно друг к другу, чтобы на них надеть шестерни,

на потенциометре сверху остаётся место для надевания колпачка, держащего ось вращения створки (рис. 1).

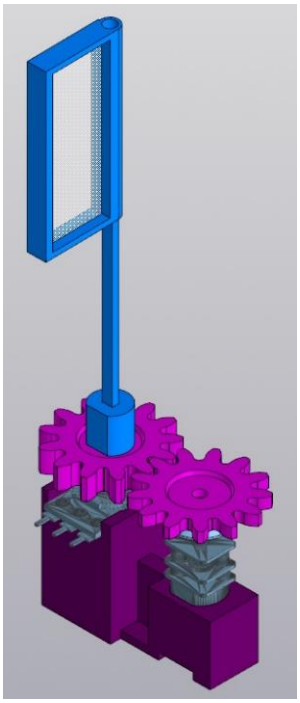


Рис. 1. Модель механизма автоматического створчатого элемента

Для дополнительного закрепления створки в рамке, сверху рамки находится небольшой скругленный выступ, который входит в цилиндрический паз, располагающийся на оси вращения сверху створки. Ось вращения помещается в отверстии внизу рамки, которое проходит сквозь стену и пол до коробки, где находится нижняя часть механизма автоматического створчатого элемента.

В модели стен нужно было включить пазы под расположение электрических компонентов и пустое пространство для подключения проводов к микроконтроллеру, находящемуся в коробке, отделенной полом модели, из-за этого цельная модель стен имеет сложную геометрию для 3d-печати и последующего использования. Было принято решение разделить стены на две части для исключения пустого пространства внутри одной детали (рис. 2) [7]. Верх составных стен закрывается заглушками, крепления которых осуществляется с помощью клинов и пазов, что дает хорошее закрепление при малом размере самого крепления. На заглушке, расположенной над стеной с дверью, находится крепление под кейс камеры Raspberry Pi (RaspiCam).

Создание деталей полов коробки начинается с позиционирования оси вращения механизма автоматических створчатых элементов из-за смежного расположения со стеной коробки, с которой нельзя допускать столкновения с шестернями механизма. Для этого на эскизе плоскости XY от начала координат строится прямоугольник к проекции отверстия в модели стены под ось вращения механизма, этот прямоугольник копируется с точкой привязки в начале координат. Создается новая деталь и строится эскиз на плоскости XY и вставляется этот прямоугольник в начало координат, где вторая точка будет являться точной позицией отверстия под ось вращения механизма. От этой оси строится все остальное таким же

способом. Такой алгоритм повторяется с каждой отдельной деталью пола. На полу корпуса коробки строится крепление, повторяющее форму крепления механизма снизу, где ось вращения сходится с осью вращения механизма автоматического створчатого элемента (рис. 3).

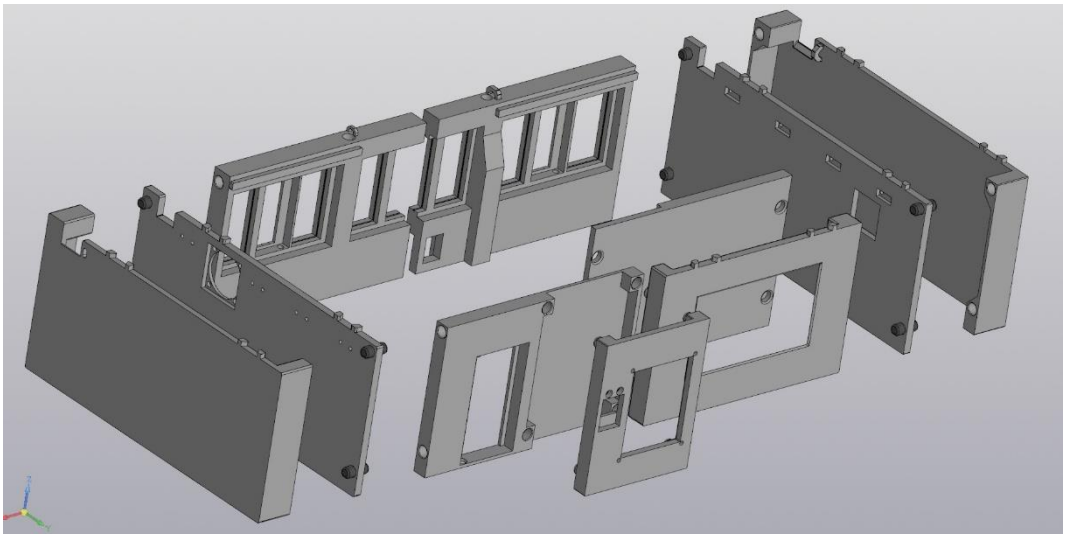


Рис. 2. Модели стен верхней части проекта, расставленные 50 мм друг от друга в месте присоединения

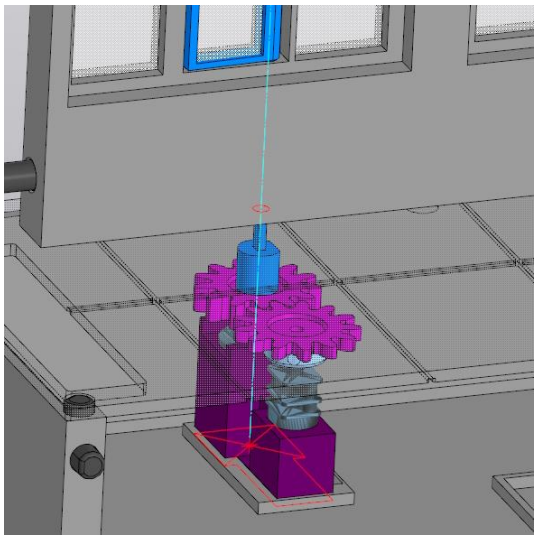


Рис. 3. Расположение оси вращения механизма автоматического створчатого элемента. Голубой эскиз – ось вращения, от которой строятся красные эскизы на деталях верхнего и нижнего пола

3d-печать полов верхней и нижней части макета оказалась затруднительной в отличие от стен макета. Простые модели при окончании печати оказываются с деформациями поверхности сверху и снизу, часто острые углы деталей отрывались от поверхности стола 3d-принтера, что оставляло деформированные угол и рядом лежащую геометрию детали [6]. Для устранения дефектов деталей полов было применено скругление острых углов с радиусом 1 мм, добавлены рейки в форме сетки на нижнюю поверхность, с которой начиналась 3d-печать, для распределения стресса на меньшие участки и для улучшения схватывания между поверхностями детали и стола 3d-принтера. Для исправления верхней поверхности было

принято решение поставить более плотное заполнение детали (с 5% до 10%) и изменение типа заполнения с grid на lines [5].

Механизм автоматических жалюзи представляет из себя вал, на котором закрепляется ткань жалюзи, и креплений dc-мотора и датчика угла поворота, на которые надевается этот вал (рис. 4). Механизм крепится над окном на прямоугольный выступ. Изначально, вместо датчика угла поворота, крепление моделировалось под поворотный потенциометр, но он может повернуться только на 270-300°, что не позволяет развернуть ткань жалюзи полностью, а датчик угла поворота без ограничений может сделать полный оборот. Разъёмы датчика угла поворота направленны вверх, чтобы провода можно было завернуть и просунуть в отверстие внутри стены, которое доходит до нижней части макета.

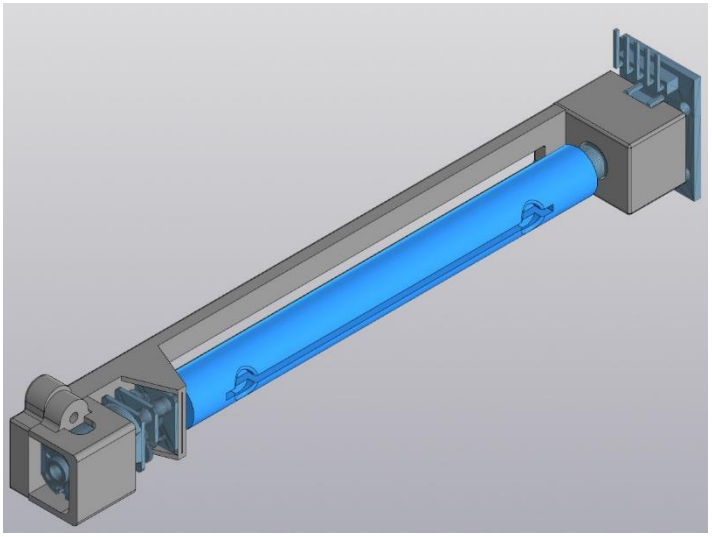


Рис. 4. Модель механизма автоматических жалюзи

Последним этапом моделирования является приведение модели к готовому виду (рис. 5), добавляя модели электрических компонентов, задуманных с самого начала, прозрачные модели стеклопакета, возможность привести в движение механизмы.

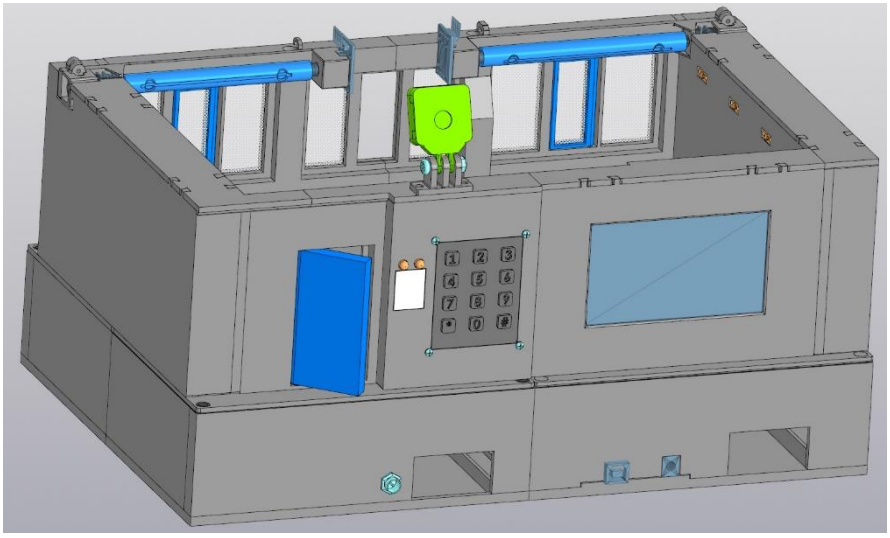


Рис. 5. Полная модель интерактивного макета умного помещения

В заключении можно сказать, что этот проект объединил в себе множество методик проектирования и разработки 3d-моделей для последующего создания на 3d-принтере. По окончании 3d-печати получился корпус интерактивного макета умного помещения, который почти не требует физической обработки деталей, что ускоряет сборку макета и размещение в нём электрических компонентов, для последующей разработки ПО системы управления умного помещения на базе Arduino.

Литература

1. Богаткин М. А. Особенности FDM 3D-печати: от идеи до готового изделия // Постулат. 2022. № 12(86). С. 1-10. URL: https://e-postulat.ru/index.php/Postulat/article/view/4710 (дата обращения: 06.03.2026).

2. Богаткин М. А. Особенности подготовки 3D-модели к печати на FDM 3D-принтере на примере программы PrusaSlicer // Постулат. 2023. № 7(93). С. 1-7. URL: https://e-postulat.ru/index.php/Postulat/article/view/5192 (дата обращения: 06.03.2026).

3. Васильева Е. Я., Петреченков М. А. Сравнительный анализ компьютерных программ 3D моделирования: solid Works, Компас 3D, AUTOCAD // Молодая фармация потенциал будущего: Сборник материалов XIV всероссийской научной конференции с международным участием Молодежного научного общества СПХФУ (Санкт-Петербург, 28 марта 2024 года). Санкт-Петербург: Санкт-Петербургский государственный химико-фармацевтический университет, 2024. С. 1086-1089.

4. Кириллов А. А., Щербакова Е. С. Автоматизированное проектирование (CAD) и автоматизированное производство (CAM) // Экономика и социум. 2021. № 4-2(83). С. 57-60.

5. Михальченко А. А., Невзорова А. Б. Повышение точности 3d-печати методом FDM путем изменения параметров 3D-принтера // Стратегия и тактика развития производственно-хозяйственных систем: сборник научных трудов. Гомель: Гомельский государственный технический университет имени П. О. Сухого, 2023. С. 140-143.

6. Поляничко К. С. Различные дефекты 3D-печати // Организация работы с молодежью. 2021. № 5. С 1-8. URL: ovv.esrae.ru/308-1399 (дата обращения: 06.03.2026).

7. Kantaros A., Ganetsos T., Pallis E., Papoutsidakis M. Toward a holistic approach for design for additive manufacturing: a perspective on challenges, practical insights, and research needs // Adv. Manuf. 2025. №2. С 1-28. https://doi.org/10.55092/am20250011

Вернигоров С.И.
Нижневартовский государственный университет
г. Нижневартовск, Россия

МЕТОДЫ ПРЕОБРАЗОВАНИЯ ТЕКСТА В РЕЧЬ И РАСПОЗНАВАНИЯ РЕЧИ С ПРЕОБРАЗОВАНИЕМ В ТЕКСТ

Автоматическое распознавание речи представляет собой одно из наиболее динамично развивающихся направлений компьютерных наук и искусственного интеллекта [1]. Технологии преобразования речевого сигнала в текстовую форму лежат в основе функционирования голосовых помощников, интеллектуальных систем управления, сервисов автоматической транскрипции, систем поддержки принятия решений и специализированных решений для лиц с ограниченными возможностями здоровья. Повышенный интерес к данной области обусловлен стремительным ростом вычислительных мощностей, развитием методов глубокого обучения и накоплением масштабных корпусов речевых данных.

С научной точки зрения задача распознавания речи является сложной междисциплинарной проблемой, объединяющей акустику, цифровую обработку сигналов, математическую статистику, теорию вероятностей и нейросетевые методы. В рамках данной работы рассматриваются физические характеристики голосовой речи, математические методы её анализа и современные нейросетевые архитектуры, обеспечивающие преобразование речевого сигнала в текст.

1. Акустическая природа и математическая модель речевого сигнала

Речь человека формируется в результате взаимодействия дыхательной системы, голосовых связок и артикуляционного аппарата. Возникающий акустический сигнал представляет собой колебательный процесс, распространяющийся в упругой среде [5]. В непрерывной форме он описывается функцией $x(t)$, отражающей изменения звукового давления во времени.

При цифровой обработке сигнал подвергается дискретизации с шагом T_s, что приводит к получению последовательности отсчётов:

$$x(n) = x(nT_s),$$

где $n \in Z$, а частота дискретизации $f_s = \frac{1}{T_s}$.

Существенной особенностью речевого сигнала является его нестационарность: статистические характеристики сигнала изменяются во времени вследствие смены фонем, интонаций и темпа произношения. Однако в пределах коротких временных интервалов (обычно 10–25 мс) сигнал можно считать квазистационарным. Это позволяет применять методы спектрального анализа, основанные на оконном преобразовании Фурье [8].

Для анализа спектрального состава используется дискретное преобразование Фурье:

$$X(k) = \sum_{n=N-0}^{1} x\,(n)e^{-j}\frac{2\pi kn}{N},$$

где N – длина окна анализа.

Спектральное представление позволяет выявить частотную структуру сигнала, включая основную частоту f_0 и форманты F_1, F_2, F_3, соответствующие резонансным характеристикам речевого тракта. Форманты играют ключевую роль в различении гласных звуков и определяют акустическую идентичность фонем.

2. Структура системы автоматического распознавания речи

Современная система распознавания речи включает несколько функционально связанных этапов. На первом этапе осуществляется предварительная обработка сигнала, включающая фильтрацию шумов и нормализацию амплитуды. Далее сигнал сегментируется на короткие фреймы, к каждому из которых применяется оконная функция, например окно Хэмминга:

$$w(n) = 0{,}54 - 0{,}46cos\left(\frac{2\pi n}{N-1}\right).$$

Оконная обработка снижает эффект спектральных утечек и обеспечивает корректность частотного анализа.

После спектрального анализа осуществляется извлечение признаков, представляющих компактное описание сигнала [6]. Далее акустическая модель сопоставляет последовательность признаков с вероятностными представлениями фонем или символов. Языковая модель учитывает статистические закономерности языка и формирует наиболее вероятную текстовую последовательность. На завершающем этапе декодер определяет оптимальный текстовый результат.

Рис. 1. Структурная схема системы автоматического распознавания речи (составлено автором)

3. Методы извлечения признаков.

Одним из наиболее распространённых методов параметризации речи являются мел-кепстральные коэффициенты (MFCC) [2]. Они основаны на учёте особенностей восприятия частоты человеческим слухом. Переход к мел-шкале осуществляется по формуле:

$$m = 2595\log_{10}(1 + \frac{f}{700}).$$

После применения набора фильтров в мел-шкале вычисляется логарифм энергии каждого фильтра и выполняется дискретное косинусное преобразование:

$$C_k = \sum_{n=1}^{M} log(s_n)\, cos\left[\frac{\pi k}{M}(n - 0{,}5)\right],$$

где s_n – энергия в n-м фильтре.

Полученные коэффициенты образуют вектор признаков, характеризующий спектральную форму сигнала. Дополнительно могут вычисляться дельта- и дельта-дельта-коэффициенты, отражающие динамику изменения спектра во времени.

4. Вероятностные модели распознавания речи.

Исторически основой систем распознавания речи являлись скрытые марковские модели [3]. Они описывают речевой процесс как последовательность скрытых состояний $Q = \{q1, q2, \ldots, qT\}$, генерирующих наблюдаемые векторы признаков O.

Модель определяется набором параметров:

$$\lambda = (A, B, \pi),$$

где A – матрица переходных вероятностей, B – вероятности эмиссии, π – начальное распределение состояний.

Вероятность наблюдаемой последовательности вычисляется как:

$$P(O \mid \lambda) = \sum_{Q} P(O, Q \mid \lambda).$$

Для поиска наиболее вероятной последовательности состояний применяется алгоритм Витерби, минимизирующий суммарную стоимость пути.

Несмотря на высокую эффективность в условиях ограниченного словаря, HMM имеют ограничения, связанные с предположением марковского свойства и независимости наблюдений.

5. Нейросетевые методы распознавания речи.

Современные системы распознавания речи базируются на глубоких нейронных сетях. Рекуррентные нейронные сети моделируют временные зависимости следующим образом:

$$ht = f(W_h h_t - 1 + W_x x_t + b),$$

где h_t – скрытое состояние.

Модификация LSTM вводит ячейку памяти:

$$C_t = f_t \odot C_t - 1 + i_t \odot \tilde{C}_t,$$

что позволяет учитывать долгосрочные зависимости и уменьшать проблему затухающего градиента.

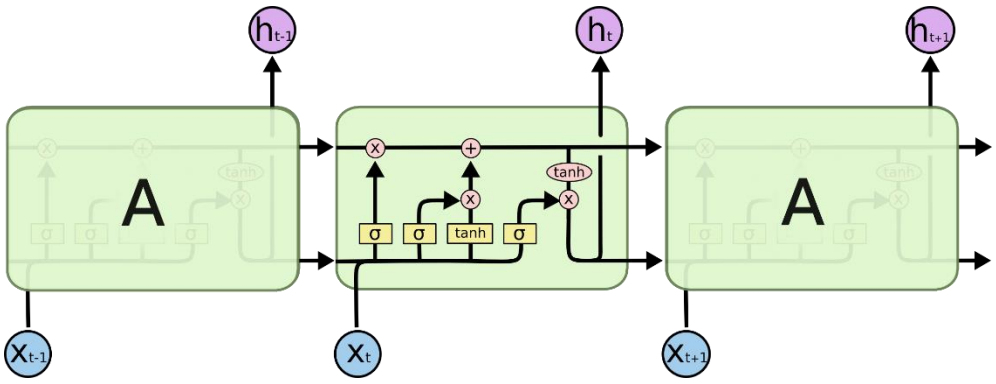


Рис. 2. Архитектура нейросетевой модели LSTM распознавания речи [7]

Архитектура основана на механизме самовнимания [4]:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{\partial_k}}\right)V.$$

Данный механизм позволяет учитывать взаимосвязи между всеми элементами входной последовательности одновременно, что существенно повышает точность распознавания.

Современные end-to-end модели используют функцию потерь CTC (Connectionist Temporal Classification):

$$\mathcal{L}_{CTC} = -lnP(Y \mid X),$$

где X – входной аудиосигнал, а Y – текстовая последовательность.

Заключение.

В работе рассмотрены основные методы автоматического распознавания речи с преобразованием в текст, охватывающие как классические статистические подходы, так и современные нейросетевые технологии. Показано, что речевой сигнал представляет собой сложный нестационарный акустический процесс, требующий применения методов цифровой обработки, спектрального анализа и параметризации. Использование оконных преобразований и мел-кепстральных коэффициентов позволяет сформировать информативное представление сигнала, пригодное для дальнейшего моделирования.

Анализ вероятностного подхода продемонстрировал, что формализация задачи распознавания в виде максимизации апостериорной вероятности $P(W|X)$ стала теоретической основой построения систем распознавания речи. Скрытые марковские модели и языковые n-граммные модели обеспечили развитие устойчивых алгоритмов декодирования, однако их ограничения обусловили переход к более гибким методам глубокого обучения.

Современные нейросетевые архитектуры, включая рекуррентные сети и модели на основе механизма самовнимания, существенно повысили точность распознавания и упростили структуру систем за счёт сквозного обучения. Тем не менее остаются актуальными задачи повышения устойчивости к шумам, адаптации к различным условиям записи и оптимизации вычислительных затрат.

Таким образом, развитие методов распознавания речи является важным направлением современных компьютерных наук и играет значительную роль в совершенствовании интеллектуальных человеко-машинных интерфейсов.

Рассмотрим простые примеры преобразования текста в речь и речи в текст.

Более интересным и глубоким для изучения является преобразование речи в текст. Приведем простой пример программы на языке Python. Конечно, имеются голосовые помощники в различных системах, для которых реализованы эти процессы. Мы привели коды собственных простых решений.

```python
import pyttsx3
# Инициализация движка
engine = pyttsx3.init()
# Получение доступных голосов
voices = engine.getProperty('voices')

# Вывод информации о доступных голосах
for index, voice in enumerate(voices):
    print(f"Voice {index}: {voice.name} - {voice.languages}")

# Выбор голоса (например, 0 для первого доступного голоса)
engine.setProperty('voice', voices[0].id)  # Измените индекс для выбора другого спикера

# Текст для озвучивания
text = "Hi! This is an example of text-to-voice conversion with speaker selection on Python."

#text="Привет!Это пример преобразования текста в голос с выбором спикера на Питоне"
#engine.say(text1)
### Проигрывание текста
#engine.setProperty('voice', voices[1].id)
#engine.say(text1)
engine.setProperty('voice', voices[0].id)
engine.say(text)
# Ожидание завершения произнесения текста
engine.runAndWait()
```

Рис. 3. Преобразование текста в голосовую речь (составлено автором).

```python
import speech_recognition as speech_r
import pyaudio
import wave
#Для начала нужно выставить параметры записи звука:
CHUNK = 1024 # определяет форму ауди сигнала
FRT = pyaudio.paInt16 # шестнадцатибитный формат задает значение амплитуды
CHAN = 1 # канал записи звука
RT = 44100 # частота
REC_SEC = 5 #длина записи
OUTPUT = "output.wav"
#Далее нужно создать объект для обращения к устройству звукозаписи:
p = pyaudio.PyAudio()
#и открыть поток для записи звука:
stream = p.open(format=FRT,channels=CHAN,rate=RT,input=True,
                frames_per_buffer=CHUNK)
# открываем поток для записи
print("rec")
frames = [] # формируем выборку данных фреймов
for i in range(0, int(RT / CHUNK * REC_SEC)):
    data = stream.read(CHUNK)
    frames.append(data)
print("done")
#и закрываем поток
stream.stop_stream() # останавливаем и закрываем поток
stream.close()
p.terminate()
#Дальше нам нужно записать оцифрованную звуковую дорожку в файл.
#Для этого нам и пригодится библиотека wave:
w = wave.open(OUTPUT,'wb')
w.setnchannels(CHAN)
w.setsampwidth(p.get_sample_size(FRT))
w.setframerate(RT)
w.writeframes(b''.join(frames))
w.close()
#В итоге мы получаем готовую звуковую дорожку записанную с микрофона устройства и готовую к распознаванию
#для этого нам потребуется библиотека Speech Recognition:
sample = speech_r.WavFile('output.wav')
#Непосредственно для распознавания текста нам потребуется класс Recognizer он имеет множество функций,
#а также определяет каким API мы будем пользоваться: #tt-это аудио
r = speech_r.Recognizer()
with sample as tt:
    content = r.record(tt)
    r.adjust_for_ambient_noise(tt)
print(r.recognize_google(tt,'en-US'))
```

Рис. 4. Запись речи с преобразованием в текст (составлено автором)

Литература

1. Бурцев М.С., Мельников А.В. Современные методы автоматического распознавания речи на основе глубокого обучения // Информационные технологии. 2022. № 4. С. 15-23.

2. Ковалёв С.А., Григорьев И.Н. Методы извлечения признаков в системах распознавания речи // Вестник компьютерных и информационных технологий. 2023. № 7. С. 42-49.

3. Лукьянов Д.В., Смирнова Е.П. Применение скрытых марковских моделей в задачах распознавания речи // Программные системы: теория и приложения. 2021. Т. 12. № 3. С. 85-97.

4. Attention is All You Need Paper – how are the Q, K, V values calculated? // ai.stackexchange.com. URL: https://clck.ru/3TLk9x (дата обращения: 18.02.2026).

5. Baevski A., Zhou H., Mohamed A., Auli M. wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations // Advances in Neural Information Processing Systems. 2021. Vol. 34. P. 12449-12460. https://doi.org/10.48550/arXiv.2006.11477

6. Gulati A., Qin J., Chiu C.C. et al. Conformer: Convolution-augmented Transformer for Speech Recognition // Interspeech. 2021. P. 5036-5040. https://doi.org/10.21437/Interspeech.2020-3015

7. Olah C. Understanding LSTM Networks // colah’s blog. 2015. URL: https://colah.github.io/posts/2015-08-Understanding-LSTMs (дата обращения: 17.02.2026).

8. Radford A., Kim J.W., Xu T. et al. Robust Speech Recognition via Large-Scale Weak Supervision // Proceedings of ICML. 2023. https://doi.org/10.48550/arXiv.2212.04356

Володкевич М.А.
Нижневартовский государственный университет
г. Нижневартовск, Россия

ОТ МАШИННОГО КОДА К ЭРЕ ПРОГРАММИРОВАНИЯ НА ЕСТЕСТВЕННОМ ЯЗЫКЕ

История программирования представляет собой непрерывный процесс абстрагирования – последовательного удаления человека от необходимости мыслить категориями машины и приближения машины к способности понимать категории человека. Каждый новый уровень абстракции – от машинного кода к ассемблеру, от ассемблера к языкам высокого уровня, от процедурного программирования к объектно-ориентированному – расширял круг людей, способных создавать программное обеспечение, и увеличивал производительность разработчиков [4].

Сегодня мы находимся на пороге качественно нового этапа этой эволюции. Термин «вайбкодинг» (vibe coding), введённый в феврале 2025 года Андреем Карпатым – бывшим директором по искусственному интеллекту Tesla и одним из основателей OpenAI, – ознаменовал легитимизацию принципиально нового подхода: человек описывает желаемый результат на естественном языке, а большая языковая модель (LLM) генерирует программный код [7]. Принципиальное отличие состоит в том, что разработчик может вообще не читать сгенерированный код, полагаясь на результат выполнения как на единственный критерий корректности.

Этот феномен поднимает фундаментальные вопросы: является ли программирование на естественном языке подлинной демократизацией технологии или иллюзией компетентности? Может ли естественный язык стать надёжным средством спецификации вычислительных процессов? Как должна измениться подготовка IT-специалистов? [1]

Актуальность исследования определяется стремительностью изменений: по данным GitHub, к началу 2025 года более 92 миллионов разработчиков использовали AI-инструменты для генерации кода [9], а Gartner прогнозирует, что к 2028 году 75% корпоративных разработчиков будут применять AI-ассистентов как основной инструмент.

Целью работы является комплексный анализ эволюции парадигм программирования – от машинного кода до программирования на естественном языке – с точки зрения последовательного повышения уровня абстракции. Задачи исследования: проследить историческую эволюцию средств программирования; проанализировать технологические основы программирования на естественном языке; определить содержание и границы феномена «вайбкодинга»; оценить преимущества, ограничения и риски данного подхода; сформулировать рекомендации по адаптации учебных программ высшего образования.

Отправной точкой эволюции является машинный код – последовательность двоичных инструкций, непосредственно исполняемых процессором. Это нулевой уровень абстракции: каждая команда однозначно соответствует физической операции аппаратного обеспечения. Первые программисты, среди которых Грейс Хоппер (компьютер Mark I, 1944), составляли

программы в виде числовых последовательностей – процесс чрезвычайно трудоёмкий и подверженный ошибкам [4].

Появление ассемблера в начале 1950-х стало первым шагом к абстрагированию: числовые коды операций заменились мнемоническими обозначениями (ADD, MOV, JMP), а адреса памяти – символическими именами. Однако ассемблер сохранял привязанность к конкретной архитектуре и требовал мышления в категориях машины – регистров, стека, флагов состояния.

Подлинная революция произошла с созданием Фортрана (1957) группой Джона Бэкуса в IBM. Впервые программист мог выражать вычисления в форме, приближённой к математической нотации. Ключевым прорывом стала идея компиляции – автоматического преобразования программы высокого уровня в машинный код, при котором компилятор берёт на себя распределение регистров, управление памятью и оптимизацию.

Вслед появились LISP (1958, функциональное программирование), COBOL (1959, приближение к английскому языку), ALGOL (1960, структурное программирование). В 1970–80-е годы Pascal, C и Ada воплотили парадигму модульного программирования. Язык C занял уникальное положение, позволив создать UNIX без обращения к ассемблеру.

Следующий скачок абстракции связан с объектно-ориентированным программированием. От Simula (1967) через Smalltalk (1972) и C++ (1985) к Java (1995) и Python – ООП позволило моделировать предметную область в терминах объектов с атрибутами и методами. Параллельно развивались декларативные подходы (SQL, HTML/CSS), позволяющие описывать что должно быть сделано, оставляя как на усмотрение системы.

На протяжении всей истории каждый новый уровень абстракции делал программу более похожей на естественный язык: конструкции типа if (balance > 0) then withdraw(amount) уже содержат значительную долю элементов английского языка.

Качественный прорыв произошёл с появлением архитектуры трансформеров (2017, статья «Attention Is All You Need») [10]. Механизм самовнимания позволил создавать модели, улавливающие сложные зависимости как в естественном языке, так и в программном коде. Масштабирование привело к появлению GPT-4, Claude, Gemini, LLaMA и других LLM, демонстрирующих высокую способность генерировать корректный код на основе естественноязычных описаний [5].

Ключевые технологические компоненты: обучение на массивных корпусах кода (модель Codex обучена на десятках миллиардов строк) [6]; инструктивная настройка и RLHF; контекстные окна до миллиона токенов; агентные системы, способные автономно выполнять код, анализировать ошибки и взаимодействовать с файловой системой. Современные генеративные модели демонстрируют способность не только генерировать код с нуля, но и дополнять существующие фрагменты [5].

А.Карпатый описал вайбкодинг так: «Ты полностью отдаёшься вайбу и забываешь, что код вообще существует. Я почти ничего не набираю сам. И я даже не читаю код» [7]. Принципиальное отличие от академического «программирования на естественном языке»:

критерием успеха является не корректность кода, а удовлетворительность поведения программы. Код – промежуточный артефакт, который может быть не прочитан создателем.

Это впервые в истории разрывает связь между программистом и кодом: между намерением и реализацией существует «чёрный ящик» – нейронная сеть, непрозрачная даже для её создателей [1].

К 2025 году сложилась экосистема инструментов: GitHub Copilot (ускорение на 55%) [9], Cursor, Claude Code, Replit Agent, Devin – от ассистирующего режима до полностью агентного, но все объединены идеей естественного языка как основного интерфейса разработки.

Преимущества подхода. Демократизация: специалисты других областей могут создавать программные инструменты без изучения языков программирования. Исследование Peng et al. показало, что AI-ассистенты позволяют новичкам решать задачи уровня среднего специалиста [9]. Ускорение прототипирования: время создания MVP сокращается с недель до часов. Снижение когнитивной нагрузки: делегирование рутины позволяет сосредоточиться на архитектуре. Мультиязычность: переключение между языками программирования. Образовательный потенциал: AI как интерактивный учебный партнёр [3].

Ограничения и риски. Галлюцинации: до 40% сгенерированного кода содержит ошибки разной степени серьёзности [2]. Безопасность: значительная доля генерируемого кода содержит известные уязвимости (SQL-инъекции, XSS) [8]. Неоднозначность языка: инструкция «сделай быстрее» допускает множество интерпретаций. Масштабируемость: кодовая база без понимания структуры крайне трудна в сопровождении – вайбкодинг автоматизирует наименее сложную часть процесса [1]. Зависимость от коммерческих сервисов и проблемы приватности. Эрозия компетенций: парадоксальная ситуация – для эффективного использования AI необходимо знание, приобретение которого AI делает казалось бы ненужным [3].

LLM предлагают эвристическое преодоление разрыва между неоднозначным естественным языком и точным программным кодом – не через формальную верификацию, а через статистически обоснованную «наиболее вероятную» интерпретацию [6]. Для многих задач это достаточно, но для критически важных систем – категорически нет [2; 8].

Формируется новая форма дискурса – промпт-инжиниринг: своеобразный «пиджин» между человеком и AI, не являющийся ни естественным, ни формальным языком, но обладающий собственной грамматикой и стилистическими конвенциями.

Импликации для высшего образования. Традиционная модель подготовки нуждается в переосмыслении [3]. Необходима переориентация с синтаксиса на вычислительное мышление; развитие навыков верификации и критической оценки сгенерированного кода; интеграция промпт-инжиниринга в учебный процесс; формирование этической и правовой грамотности. Как отмечается в современных исследованиях, «способность взаимодействовать с генеративным AI становится ключевой компетенцией, не заменяющей фундаментальное понимание принципов вычислений» [1; 3].

Вероятно сочетание трёх сценариев: демократизация для простых задач, профессиональная стратификация («вайбкодеры» – «AI-оркестраторы» – «глубокие инженеры»), периодические коррекции ожиданий.

Эволюция представляет собой последовательность уровней с определённым соотношением выразительности, контролируемости и доступности. Машинный код: максимальная выразительность и контролируемость, минимальная доступность. Естественный язык: максимальная доступность, но резкое снижение контролируемости. Каждый уровень надстраивается над предыдущим, не отменяя его.

Переход к программированию на естественном языке ставит глубокие теоретические вопросы на стыке информатики, лингвистики и философии.

С точки зрения формальных языков, программирование исторически основывалось на использовании формальных языков – языков с точно определённым синтаксисом и семантикой, допускающих однозначный разбор и интерпретацию. Естественный язык, напротив, является языком с размытой семантикой, контекстуальной зависимостью и конвенциональными элементами. Традиционный взгляд на программирование предполагает, что этот разрыв непреодолим: невозможно построить надёжную систему на основе ненадёжной спецификации. Однако LLM, по сути, предлагают эвристическое преодоление этого разрыва: не через формальную верификацию, а через статистически обоснованную «наиболее вероятную» интерпретацию [6].

Это порождает фундаментальный вопрос: является ли «наиболее вероятная» интерпретация достаточной для создания надёжного программного обеспечения? Для многих практических задач ответ, по-видимому, положительный – модели генерируют корректный код в значительной доле случаев, и итеративный процесс уточнения через диалог позволяет устранить большинство неверных интерпретаций. Однако для критически важных систем – медицинских, финансовых, транспортных – статистическая «наиболее вероятная» корректность может быть категорически недостаточна [2; 8].

С лингвистической точки зрения, вайбкодинг создаёт новую форму дискурса – «промпт-инжиниринг» (prompt engineering), – которая не является ни естественным языком в его обыденном употреблении, ни формальным языком программирования. Эффективный промпт должен быть одновременно достаточно естественным для удобства человека и достаточно структурированным для адекватной интерпретации моделью. Формируется своеобразный «пиджин» – упрощённый контактный язык между человеком и AI, обладающий собственной грамматикой, прагматикой и стилистическими конвенциями.

Описанные изменения имеют глубокие последствия для системы высшего образования в области информационных технологий. Традиционная модель подготовки программистов, основанная на последовательном изучении синтаксиса языков программирования, алгоритмов и структур данных, парадигм программирования и технологий разработки, нуждается в существенном переосмыслении [3].

Переориентация с синтаксиса на мышление. Если генерация синтаксически корректного кода может быть делегирована AI, фокус обучения должен сместиться на

формирование вычислительного мышления (computational thinking) – способности декомпозировать сложные задачи, выявлять закономерности, конструировать абстракции и разрабатывать алгоритмы на концептуальном уровне [4]. Эти навыки необходимы как для традиционного программирования, так и для эффективного формулирования промптов и оценки сгенерированного кода.

Развитие навыков верификации и критической оценки. Студенты должны научиться критически оценивать код, сгенерированный AI: проверять его корректность, выявлять потенциальные уязвимости, оценивать производительность и масштабируемость [5; 8]. Это требует глубокого понимания основ информатики – алгоритмической сложности, теории типов, принципов безопасности, – парадоксальным образом делая фундаментальное образование более важным, а не менее.

Интеграция промпт-инжиниринга в учебный процесс. Навыки эффективного взаимодействия с AI-ассистентами – формулирования точных промптов, итеративного уточнения требований, декомпозиции сложных задач для AI – становятся профессионально значимыми и должны быть включены в учебные программы [1]. Однако это должно дополнять, а не заменять традиционные компетенции.

Этическая и правовая грамотность. Использование AI для генерации кода поднимает вопросы авторского права (кому принадлежит сгенерированный код?), ответственности (кто несёт ответственность за ошибки в сгенерированном коде?), академической честности (является ли использование AI при выполнении учебных заданий плагиатом?) и этики (как обеспечить справедливость и непредвзятость систем, созданных с помощью AI?). Эти вопросы должны стать неотъемлемой частью подготовки специалистов [3].

Прогнозирование развития технологий, связанных с AI, сопряжено с высокой степенью неопределённости. Тем не менее можно обозначить несколько вероятных сценариев.

Оптимистический сценарий: программирование как универсальная грамотность. Дальнейшее развитие LLM приведёт к тому, что создание программного обеспечения станет настолько же доступным, как создание документов в текстовом редакторе. Каждый специалист сможет создавать программные инструменты для своих нужд, программирование перестанет быть узкоспециальным навыком, а профессиональные разработчики сосредоточатся на сложных архитектурных задачах и обеспечении надёжности систем.

Реалистический сценарий: новая стратификация профессии. Профессия разработчика программного обеспечения расслоится на несколько уровней: «вайбкодеры», создающие относительно простые приложения с помощью AI; «AI-оркестраторы», управляющие сложными процессами разработки с участием множества AI-агентов; и «глубокие инженеры», работающие на уровне системной архитектуры, производительности и безопасности, где AI-инструменты пока недостаточно надёжны [1].

Осторожный сценарий: пузырь «магического» программирования. Массовое внедрение вайбкодинга приведёт к лавинообразному росту программного обеспечения низкого качества, увеличению числа уязвимостей и инцидентов безопасности [2; 8],

разочарованию в AI-ассистентах и коррекции ожиданий. Фундаментальные навыки программирования вновь станут высоко цениться как гарантия качества и надёжности.

Наиболее вероятным представляется сочетание элементов всех трёх сценариев: демократизация для простых задач, новая профессиональная стратификация для сложных и периодические коррекции ожиданий по мере обнаружения ограничений текущего поколения AI-инструментов.

Для системного осмысления эволюции программирования целесообразно представить её как последовательность уровней абстракции, каждый из которых характеризуется определённым соотношением между выразительностью, контролируемостью и доступностью.

На уровне машинного кода выразительность максимальна (можно описать любую операцию, поддерживаемую аппаратным обеспечением), контролируемость абсолютна (каждая инструкция определяет точное действие), но доступность минимальна (требуются глубокие знания архитектуры). На уровне ассемблера выразительность и контролируемость практически не изменились, но доступность несколько возросла за счёт мнемонических обозначений. Языки высокого уровня существенно повысили доступность и частично пожертвовали контролируемостью (программист не контролирует конкретные машинные инструкции). Объектно-ориентированные и декларативные языки продолжили этот тренд.

Программирование на естественном языке и вайбкодинг доводят эту тенденцию до логического предела: доступность максимальна (любой носитель языка может сформулировать промпт), но контролируемость резко снижается (разработчик не контролирует и часто даже не знает конкретную реализацию). Выразительность ограничена способностями конкретной LLM и может быть как выше, так и ниже, чем у традиционных средств, – в зависимости от задачи [5].

Этот анализ показывает, что каждый уровень абстракции не отменяет предыдущий, а надстраивается над ним: программирование на естественном языке невозможно без языков высокого уровня, те – без ассемблера, а ассемблер – без машинного кода. Вопрос не в замене одного уровня другим, а в том, на каком уровне работает каждый конкретный специалист для каждой конкретной задачи.

Выводы. Проведённое исследование позволяет сформулировать следующие основные выводы. Эволюция от машинного кода к программированию на естественном языке представляет собой закономерный процесс повышения уровня абстракции, в рамках которого вайбкодинг выступает логическим продолжением исторически сложившейся тенденции, а не случайным артефактом технологического развития [4; 10]. Технологическую основу этого явления составляют большие языковые модели, построенные на архитектуре трансформеров и обеспечивающие статистическую аппроксимацию отображения человеческих намерений в программную реализацию [6; 10]. При этом вайбкодинг, обладая значительным потенциалом для демократизации и ускорения процесса разработки [9], одновременно сопряжён с существенными рисками, включающими галлюцинации моделей [2], уязвимости безопасности [8], трудности масштабирования и эрозию профессиональных компетенций [1]. Вместе с тем программирование на естественном языке не заменяет традиционное, а

формирует новый уровень абстракции, при котором для критически важных систем фундаментальное понимание информатики по-прежнему остаётся незаменимым. В связи с этим образование должно сместить акцент на развитие вычислительного мышления, навыков критической оценки кода, промпт-инжиниринга и этико-правовой грамотности [3], поскольку фундаментальное образование приобретает новое измерение, трансформируясь из средства создания кода в средство контроля кода, создаваемого искусственным интеллектом. В конечном счёте переход к программированию на естественном языке является эпистемологическим явлением, меняющим саму природу знания в сфере информационных технологий: от владения формальным языком – к способности точно формулировать задачи и управлять процессом, в котором формальные детали реализации скрыты за вероятностной моделью.

Литература

1. Воронцов К.В. Машинное обучение и искусственный интеллект в разработке программного обеспечения // Программная инженерия. 2024. Т. 15, № 3. С. 112-125.

2. Гаврилова Т.А., Кудрявцев Д.В. Проблемы надёжности систем искусственного интеллекта в критически важных приложениях // Искусственный интеллект и принятие решений. 2023. № 4. С. 78-91.

3. Кузнецов А.С., Петрова М.В. Трансформация IT-образования в эпоху генеративного искусственного интеллекта // Высшее образование в России. 2024. Т. 33, № 6. С. 45-62.

4. Лавров С.С. Программирование. Математические основы, средства, теория. Санкт-Петербург: БХВ-Петербург, 2001. 320 с.

5. Николенко С.И., Кадурин А.А., Архангельская Е.О. Глубокое обучение: погружение в мир нейронных сетей. Санкт-Петербург: Питер, 2023. 480 с.

6. Chen M., Tworek J., Jun H. et al. Evaluating Large Language Models Trained on Code // arXiv preprint. 2021. arXiv:2107.03374.

7. Karpathy A. Vibe coding // X: [social network]. 2025. 3 Feb. URL: https://x.com/karpathy/status/1886192184808149383 (accessed: 15.06.2025).

8. Pearce H., Ahmad B., Tan B. et al. Asleep at the Keyboard? Assessing the Security of GitHub Copilot's Code Contributions // Proceedings of the 43rd IEEE Symposium on Security and Privacy (S&P). 2022. P. 754-768.

9. Peng S., Kalliamvakou E., Cihon P., Demirer M. The Impact of AI on Developer Productivity: Evidence from GitHub Copilot // arXiv preprint. 2023. arXiv:2302.06590.

10. Vaswani A., Shazeer N., Parmar N. et al. Attention Is All You Need // Advances in Neural Information Processing Systems. 2017. Vol. 30. P. 5998-6008.

Габулян А.Н.
Нижневартовский государственный университет
г. Нижневартовск, Россия

ИНТЕЛЛЕКТУАЛЬНЫЙ ПОИСК ПО ДОКУМЕНТАМ НА ОСНОВЕ RAG: АРХИТЕКТУРА, ИНДЕКСИРОВАНИЕ И ОЦЕНКА КАЧЕСТВА

Рост объёмов корпоративной документации – регламентов, инструкций, баз знаний, отчётов – делает поиск по ключевым словам всё менее удобным в повседневной работе. Один и тот же запрос пользователи формулируют по-разному: используют синонимы, сокращения, перефразируют, уточняют детали. При этом нужный ответ нередко «распылён» по нескольким документам или разделам. В результате поиск превращается не в быстрое получение ответа, а в последовательный просмотр выдачи, ручной отбор фрагментов и последующую проверку контекста, что замедляет принятие решений и повышает вероятность ошибок.

Большие языковые модели (LLM) умеют выдавать связный ответ на естественном языке, однако без надёжной опоры на проверяемые источники могут допускать недостоверные утверждения (галлюцинации) и не обеспечивать воспроизводимость результата для пользователя [1, с. 1; 7, с. 1]. Для снижения этой проблемы применяется подход Retrieval-Augmented Generation (RAG), в котором генерация ответа дополняется поиском релевантных фрагментов: сначала ретривер извлекает материалы из коллекции, затем модель формирует ответ, опираясь на найденный контекст как на «доказательства» [5, с. 1; 10, с. 1].

Цель работы – описать практическую архитектуру интеллектуального поиска по документам на основе RAG, выделить ключевые этапы индексирования и обработки запроса и предложить компактную методику оценки качества ретривера по метрикам hit-rate@k и MRR. Дополнительно приводится демонстрационный эксперимент, в котором показано, как параметры разбиения текста на фрагменты (chunking) и величина top-k влияют одновременно на качество поиска и на размер индекса.

Архитектура решения

Типичная RAG-система включает четыре логических слоя:

– слой данных (набор документов);
– ingestion pipeline (подготовка и индексирование);
– слой ретривера (поиск top-k фрагментов);
– слой генерации ответа (LLM) с оформлением ссылок на источники (рис. 1).

На практике компоненты разворачиваются как отдельные сервисы, что позволяет независимо масштабировать индексирование и обработку запросов. В рамках данной работы была реализована демонстрационная версия RAG-конвейера, ориентированная на воспроизводимый эксперимент и измерение качества ретривера. Коллекция включала 12 документов, приведённых к единому текстовому представлению; для каждого документа фиксировались идентификатор (doc_id), название и ссылка на источник. На этапе индексирования текст каждого документа разбивался на фрагменты фиксированной длины в словах с перекрытием, а затем для каждого фрагмента формировалось поисковое

представление и сохранялись метаданные, необходимые для трассировки результата (doc_id, номер фрагмента, диапазон слов и ссылка на первоисточник) [10, с. 2].

Поскольку цель эксперимента заключалась в сравнении параметров chunking и top-k, в качестве ретривера использован простой лексический подход TF-IDF с ранжированием по близости (косинусная мера) между вектором запроса и векторами фрагментов. Такой выбор позволил изолировать влияние параметров разбиения и глубины выдачи без зависимости от конкретной эмбеддинговой модели. Генератор ответа в рамках работы рассматривается как следующий слой системы (LLM), однако количественная оценка выполнялась именно для ретривера, то есть для качества подбора «доказательных» фрагментов, которые затем могут быть использованы генератором при формировании ответа [6, с. 1].

Ключевой принцип RAG-системы – чёткое разделение функций между компонентами. Ретривер решает задачу отбора: он должен находить действительно релевантные фрагменты, которые могут служить основанием для ответа. Генератор, в свою очередь, отвечает за представление результата: он формирует связный текст и сохраняет привязку каждого утверждения к использованным источникам, чтобы пользователь мог быстро проверить выводы.

Для промышленного применения одного «поиска по смыслу» недостаточно: критически важно работать с метаданными документов. Идентификатор и версия документа, раздел, дата обновления, а также права доступа (ACL) позволяют ограничивать выдачу только теми материалами, которые актуальны и доступны конкретному пользователю. Такой подход упрощает аудит (что именно было использовано при формировании ответа) и делает систему пригодной для эксплуатации в организациях с требованиями к контролю доступа и трассируемости результатов [3, с. 2; 8, с. 4; 10, с. 2].

Рис. 1. Архитектура интеллектуального поиска на основе RAG

Индексирование документов

Индексирование можно рассматривать как подготовку «поискового представления» коллекции: исходные файлы переводятся в единый, стабильный формат, с которым затем работает ретривер. Сначала выполняется извлечение текста из разных типов документов (PDF, DOCX, HTML), после чего данные приводятся к нормализованному виду: удаляются технические артефакты, выравниваются пробелы и переносы строк, устраняются проблемы кодировок. Для сложных PDF полезно сохранять связь извлечённого текста с первоисточником – как минимум номер страницы, а при необходимости и координаты фрагмента на странице. Это упрощает формирование корректных ссылок и последующую проверку источников пользователем.

Для повышения воспроизводимости был зафиксирован единый формат промежуточных данных. На входе пайплайна каждый документ приводился к «чистому тексту», после чего

сохранялись: doc_id, источник (URL), а также служебные признаки, позволяющие связать фрагменты с оригиналом. После разбиения формировались чанки, для которых хранились: идентификатор документа, порядковый номер чанка, длина в словах, параметры overlap, а также границы фрагмента (например, start_word и end_word). Такая структура метаданных нужна не только для фильтрации, но и для объяснимости: по ней можно однозначно восстановить, из каких частей коллекции была собрана контекстная «подложка» для ответа [10, с. 3].

В демонстрационном прототипе права доступа (ACL) рассматривались как часть целевой промышленной архитектуры и учитывались на уровне модели данных, однако в эксперименте использовалась единая коллекция без раздельных политик доступа. При переносе подхода в корпоративную среду эти же поля (раздел, версия, уровень доступа) позволяют ограничивать выдачу актуальными и разрешёнными документами и обеспечивать аудит результата, то есть возможность проверить, какие именно материалы использованы системой [3, с. 2].

Далее текст делится на фрагменты (чанки), которые станут элементами индекса. На практике часто используют фиксированный размер чанка с перекрытием (overlap): перекрытие снижает риск потери смысла на границах фрагментов и повышает вероятность того, что связанный контекст окажется в выдаче. Альтернативный вариант – разбиение по структуре документа (заголовки, абзацы, предложения), однако такой подход зависит от качества парсинга и может давать нестабильный результат на разнородных форматах. Для каждого чанка сохраняются служебные поля: идентификатор документа, раздел, версия, а также параметры доступа (ACL). Эти метаданные нужны для фильтрации и позволяют обновлять индекс инкрементально, не пересобирая коллекцию целиком [10, с. 3].

На следующем шаге для каждого чанка вычисляется векторное представление (эмбеддинг), после чего данные записываются в векторный индекс. Векторная база хранит связку «эмбеддинг + метаданные» и реализует быстрый поиск ближайших соседей (ANN), что особенно важно при росте объёма коллекции [2, с. 2; 4, с. 2]. В промышленной конфигурации часто добавляют гибридную схему (BM25 вместе с векторным поиском) и/или этап reranking, чтобы повысить устойчивость к редким терминам, аббревиатурам и техническим обозначениям (рис. 2) [9, с. 1; 10, с. 4].

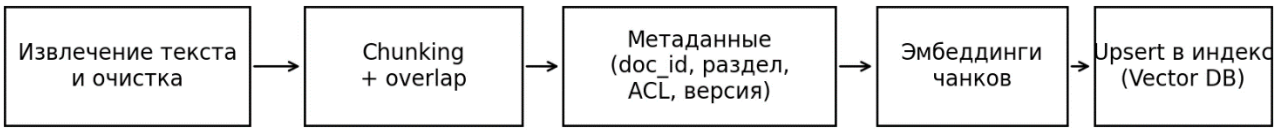


Рис. 2. Пайплайн индексирования документов

Обработка запроса и формирование ответа

На этапе обработки запроса система сначала переводит вопрос пользователя в векторное представление и использует его как ключ для поиска по индексу: вычисляется близость между вектором запроса и векторами чанков. На выходе формируется список из k наиболее релевантных фрагментов, которые затем собираются в единый контекст для генерации ответа. Выбор k задаёт компромисс между полнотой и «зашумлением»: слишком малое значение

повышает вероятность пропуска нужного факта, тогда как чрезмерно большое значение добавляет в контекст лишние, слабо связанные с вопросом фрагменты и ухудшает качество итогового ответа [6, с. 2; 10, с. 4].

При необходимости между первичным поиском и генерацией вводится дополнительный слой отбора. Наиболее распространённый вариант – reranking, когда найденные кандидаты пересортировываются более точной (но обычно более дорогой) моделью. Другой подход – формирование контекста с учётом разнообразия, например по схеме MMR (Maximal Marginal Relevance), чтобы не включать несколько почти одинаковых фрагментов и тем самым эффективнее использовать ограниченный объём контекста. После отбора формируется промпт: в него включаются инструкция (правила ответа), исходный вопрос и выбранные фрагменты. Далее LLM генерирует итоговый текст. Для повышения проверяемости результата в инструкциях стоит явно требовать ссылки на использованные источники и задать единый формат цитирования (рис. 3) [6, с. 3; 10, с. 5].

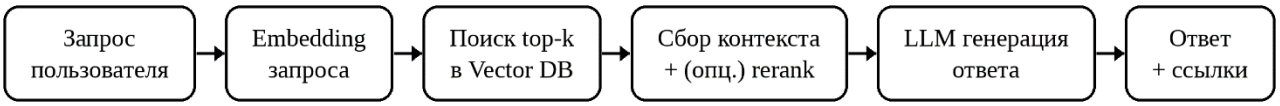


Рис. 3. Пайплайн обработки запроса в RAG

Методика оценки качества

Оценивание RAG-системы удобно рассматривать как две взаимосвязанные задачи: проверку качества поиска (retrieval) и проверку качества генерации (generation). В данной работе внимание сосредоточено на ретривере, поскольку именно он определяет, попадут ли в контекст фрагменты, на которые затем сможет опереться модель при формировании ответа. Если релевантные материалы не извлечены на этапе поиска, генератор не сможет корректно восстановить факт без риска “додумывания”. Для количественной оценки требуется тестовый набор вопросов и заранее подготовленный эталон релевантности – список документов или фрагментов, которые считаются правильными для каждого вопроса [6, с. 1].

В качестве базовых показателей качества использованы hit-rate@k и MRR. Метрика hit-rate@k фиксирует долю запросов, для которых в первых k результатах нашёлся хотя бы один релевантный элемент. Метрика MRR (Mean Reciprocal Rank) дополнительно учитывает порядок выдачи: она растёт, если первый релевантный результат появляется ближе к началу списка, и падает, если он смещается вниз. В совокупности эти показатели позволяют сравнивать настройки разбиения текста (chunking), значение top-k и оценивать, как изменение параметров влияет на долю «успешных» запросов и на ранжирование релевантных результатов при наличии шума в выдаче [6, с. 1].

Эксперимент и результаты

Чтобы наглядно оценить влияние параметров разбиения текста и величины top-k, был подготовлен небольшой экспериментальный корпус из 12 документов, посвящённых RAG и связанным практикам построения поисково-генеративных систем. На основе этого корпуса составлен набор из 12 тестовых вопросов; для каждого вопроса заранее определён перечень

релевантных документов, который использовался как эталон при расчёте метрик. В роли базового ретривера выбран лексический метод TF-IDF – как простая и воспроизводимая отправная точка, позволяющая сравнивать конфигурации без использования эмбеддингового поиска.

В ходе эксперимента варьировались два основных параметра: размер чанка (25, 40, 60 и 80 слов) при фиксированном перекрытии 10 слов и глубина выдачи top-k (3, 5 и 8). Дополнительно учитывалась «стоимость» индекса в виде числа сформированных чанков: уменьшение размера фрагмента увеличивает количество элементов индекса и, следовательно, может приводить к росту требований к памяти и времени поиска.

Итоговые значения метрик для нескольких наиболее показательных сочетаний параметров представлены в таблице.

Таблица

Метрики качества ретривера для выбранных конфигураций

chunk, слов	overlap, слов	top-k	hit-rate@k	MRR	чанков
25	10	3	0.917	0.819	42
40	10	3	0.917	0.875	24
40	10	5	1.000	0.896	24
80	10	5	1.000	0.896	12

Полученные результаты показывают, что увеличение top-k с 3 до 5 повышает hit-rate@k до 1.0 на демонстрационном наборе вопросов, а также улучшает MRR за счёт более раннего появления релевантного документа в выдаче. При этом уменьшение размера чанка увеличивает количество элементов индекса (например, 42 чанка при 25 словах против 12 при 80 словах), что может ухудшать производительность при масштабировании.

В реальной системе вместо TF-IDF следует применять эмбеддинги и векторный поиск, что повышает устойчивость к перефразированию и синонимам, а также рассмотреть гибридный режим (BM25 + векторы) и reranking [9, с. 1; 10, с. 4]. Тем не менее предложенная методика оценки и форма представления результатов остаются применимыми: можно сравнивать настройки chunking, параметры top-k, модели эмбеддингов и варианты reranking на одном и том же тестовом наборе.

Заключение

В статье рассмотрен практический подход к интеллектуальному поиску по документам на основе RAG, включающий этапы индексирования, извлечения контекста и формирования ответа с опорой на источники. Показано, что качество ретривера существенно влияет на итоговую надёжность системы, а параметры chunking и top-k определяют баланс между полнотой, шумом контекста и размером индекса.

Предложенная методика оценки (hit-rate@k и MRR) позволяет быстро получать измеримые результаты и сравнивать конфигурации системы. Дальнейшее развитие работы может включать использование эмбеддинговых моделей, гибридного поиска и reranking, а

также расширение оценки на уровень генератора (корректность фактов и точность цитирования) [6, с. 4].

Литература

1. Айсин Т.Р., Шамардина Т.В. Детекция галлюцинаций на основе внутренних состояний больших языковых моделей. 2025. Электронные библиотеки. URL: https://rdl-journal.ru/article/view/977 (дата обращения: 15.02.2026).

2. Анатомия векторного поиска в Elasticsearch. Хабр. URL: https://habr.com/ru/articles/807769 (дата обращения: 15.02.2026).

3. Архитектура нулевого доверия (Zero Trust Architecture). Хабр. URL: https://habr.com/ru/articles/932274 (дата обращения: 15.02.2026).

4. Векторные базы данных: простым языком про устройство и принцип работы. Хабр. URL: https://habr.com/ru/companies/tochka/articles/809493 (дата обращения: 15.02.2026).

5. Оболенский Д.М., Шевченко В.И. Использование метода RAG и больших языковых моделей в интеллектуальных образовательных экосистемах // Экономика. Информатика. 2024. URL: https://econom-inform-journal.ru/index.php/journal/article/view/390 (дата обращения: 15.02.2026).

6. Полное руководство по оценке компонентов системы RAG. Хабр. URL: https://habr.com/ru/articles/860390 (дата обращения: 15.02.2026).

7. Проблема «галлюцинирования» в больших языковых моделях. Хабр. URL: https://habr.com/ru/companies/sberbank/articles/812775 (дата обращения: 15.02.2026).

8. Установка, настройка и эксплуатация стэка OpenSearch в классической среде. Хабр. URL: https://habr.com/ru/articles/662527 (дата обращения: 15.02.2026).

9. BM25: как работает алгоритм ранжирования (перевод). Хабр. URL: https://habr.com/ru/articles/860830 (дата обращения: 15.02.2026).

10. RAG: от теории к практике. Основы и продвинутые техники. Хабр. URL: https://habr.com/ru/articles/871226 (дата обращения: 15.02.2026).

Датский А.К.
Волгоградский государственный университет
г. Волгоград, Россия

РАЗРАБОТКА ПРОТОТИПА СКЛАДНОЙ РАМЫ КВАДРОКОПТЕРА НА КАРБОНОВЫХ ТРУБКАХ

Интенсивное развитие беспилотных летательных аппаратов (БПЛА) мультироторного типа предъявляет новые требования к их конструкции, среди которых ключевыми становятся не только летные характеристики, но и удобство транспортировки, обслуживания и адаптации под различные задачи. Возможность быстрого складывания рамы существенно расширяет сферу применения аппарата, позволяя оператору оперативно разворачивать его в полевых условиях и переносить в компактном рюкзаке [2]. Использование карбоновых трубок в сочетании с узлами, изготовленными методом FDM–печати, является перспективным подходом, обеспечивающим высокую удельную прочность конструкции и гибкость при внесении изменений [1, 3]. Развитие аддитивных технологий открывает новые возможности для создания сложных геометрических форм, недоступных при традиционных методах обработки [6].

Проектируемый квадрокоптер относится к классу аппаратов с диагональю 10 дюймов и ориентирован на использование силовой установки на базе двигателей Sunnysky 2212, стека электроники SpeedyBee v4 и цифровой FPV-системы DJI O4 Air Unit (Lite). Основным требованием к раме является возможность складывания лучей для уменьшения габаритов при транспортировке в рюкзаке, при этом в разложенном состоянии конструкция должна обеспечивать жесткую и воспроизводимую фиксацию геометрии для стабильного полета [7].

В качестве базовой конструктивной схемы выбрана рама на двух параллельных карбоновых трубках диаметром 16 мм, расположенных вдоль продольной оси. Такая схема позволяет эффективно воспринимать изгибающие нагрузки и служит основой для крепления центрального модуля с электроникой и аккумулятором. Пространственная компоновка, представленная на рисунке 1, реализует комбинированную кинематику складывания: передние лучи вращаются вокруг оси, расположенной вблизи трубки, а задние – на вынесенных назад кронштейнах. Это обеспечивает компактное сложение лучей без взаимного перекрытия по высоте [4].

Ключевой проблемой при сопряжении жестких карбоновых трубок и печатных зажимов является разброс фактических диаметров трубок (15,97–16,03 мм) и усадка пластика при FDM-печати. Для компенсации этих отклонений и обеспечения равномерного обжима были разработаны промежуточные втулки из термопластичного полиуретана (TPU) твердостью 95A. В ходе экспериментального подбора геометрических параметров установлено, что оптимальное усилие фиксации и виброразвязка достигаются при номинальном внешнем диаметре втулки 18,5 мм (под посадочное отверстие зажима 18 мм) и внутреннем диаметре 16 мм. Усадка TPU при печати (около 0,2 мм) создает необходимый натяг, обеспечивающий фрикционное соединение без повреждения композитной трубки.

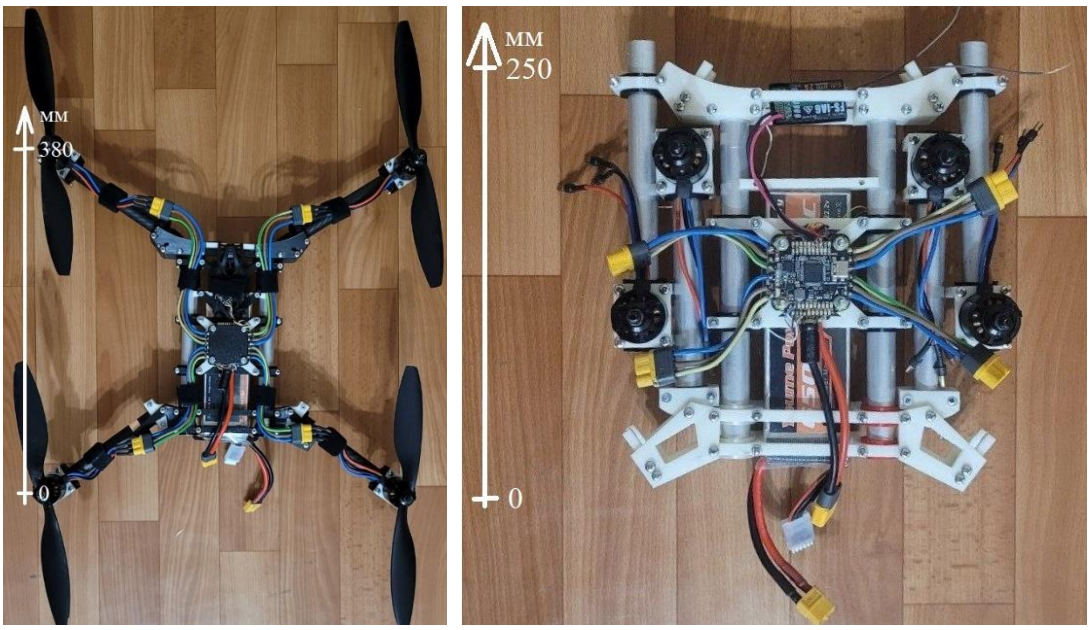


Рис. 1. Прототип складной рамы квадрокоптера в разложенном и сложенном состояниях

Наиболее критичным узлом, определяющим удобство и надежность эксплуатации, является механизм фиксации лучей в рабочем положении. Первоначальная геометрия защелки прямоугольной формы (рис. 2а) с ослабленным сечением показала низкую износостойкость: разрушение PET–G деталей наступало после 5–6 циклов складывания. Вторая итерация, выполненная по принципу трубной клипсы с равномерной радиальной частью (рис. 2б), позволила повысить ресурс, однако выявила недостаток в виде нестабильности положения из-за одноточечного крепления [5].

В третьей итерации (рис. 2в) данный недостаток был устранен путем введения дополнительного опорного выступа со второй осью крепления, что обеспечило кинематическую устойчивость узла и исключило его выворачивание при установке луча. Также было экспериментально подтверждено, что ориентация линий заполнения при 3D–печати, повторяющая контур силовой дуги защелки, значительно повышает ее стойкость к циклическим нагрузкам за счет более равномерного восприятия растягивающих напряжений.

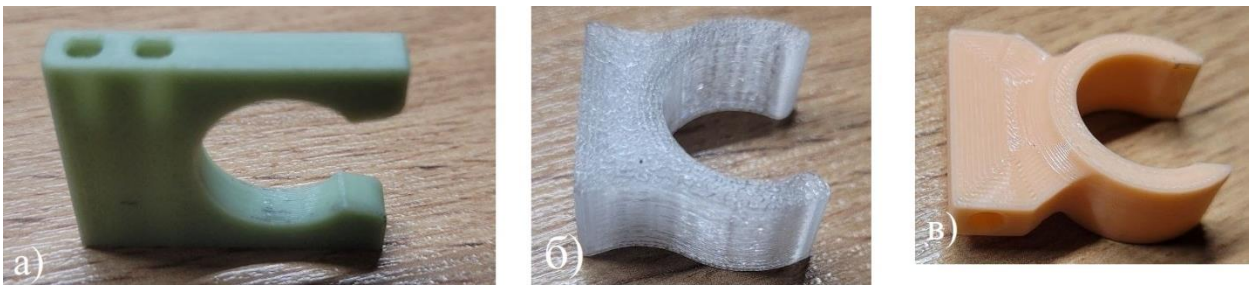


Рис. 2. Эволюция геометрии защёлки механизма фиксации лучей:
а) первая итерация прямоугольной формы с ослабленной зоной разгиба;
б) вторая итерация по принципу клипсы для труб с равномерной радиальной частью;
в) третья итерация с усилением и введением второй оси крепления

В итоговой конфигурации прототипа, помимо базового силового каркаса, были реализованы дополнительные печатные элементы: крышка зоны полетного контроллера для

стабилизации показаний барометра и переходная планка для жесткой установки видеосистемы DJI O4 Air Unit (Lite) со смещением вперед для оптимизации компоновки. Сборка прототипа подтвердила воспроизводимость геометрии и отсутствие люфтов в шарнирных соединениях при использовании оптимизированных TPU–втулок и доработанных защелок.

На рисунке 3 представлен моторный луч в сборе, наглядно демонстрирующий реализацию предложенных конструктивных решений. Здесь хорошо видно, что TPU–втулка (выделена серым цветом) устанавливается непосредственно в посадочное гнездо моторной пластины, охватывая карбоновую трубку по всему периметру. Такая схема крепления не только исключает прямой контакт «пластик–карбон», но и формирует упругий демпфирующий слой, который эффективно гасит высокочастотные вибрации, возникающие при работе двигателя и вращении винтов. Благодаря своей эластичности, втулка деформируется под нагрузкой, поглощая энергию колебаний и предотвращая ее передачу на плечо луча и далее на центральный модуль с чувствительной электроникой. Таким образом, применение TPU-втулок в сочетании с оптимизированной геометрией моторных пластин позволяет решить комплексную задачу: обеспечить надежную и регулируемую фиксацию, компенсировать технологические допуски и создать эффективную виброизоляцию силовой установки, что критически важно для стабильной работы полетного контроллера и качества FPV-изображения.

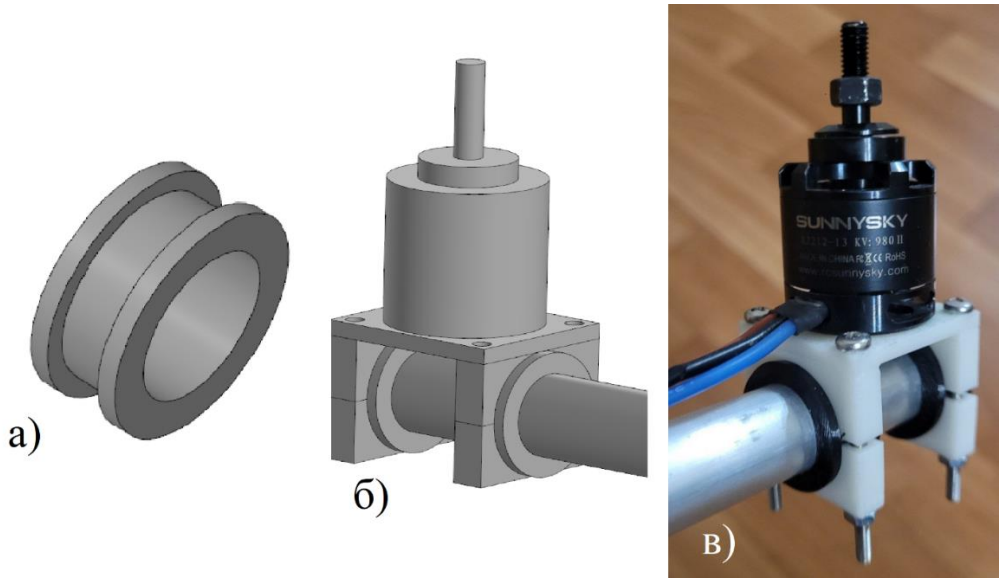


Рис. 3. Демонстрация конструкции и использования втулки из TPU для снижения вибраций и улучшения фиксации компонентов: а) модель втулки из TPU, используемой для фиксации и демпфирования вибраций; б) сборка трубки с креплением мотора, установленной на втулке; в) реальный вид установки втулки на трубке с мотором, готовой к эксплуатации

Таким образом, в ходе работы был реализован полный цикл разработки прототипа складной рамы квадрокоптера. Экспериментально подтверждена эффективность применения TPU–втулок в качестве интерфейсного слоя для компенсации технологических допусков и демпфирования вибраций. Итеративная модернизация механизма фиксации лучей показала, что его надежность определяется совокупностью рациональной геометрии, исключающей

концентраторы напряжений, кинематической устойчивостью и правильным выбором параметров печати. Полученные результаты могут быть использованы при проектировании аналогичных модульных конструкций БПЛА.

Литература

1. Лиханов О.О., Зайцева Т.А., Яндышев А.Д. Возможности использования композитных материалов в авиационной технике // Международный журнал гуманитарных и естественных наук. 2024. Т. 10, № 97. С. 7-10.

2. Тарасов Я.О. Квадрокоптеры: проектирование, конструирование и испытания // Известия Тульского государственного университета. Технические науки. 2023. № 5. С. 21-27.

3. Узбеков Р.Ш., Челноков В.В., Аверина Ю.М. Применение систем CAD/CAM/CAE в аддитивной технологии 3D-печати // Наука, техника и образование. 2023. Т. 4, № 92. С. 8-11.

4. Bennaceur S., Azouz N. Modelling and control of a quadrotor with flexible arms // Alexandria Engineering Journal. 2022. Vol. 65, no. 6. P. 209-231.

5. Croccolo D., Agostinis M., Fini S. Fretting Fatigue in Mechanical Joints: A Literature Review // Lubricants. 2022. Vol. 10, no. 4. P. 53.

6. Gibson I., Rosen D., Stucker B. Additive Manufacturing Technologies. 3rd ed. Cham: Springer, 2021. 498 p.

7. Goli S., Kurtulus D.F., Alhems L.M. Experimental study on efficient propulsion system for multicopter UAV design applications // Results in Engineering. 2023. Vol. 20. Article 101548.

Зайцев А.В.
Нижневартовский государственный университет
г. Нижневартовск, Россия

РАЗРАБОТКА СИСТЕМЫ АВТОНОМНОГО УПРАВЛЕНИЯ БПЛА НА ОСНОВЕ ИНТЕРПРЕТАЦИИ КОМАНД G-CODE ИЗ ВИЗУАЛЬНЫХ МАРКЕРОВ

Введение

Современное развитие беспилотных авиационных систем (БАС) характеризуется переходом от дистанционного управления к полной автономности. Особую сложность представляют задачи навигации в закрытых пространствах или зонах с подавленным сигналом ГНСС (GPS/ГЛОНАСС) [3, с. 101-112]. Одним из перспективных направлений решения данной проблемы является использование методов визуальной одометрии и навигации по внешним ориентирам (маркерам) [4].

В данной работе предлагается подход, при котором визуальный маркер (QR-код) выступает не только как точка позиционирования, но и как носитель исполняемого полетного задания. Использование синтаксиса G-code, традиционно применяемого в станках с ЧПУ и 3D-принтерах, позволяет унифицировать процесс программирования движений БПЛА и упростить интеграцию робототехнических комплексов в автоматизированные производственные системы [2].

1. Программно-аппаратная архитектура комплекса

Исследование проводилось на базе лаборатории БПЛА НВГУ с использованием платформы «Клевер» (COEX). Выбор данной платформы обусловлен её открытой архитектурой и интеграцией с ROS (Robot Operating System) [5].

Состав используемого комплекса:

- Полетный контроллер: COEX Pix (стек PX4).
- Бортовой компьютер: Raspberry Pi 4 (ОС Linux Ubuntu с преднастроенным образом Clover).
- Сенсоры: Камера для Raspberry Pi с 5Мп сенсором OV5647(для компьютерного зрения), лазерный дальномер (для удержания высоты).
- Среда разработки: Python 3, OpenCV [1], библиотеки MAVROS для взаимодействия с полетным контроллером.

2.1. Разработка системы интерпретации визуальных команд

Ключевой особенностью работы является программная реализация интерпретатора G-code, адаптированного для полетных задач. В QR-код записывается команда, определяющая траекторию движения и состояние аппарата.

Таблица

Подмножество реализованных команд G-code для БПЛА

Команда	Описание	Аргументы	Эквивалент в Clover API
G0 / G1	Перемещение в точку относительно дрона	X, Y, Z (метры)	navigate = rospy.ServiceProxy('navigate', srv.Navigate) navigate(x, y, z, frame_id='body')
G4	Задержка (ожидание)	P (миллисекунды)	rospy.sleep()
G92	Перемещение в точку по координатам (долгота, широта, высота)	X, Y, Z	rospy.ServiceProxy('navigate', srv.Navigate) navigate(x, y, z, frame_id='map')
M06	Смена цвета и эффекта LED-ленты. Цветовая схема: RGB Эффект: fill, blink, blink_fast, fade, wipe, flash, rainbow, rainbow_fill	R, G, B, effect	set_led = rospy.ServiceProxy('led/set_effect', srv.SetLEDEffect) set_led(r, g, b, effect)
M18	Посадка и разоружение	–	land = rospy.ServiceProxy('land', Trigger) land()
M112	Экстренная остановка	–	rosservice call /mavros/cmd/arming (выключение моторов)

Программный модуль, разработанный на языке Python, функционирует как отдельная ROS-нода. Алгоритм работы включает следующие этапы:

1. Захват видеопотока с камеры (частота 25 fps).
2. Распознавание QR-кода библиотекой OpenCV (QRCodeDetector) [5].
3. Декодирование текстовой строки и парсинг G-code.
4. Вызов соответствующих функций библиотеки clover.srv.

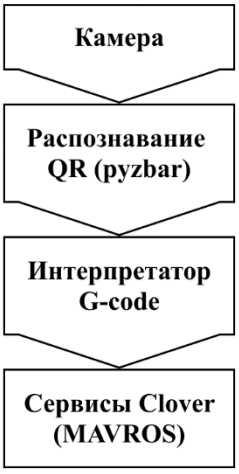


Рис. 1. Визуальное представление структуры алгоритма, «составлено автором»

2.2. Объяснение кода интерпретатора

Программная реализация обработки команд, заложенных в визуальные маркеры (QR-коды), базируется на принципах сервисориентированной архитектуры ROS (Robot Operating System) с открытым исходным кодом. В качестве примера рассмотрим алгоритм

интерпретации специализированной команды M06, отвечающей за световую индикацию БПЛА.

Ниже представлено описание логики работы программного модуля на языке Python:

1. Инициализация интерфейса взаимодействия: Для управления периферийным оборудованием (светодиодной лентой) используется объект rospy.ServiceProxy. Он устанавливает соединение с системным сервисом led/set_effect, определенным в пакете clover.srv. Это позволяет вызывать функции изменения состояния светодиодов как удаленные процедуры.

2. Обработка входных данных (Парсинг): Декодированная из QR-кода строка (b_data) подвергается декомпозиции методом split(). Согласно разработанному в данной работе протоколу, первый элемент результирующего списка классифицируется как идентификатор команды (аналог G-code), а последующие элементы, как ее аргументы (параметры).

3. Логический блок трансляции: При идентификации команды M06 система выполняет приведение типов данных (из строкового в целочисленный для цветовых каналов и строковый str для типа эффекта). Параметры r (red), g (green) и b (blue) определяют цветовое пространство индикации, а параметр effect, динамику отображения (например, статичное свечение, мигание или «радуга»).

4. Обеспечение отказоустойчивости: Код обернут в конструкцию try…except, что является критически важным для автономных систем. Это предотвращает аварийное завершение управляющего скрипта в случае считывания некорректно сформированного QR-кода (например, при недостаточном количестве аргументов в строке), обеспечивая живучесть программного обеспечения дрона.

```
119  set_led = rospy.ServiceProxy('led/set_effect', srv.SetLEDEffect)
120  b_data = 'M06'
121  try:
122      parts = b_data.split(" ")                                   # Разбиваем полученный текст с QR на фрагменты
123      cmd = parts[0]                                              # 1 часть всегда будет командой
124      args = parts[1:]                                            # Остальные части аргументами команды
125      if cmd == "M06":                                            # Если команда соответствует
126          set_led(r=int(args[0]), g=int(args[1]), b=int(args[2]), effect=str(args[3]))  # Присваеваем ленте цвета и эффекты
127          print("Команда", cmd, "выполнена!")                     # Отправляем в консоль, что команда выполнена
128          success = True                                          # Поднимаем флаг успеха
129  except Exception as e:
130      print(f"Error: {e}")
```

Рис. 2. Пример части кода интерпретатора команд

3. Экспериментальное исследование

Эксперимент проводился в условиях полигона НВГУ. На поверхности были размещены QR-коды размером 20x20 см, содержащие команды смены эффекта LED-ленты. Для отладки в программу была добавлена функция отметки распознанного QR-кода на полученном с камеры изображении и обновления этого изображения в отдельном топике. В виду ограничения полетной зоны в лаборатории и невозможностью запуска в открытом пространстве, было решено проводить эксперимент используя только QR-коды смены эффекта LED-ленты, для визуального отслеживания отклика программы.

Методика эксперимента: БПЛА совершал взлет в автономном режиме, после чего входил в режим поиска маркера. При обнаружении кода «M06 255 0 0 fill» аппарат выполнял

смену цвета LED-ленты на статический красный. В ходе испытаний фиксировалась точность распознавания кода в зависимости от высоты (Z) и освещенности.

Результаты:

- Стабильное считывание кода обеспечено на высоте до 2.5 метров при разрешении камеры 640x480.

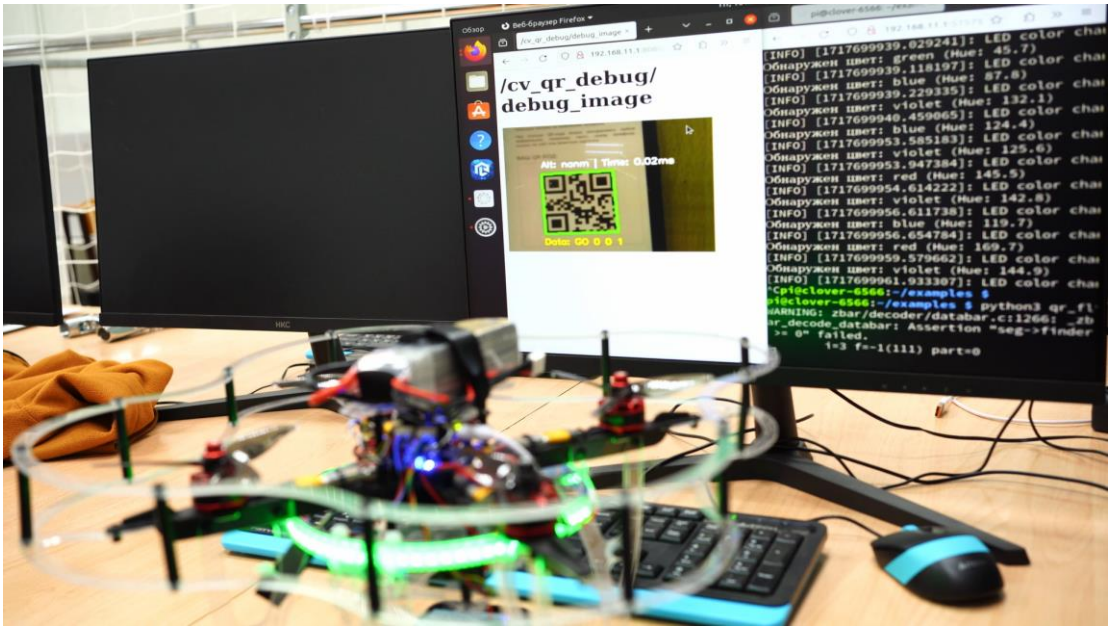


Рис. 3. Пример работы программы

- Среднеквадратичное отклонение при позиционировании над маркером составило ±7 см, что является допустимым для систем данного класса.
- Время интерпретации команды после захвата кадра составило не более 120 мс.

Заключение

В ходе работы была успешно реализована система управления БПЛА «Клевер», использующая QR-коды в качестве динамических носителей полетных заданий. Использование G-code протокола доказало свою эффективность для описания простых навигационных миссий. Предложенная система может быть масштабирована для задач инспекции складов, где дрон перемещается от одного стеллажа к другому, считывая инструкции непосредственно с маркировки стеллажей.

Литература

1. Кэлер А., Брэдски Г. Изучаем OpenCV 3. Разработка программ компьютерного зрения на C++ с применением библиотеки OpenCV. Издательство «ДМК Пресс», 2017. 826 с.

2. Майоров Н.Н., Костин А.С. Автоматизация процесса идентификации объектов при выполнении автономных полетных заданий беспилотной авиационной системой // Известия ТулГУ. Технические науки. 2022. № 2. С. 640-644.

3. Першина Ж.С., Каздорф С.Я., Лопота А.В. Методы визуальной навигации мобильного робота и построения картографических моделей внешней среды // Автометрия. 2019. Т. 55, № 2. С. 92-102.

4. Телегин А.И., Гусев Е.В., Кодкин В.Л., Ширяев В.И. Х3d-прототипирование моделей тел манипуляционных роботов // Вестник ЮУрГУ. Серия: Компьютерные технологии, управление, радиоэлектроника. 2024. № 4. С. 16-30.

5. Черноножкин В.А., Половко С.А. Система локальной навигации для наземных мобильных роботов // Научно-технический вестник информационных технологий, механики и оптики. 2008. № 57. С. 13-22.

Кнестяпин Р.А.
Самарский национальный исследовательский университет им. академика С.П. Королева
г. Самара, Россия

АЛГОРИТМЫ ОПРЕДЕЛЕНИЯ ПРОСТРАНСТВЕННОГО ПОЛОЖЕНИЯ БЕСПИЛОТНОЙ НАЗЕМНОЙ ПЛАТФОРМЫ С ИСПОЛЬЗОВАНИЕМ КОМПЬЮТЕРНОГО ЗРЕНИЯ

В современной технике нейронные сети занимают ключевое место в обработке видеоданных. Разработка методов поиска объектов на видеопотоке и определения их пространственного положения является актуальной задачей для навигации беспилотных наземных платформ. Цель данной работы – изучение и анализ возможностей использования нейронных сетей для обнаружения объектов и оценки расстояния до них, а также определение координат платформы на основе распределения обнаруженных объектов. В работе решаются задачи создания и обучения модели обнаружения объектов на базе YOLO, оценки расстояния с помощью карт глубин MiDaS и разработки алгоритма локализации платформы методом наименьших квадратов. Научная новизна работы заключается в экспериментальной оценке применимости связки методов (YOLOv8x + MiDaS) для задач навигации наземных платформ в условиях ограниченного пространства. В отличие от существующих работ, основное внимание уделяется не просто детекции, а количественному анализу погрешности калибровки карт глубин на малых (до 4 м) и средних (6–8 м) дистанциях, а также оценке влияния ошибок детекции и измерения глубины на конечную точность локализации платформы методом наименьших квадратов. Практическая значимость заключается в получении пороговых значений применимости монокулярного метода оценки глубины для навигации внутри помещений.

Создание и обучение модели обнаружения объектов

Для эксперимента использован видеопоток, снятый с платформы, движущейся внутри помещения (рис. 1). Необходимо было детектировать объекты окружения: стулья, столы, колонны и т. п. Был создан датасет из кадров видеоряда, размеченных с помощью сервиса Roboflow (https://clck.ru/3S6uig). Пример размеченного изображения показан на рисунке 2. Для расширения выборки использован дополнительный открытый датасет Indoor-objects (https://clck.ru/3S6uoA).

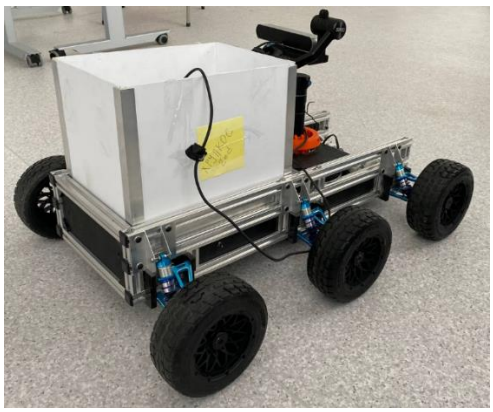


Рис. 1. Используемая платформа с установленной камерой

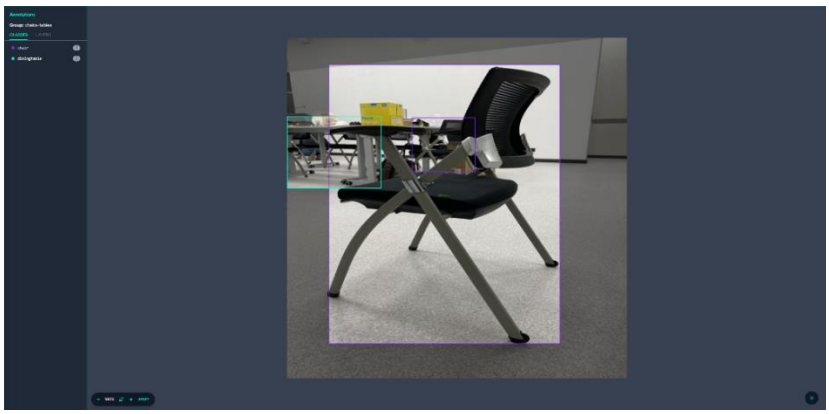


Рис. 2. Пример размеченной фотографии из датасета

Обучение проводилось на базе архитектуры YOLOv8x – одной из наиболее точных версий детектора семейства YOLO [4; 5]. Использовалась видеокарта NVIDIA RTX 4070Ti. На рисунках 3 и 4 показаны графики потерь и метрик после обучения на собственном и на расширенном датасетах соответственно. Результаты обучения показывают, что исходный датасет обладает недостаточной репрезентативностью: модель показывает высокие потери и низкую точность. Расширенный датасет позволил снизить потери и улучшить метрики.

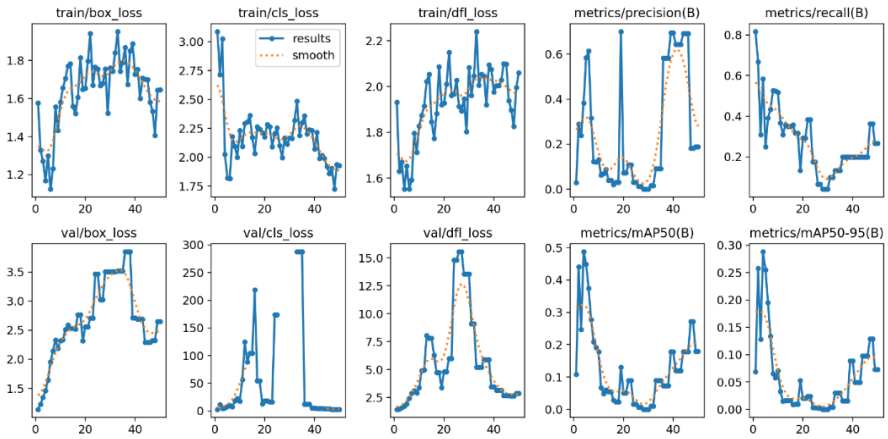


Рис. 3. Графики потерь и метрики после обучения на собственном датасете, насыщение кривых потерь при высоких значениях указывает на недостаточную репрезентативность выборки

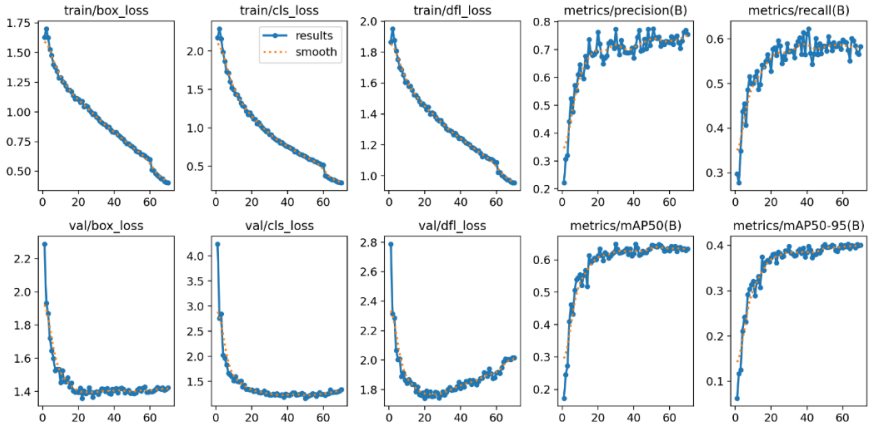


Рис. 4. Графики потерь и метрики после обучения на расширенном датасете

Сравнение работы моделей на одном кадре (рис. 5-7) показало, что модель, обученная только на собственном датасете, генерирует много ложноположительных срабатываний (рис. 5). Дообучение на расширенном датасете Indoor-objects повышает качество детекции, но всё ещё уступает предобученной на COCO модели YOLO (рис. 7). Тем не менее, даже улучшенная модель позволяет выделить большинство значимых объектов.

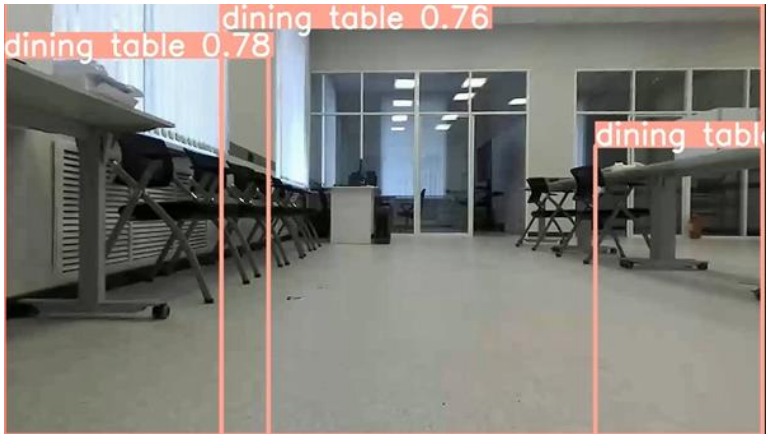


Рис. 5. Результат модели, обученной на собственном датасете

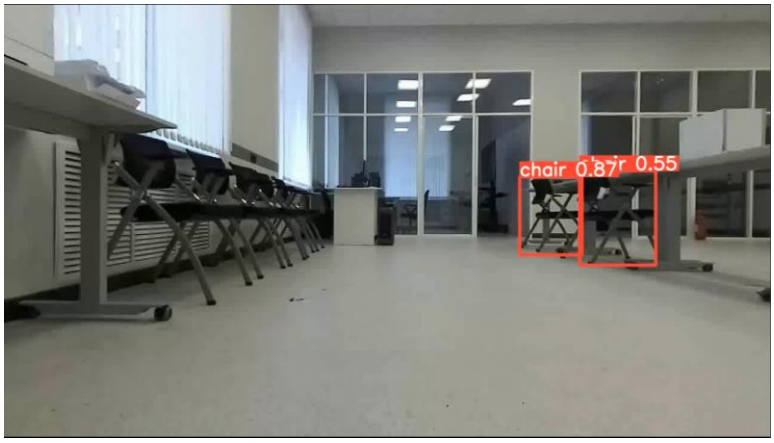


Рис. 6. Результат модели, обученной на расширенном датасете

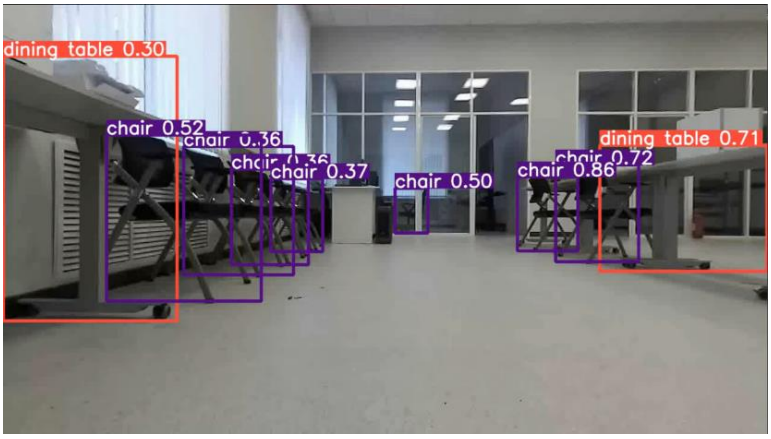


Рис. 7. Результат предобученной модели COCO

Оценка расстояния до объектов с использованием карт глубин

Для определения расстояния от камеры до обнаруженных объектов применялся алгоритм построения карт глубины MiDaS [2]. Подробное описание реализации и последней версии 3.1 доступно в официальном репозитории Intel Labs (https://clck.ru/3S6uoV). Он возвращает значения глубины в условных единицах, поэтому требуется калибровка. По изображению на рисунке 7 была построена карта глубины (рисунок 8). Реальные расстояния до двух стульев составляли 3,75 м и 3,2 м, а соответствующие значения на карте глубины – 1620,5 и 1845,6 усл. ед. Используя фокусное расстояние камеры fx = 699,73, по формуле

$$distance = \frac{fx \times scale}{depth\_distance}$$

был вычислен масштабный коэффициент $scale \approx 8{,}56$.

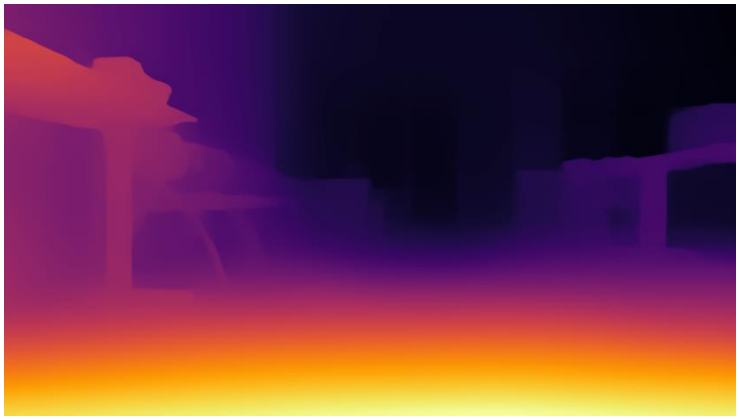


Рис. 8. Карта глубины калибровочного изображения

С помощью полученного коэффициента оценены расстояния до других объектов. Результаты представлены в таблице. Видно, что для объектов, удалённых до 4 м, погрешность не превышает 7%. Однако при увеличении дальности (6–8 м) ошибка возрастает до 50%, что говорит о необходимости совершенствования методики либо использования дополнительных датчиков для больших дистанций. Полученные результаты согласуются с современными исследованиями, подтверждающими высокий потенциал MiDaS для задач метрических измерений, однако требующими осторожного подхода к масштабированию на разные дистанции. Аналогичные выводы о границах применимости монокулярной глубины и необходимости корректной калибровки представлены в работах [2; 3].

Таблица

Оценка погрешности вычисления расстояния

Реальное расстояние, м	Предсказанное расстояние, м	Погрешность, %
3,75	3,697	1,4
3,2	3,246	1,4
1,0	1,072	7,2
3,4	3,375	0,7
3,7	3,57	3,0
7,86	3,69	53
6,3	3,4	46

Определение координат платформы методом наименьших квадратов

Имея координаты нескольких неподвижных объектов в глобальной системе и измеренные расстояния до них (полученные по картам глубин), можно оценить местоположение платформы. Задача решается методом наименьших квадратов (МНК). Пусть известны координаты (x_i, y_i) объектов, а измеренные расстояния до платформы равны d_i. Необходимо найти координаты (x,y), минимизирующие сумму квадратов невязок:

$$S(x,y) = \sum_{i=1}^{n}(\widehat{d_i} - \sqrt{(x - x_i)^2 + (y - y_i)^2})^2.$$

Минимизация выполнялась численно с помощью библиотеки SciPy (функция *least_squares*). Начальное приближение задавалось произвольно. Использование монокулярной глубины совместно с оценкой положения платформы рассматривается в работе [1].

Для тестирования использован кадр, представленный на рисунке 9. На нём детектированы три объекта с известными координатами (привязка выполнена относительно точки старта платформы, принятой за (0,0)).

Расстояния до объектов, измеренные с помощью карты глубины, составили: 1,78 м до объекта в точке (-1,6; 4,3); 2,25 м до объекта (-1,6; 3,8); 3,21 м до объекта (-1,6; 3,3).

Результат оценки координат платформы: (−0,256;5,518). Фактическое положение платформы в момент съёмки – (0,5;5,3). Расхождение между оценкой и реальными координатами равно примерно 0,79 м. Данная точность, хотя и уступает современным комплексным системам позиционирования (например, в работе [1] приведена ошибка позиционирования 0,104 м), является вполне приемлемой для базового навигационного решения и открытой для дальнейших улучшений.

Полученная точность (погрешность менее 1 м) является приемлемой для задач навигации в помещении. Уменьшить ошибку можно за счёт использования большего числа опорных объектов и повышения точности измерения расстояний.

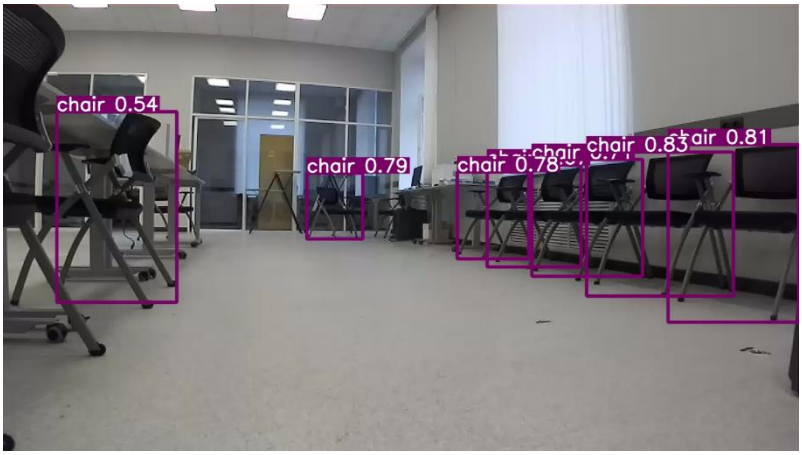


Рис. 9. Кадр для тестирования с выделенными объектами

Заключение

В ходе работы проведён анализ методов обнаружения объектов и построения карт глубины. Создана и обучена модель YOLO для детекции предметов в условиях помещения,

однако качество детекции ограничено объёмом обучающей выборки. Исследована возможность оценки расстояния с помощью MiDaS: метод обеспечивает приемлемую погрешность (менее 7%) только на дистанциях до 4 метров. При увеличении расстояния до 6–8 метров погрешность возрастает до 50%, что является критическим ограничением монокулярного подхода без дополнительной калибровки или сенсоров. Разработана программа локализации платформы на основе МНК, тестирование на реальных данных дало погрешность около 0,79 м. Перспективность интеграции методов компьютерного зрения, таких как YOLO и MiDaS, в навигационные системы подтверждается рядом работ последних лет [1-5]. Дальнейшие исследования должны быть направлены на улучшение качества детекции за счёт расширения датасета и компенсацию нелинейности карт глубины.

Литература

1. Liang C., Wang J., Li S., Sou K.W., Luo X., Ding W. MonoLDP: LED Assisted Indoor Mobile Bot Monocular Depth Prediction and Pose Estimation System // 2025 IEEE International Conference on Robotics and Automation (ICRA). 2025. P. 12251-12257. https://doi.org/10.1109/ICRA55743.2025.11128305

2. Ranftl R., Lasinger K., Hafner D., Schindler K., Koltun V. Towards Robust Monocular Depth Estimation: Mixing Datasets for Zero-shot Cross-dataset Transfer // IEEE Transactions on Pattern Analysis and Machine Intelligence. 2022. Vol. 44. № 3. P. 1623-1637. https://doi.org/10.1109/TPAMI.2020.3019967

3. Tang Y., Zhao C., Wang J., Zhang C., Sun Q., Zheng W.X., Du W., Qian F., Kurths J. Perception and navigation in autonomous systems in the era of learning: a survey // IEEE Transactions on Neural Networks and Learning Systems. 2023. Vol. 34. № 12. P. 9604-9624. https://doi.org/10.1109/TNNLS.2022.3167688

4. Terven J., Córdova-Esparza D. A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS // Machine Learning and Knowledge Extraction. 2023. Vol. 5. № 4. P. 1680–1716. https://doi.org/10.3390/make5040083

5. Wang C.Y., Bochkovskiy A., Liao H.Y.M. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2023. P. 7464-7475. https://doi.org/10.1109/CVPR52729.2023.00721

Лагерев А.П.
Иркутский государственный университет
г. Иркутск, Россия

АНАЛИЗ УЯЗВИМОСТЕЙ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ 2020–2025 ГОДОВ

В современном цифровом мире кибербезопасность становится одной из ключевых проблем, определяющих стабильность и развитие информационного общества. Согласно данным, собранным в рамках официального репозитория уязвимостей (CVE), за период 2020–2025 годов наблюдается экспоненциальный рост количества зарегистрированных уязвимостей в программном обеспечении. Количество угроз увеличилось с 14400 в 2020 году до 48112 в 2025 году что представляет собой более чем трехкратный рост за шестилетний период.

Этот тренд вызывает серьезную обеспокоенность среди специалистов по информационной безопасности, поскольку каждая новая уязвимость представляет потенциальную угрозу для инфраструктуры организаций и конфиденциальности данных пользователей. Несмотря на то, что средний показатель критичности угроз остается относительно стабильным в диапазоне 6,5–6,7 по шкале CVSS v3.1 (Common Vulnerability Scoring System), абсолютное количество высококритичных уязвимостей продолжает расти, что требует пересмотра к управлению рисками и усилению мер защиты.

Целью данного исследования является проведение комплексного анализа существующих киберугроз за период 2020–2025 годов на основе данных из официального репозитория уязвимостей CVE, выявить ключевые тенденции и разобрать рекомендации для повышения уровня безопасности инфекционных систем.

Для достижения поставленной цели необходимо решить следующие задачи:

1. Проанализировать динамику количества угроз за указанный период и выявить факторы, влияющие наих рост.
2. Исследовать распределение угроз по уровням критичности и определить тенденции в изменении структуры уязвимостей.
3. Выявить наиболее уязвимые вендоры и продукты, а также определить причины их повышенной уязвимости.
4. Проанализировать распространенные типы уязвимостей (согласно классификации CWE) и векторы атак.
5. Изучить влияние угроз на конфиденциальность, целостность и доступность информационных систем.

Для проведения настоящего исследования использовался официальный репозиторий уязвимостей CVE Project (Common Vulnerabilities and Exposures), который представляет собой наиболее авторитетный и полный источник информации об уязвимостях в программном обеспечении. Данные были получены из официального репозитория cvelistV5, который содержит структурированные данные в формате JSON.

Данные за период исследуемый период были отфильтрованы и обработаны для исключения дубликатов и некорректных записей. В результате обработки было получено 177299 уникальных записей об уязвимостях, что составило основу для последующего анализа.

Для сбора и обработки данных была разработана специализированная система, состоящая из нескольких этапов:

1. Распаковка и подготовка данных: использование архивов cvelistV5 для получения исходных данных, распаковка и структурирование файловой системы.
2. Парсинг данных: извлечение ключевой информации из структурированных JSON-документов, включая данные о вендорах, продуктах, версиях, метриках CVSS и типах уязвимостей по классификации CWE.
3. Очистка и нормализация данных: удаление дубликатов и некорректных записей, нормализация названий вендоров и продуктов, преобразование текстовых данных в структурированный формат.
4. Агрегация данных: группировка уязвимостей по годам, агрегация по уровням критичности, создание сводных таблиц для вендоров, продуктов и типов угроз.

Для анализа собранных данных были применены следующие методы:

1. Статистический анализ: расчет статистических показателей, анализ распределения угроз по годам, вычисления доли угроз по уровням критичности.
2. Визуализация данных: построение временного ряда, создание круговых диаграмм, гистограмм и столбчатых диаграмм для наглядного представления результатов анализа.
3. Сравнительный анализ: сопоставление данных по различным категориям для выявления закономерностей и трендов.
4. Корреляционный анализ: исследование взаимосвязей между различными параметрами угроз.

Для оценки угроз и их критичности использовались следующие метрики:

1. CVSS: базовый балл (0–10), уровень критичности (CRITICAL, HIGH, MEDIUM, LOW), векторная строка.
2. Классификация типов угроз.
3. Векторы атак: сетевые, локальные, атаки в соседней сети, физические атаки.
4. Воздействие на систему: конфиденциальность, целостность, доступность.

Анализ данных за период 2020–2025 годов выявил тревожную тенденцию экспоненциального роста количества зарегистрированных уязвимостей в программном обеспечении. Согласно полученным результатам, количество угроз увеличилось с 14400 в 2020 году до 48112 в 2025 году. Динамика роста угроз по годам представлена на рисунке.

Анализ данных показывает, что рост угроз не является линейным, а имеет ускоряющийся характер.

Рис. Количество угроз по годам (2020-2025), составлено автором

Линейная регрессия, проведенная на основе данных за исследуемый период, показывает устойчивый тренд с коэффициентом корреляции $R^2 = 0{,}987$, что указывает на высокую степень достоверности прогноза. Экстраполяция данных на 2026 год позволяет предположить, что количество угроз может превысить 56000, что требует адаптации мер безопасности к растущему объему угроз.

Такой рост зарегистрированных уязвимостей можно объяснить ускорением цифровизацией во время пандемии и после нее, приведшее к значительному расширению атакуемой поверхности. Согласно исследованиям специалистов, переход на удаленную работу и активное внедрение облачных технологий увеличило количество потенциальных точек входа для злоумышленников на 40–60% [4, с. 528]. Каждое новое приложение, сервис или устройство может иметь уязвимости, которыми могу воспользоваться злоумышленники.

Также рост зарегистрированных уязвимостей можно объяснить повышением осведомленности о безопасности, ростом количества исследователей безопасности и улучшением методов обнаружения уязвимостей. По данным российского рынка труда в сфере информационной безопасности, количество исследователей выросло на 35% с 2020 года [2, с. 50]. Развитие автоматизированных инструментов анализа безопасности, таких как статические и динамические анализатора кода, позволило значительно повысить эффективность обнаружения уязвимостей [5, с. 1130].

На повышение осведомленности также повлияло повышение требование к организациям, которое в свою очередь привило к увеличению количества проводимых аудитов и тестов на проникновение.

Анализ распределения угроз по уровням критичности за данный период интересную структуру, которая имеет важные практические последствия для управления рисками информационной безопасности. Согласно полученным данным, распределение угроз по уровню критичности представлено в таблице 1.

Таблица 1

Распределение угроз по уровням критичности (2020–2025), составлено автором

Уровень критичности	Процент от общего числа, %
CRITICAL (9.0-10.0)	8,3
HIGH (7.0-8.9)	37,6
MEDIUM (4.0-6.9)	49,2
LOW (0.1-3.9)	4,9

Данные таблицы 1 показывают, что подавляющее большинство угроз (86,8%) имеют средний и высокий уровни критичности. Это указывает на то, что большинство угроз представляют существенную угрозу для информационных систем и требуют своевременного реагирования.

Интересным наблюдением является относительно небольшая доля критичности угроз (8,3%), что может интерпретироваться как положительный тренд. Однако важно учитывать, что даже небольшой процент критических угроз может нанести непоправимый вред информационным систем, и такие уязвимости требуют внимательного рассмотрения и мониторинга.

Анализ данных выявил десятку разработчиков программного обеспечения (вендоров), на которые приходится наибольшее количество зарегистрированных уязвимостей. Данные представлены в таблице 2

Таблица 2

Топ-10 вендоров по количеству уязвимостей (2020–2025), составлено автором

Вендор	Количество уязвимостей	Процент от общего числа уязвимостей, %
Linux	9967	5.6
Adobe	3487	2
Oracle Corporation	2749	1,6
Google	2723	1,5
IBM	2575	1,5
Cisco	2043	1,2
SourceCodester	1596	0,9
Qualcom, Inc.	1457	0,8
Sasung Mobile	1244	0,7
Неизвестные разработчики	4229	2,4

Данные данной таблицы показывают, что на долю приходится 17,7% от общего числа уязвимостей. При это лидером является сообщество разработчиков операционной системы Linux с 9967 уязвимостями, что составляет 5,6% от общего числа.

Такое количество угроз у Linux можно объяснить открытостью исходного кода: доступность исходного кода позволяет исследователям безопасности находить уязвимости, но также этим могут воспользоваться и злоумышленники. Linux также является основой для большинства серверов, облачных платформ и встраиваемых систем, и поэтому является целью для атак и исследований безопасности.

Также за исследуемый период были выявлены 10 наиболее распространенных типов уязвимостей, представленные в таблице 3.

Таблица 3

Топ-10 типов уязвимостей (2020–2025), составлено автором

CVW ID	Название уязвимости	Количество уязвимостей	Процент от общего числа уязвимостей, %
CWE-79	Cross-Site-Scripting (XSS)	16285	9,2
CWE-89	SQL Injection	4727	2,7
CWE-862	Missing Authorization	3856	2,2
CWE-20	Improper Input Validation	3785	2,1
CWE-78	OS Command Injection	2466	1,4
CWE-352	Cross-Site Request Forgery (CSRF)	2336	1,3
CWE-434	Unrestricted Upload of File with Dangerous Type	2313	1,3
CWE22	Path Traversal	2223	1,3
CWE-287	Improper Authentication	1962	1,1
CWE-798	Use of Hard-coded Credentials	1798	1

Эти данные показывают, что на долю приведенных уязвимостей приходится около четверти (23,6%) от общего числа уязвимостей, а лидером является уязвимость типа Cross-Site Scripting (CWE-79).

Лидерство данной уязвимости обуславливается ростом количества веб-приложений, сложностью этих приложений и недостаточной валидацией пользовательского ввода. Современные веб-приложения становятся более комплексными, что увеличивает вероятность проявления уязвимости. Например, разработчики могут использовать в своем продукте библиотеку, которая содержит уязвимость. Также для разработчиков главным является работа их приложения, а безопасности уделяется недостаточно времени, что также приводит к увеличению количество уязвимостей.

Результатом анализа данных также является таблица 4, в которой представлены распределения угроз по векторам атак.

Таблица 4

Распределение угроз по векторам атак (2020–2025), составлено автором

Вектор атаки	Процент от общего числа, %
NETWORK	73,8
LOCAL	21,3
ADJACENT_NETWORK	3,9
PHYSICAL	1

Данная таблица показывает, что подавляющее большинство угроз (73,8%) имеют сетевой вектор атаки. Это указывает на эксплуатацию большинства уязвимостей удаленно через сеть, что значительно увеличивает их потенциальную опасность.

Такое доминирование сетевых атак можно объяснить распространением облачных технологий и сервисов, а также развитием интернета вещей (IoT). Так внедрение облачных сервисов и сервисов для удаленной работы создало огромное количество сетевых интерфейсов, доступных через интернет, который представляет потенциальную точку входа для злоумышленников. Развитие IoT также создало миллионы новых сетевых точек входа, так как многие устройства имеют недостаточный уровень защиты и могут быть использованы для атак на другие системы.

Также был проведён анализ угроз по воздействию их на конфиденциальность информации. Данные представлены в таблице 5.

Таблица 5

Воздействие угроз на конфиденциальность (2020–2025)

Уровень воздействия	Процент от общего числа, %
HIGH	44
LOW	31,7
NONE	24,3

Эти данные показывают, что 44% угроз оказывают высокое воздействие на конфиденциальность информации, что составляет около 80 тысяч уязвимостей. Это указывает на то, что защита конфиденциальности информации является одной из ключевых задач в области информационной безопасности.

Высокая доля угроз с высоким воздействием на конфиденциальность информации может обусловлена следующими факторами:

1. Рост объема обрабатываемых данных: количество данных, обрабатываемых в частных и государственных организациях, значительно выросло за анализируемый период. По данным исследований, объем данных в средней организации увеличился на 200% за период 2020–2025 годов [5]. Большой объем данных создает больше возможностей для утечки информации.

2. Увеличение ценности конфиденциальной информации: конфиденциальная информации становится более ценной на черном рынке, что увеличивает мотивацию злоумышленников к поиску уязвимостей, позволяющих получить доступ к такой информации [1, с. 62].

Также следует заметить, что почти четверть всех угроз (24,3%) за рассматриваемый период не оказывает никого воздействия на конфиденциальность информации. Это может быть связано с уязвимостями, которые влияют только на целостность и/или доступность информационных систем, но не приводят к утечке информации.

Все эти данные указывают на необходимость усиления информационной безопасности, используя следующие средства:

1. Усиление сетевой безопасности путем внедрения современных систем сетевой защиты, таких как сертифицированные межсетевые экраны с глубокой проверкой пакетов и систем предотвращения вторжений.

2. Регулярное обновление ПО и мониторинг уязвимостей версий ПО с возможностью отката обновления.

3. Усиление защиты конфиденциальности информации внедрением современных методов шифрования и контроля доступа к информационным ресурсам.

4. Внедрение комплексного подхода к управлению уязвимостями путем внедрения систем управления уязвимостями, которые позволяют приоритизировать угрозы на основе их критичности и потенциального ущерба

Проведенный анализ данных о уязвимостях за период 2020–2025 годов позволил выявить ключевые тенденции в эволюции угроз и разработать практические рекомендации для повышения уровня информационной безопасности. Основные выводы анализа:

1. Наблюдается ускоряющийся рост количества угроз, что требует адаптации мер безопасности к увеличивающемуся объему угроз.

2. Подавляющее большинство угроз имеют сетевой вектор атаки, что указывает на необходимость усиления сетевой безопасности.

3. Концентрация уязвимостей в продуктах ведущих вендоров требует особого внимания к обновлению и защите этих продуктов.

Полученные результаты имеют как теоретическое, так и практическое значение. Теоретически, исследование вносит вклад в понимание эволюции уязвимостей и факторов, влияющих на их динамику. Практически, результаты исследования могут быть использованы для разработки стратегий и тактик защиты информационных систем от киберугроз.

Литература

1. Бакеев Р.Н., Кузьмин В.Н., Менисов А.Б., Сабиров Т.Р. Метод определения уязвимостей программного кода на основе кластерного анализа и контекстной адаптации больших языковых моделей // Информационно-управляющие системы. 2025. № 4. С. 58-70.

2. Шестаков В.А., Чеботарь А.С., Наметившиеся тенденции в развитии киберпреступности // Вестник Воронежского государственного университета. Серия: Право. 2025. № 3. С. 45-58.

3. Gopal S.S., et al. Mining Five Years of Actively Exploited Vulnerabilities // Proceedings of the 2025 IEEE Symposium on Security and Privacy. IEEE, 2025. P. 1123-1138.

4. Hassan S.A.H., Rasheed A.A., Mousa A.A., Hussein Z.A., Ambudkar B. The Rising Cost of Cyberattacks: Trends and Impacts across Industries // HighTech and Innovation Journal. 2025. Vol. 6, № 2. P. 524-536.

5. Tejasree G., Nithin G., Sai Priya K., Ajay B., Santosh B. Advanced Machine Learning Framework for Detecting Known and Zero-Day Software Vulnerabilities // International Journal of Communication Networks and Information Security. 2025. Vol. 17, No. 5. P. 205-212.

Масалимова В.Р., Насонов И.П., Ерохин В.В.
Иркутский государственный университет путей сообщения
г. Иркутск, Россия

СРАВНИТЕЛЬНЫЙ АНАЛИЗ АЛГОРИТМОВ ПОИСКА В БОЛЬШИХ ОБЪЕМАХ ДАННЫХ

В условиях быстрой цифровизации анализ и обработка больших объемов данных становится неотъемлемой частью всех вычислительных процессов. Огромное значение имеет определение оптимального алгоритма поиска, так как они будут определять эффективность работы программных комплексов, поскольку операции поиска выполняются многократно при обработке массивов данных, работе с базами данных, информационно-поисковыми системами и аналитическими платформами [1].

Выбор наиболее подходящего алгоритма поможет сократить время обработки запросов, а также повышает производительность программных систем. Практическая эффективность алгоритмов может значительно отличаться от теоретических оценок времени сложности определенных алгоритмов, в первую очередь это будет зависеть от структуры данных, помимо этого от особенностей аппаратной платформы и реализации алгоритмов.

В связи с этим задача экспериментального исследования производительности различных алгоритмов поиска на массивах большого размера будет весьма актуальной и востребованной. Полученные результаты позволят не только подтвердить теоретические оценки, но и выявить практические особенности их дальнейшего применения [6].

Целью данной работы является анализ производительности алгоритмов линейного, бинарного и хеш-поиска при обработке массивов данных различного объема с помощью программы написанной на языке программирования Python.

Для достижения поставленной цели были выдвинуты следующие задачи:

– выбрать алгоритмы поиска, которые будут использоваться в эксперименте;

– написать программу на языке Python для демонстрации работы выбранных алгоритмов;

– провести тестирование алгоритмов на массивах различного размера;

– выполнить анализ полученных результатов.

В данной работе выбраны следующие алгоритмы поиска: линейный поиск, бинарный поиск и хеш-поиск. Далее каждый из алгоритмов будет разобран подробнее.

Линейный поиск заключается в последовательном просмотре элементов массива с целью нахождения заданного значения. Алгоритм последовательно сравнивает каждый элемент массива с искомым значением до момента его обнаружения либо до завершения перебора всех элементов [7].

Временная сложность линейного поиска среднем составляет O(n), где n – количество элементов массива. Это означает, что при увеличении размера массива время выполнения алгоритма будет расти линейно, что делает данный метод неэффективным при работе с большими объемами данных.

Бинарный поиск применяется только к предварительно отсортированным массивам. Алгоритм основан на принципе деления массива пополам и последовательного сокращения области поиска [7].

На каждом шаге производится сравнение искомого значения со средним элементом массива, после чего поиск продолжается либо в левой, либо в правой части массива. Временная сложность бинарного поиска составляет O(log n), что значительно быстрее линейного поиска при работе с большим объемом данных [2, с. 70].

Хеш-поиск основан на использовании хеш-таблиц, обеспечивающих очень быстрое нахождение элементов независимо от размера массива [5]. При использовании корректной хеш-функции и достаточного объема памяти операция поиска выполняется за время O(n=1).

Основным преимуществом хеш-поиска является высокая скорость, однако он требует дополнительных затрат памяти и предварительной обработки данных для построения хеш-таблицы.

Для проведения вычислительного эксперимента была написана программа на языке Python с использованием стандартных библиотек. Для измерения времени выполнения алгоритмов применялся высокоточный таймер time.perf_counter(), обеспечивающий высокую точность замеров. В данном случае хеш-поиск был реализован с использованием встроенной структуры данных set.

Для точности эксперимента были сгенерированы массивы случайных целых чисел следующих размеров: 10000 элементов, 100000 элементов, 1000000 элементов, 5000000 элементов, 10000000 элементов.

Сгенерированные массивы сохранялись в текстовые файлы и использовались повторно при выполнении всех тестов, что исключало влияние случайных факторов.

Для бинарного поиска массивы предварительно сортировались. Для хеш-поиска создавалась хеш-таблица на основе множества элементов массива [3, 230 с.].

Для каждого массива проводилось 100 повторных операций поиска одного и того же элемента, после чего вычислялось суммарное время выполнения. Такой подход позволил минимизировать влияние случайных флуктуаций и повысить точность измерений [4, 486 с.].

Основной код программы представлен на рисунке 1.

В результате выполнения серии вычислительных экспериментов были получены временные характеристики работы алгоритмов поиска. Результаты временных характеристик алгоритмов поиска представлены в таблице.

```python
# ========== Алгоритмы поиска ==========
def linear_search(arr, x):
    for v in arr:
        if v == x:
            return True
    return False

def binary_search(arr, x):
    i = bisect.bisect_left(arr, x)
    return i != len(arr) and arr[i] == x

def hash_search(s, x):
    return x in s
# ========== Эксперимент ==========
sizes = [10**4, 10**5, 10**6, 5 * 10**6, 10**7]
repeats = 100

generate_and_save_arrays(sizes)

linear_times = []
binary_times = []
hash_times = []

for n in sizes:
    print(f"\nТестирование массива: {n}")
    arr = load_array(f"datasets/array_{n}.txt")
    arr_sorted = sorted(arr)
    s = set(arr)
    target = random.choice(arr)
    # Линейный поиск
    start = time.perf_counter()
    for _ in range(repeats):
        linear_search(arr, target)
    linear_times.append(time.perf_counter() - start)
    # Бинарный поиск
    start = time.perf_counter()
    for _ in range(repeats):
        binary_search(arr_sorted, target)
    binary_times.append(time.perf_counter() - start)
    # Хеш-поиск
    start = time.perf_counter()
    for _ in range(repeats):
        hash_search(s, target)
    hash_times.append(time.perf_counter() - start)
```

Рис. 1. Программа на языке Python (составлено автором)

Таблица [«Составлено автором»]

Размер массива	Линейный поиск (с)	Бинарный поиск (с)	Хеш-поиск (с)
10000	0.011190	0.000041	0.000010
100000	0.091925	0.000039	0.000010
1000000	0.364270	0.000045	0.000010
5000000	9.157202	0.000052	0.000011
10000000	5.239883	0.000050	0.000011

Для наглядного сравнения были построены графики зависимости времени выполнения от размера массива. На рисунке 2 представлено общее сравнение алгоритмов с использованием логарифмической шкалы, что позволило корректно отобразить различия между алгоритмами с разной асимптотической сложностью.

На рисунке 3 приведено детальное сравнение бинарного и хеш-поиска. Использование отдельного графика позволило наглядно продемонстрировать различия во временных характеристиках данных алгоритмов, которые слабо заметны на общем графике. Полученные результаты показывают, что хеш-поиск обладает более стабильным и низким временем

выполнения по сравнению с бинарным поиском, особенно при увеличении объема обрабатываемых данных.

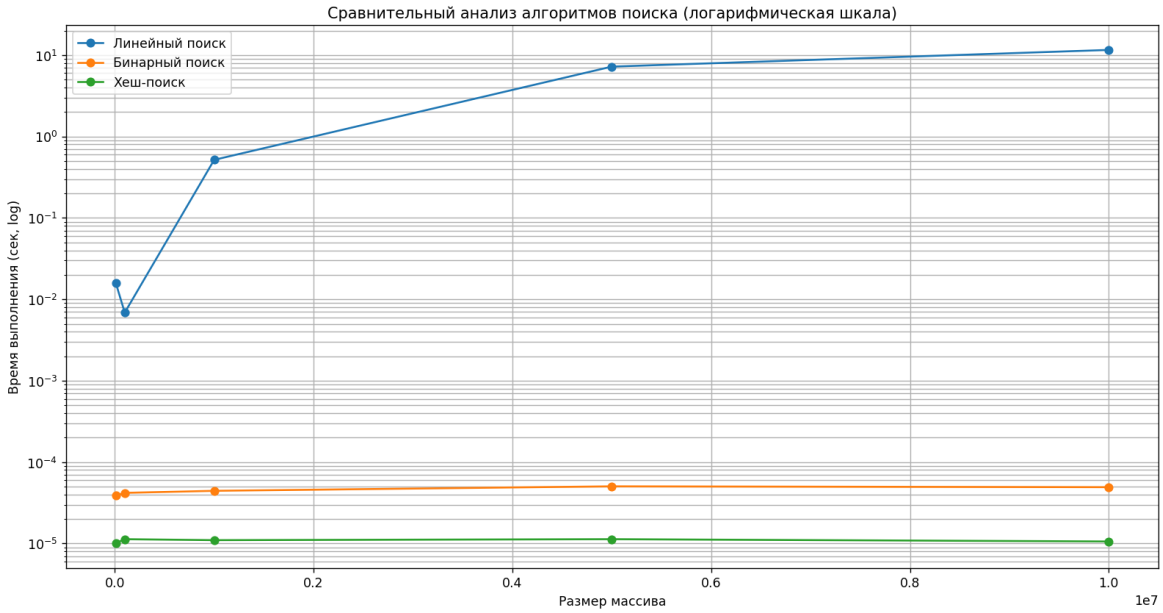


Рис. 2. Общее сравнение алгоритмов с использованием логарифмической шкалы (составлено автором)

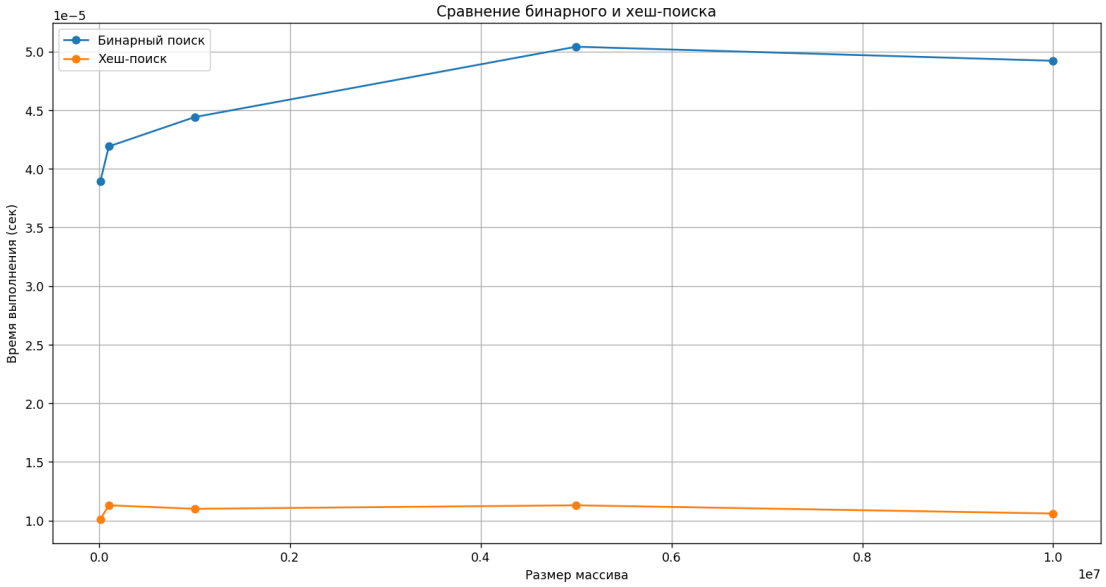


Рис. 3. Сравнение бинарного и хеш-поиска (составлено автором)

Анализ экспериментальных данных подтверждает теоретические оценки временной сложности исследуемых алгоритмов. Время выполнения линейного поиска демонстрирует линейный рост при увеличении объема данных, что делает данный алгоритм наименее эффективным для обработки больших массивов.

Бинарный поиск показывает значительно лучшие результаты, поскольку его временная сложность логарифмическая. Даже при увеличении размера массива до 10^7 элементов рост времени выполнения остается незначительным.

Наилучшие показатели продемонстрировал хеш-поиск, обеспечив практически постоянное время выполнения независимо от размера массива. Это подтверждает его высокую эффективность в задачах, требующих быстрого доступа к данным.

Однако следует отметить, что использование хеш-таблиц связано с дополнительными затратами памяти и необходимостью предварительного построения структуры данных, что может быть критичным в условиях ограниченных ресурсов.

Таким образом, выбор алгоритма поиска должен осуществляться с учетом характера задачи, объема данных, требований к быстродействию и доступных вычислительных ресурсов.

В данной работе был проведен экспериментальный анализ производительности линейного, бинарного и хеш-поиска на массивах большого размера. В ходе исследования была разработана программная система для проведения вычислительных экспериментов и получены оригинальные экспериментальные данные.

Результаты исследования показали, что линейный поиск обладает наименьшей эффективностью и может применяться лишь при работе с небольшими объемами данных. Бинарный поиск обеспечивает значительное ускорение обработки запросов, однако требует предварительной сортировки массива. Наиболее эффективным алгоритмом является хеш-поиск, обеспечивающий минимальное время доступа к элементам массива.

Практическая значимость исследования заключается в возможности применения полученных результатов при проектировании программных систем, работающих с большими объемами данных, включая базы данных, информационно-поисковые системы, аналитические платформы и высоконагруженные веб-сервисы.

Материалы работы также могут быть использованы в образовательном процессе при изучении дисциплин, связанных с алгоритмами, структурами данных и программированием.

Литература

1. Алейникова Е.М., Гринько А.М., Лукьянова Г.В. Машинное обучение в анализе больших объемов данных // Проблемы проектирования, применения и безопасности информационных систем в условиях цифровой экономики: Материалы XXIII Международной научно-практической конференции (Ростов-на-Дону, 25–26 ноября 2024 года). Ростов-на-Дону: Ростовский государственный экономический университет (РИНХ), 2024. С. 243-251.

2. Венгроу Д. Прикладные структуры данных и алгоритмы. Прокачиваем навыки / Переводчик С. Черников. СПб.: Питер, 2023. 869 с.

3. Зингаро Д. Python без проблем: решаем реальные задачи и пишем полезный код / Переводчик С. Черников. СПб.: Питер, 2023. 324 с.

4. Зингаро Д. Алгоритмы на практике / Переводчик Д. Брайт. СПб.: Питер, 2023. 536 с.

5. Леонтьев П.Н. Хеш-таблица как одна из наиболее эффективных структур хранения данных // Технологии Microsoft в теории и практике программирования (Томск, 21–22 марта 2012 года). Томск, Национальный исследовательский Томский политехнический университет, 2012. С. 233-235.

6. Полуэктов А.В., Попова Е.А., Журавлева И.В. Большие данные (BIG DATA): обработка, хранение и анализ больших объемов данных // Перспективные аспекты моделирования систем и процессов: Материалы Всероссийской научно-практической конференции (Воронеж, 07 октября 2023 года). Воронеж: Воронежский государственный лесотехнический университет им. Г.Ф. Морозова, 2023. С. 368-374. https://doi.org/10.58168/PAMSP_368-374

7. Тихова В.В. Поисковые алгоритмы: линейный и бинарный поиск // Молодые исследователи – современной науке: сборник статей XI Международной научно-практической конференции (Петрозаводск, 05 января 2026 года). Петрозаводск: Новая Наука, 2026. С. 25-32.

Огурцова М.А.
Национальный исследовательский технологический университет «МИСИС»
г. Москва, Россия

ПРОБЛЕМА КАТАСТРОФИЧЕСКОГО ЗАБЫВАНИЯ (CATASTROPHIC FORGETTING) В МАШИННОМ ОБУЧЕНИИ: ТЕОРЕТИЧЕСКИЙ ОБЗОР

Стремительное развитие больших языковых моделей (Large Language Models, далее – LLM) и их внедрение в динамичные среды требует способности к непрерывному обучению (continual learning). Современная нейронная сеть должна последовательно осваивать несколько задач, сохраняя при этом ранее приобретенные знания. Более того, появление мощных и общедоступных базовых моделей, таких как LLaMA и Mistral, дало толчок волне специализации. Чтобы оптимизировать эти модели для конкретных областей, например медицины, права или финансов, часто применяется дообучение с учителем (supervised fine-tuning, далее – SFT) на соответствующих тематических датасетах. Одно из препятствий на пути к созданию адаптивных и устойчивых систем искусственного интеллекта – проблема катастрофического забывания, которая характеризуется резкой потерей моделями ранее усвоенных знаний при освоении новой информации. Преодоление данной проблемы имеет важнейшее значение для создания масштабируемых и надежных агентных систем, способных адаптироваться к динамически меняющимся условиям [3, с. 128].

Впервые феномен катастрофического забывания был описан М. Макклоски и Н. Коэном еще в 1989 году. Исследователи определили его как “значительную деградацию способностей модели решать задачи или работать с языками, которые не были представлены в данных для дообучения” [9, с. 110].

Р. Френч считал, что тенденция нейронных сетей к снижению производительности является следствием перезаписи весов сети, полученных в ходе решения ранних задач, при обучении на последующей, отличной задаче [3, с. 139].

Катастрофическое забывание не только замедляет процесс дообучения модели, но и создает ряд проблем:

1) неспособность к инкрементальному обучению без забывания делает систему хрупкой и неспособной адаптироваться к меняющимся условиям, что ограничивает ее автономность;

2) выборочное забывание информации может привести к потере критически важных данных, в то же время модель может сохранить неэтичные или некорректные факты из исходного датасета, создавая тем самым большой риск для галлюцинаций;

3) дообучение на нейтральных наборах данных может непреднамеренно ухудшать ранее усвоенные паттерны безопасного поведения, в связи с этим модель теряет настройки алайнмента и становится уязвимой для генерации небезопасного контента [11].

Катастрофическое забывание может быть вызвано рядом факторов:

1) соотношение параметров предобученной модели и объема данных для дообучения (например, небольшие модели, дообученные на крупных датасетах, более подвержены забыванию);

2) размер модели (модели большего размера демонстрируют большую устойчивость к катастрофическому забыванию. Так, модель с 0,6 млрд параметров показывает результат лучше, чем версия с 0,2 млрд параметров);

3) размер датасета для дообучения (дообучение на большом объеме данных приводит к забыванию вне зависимости от размера LLM, при постепенном увеличении датасета растет количество ошибок) [7, с. 348];

4) дообучение модели на одной определенной предметной области (например, в задачах машинного перевода дообучение может вызвать забывание непредставленных языковых пар даже в моделях с широким лингвистическим охватом, таких как M2M-124 и mBART-50) [1, с. 157].

Для борьбы с проблемой катастрофического забывания были предложены различные подходы и методы.

В скором времени после появления термина, Э. Робинс представил метод репетиции (rehearsal), который был направлен на смягчение забывания путем повторного предъявления модели данных или репрезентаций из ранее изученных задач [12, с. 126].

Эта идея получила дальнейшее развитие в таких методах как:

1) Воспроизведение опыта (Experience Replay): метод, основанный на хранении выборок примеров из прошлых задач и периодическом переобучении модели на смеси этих исторических примеров и новых данных [13].

2) Генеративная репетиция (Generative Replay): для интеграции в процесс обучения новой задаче используются генеративные модели, создающие синтетические примеры предшествующих задач. Предлагается архитектура с двумя компонентами: 1) Generator – генерирует псевдопримеры старых задач (например, с помощью GAN или VAE); 2) Solver – обучается на новых данных вместе с псевдопримерами [14].

3) Самоформируемая репетиция (Self-Synthesized Rehearsal, SSR): метод, в котором происходит генерация собственных синтетических экземпляров внутри модели на основе внутриконтекстного обучения базовой LLM и отбор наиболее разнообразных и качественных примеров для последующего повторения [5, с. 1416].

Другой класс методов основан на регуляризации. Такие методы вводят дополнительные ограничения на изменение параметров модели при обучении новым задачам. Регуляризация направлена на сохранение ранее выученных знаний путем ограничения обновления параметров, которые были критически важны для предыдущих задач. В частности, используется упругое закрепление весов (Elastic Weight Consolidation, EWC) – каждому параметру модели приписывается мера важности, вычисляемая на основе диагональных элементов матрицы информации Фишера. Параметры, имеющие высокую важность для предыдущих задач, получают более сильный штраф при изменении во время обучения новой задаче.

Регуляризованная функция потерь в модели авторов имеет следующий вид:

$$L_{EWC}(\theta_t) = L_{ER}(\theta_t) + \lambda \sum_i F_i(\theta_{t,i} - \theta_{t-1,i})^2$$

где $\theta t, i$ и $\theta t - 1$ обозначают значения параметра i на текущем и предыдущем этапах обучения соответственно, Fi – важность параметра, а λ– коэффициент, контролирующий силу регуляризации. Авторы отмечают, что новые задачи могут существенно различаться по сложности и степени сходства с предыдущими, поэтому фиксированное значение коэффициента λ является неоптимальным [10, с. 3].

Исследователи из Корейского института передовых технологий (KAIST) выделяют несогласованность направлений градиента между задачами как одну из причин катастрофического забывания. Для устранения этого фактора они предлагают метод Sequential Reptile, который представляет собой мета-оптимизационный алгоритм, направленный на согласование градиентов между задачами для уменьшения негативного трансфера и катастрофического забывания [6, с. 1].

Методы, основанные на архитектуре моделей, включают стратегии изменения структуры нейросетей таким образом, чтобы удерживать знания о старых задачах, не мешая обучению новым. В одном из исследований был предложен подход подход, реализованный через итеративное сочетание трёх механизмов: прунинг (Network Pruning), расширение (Network Expanding) и маскирование (Task-Specific Masking) в рамках модели TPEM – изменение архитектуры нейросети по ходу обучения так, чтобы сохранять и выделять параметры, специфичные для каждой задачи:

1) Network Pruning: сжатие сети после обучения (удаление менее значимых весов, переобучение после их обнуления, восстановление производительности по старым задачам, сохранение выученной структуры).

2) Network Expanding: расширение архитектуры для обеспечения достаточной емкости для новых задач (добавление новых свободных весов, масштабирование скрытых слоев в зависимости от количества свободных параметров и объема данных для текущей задачи).

3) Task-Specific Masking: введение масок, которые действуют как бинарные матрицы (селективно скрывают те старые веса, которые препятствуют обучению текущей задаче, при этом истинные сохранённые веса остаются зафиксированными, что обеспечивает стабильность представлений для всех предыдущих задач) [4, с. 518].

Методы, основанные на памяти, решают проблему забывания путем сохранения и повторного предъявления примеров из прошлых задач в процессе обучения, тем самым снижая интерференцию. Д. Лопез-Паз и М. Ранзато предлагают подход градиентная эпизодическая память (Gradient Episodic Memory, GEM), который использует примеры из прошлых задач в сочетании с оптимизационными ограничениями для предотвращения ухудшения производительности на этих задачах при обучении новым. Ключевой компонент GEM – это эпизодическая память, которая содержит ограниченное подмножество примеров, полученных при обучении каждой задачи. Во время обучения GEM накладывает ограничения на обновление параметров, чтобы не допустить увеличения потерь на примерах из episodic memory для предыдущих задач [8, с. 6468].

Ф. Динг и Б. Ванг предложили свой собственный подход к решению проблемы катастрофического забывания, который основан на работе с данными (data-centric pipeline).

Они предложили улучшенную схему SFT, которая не требует доступ к оригинальным данным. На основе поведения и параметров базовой LLM, восстанавливается приближенное распределение SFT-инструкций, которые, вероятно, были использованы для первоначального обучения. С помощью других моделей выбираются лучшие примеры, которые затем смешиваются с новыми данными. Предложенный подход сохраняет или улучшает общие способности LLM в широких доменах без доступа к оригинальным SFT данным [2].

Таким образом, катастрофическое забывание является одной из ключевых проблем в области машинного обучения, особенно в сценариях последовательного и адаптивного обучения. Как показывает анализ, не существует универсального метода для устранения забывания. Регуляризационные методы наиболее уместны в ситуациях, где требуется сохранить параметры, критически важные для предыдущих задач, при минимальном изменении архитектуры и данных. Градиентно-ориентированные подходы целесообразны, когда забывание обусловлено конфликтующими обновлениями, и необходим контроль направления оптимизации. Архитектурные методы демонстрируют эффективность в сценариях с четко разделенными задачами и возможностью увеличения емкости модели, однако требуют дополнительных вычислительных и инженерных затрат. Data-centric подходы особенно актуальны для больших языковых моделей, где доступ к исходным данным часто отсутствует, а прямое вмешательство в параметры или архитектуру затруднено.

Решение проблемы катастрофического забывания требует осознанного проектирования процесса дообучения, где выбор метода определяется конкретными ограничениями и целями. Такой подход способствует более надежному и безопасному применению моделей в условиях непрерывного обучения.

Литература

1. Dakwale P., Monz C. Fine-tuning for neural machine translation with limited degradation across in- and out-of-domain data // Proceedings of the Machine Translation Summit XVI: Research Track (Nagoya, Japan). Nagoya. 2017. P. 156-169.

2. Ding F., Wang B. Improved supervised fine-tuning for large language models to mitigate catastrophic forgetting // arXiv.org. URL: https://arxiv.org/abs/2506.09428 (дата обращения: 25.01.2026).

3. French R. Catastrophic forgetting in connectionist networks // Trends in Cognitive Sciences. 1999. Vol. 3, № 4. P. 128-135. https://doi.org/10.1016/S1364-6613%2899%2901294-2

4. Geng B., Yuan F., Xu Q., Shen Y., Xu R., Yang M. Continual learning for task-oriented dialogue system with iterative network pruning, expanding and masking // Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing. 2021. P. 517-523. https://doi.org/10.18653/v1/2021.acl-short.66

5. Huang J., Cui L., Wang A., Yang C., Liao X., Song L., Yao J., Su J. Mitigating catastrophic forgetting in large language models with self-synthesized rehearsal // Proceedings of the 62nd Annual

Meeting of the Association for Computational Linguistics (Bangkok, Thailand). Bangkok. 2024. P. 1416-1428. https://doi.org/10.18653/v1/2024.acl-long.77

6. Lee S., Lee H.B., Lee J., Hwang S.J. Sequential reptile: Inter-task gradient alignment for multilingual learning [Электронный ресурс] // arXiv.org. URL: https://arxiv.org/abs/2110.02600 (дата обращения: 27.01.2026).

7. Liu D., Niehues J. Conditions for catastrophic forgetting in multilingual translation // Proceedings of the 5th Workshop on Multilingual Representation Learning (MRL 2025, Suzhuo, China). Suzhuo. 2025. P. 347-359. https://doi.org/10.18653/v1/2025.mrl-main.23

8. Lopez-Paz D., Ranzato M. Gradient episodic memory for continual learning // Advances in Neural Information Processing Systems (NeurIPS 2017, Long Beach, CA, USA). Long Beach. 2017. P. 6467-6476.

9. McCloskey M., Cohen N.J. Catastrophic interference in connectionist networks: The sequential learning problem // Psychology of Learning and Motivation. 1989. Vol. 24. P. 109-165.

10. Mi F., Chen L., Zhao M., Huang M., Faltings B. Continual learning for natural language generation in task-oriented dialog systems // arXiv.org. URL: https://arxiv.org/abs/2010.00910 (дата обращения: 27.01.2026).

11. Qi X., Zeng Y., Xie T., Chen P.-Y., Jia R., Mittal P., Henderson P. Finetuning aligned language models compromises safety, even when users do not intend to! // Proceedings of the International Conference on Learning Representations (ICLR). 2024.

12. Robins A. Catastrophic forgetting, rehearsal and pseudorehearsal // Connection Science. 1995. Vol. 7, № 2. P. 123–146. https://doi.org/10.1080/09540099508915711

13. Rolnick D., Ahuja A., Schwarz J., Lillicrap T., Wayne G. Experience replay for continual learning // Advances in Neural Information Processing Systems. 2019. Vol. 32.

14. Shin H., Lee J.K., Kim J., Kim J. Continual learning with deep generative replay // Advances in Neural Information Processing Systems. 2017. Vol. 30.

Павленко Е.Е.
Нижневартовский государственный университет
г. Нижневартовск, Россия

СРАВНЕНИЕ РЕКУРРЕНТНЫХ НЕЙРОННЫХ СЕТЕЙ С МЕХАНИЗМОМ ВНИМАНИЯ И БЕЗ ДЛЯ РЕШЕНИЯ ЗАДАЧ ПРОГНОЗИРОВАНИЯ ПРОДАЖ

Введение

Обзор современных методов и алгоритмов прогнозирования.

В наше время анализ и прогнозирование продаж крайне актуальны для того, чтобы выявлять тенденции, наилучшие условия для увеличения продаж. Прогнозирование возможно делать множеством способов, например, с помощью экспертной оценки или используя знания о поведении продаж в уже известной похожей ситуации, кроме этого, возможно использование построения логических сценариев для выявления прогноза или использование вычислительных ресурсов электронных вычислительных машин, если имеется статистика продаж компании за последнее время [4]. Один из подходов прогнозирования с использованием ЭВМ – это применение нейронных сетей. В данной статье будет проведен анализ одного из таких подходов.

Некоторые из современных методов прогнозирования на основе нейронных сетей:

1. Сверточные нейронные сети (CNN). В такой искусственной нейронной сети применяется комбинация следующих слоев, идущих последовательно: 1) Сверточный слой, необходимый для извлечения информативных признаков из выборки. Этот слой обрабатывает входные данные или результат предыдущего слоя с помощью фильтра, называемого ядром свертки и извлекает из данных признаки; 2) Слой пулинга, то есть пространственного сжатия данных. Этот слой обеспечивает сокращение объема обрабатываемых данных, при этом сохраняя их значимость; 3) Полносвязный слой, где каждый нейрон связан со всеми нейронами предыдущего слоя [3].

2. Рекуррентная нейронная сеть (RNN). В архитектуре рекуррентной НС существуют обратные связи между элементами, поэтому, когда принимается решение, учитываются и только что вошедшие данные, и предыдущие. Данные обрабатываются в цикле, и при каждом поступлении новых входных данных, сеть собирает информацию из цикла и выдает прогноз [9]. В RNN применяются полносвязные слои для преобразования признаков, полученных на предыдущих этапах, в результат (прогноз). Самая простая рекуррентная сеть может иметь единственный слой нейронов, каждый из которых будет направлять свой выходной сигнал на входы остальных слоев нейронов [8, с. 56-58].

Одной из распространенных архитектур рекуррентных сетей является сеть LSTM (Long Short-Term Memory) – долгая краткосрочная память. При обучении модель способна учитывать детали прошлого контекста и хранить их, пока они актуальны.

Описание архитектуры LSTM.

Один блок модели LSTM состоит из четырех взаимодействующих слоев и вектора, сохраняющего контекст всей последовательности. Когда информация проходит через взаимодействующие слои, с помощью сигмоидальной функции определяется, какую долю

блока информации необходимо пропустить, а какую следует учитывать дальше, что позволяет контролировать состояние ячейки. После определения, какие значения необходимо обновить, а какие добавить, вектор состояния обновляют, сначала умножая старое состояние на то, что следует забыть, а после прибавляя добавляемые значения. Само же состояние блока в текущий момент времени вычисляется с помощью других фильтров [10].

Описание архитектуры GRU.

Существует еще одна модель рекуррентной нейронной сети, называемой Gated Recurrent Unit, что значит управляемый рекуррентный блок. Она является упрощением LSTM, но при этом сохраняет свою эффективность. GRU использует меньше взаимодействующих слоев или фильтров. В этой модели нейронной сети также определяется, какую информацию необходимо забыть, этот фильтр называется фильтром сброса состояния. Выходное значение получается из промежуточного значения и фильтра сброса состояния. GRU имеет меньше параметров, поэтому может обучаться быстрее и требовать меньшее количество данных для анализа [5].

Механизм внимания.

В машинном обучении также применяется принцип, называемый механизмом внимания, который чаще всего используется в архитектурах некоторых рекуррентных нейронных сетей. Суть механизма внимания заключается в выделении части входных данных для более детальной обработки, он осуществляет поиск взаимосвязей между частями входных и выходных данных. Функцию внимания можно описать как сопоставление запроса (query) и набора пар ключ-значение (key-value) с выходными данными, где запрос, ключи, значения и выходные данные являются векторами. Результат вычисляется как взвешенная сумма значений, где вес, присвоенный каждому значению, вычисляется функцией совместимости запроса и соответствующего ключа [11].

Предобработка данных.

Еще одной важной задачей, выполняемой перед началом прогнозирования, является предобработка данных, то есть преобразование необработанных данных в понятный вид. От этого действия зависит качество результатов анализа данных и прогнозирования. Предобработка включает очистку, сокращение объема данных, их масштабирование, то есть приведение к определенному диапазону, и разделение на подмножества для глубокого анализа [2].

Наиболее важная часть предобработки данных – это их нормализация. Нормализация данных – это преобразование численных данных в диапазон со значениями от 0 до 1, реже от -1 до 1. Категориальные данные не нормализуются [7].

Линейная нормализация осуществляется с использованием функции «минимакс», имеющей вид: $y = (x - \min(x)) / (\max(x) - \min(x))$, где x – исходное множество данных, y – преобразованное множество данных, min и max – операции вычисления минимального и максимального значений [7].

Описание архитектуры нейронной сети и эксперимента

При подготовке статьи был проведен компьютерный эксперимент, и сделан анализ работы нейронных моделей с архитектурой LSTM с механизмом внимания и без, а также моделей сети с архитектурой GRU с вниманием и без. Для разработки нейронных сетей был использован язык программирования Python с применение библиотек tensorflow, sklearn и numpy для обработки входных данных и построения моделей нейронных сетей.

Предполагается, что модели нейронных сетей с механизмом внимания будут более эффективными, чем без механизма внимания. В добавок, можно выдвинуть предположение о том, что скорость работы моделей с GRU будет быстрее, чем с LSTM, в виду того, что архитектура GRU содержит меньшее количество слоев.

В проведенном при подготовке статьи эксперименте данные, по которым осуществляется прогнозирование, содержат номера магазинов и количество продаж в определенный день, объем данных – 7300 значений. Предобработка данных, приведение их к одному диапазону (от 0 до 1) осуществляется с использованием функции «минимакс». Далее осуществляется разделение данных на временные ряды (по 50 значений на ряд), обучающие и тестовые выборки.

Первые две модели рекуррентных нейронных сетей LSTM и GRU без механизма внимания содержат:

- Слой LSTM или GRU соответственно с 32 ячейками памяти, на вход этому слою подаются два вектора (номеров магазинов и продаж);
- Полносвязный слой с сигмоидальной функцией активации;

Сравниваемые с ними модели нейронных сетей LSTM и GRU с механизмом внимания включают:

- Слой LSTM или GRU с 32 ячейками памяти, на вход этому слою подаются два вектора (номеров магазинов и продаж);
- Реализованный механизм внимания, который глубже обрабатывает входные данные;
- Полносвязный слой с сигмоидальной функцией активации;

Функция потерь в компилируемых моделях – средняя квадратичная ошибка, метрика – средняя абсолютная ошибка. Модели проходят по 10 эпох обучения, где 20% данных выделяются на тест.

Таблица 1

Результаты метрик после тестирования

Критерий	Модель LSTM без внимания	Модель LSTM с вниманием	Модель GRU без внимания	Модель GRU с вниманием
Скорость обучения (средняя)	14,7 мс/шаг	15,6 мс/шаг	31,5 мс/шаг	20,4 мс/шаг
Средняя квадратичная ошибка обучающей выборки	0.0111	0.0159	0.0110	0.0135
Средняя квадратичная ошибка тестовой выборки	0.0126	0.0172	0.0127	0.0152

Средняя абсолютная ошибка обучающей выборки	0.0804	0.0993	0.0803	0.0901
Средняя абсолютная ошибка тестовой выборки	0.0859	0.1040	0.0862	0.0957

Выводы

Результаты эксперимента показывают очень маленькую разницу в эффективности моделей рекуррентных нейронных сетей с вниманием и без. У нейронной сети могло возникнуть недообучение, ситуация, при которой модель не может эффективно обучиться на имеющихся данных и выдает низкую производительность. Это может быть связано со множеством факторов. Модель подвержена колебаниям от нескольких функций, что может влиять на процесс обработки данных и не позволять модели выявить необходимые закономерности во входных данных. Также можно предположить, что для более глубокого анализа необходимо использование большего количества слоев нейронных сетей, то есть более сложной архитектуры сети, и большего числа эпох обучения [6].

Дополнительно был проведен эксперимент с увеличенным количеством эпох обучения, а именно с 30 эпохами. Однако значения средней квадратичной ошибки и средней абсолютной ошибки для тестовых и обучающих выборок были очень близки к результатам нейронных сетей с меньшим количеством эпох. Также от увеличения эпох не наблюдалось переобучение нейронных сетей, что могло бы произойти из-за большого количества обучающих эпох. Переобучением называют такую проблему, когда модель демонстрирует отличные результаты на обучающем наборе данных, а на тестовых данных результаты значительно ниже [6].

Еще один фактор, влияющий на результат, это нормализация данных. Необходимо правильно подобрать функцию, которая приведет данные к единому диапазону.

Небольшое отклонение в результатах для обучающей и тестовой выборки показывает, что нейронная сеть не подвержена переобучению, то есть она не адаптирована под хорошее распознавание только лишь обучающих данных, а обучена в целом классифицировать примеры [1].

Также было предположено, что модели с механизмом внимания будут эффективнее, однако результаты эксперимента показывают обратное. Более плохие показатели метрик в моделях рекуррентных нейронных сетей с механизмом внимания могут быть связаны с тем, что моделям на вход подаются всего два вектора с данными (номер магазина и продажи), поэтому модели недостаточно данных для более глубокого анализа и увеличения точности предсказаний.

Не подтвердилось и предположение о том, что скорость обработки данных рекуррентными нейронными сетями с архитектурой GRU будет быстрее скорости обработки сетями с архитектурой LSTM. Из результатов метрик после тестирования нейронных сетей можно заметить, что точность обработки данных с использованием сетей с GRU увеличилась незначительно. Следовательно, ту или иную рекуррентную нейронную сеть следует подбирать под условия конкретной поставленной задачи, а в данном случае нейронные сети с

управляемым рекуррентным блоком, то есть GRU, оказались неэффективными в сравнении с сетями с LSTM.

Литература

1. Абаляев А.Ю., Грунская Л.В., Галактионов С.А. Снижение вероятности переобучения рекуррентной нейронной сети в задаче прогнозирования среднего уровня аварийности на дорогах общего пользования // Актуальные проблемы эксплуатации автотранспортных средств: Материалы XXVI Международной научно-практической конференции, посвящённой памяти профессора Юрия Васильевича Баженова (Владимир, 21–22 ноября 2024 года). Владимир: Владимирский государственный университет им. А.Г. и Н.Г. Столетовых, 2024. С. 296-298.

2. Акимов А.А., Валитов Д.Р., Кубряк А.И. Предварительная обработка данных для машинного обучения // Научное обозрение. Технические науки. 2022. №. 2. С. 26-31. https://doi.org/10.17513/srts.1391

3. Вильданов Н.Р. Исследование параметров архитектуры сверточной нейронной сети при прогнозировании временного ряда // Актуальные проблемы науки и образования в условиях современных вызовов: Сборник материалов XI Международной научно-практической конференции (Москва, 24 мая 2022 года). М.: ИРОК, ИП Овчинников Михаил Артурович (Типография Алеф), 2022. С. 28-32.

4. Каипов И.К., Чигвинцев К.А. Обзор методов машинного обучения для краткосрочного прогнозирования продаж в обувном ритейле // Веб-программирование и интернет-технологии WebConf2021: материалы 5-й Международной научно-практической конференции (Минск, 18–21 мая 2021 года). Минск: Белорусский государственный университет, 2021. С. 102-104.

5. Козлов С.В., Седенков С.А. Анализ LSTM и GRU моделей для построения прогнозов временных рядов // International Journal of Open Information Technologies. 2024. Т. 12. №. 7. С. 43-50.

6. Навроцкий А.А., Шведко А.О., Сакович В.И. Проблематика машинного обучения: влияние гиперпараметров, переобучение и недообучение нейронных сетей // Big Data и анализ высокого уровня: сборник научных статей XI Международной научно-практической конференции (Минск, 23–24 апреля 2025 года). Минск: Белорусский государственный университет информатики и радиоэлектроники, 2025. С. 361-364.

7. Старовойтов В.В., Голуб Ю.И. Нормализация данных в машинном обучении // Информатика. 2021. Т. 18, № 3. С. 83-96. https://doi.org/10.37661/1816-0301-2021-18-3-83-96

8. Хайкин С. Нейронные сети: полный курс, 2-е издание. М.: Издательский дом Вильямс, 2008.

9. Чуб В.С. Рекуррентные нейронные сети. Обзор методов и приложений // Технологии и техника: пути инновационного развития: сборник научных статей Международной научно-технической конференции (Воронеж, 09 июня 2023 года. Воронеж). Воронеж, Воронежский государственный технический университет, 2023. С. 545-559.

10. Hochreiter S., Schmidhuber J. Long short-term memory // Neural computation. 1997. Т. 9. №. 8. С. 1735-1780.

11. Vaswani A. Attention is all you need // Advances in neural information processing systems. 2017. Т. 30.

Плотников А.А.
Нижневартовский государственный университет
г. Нижневартовск, Россия

ОСОБЕННОСТИ ТЕСТИРОВАНИЯ БЕЗОПАСНОСТИ

Современный этап развития экономики характеризуется стремительной цифровизацией всех сфер деятельности. Одновременно с этим растёт число киберугроз, что заставляет и коммерческие компании, и государственные учреждения уделять первостепенное внимание защите своих программных продуктов. Под тестированием безопасности понимается комплекс мер, направленных на выявление слабых мест в коде, серверной архитектуре и бизнес-логике приложений. Использование таких уязвимостей злоумышленниками может привести к серьёзным последствиям: финансовым потерям, утрате клиентской базы, судебным разбирательствам и падению репутации [3, с. 304]. В настоящей работе систематизированы основные подходы, методы и инструменты, используемые специалистами для оценки защищённости информационных систем.

Интеграция России в глобальное цифровое пространство и автоматизация ключевых бизнес-процессов обусловили переход кибербезопасности в разряд стратегических задач. Статистика показывает, что подавляющее большинство компаний регулярно подвергается атакам, причем основным вектором проникновения зачастую служат бреши в веб-интерфейсах [5, с. 2]. Приведенные факты подтверждают необходимость внедрения системного подхода к аудиту защищенности на всех этапах жизненного цикла ПО – от проектирования до практической эксплуатации.

Зачем тестировать безопасность?

1. Предотвращение утечек данных. Информационные системы накапливают огромные массивы персональных и корпоративных данных. Регулярные проверки позволяют удостовериться в надёжности их защиты и исключить возможность несанкционированного доступа.
2. Гарантия целостности информации. Важно исключить любую возможность несанкционированной модификации критически значимых данных – как со стороны внешних злоумышленников, так и из-за ошибок внутренних пользователей.
3. Обеспечение непрерывности работы. Анализ устойчивости систем к атакам типа «отказ в обслуживании» (DoS) помогает предотвратить остановку сервисов и сохранить их доступность для пользователей.
4. Экономия ресурсов. Устранение уязвимостей на ранних стадиях разработки обходится значительно дешевле, чем ликвидация последствий реальной кибератаки, включая выплаты пострадавшим и восстановление инфраструктуры.
5. Повышение доверия. Стабильная и безопасная работа онлайн-сервисов положительно влияет на репутацию компании и укрепляет лояльность клиентов и партнёров [3, с. 307].

Нормативно-правовая база и отраслевые стандарты

В настоящее время организации функционируют в условиях жесткого нормативного давления в сфере информационной безопасности. Российское законодательство (в частности, Федеральный закон № 152-ФЗ «О персональных данных») предъявляет строгие требования к операторам, осуществляющим обработку личной информации граждан. Несоблюдение предписанных мер защиты грозит существенными административными штрафами и потерей деловой репутации.

На глобальном уровне широко признаны такие стандарты, как ISO/IEC 27001 (системы менеджмента информационной безопасности), PCI DSS (безопасность платёжных карт) и GDPR (защита персональных данных в Европейском союзе). Чтобы подтвердить соответствие этим стандартам, компании обязаны регулярно проводить аудиты и тестирование на проникновение.

Кроме того, существуют отраслевые требования: для финансового сектора – это нормативы Банка России, для медицинских учреждений – правила защиты врачебной тайны, для госсектора – работа с системами, имеющими классификацию по защите гостайны. Качественное тестирование безопасности служит инструментом верификации соблюдения этих узкоспециализированных норм.

Уровни тестирования безопасности

Тестирование на уровне кода

Проверка исходного кода на наличие уязвимостей, таких как:

- Инъекции (SQL, команды, XSS)
- Несоблюдение принципов безопасного кодирования
- Неправильная обработка ошибок и исключений
- Уязвимости в управлении памятью (особенно в C/C++)

Инструменты: SonarQube, Fortify, Checkmarx, Snyk.

Применение инструментов статического анализа даёт возможность обнаруживать дефекты безопасности ещё до того, как код будет запущен. Чем раньше найдена проблема, тем меньше ресурсов требуется на её исправление. По некоторым оценкам, стоимость устранения уязвимости на стадии написания кода может быть на порядок ниже, чем после выхода продукта в промышленную эксплуатацию. Дополнительный плюс такого подхода – повышение осознанности разработчиков в вопросах безопасного программирования.

Примеры типичных уязвимостей, выявляемых при анализе кода:

- Неправильная валидация пользовательского ввода;
- Утечки памяти и ресурсов;
- Небезопасное использование криптографических функций;
- Отсутствие проверки прав доступа;
- Логические ошибки в бизнес-процессах.

Тестирование на уровне приложения

Проверка самого приложения на уязвимости, возникающие из-за логики работы, архитектуры и взаимодействия компонентов. Это включает:

- Проверку на SQL-инъекции, XSS, CSRF;
- Тестирование авторизации и аутентификации;
- Анализ управления сессиями;
- Проверку на уязвимости в API.

Инструменты: OWASP ZAP, Burp Suite, Acunetix, Nessus.

Тестирование на уровне сети

Проверка операционной системы, баз данных, служб на наличие уязвимостей:

- Недостаточное обновление ПО;
- Открытые файлы и директории;
- Несоблюдение политик безопасности (например, права доступа);
- Уязвимости в драйверах и ядре ОС.

Инструменты: OpenSCAP, Lynis, OSSEC.

Пентестинг

Симуляция реальной атаки со стороны злоумышленника. Цель – найти и эксплуатировать уязвимости, чтобы оценить реальный уровень безопасности системы.

Инструменты: Metasploit Framework, Kali Linux, Cobalt Strike (для продвинутых атак).

Виды тестирования безопасности

Статическое анализ кода SAST (Static Application Security Testing)

Анализ исходного кода без его выполнения. Позволяет находить уязвимости на ранних этапах разработки.

Пример: Инструмент SonarQube может выявить уязвимость типа "SQL-инъекция" в строке кода query = "SELECT * FROM users WHERE name = '" + username + "'"[1, с. 400].

Динамическое тестирование (DAST - Dynamic Application Security Testing)

Тестирование работающего приложения. Инструменты отправляют запросы и анализируют ответы на наличие уязвимостей.

Пример: Инструмент OWASP ZAP автоматически сканирует веб-приложение, отправляя различные нагрузки в формы и URL, чтобы обнаружить XSS или SQL. Динамическое тестирование приложений особенно важно для веб-сервисов и мобильных приложений, которые напрямую взаимодействуют с пользователями через интернет. Согласно отчету OWASP, наиболее распространенными уязвимостями в веб-приложениях являются инъекции, нарушения аутентификации и неправильная конфигурация безопасности. [5, с. 5]

При тестировании API особое внимание уделяется:

- Проверке аутентификации и авторизации;
- Валидации входных данных;
- Защите от атак типа «человек посередине» (MITM);
- Корректной обработке ошибок и исключений;
- Безопасности передачи данных (использование HTTPS, шифрование) [2, с. 3-4].

Сканирование уязвимостей (Vulnerability Scanning)

Автоматизированное сканирование систем и приложений на наличие известных уязвимостей (например, уязвимости в версиях библиотек, устаревшие протоколы).

Инструменты: Nessus, OpenVAS, Qualys.

Социальная инженерия (Social Engineering Testing)

Тестирование человеческого фактора. Например, фишинговые атаки на сотрудников для проверки их осведомлённости о безопасности.

Методы тестирования безопасности

- Чёрный ящик (Black-box Testing): Тестировщик не знает внутреннюю структуру системы. Имитирует действия внешнего злоумышленника. Наиболее реалистичный подход для оценки защиты. Этот метод наиболее реалистично имитирует действия внешнего злоумышленника, который не имеет доступа к внутренней структуре системы. Преимущества метода включают:
  - Обнаружение уязвимостей, доступных извне;
  - Проверка эффективности защитных механизмов;
  - Оценка реальных рисков для бизнеса.
- Белый ящик (White-box Testing): При этом подходе тестировщик имеет полный доступ к исходному коду, архитектуре и документации системы. Преимущества:
  - Глубокий анализ логики приложения;
  - Выявление сложных уязвимостей и логических ошибок;
  - Проверка соответствия принципам безопасного кодирования.
- Серый ящик (Gray-box Testing): Комбинированный подход. Тестировщик имеет частичную информацию о системе (например, архитектуру, но не весь код). Комбинированный подход сочетает преимущества обоих предыдущих методов. Тестировщик имеет частичную информацию о системе, что позволяет:
  - Эффективно находить уязвимости на стыке компонентов;
  - Проверять как внешние, так и внутренние аспекты безопасности;
  - Оптимизировать время тестирования.

Инструменты для тестирования безопасности

- OWASP ZAP – бесплатный инструмент для тестирования веб-приложений (DAST).
- Burp Suite – мощный инструмент для анализа и тестирования веб-приложений (премиум-версия).
- Nmap – сканер сетей, определяющий открытые порты и службы.
- Metasploit Framework – платформа для разработки и запуска эксплойтов.
- SonarQube – инструмент для статического анализа кода (SAST).
- Nessus – профессиональный сканер уязвимостей.
- Wireshark – анализатор сетевого трафика.
- Kali Linux – сборка Linux с множеством инструментов для тестирования безопасности.

Почему важно тестировать безопасность?

1. Снижение рисков: Выявление уязвимостей до того, как они будут использованы злоумышленниками.

2. Соответствие законодательству: Обеспечение соответствия требованиям по защите персональных данных и другим нормативам.

3. Сохранение репутации: Предотвращение инцидентов, которые могут привести к утрате доверия клиентов.

4. Финансовая защита: Избежание штрафов, потерь от кражи данных и затрат на восстановление после атак.

5. Улучшение качества продукта: Безопасность становится частью качества ПО, а не дополнительной задачей.

Заключение

Тестирование безопасности – это не разовая процедура, а непрерывный процесс, который должен быть интегрирован в культуру организации. Успешные компании рассматривают безопасность как стратегическое преимущество, а не как дополнительную нагрузку или обязательство. Инвестиции в тестирование безопасности окупаются многократно за счет предотвращения инцидентов, защиты репутации и обеспечения доверия клиентов. Будущее тестирования безопасности связано с развитием технологий искусственного интеллекта и машинного обучения. Уже сегодня появляются инструменты, способные автоматически выявлять сложные уязвимости и предсказывать потенциальные векторы атак на основе анализа больших данных. Однако технологии не заменят человеческий фактор – квалифицированные специалисты по безопасности останутся ключевым элементом эффективной защиты [4, с. 1608].

Рекомендации для организаций, начинающих внедрять тестирование безопасности:

1. Начать с оценки текущего уровня зрелости процессов безопасности;
2. Разработать стратегию тестирования, соответствующую бизнес-целям;
3. Инвестировать в обучение сотрудников и формирование культуры безопасности;
4. Регулярно обновлять инструменты и методы тестирования;
5. Интегрировать тестирование безопасности в процессы разработки (DevSecOps);
6. Проводить регулярные аудиты и оценку эффективности программ тестирования.

Список литературы

1. Бахадуров Б., Баллыев А., Байрамова Дж. Методы тестирования безопасности программного обеспечения: от статического анализа до пентестинга // Вопросы кибербезопасности. 2024. С. 399-401. URL: https://clck.ru/3TLneD (дата обращения: 15.02.2026).

2. Медаев М.К. Тестирование API // Информационные технологии. 2024. URL: https://cyberleninka.ru/article/n/testirovanie-api (дата обращения: 15.02.2026).

3. Позин Б.А. Тестирование в жизненном цикле автоматизированных систем // Трудцы ИСП РАН. 2025. Т. 37. Вып. 3. С. 303-309. URL: https://clck.ru/3TLnf8 (дата обращения: 15.02.2026).

4. Хусаинова Э.Р. Современные тенденции в области тестирования безопасности // Вестник науки. 2025. Т1, № 6(87). С. 1601-1611. URL: https://clck.ru/3TLncn (дата обращения: 15.02.2026).

5. Шатурный М.В. Анализ актуальных угроз и разработка подходов к защите веб-приложений // Инженерный вестник Дона. 2024. № 7. URL: https://clck.ru/3TLndR (дата обращения: 15.02.2026).

Плотникова А.Ю.
Нижневартовский государственный университет
г. Нижневартовск, Россия

ОСОБЕННОСТИ ТЕСТИРОВАНИЯ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА И МАШИННОГО ОБУЧЕНИЯ

Искусственный интеллект (ИИ) и машинное обучение (МО) становятся неотъемлемой частью современных программных систем – от рекомендательных сервисов и чат-ботов до систем аналитики и автоматизации тестирования. В отличие от традиционного программного обеспечения, поведение ИИ-систем определяется не жёстко заданными алгоритмами, а обученными моделями, что вносит принципиальные отличия в подходы к их тестированию. В данной статье рассматриваются особенности тестирования ИИ и МО, основные уровни и методы проверки качества таких систем, а также инструменты, применяемые на практике. Важно отметить, что тестирование ИИ-компонентов требует пересмотра классических подходов и включения в процесс проверки не только функциональности, но и поведения модели в условиях неопределённости.

Современные ИИ-системы, особенно на основе глубоких нейронных сетей и больших языковых моделей, демонстрируют высокую эффективность, однако их сложность и непрозрачность порождают новые риски. Ошибки могут проявляться не только в виде неверных ответов, но и в форме неожиданного поведения при незначительных изменениях входных данных, что требует разработки специализированных подходов к верификации. В связи с этим тестирование ИИ становится не просто этапом разработки, а критически важным процессом, обеспечивающим доверие к таким системам и их безопасное применение в реальных условиях.

Зачем тестировать системы ИИ и МО?

Проверка качества систем на базе ИИ преследует особые цели, которые отличаются от тестирования обычного ПО, обусловленных природой этих технологий. Непреднамеренные ошибки в алгоритмах, предвзятость моделей и угрозы со стороны злоумышленников могут привести к серьезным последствиям – от экономических потерь до рисков для жизни человека [2, с. 109]. Анализ практических проектов, включающих ИИ-модули, показывает, что стандартные тест-кейсы часто оказываются недостаточными. Требуется проверка не только корректности работы, но и надёжности, справедливости и безопасности функционирования модели. Ниже перечислены ключевые цели тестирования.

1. Обеспечение корректности предсказаний:

Модель должна выдавать точные и обоснованные результаты на новых, ранее неизвестных данных. Неточности в прогнозах способны повлечь за собой критические последствия, особенно в критически важных сферах – здравоохранении, финансах или автономном управлении [2, с. 109].

2. Проверка устойчивости к шуму и аномалиям:

ИИ-системы должны корректно реагировать на некорректные, зашумлённые или аномальные входные данные, не допуская катастрофических сбоев. Одним из важных

направлений является тестирование устойчивости ИИ к изменениям входных данных, включающее анализ поведения системы при воздействии так называемых «враждебных примеров» – искусственно созданных входных данных, которые способны вводить алгоритмы в заблуждение [2, с. 109].

3. Обеспечение справедливости и отсутствия предвзятости:

Модели могут воспроизводить или даже усиливать социальные предубеждения, заложенные в обучающих данных. Тестирование на справедливость (fairness testing) помогает выявить дискриминационные паттерны. Предвзятость алгоритмов, вызванная ограничениями обучающих данных или архитектурой моделей, может приводить к дискриминации или несправедливым решениям. Например, системы кредитного скоринга или подбора кандидатов на работу могут демонстрировать тенденции к предпочтению одних групп людей над другими [2, с. 111].

4. Воспроизводимость и интерпретируемость:

Важно, чтобы результаты модели были воспроизводимы при одинаковых входных условиях, а также по возможности объяснимы для разработчиков и пользователей. Это требует особого подхода к документированию тестовых сценариев и условий выполнения проверок. Искусственные нейронные сети могут быть применены для решения задачи автоматизированного тестирования, включая генерацию тестовых сценариев, способных выявлять неисправности [1, с. 65].

5. Соответствие требованиям безопасности и конфиденциальности:

ИИ-системы могут быть уязвимы к атакам типа adversarial examples (враждебных примеров) или утечкам данных через выводы модели. Adversarial examples – это специально модифицированные входные данные, в которые вносятся минимальные, часто незаметные для человека изменения, приводящие к ошибочным предсказаниям модели. Например, незначительное искажение пикселей на изображении может заставить модель классификации ошибочно идентифицировать объект [2, с. 109]. Тестирование должно включать проверку на устойчивость к подобным угрозам, что добавляет новый тип проверок – целенаправленную генерацию «вредоносных» входных данных для оценки устойчивости модели. Для повышения реалистичности тестовых данных используются генеративно-состязательные сети (GAN), которые преобразуют данные симуляторов в реалистичные входные изображения, позволяя более эффективно оценивать точность Deep Neural Networks (DNN) и дообучать модель [3, с. 567].

Уровни тестирования ИИ и МО

Многоуровневый подход к тестированию ИИ позволяет системно оценивать качество системы на всех этапах её жизненного цикла, что особенно важно при интеграции ИИ-модулей в существующие программные продукты. На протяжении жизненного цикла разработки и сопровождения программного обеспечения тестирование осуществляется на разных его уровнях. Уровень определяет, какой объект или элемент тестируемого продукта рассматривается в ходе проверки. Принято выделять четыре типовых уровня тестирования: модульное, интеграционное, системное и приёмочное тестирование [1, с. 62].

Тестирование данных (Data Testing)

Качество модели напрямую зависит от качества обучающих данных. На этом уровне проверяются:

- Полнота и сбалансированность датасета;
- Отсутствие дубликатов и аномалий;
- Корректность разметки (в случае обучения с учителем).

Инструменты: Great Expectations, Pandas Profiling, TensorFlow Data Validation.

Многие проблемы в работе ИИ-моделей связаны с низким качеством или несбалансированностью данных, что подчёркивает критическую важность этого этапа тестирования. Эксперименты подтверждают эффективность генерации тестовых сценариев при помощи искусственных нейронных сетей для выявления неисправностей [1, с. 65].

Тестирование модели (Model Testing)

Этот уровень включает оценку метрик качества модели: точность (accuracy), полнота (recall), F1-мера, AUC-ROC и др. Также проверяется:

- Обобщающая способность (generalization) на валидационных и тестовых выборках;
- Стабильность метрик при изменении гиперпараметров;
- Устойчивость к переобучению (overfitting) и недообучению (underfitting).

Инструменты: Scikit-learn, MLflow, Weights & Biases.

Метод ABS (Adaptive Beam Search) демонстрирует высокую эффективность при тестировании больших языковых моделей: незначительно изменённый текст может ввести ChatGPT в заблуждение, заставив изменить метку входного текста с «NEGATIVE» (с уверенностью 91%) на «POSITIVE» (с уверенностью 96%) [4, с. 1]. ABS достигает среднего показателя успешности 86.138%, значительно превосходя базовый метод Probability Weighted Word Saliency (PWWS) (68.177%) [4, с. 13].

Тестирование интеграции модели в систему

После обучения модель интегрируется в программное приложение. Здесь проверяется:

- Корректность взаимодействия модели с API и базами данных;
- Обработка ошибок при недоступности модели;
- Латентность и производительность предсказаний.

Инструменты: Postman, Pytest, Locust (для нагрузочного тестирования API).

На этом этапе применяются классические методики интеграционного и нагрузочного тестирования, адаптированные с учётом специфики работы ИИ-компонентов.

Тестирование в production (A/B-тестирование и мониторинг)

После развёртывания модель продолжает тестироваться в реальных условиях. Проводится A/B-тестирование разных версий модели, мониторинг дрейфа данных и дрейфа концепции, сбор обратной связи от пользователей. Фреймворк SAFuzz, предназначенный для тестирования кода, сгенерированного большими языковыми моделями, улучшает дискриминацию уязвимостей, отсеивая в 2 раза больше не уязвимых целей, и достигает ускорения в 1.7 раза по сравнению с базовыми методами [5, с. 7]. Гибридная модель, лежащая в основе SAFuzz, отсеивает 82.5% безопасного кода, отфильтровывая лишь 9.8% ошибочного

кода, что позволяет значительно экономить ресурсы при фаззинге без существенного влияния на обнаружение ошибок [5, с. 6]. Непрерывный мониторинг позволяет своевременно выявлять деградацию качества модели, что особенно актуально для систем, работающих с динамичными данными.

Методы тестирования ИИ и МО

Для эффективного тестирования ИИ-систем необходим комплекс методов, включающий как классические подходы, так и специализированные методики, адаптированные под недетерминированную природу моделей.

- Метод чёрного ящика: Проверка поведения модели по входу-выходу без анализа внутренней структуры. Применяется при тестировании готовых API.
- Метод белого ящика: Анализ внутренних весов, градиентов, слоёв нейросети. Используется для отладки и интерпретации.
- Adversarial testing: Специальное тестирование на устойчивость к враждебным (adversarial) примерам – данным, искусственно изменённым для введения модели в заблуждение.
- Property-based testing: Проверка выполнения заданных свойств модели.

Инструменты для тестирования ИИ и МО

Современный инструментарий позволяет автоматизировать многие аспекты тестирования ИИ, что повышает эффективность и воспроизводимость проверок. Для валидации качества данных применяются инструменты Great Expectations, Pandas Profiling, TensorFlow Data Validation. Для мониторинга дрейфа и качества моделей используется Evidently AI, Prometheus + Grafana, Amazon SageMaker Model Monitor. Управление жизненным циклом моделей и экспериментов осуществляется с помощью MLflow, Weights & Biases. Для генерации синтетических данных применяется Faker, а для написания автоматизированных тестов для ML-пайплайнов – Pytest.

Почему важно тестировать ИИ и МО?

Комплексное тестирование ИИ-систем является критически важным элементом обеспечения их качества и снижения сопутствующих рисков.

1. Снижение рисков: Ошибки ИИ могут иметь этические, юридические и финансовые последствия.
2. Повышение доверия: Прозрачные и тестируемые модели вызывают больше доверия у пользователей и регуляторов.
3. Поддержка жизненного цикла: Тестирование позволяет поддерживать актуальность модели в меняющихся условиях.
4. Соответствие стандартам: Во многих отраслях (медицина, финансы) регулирование требует строгой верификации ИИ-решений.

Заключение

Тестирование систем искусственного интеллекта и машинного обучения требует междисциплинарного подхода, сочетающего методы классического тестирования ПО, статистического анализа и специализированных техник, разработанных для ИИ. Учитывая

растущую роль ИИ в повседневной жизни, обеспечение его надёжности, справедливости и безопасности становится не просто технической, но и этической задачей. Особую значимость приобретает разработка методов, способных выявлять не только очевидные ошибки, но и скрытые дефекты, связанные с недетерминированностью поведения моделей. Это требует постоянного обновления инструментария и методик в соответствии с развитием самих ИИ-систем. Использование нейросетевых моделей для генерации тестовых сценариев [1, с. 65], применение GAN для повышения реалистичности симуляций [3, с. 567] и адаптивный фаззинг LLM-кода с помощью SAFuzz [5, с. 6-7] позволяют значительно повысить эффективность обнаружения дефектов и сократить временные затраты. Учёт потенциальных уязвимостей и предвзятости алгоритмов [2, с. 109, 111] остаётся критически важным для обеспечения безопасности и справедливости ИИ-систем. Современные инструменты и практики позволяют систематизировать этот процесс, делая его неотъемлемой частью разработки ИИ-продуктов.

Дальнейшее развитие методов тестирования ИИ неразрывно связано с совершенствованием стандартов и нормативной базы. Внедрение регуляторных требований стимулирует создание унифицированных протоколов проверки, особенно для систем высокого риска. Важную роль играет также междисциплинарное взаимодействие разработчиков, тестировщиков, юристов и этиков, позволяющее учитывать не только технические аспекты, но и социальные последствия применения ИИ. Только комплексный подход обеспечит создание по-настоящему надёжных и безопасных интеллектуальных систем. Освоение методологий тестирования ИИ представляет собой важное направление развития профессиональных компетенций в области обеспечения качества программного обеспечения. Практика показывает, что системное тестирование ИИ-компонентов значительно снижает риски их внедрения и способствует созданию более надёжных технологических решений.

Литература

1. Данилов А.Д., Мугатина В.М. Верификация и тестирование сложных программных продуктов на основе нейросетевых моделей // Вестник Воронежского государственного технического университета. 2016. Т 12, № 6.С. 62-67.

2. Нуреев Д.Р. Разработка универсальных методов тестирования и проверки безопасности искусственного интеллекта // Парадигма. 2025. № 2.С. 108-113.

3. Attaoui M.O., Pastore F., Briand L.C. Search-based DNN Testing and Retraining with GAN-enhanced Simulations // IEEE Transactions on Software Engineering. 2025. Vol. 51, No. 3. P. 567-582. URL: https://arxiv.org/abs/2406.13359 (дата обращения 15.02.2026).

4. Xiao M., Xiao Y., Ji S., Cai H., Xue L., Zhang P. Assessing the Robustness of LLM-based NLP Software via Automated Testing // ACM Transactions on Software Engineering and Methodology. 2025. Vol. 34, No. 2. P. 1-28. URL: https://arxiv.org/abs/2412.21016 (дата обращения 15.02.2026).

5. Yang Z., Inani K., Kabra K., Gupta V., Iyer A.P. SAFuzz: Semantic-Guided Adaptive Fuzzing for LLM-Generated Code // arXiv preprint. 2026. arXiv:2602.11209. URL: https://arxiv.org/abs/2602.11209 (дата обращения 15.02.2026).

Поротиков А.П., Тагиров К.М.
Нижневартовский государственный университет
г. Нижневартовск, Россия

ВИЗУАЛИЗАЦИЯ ДАННЫХ С КАМЕРЫ БПЛА НА LED-ПАНЕЛЬ В РЕАЛЬНОМ ВРЕМЕНИ

В современных исследованиях и образовательных проектах в области робототехники и автономных систем всё большую роль играют доступные и гибкие программно-аппаратные платформы [3; 5]. Одной из таких платформ является «Клевер» – открытый учебный конструктор на базе квадрокоптера, оснащённый полетным контроллером PX4, одноплатным компьютером Raspberry Pi и камерой для реализации задач компьютерного зрения. Его ключевое преимущество – это открытая архитектура и программное обеспечение, которые позволяют исследователям и студентам не просто управлять дроном, а программировать сложные автономные сценарии, включая навигацию по визуальным меткам, построение карт и работу в роях. В то же время, эффективная интерпретация данных, собираемых такими автономными системами, остаётся важной задачей [2]. Для анализа полётной информации, отладки алгоритмов и демонстрации работы системы в реальном времени стандартных мониторов часто недостаточно [4]. Использование крупноформатных LED-панелей позволяет создать наглядную, масштабную и удобную для коллективного наблюдения визуализацию, что критически важно для образовательного процесса, совместной работы над проектами и публичных демонстраций.

Целью данного исследования является разработка и реализация программного комплекса для передачи и визуализации видеопотока и телеметрии с автономного квадрокоптера «Клевер» на LED-панель в реальном времени. Для достижения поставленной цели были определены следующие задачи:

1. Анализ архитектуры и программных интерфейсов (API) платформы «Клевер» для доступа к видеопотоку с бортовой камеры и данным телеметрии.
2. Разработка клиент-серверного программного модуля, осуществляющего изъятие данных с бортового компьютера Raspberry Pi дрона, их обработку и передачу на led-ленту.
3. Экспериментальная оценка работоспособности системы, измерение задержки «от камеры до экрана» и устойчивости канала передачи данных.

В качестве базовой аппаратной платформы использовался учебный квадрокоптер «Клевер 4». Данный конструктор включает полетный контроллер COEX Pix на базе PX4 Autopilot для управления полетом и стабилизации, а также одноплатный компьютер Raspberry Pi 4 Model B, выполняющий роль бортового вычислителя высокого уровня. На Raspberry Pi предустановлен специализированный образ операционной системы с ROS 1 (Robot Operating System, версия Melodic или Noetic), что обеспечивает стандартизированные интерфейсы для взаимодействия с компонентами дрона. Сенсоры: камера для Raspberry Pi с 5Мп сенсором OV5647ё, лазерный дальномер (для удержания высоты), RGB-светодиодная лента типа ws281x.

Исследование проводилось на основе учебного квадрокоптера «Клевер 4» (Clover 4), который представляет собой открытую программно-аппаратную платформу для разработки автономных систем. Конструкция дрона включает два ключевых вычислительных модуля, взаимодействующих по протоколу MAVLink [1]:

- Низкоуровневый полетный контроллер COEX Pix, на котором функционирует открытое полетное стекло PX4 Autopilot. Данный модуль отвечает за стабилизацию, навигацию по данным инерциального измерительного блока (IMU) и непосредственное управление бесколлекторными двигателями.
- Роль полётного вычислителя верхнего уровня в рассматриваемой конфигурации выполняет одноплатный компьютер Raspberry Pi 4 Model B. На его борту развёрнута среда Ubuntu с интегрированным фреймворком Robot Operating System (ROS 1, дистрибутив Noetic), который берёт на себя функции унификации доступа к аппаратному обеспечению. Благодаря ROS становятся доступны стандартизованные каналы обмена данными (топики и сервисы) для работы с сенсорным оборудованием, в том числе с камерой, подключённой через интерфейс CSI.

Для визуализации применялась светодиодная (LED) панель, совместимая с протоколом управления Digital Multiplex (DMX) через адаптер. Управляющие команды генерировались программным узлом на бортовом компьютере дрона, что исключало необходимость в дополнительном наземном сервере и упрощало архитектуру системы.

2.2. Архитектура разработанной системы

Разработанная система представляет собой **распределенное ROS-приложение**, функционирующее по принципу издатель-подписчик (publisher-subscriber). Архитектура системы включает следующие ключевые модули, образующие замкнутый контур управления «восприятие-визуализация» (рис. 1). Обработка изображения и управление светодиодами реализованы в рамках единого узла color_detection_led_control.

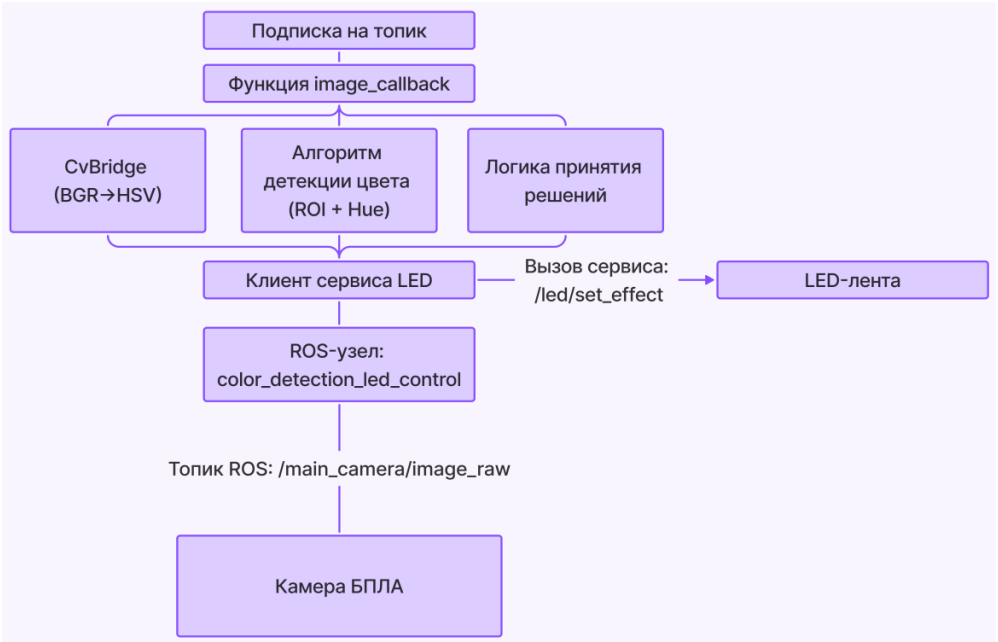


Рис. 1. Архитектура ROS-системы для визуализации данных с камеры БПЛА

1. **Модуль захвата и публикации изображения:** Встроенный ROS-драйвер камеры непрерывно публикует необработанные изображения в топик /main_camera/image_raw с типом сообщения sensor_msgs/Image.
2. **Модуль обработки изображения и принятия решений:** Центральный **ROS-узел** color_detection_led_control, написанный на Python. Узел инициализирует подписку на топик /main_camera/image_raw и назначает **функцию-обработчик** image_callback(msg) для каждого входящего сообщения с изображением. Внутри данной функции реализован полный алгоритм обработки кадра, от конвертации до определения доминирующего цвета. Таким образом, узел является единым исполняемым модулем, совмещающим в себе подписку на данные, их обработку и вызов управляющих сервисов.
3. **Модуль управления LED-панелью:** После обработки кадра узел формирует соответствующую команду, вызывая **ROS-сервис** /led/set_effect с типом сообщения clover/srv/SetLEDEffect. Данный сервис является частью штатного пакета clover платформы «Клевер» и напрямую управляет светодиодной подсветкой.

2.3. Алгоритм обработки видеопотока и управления визуализацией

Разработанный программный модуль является центральным ROS-узлом с именем color_detection_led_control. Его основная функция – обработка изображений в реальном времени и преобразование визуальной информации в управляющие команды для системы отображения. Алгоритм работы узла представляет собой последовательность детерминированных этапов (рис. 2), инициируемых каждым новым кадром, поступающим от камеры.

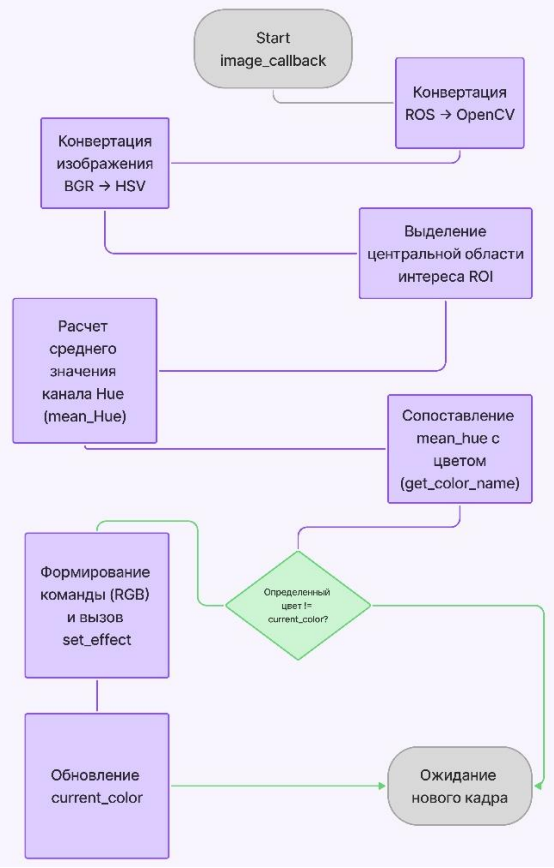


Рис. 2. Алгоритм работы узла

1. **Инициализация и настройка ROS-интерфейсов.** При запуске узел инициализирует библиотеку CvBridge для преобразования ROS-сообщений в формат numpy, пригодный для обработки OpenCV. Далее узел создает клиента для ROS-сервиса /led/set_effect типа SetLEDEffect из пакета clover, который является стандартным интерфейсом управления бортовой светодиодной подсветкой платформы «Клевер».

2. **Предварительная обработка кадра.** Каждое входящее изображение из топика /main_camera/image_raw (тип sensor_msgs/Image) конвертируется внутри функции-обработчика image_callback(msg). Сообщение преобразуется из формата ROS в формат BGR библиотеки OpenCV, а затем в цветовое пространство HSV (Hue, Saturation, Value). Это преобразование критически важно, так как канал тона (Hue) инвариантен к изменениям освещенности, что повышает устойчивость алгоритма к переменным внешним условиям.

3. **Сегментация области интереса (ROI) и анализ цвета.** Для анализа выделяется центральная квадратная область кадра с фиксированным радиусом (r = 20 пикселей). Такой подход фокусирует алгоритм на ключевой точке обзора, минимизирует влияние фоновых помех и снижает вычислительную нагрузку на бортовой компьютер Raspberry Pi. Для выделенной ROI вычисляется среднее арифметическое значение канала тона (mean_hue). Данное значение передается в функцию get_color_name(mean_hue), которая на основе пороговой логики (см. Таблицу 1) производит классификацию, относя тон к одному из семи предопределенных категорий цветов: красный, оранжевый, желтый, зеленый, синий, фиолетовый.

4. **Принятие решения и формирование управляющего воздействия.** Полученное семантическое название цвета сравнивается с глобальной переменной current_color. При фиксации изменения вызывается функция set_led_color(color_name). Данная функция использует статический словарь color_map для преобразования названия цвета в соответствующие значения интенсивности для красного (R), зеленого (G) и синего (B) каналов модели RGB. Сформированный набор параметров (эффект 'fill' и значения R, G, B) передается через ранее созданного клиента в сервис /led/set_effect, что приводит к немедленному изменению цвета всей светодиодной подсветки дрона.

В ходе выполнения работы было завершено исследование, направленное на создание программно-аппаратного комплекса для визуализации видеопотока с камеры беспилотного летательного аппарата на светодиодную панель в режиме реального времени. В качестве базовой платформы использован открытый учебный конструктор квадрокоптера «Клевер 4», что позволило обеспечить высокую степень гибкости и воспроизводимости разработанных решений.

Заключение. В ходе выполнения работы было завершено исследование, направленное на создание программно-аппаратного комплекса для визуализации видеопотока с камеры беспилотного летательного аппарата на светодиодную панель в режиме реального времени. В качестве базовой платформы использован открытый учебный конструктор квадрокоптера «Клевер 4», что позволило обеспечить высокую степень гибкости и воспроизводимости разработанных решений.

Основные научные и практические результаты работы:

1. Проведен анализ архитектуры и программных интерфейсов платформы «Клевер». Установлено, что среда Robot Operating System (ROS) предоставляет стандартизированные топики для доступа к видеопотоку (/main_camera/image_raw) и сервисы для управления исполнительными устройствами (/led/set_effect), что делает её эффективной основой для построения систем подобного класса.

2. Разработан и реализован программный модуль – ROS-узел color_detection_led_control на языке Python. В его состав входит функция-обработчик image_callback, интегрирующая полный цикл обработки данных: захват кадра, конвертацию цветовых пространств (BGR → HSV), сегментацию центральной области интереса (ROI), вычисление среднего значения тона (Hue) и классификацию цвета на основе пороговых значений. Предложенный алгоритм отличается низкой вычислительной сложностью, что позволяет выполнять его непосредственно на бортовом компьютере Raspberry Pi без привлечения внешних вычислительных мощностей.

3. Осуществлена интеграция с устройством визуализации. Управляющий сигнал, сформированный на основе детектируемого цвета, передается на светодиодную ленту типа WS281x посредством вызова ROS-сервиса set_effect с параметром fill. Данный подход обеспечивает синхронное изменение состояния индикации при смене наблюдаемого объекта в поле зрения камеры.

Экспериментальная апробация разработанного программного модуля проводилась в условиях лабораторного полигона, развёрнутого на базе факультета информационных технологий и математики ФГБОУ ВО «Нижневартовский государственный университет». Целью испытаний являлась проверка корректности детекции базовых цветов при наведении камеры беспилотного аппарата на соответствующие объекты-маркеры. На рисунках 3 и 4 представлены характерные кадры, фиксирующие момент распознавания цвета и последующее изменение состояния светодиодной индикации.

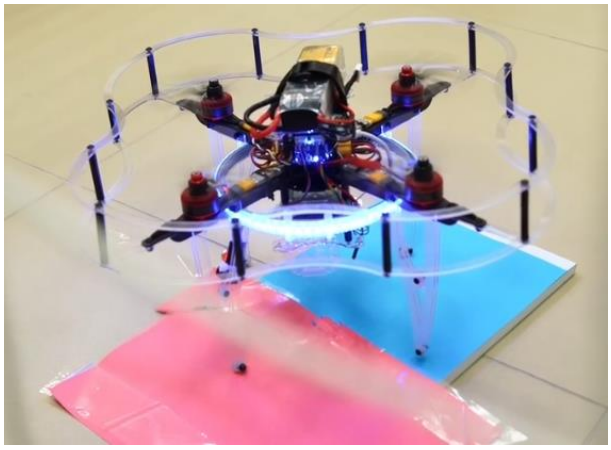


Рис. 3. Определение дроном синего цвета в полете

Рис. 4. Определение дроном красного и фиолетового цвета

Перспективы дальнейшей работы. Полученные результаты открывают направления для развития представленной системы:

- Интеграция с телеметрией: дополнение системы отображением полётных параметров (высота, заряд батареи, координаты) непосредственно на LED-панели, что повысит информативность визуализации при проведении испытаний и демонстраций.
- Оптимизация временных задержек: проведение детального профилирования системы с целью минимизации латентности «от камеры до панели», что является критическим фактором для задач оперативного мониторинга.

Таким образом, поставленная цель исследования достигнута: создан и экспериментально апробирован действующий прототип системы визуализации, демонстрирующий возможность эффективного использования открытой платформы «Клевер» в задачах интерактивного отображения данных. Разработанное программное обеспечение может быть рекомендовано к применению в образовательном процессе при изучении основ робототехники, компьютерного зрения и встраиваемых систем, а также в качестве основы для дальнейших научно-исследовательских работ в области автономной навигации БПЛА.

Литература

1. Баранов В.С. Разработка алгоритмов построения кратчайших маршрутов и управления параметрами полета гражданских БПЛА // Вестник науки. 2025. № 5(86). С. 1331-1336. URL: https://clck.ru/3TLo8g (дата обращения: 23.01.2026).

2. Вардан С., Ваагн М., Севак С. Локализация целей в реальном времени на БПЛА с лазерным дальномером на подвесе // Труды ИСП РАН. 2025. № 4-1. С. 189-198. URL: https://clck.ru/3TLoAS (дата обращения: 13.01.2026).

3. Костин А.С., Майоров Н.Н. Разработка автоматизированных решений для исследования вариантов маршрутов доставки при совместном использовании транспортного средства и беспилотной авиационной системы в границах города // Известия ТулГУ. Технические науки. 2022. № 7. С. 348-356. URL: https://clck.ru/3TLoBC (дата обращения: 11.03.2026).

4. Майоров Н.Н., Костин А.С. Автоматизация процесса идентификации объектов при выполнении автономных полетных заданий беспилотной авиационной системой // Известия

ТулГУ. Технические науки. 2022. № 2. С. 640-646. URL: https://clck.ru/3TLoCQ (дата обращения: 23.01.2026).

5. Сацюк А.В., Швалов Д.В. Автономное наведение бпла с использованием компьютерного зрения: проблема точного управления рулями // Автоматика на транспорте. 2024. № 4. С. 372-381.

Спектор И.В., Минаев Е.Ю.
Самарский национальный исследовательский университет им. академика Королева
г. Самара, Россия

ИССЛЕДОВАНИЕ И РАЗРАБОТКА МЕТОДОВ СШИВАНИЯ ИЗОБРАЖЕНИЙ С БПЛА ДЛЯ КОНТРОЛЯ ГРАНИЦ ЗОНЫ ПОЛЁТА

Беспилотные летательные аппараты (БПЛА) активно применяются в сельском хозяйстве для мониторинга полей, картографии и других задач. Одной из ключевых проблем является построение точных панорамных изображений на основе видеоряда с БПЛА. Это необходимо для контроля границ зоны полёта и последующего анализа данных.

Цель работы – разработка и оптимизация алгоритмов сшивания изображений, обеспечивающих высокую точность и скорость обработки. В статье рассматриваются методы SIFT и FLANN, их реализация на Python и C++, а также предложены модификации для работы в реальном времени.

Сшивание изображений с БПЛА [3; 5; 8] – это процесс объединения множества перекрывающихся снимков в единое панорамное изображение с высокой детализацией. Эта задача актуальна для картографии, мониторинга сельского хозяйства [1; 2; 4; 6; 10], поисково-спасательных операций и других областей.

Существует множество онлайн-ресурсов для сшивания изображений, однако большинство из них платные и не могут работать в полевых условиях. Есть и решения о сшивании панорам в открытых источниках, но они не способны работать в режиме реального времени. Нейросетевые методы глубокого обучения [12] перспективны и показывают высокую точность, но требуют больших вычислительных мощностей. Нейронные сети с End-to-End подходом [11] до сих пор уступают классическим методам в точности и стабильности. Основная проблема применения нейронных сетей заключается в отсутствии обучающих данных в достаточном количестве.

Традиционные методы сшивания изображений включают в себя SIFT [9] (Scale-Invariant Feature Transform), SURF [7] (Speeded-Up Robust Features) и ORB (Oriented FAST and Rotated BRIEF). Для задачи сшивания изображений сельскохозяйственных полей требуется высокая точность, которую можно получить, лишь используя SIFT. Поэтому необходимо было модернизировать метод, чтобы с помощью него можно было получать панораму в реальном времени.

Метод SIFT (Scale-Invariant Feature Transform) – это алгоритм выделения и описания локальных особенностей изображения с помощью дескрипторов, предложенный Дэвидом Лоу в 1999 году. Основное преимущество SIFT – инвариантность к масштабу, повороту, изменению освещенности и частичная устойчивость к аффинным искажениям.

Первоначальная версия алгоритма использовала OpenCV SIFT для детектирования ключевых точек и их сопоставления. Основные этапы алгоритма:

1. преобразование кадров в градации серого;
2. применение SIFT для выделения дескрипторов;
3. сопоставление точек с помощью FLANN;

4. фильтрация ложных соответствий с использованием теста Лоу;

5. построение панорамы с применением матрицы гомографии.

Тестирование проводилось на пяти видеозаписях длительностью от 10 до 50 секунд на Raspberry Pi 5 с 8 Гб оперативной памяти и процессором Cortex-A76, который можно установить на беспилотные летательные аппараты. Первоначальные результаты, представленные на рисунке 1, показали, что при увеличении длительности видеозаписи, время обработки возрастало в геометрической прогрессии. Причиной этому являлось увеличение размера панорамного изображения по мере работы алгоритма.

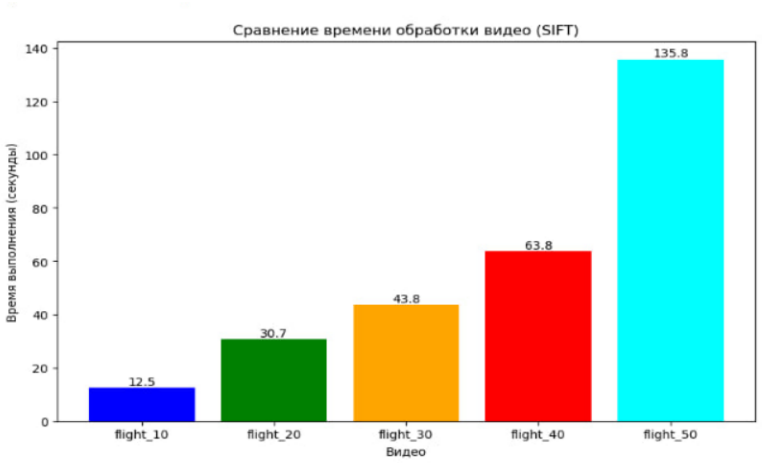


Рис. 1. Время выполнения обработки видео на Python без модификаций

Для ускорения обработки была введена концепция области интереса (Region of Interest или ROI). Вместо обработки всей панорамы, алгоритм фокусируется на зоне последнего сшивания, что сокращает время вычислений и уменьшает количество ложных соответствий. Результаты тестирования, представленные на рисунке 2, демонстрируют линейную зависимость времени работы алгоритма от длительности видеоряда.

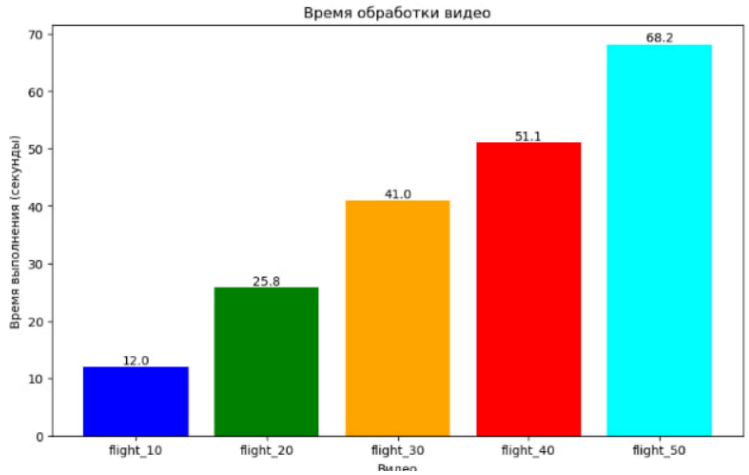


Рис. 2. Время выполнения обработки видео на Python с Region of Interest

Переход на C++ позволил значительно ускорить обработку за счёт оптимизации памяти и использования возможностей OpenCV. Дополнительное распараллеливание вычислений на

процессоре позволило сократить время выполнения алгоритма до реального времени, как представлено на рисунке 3.

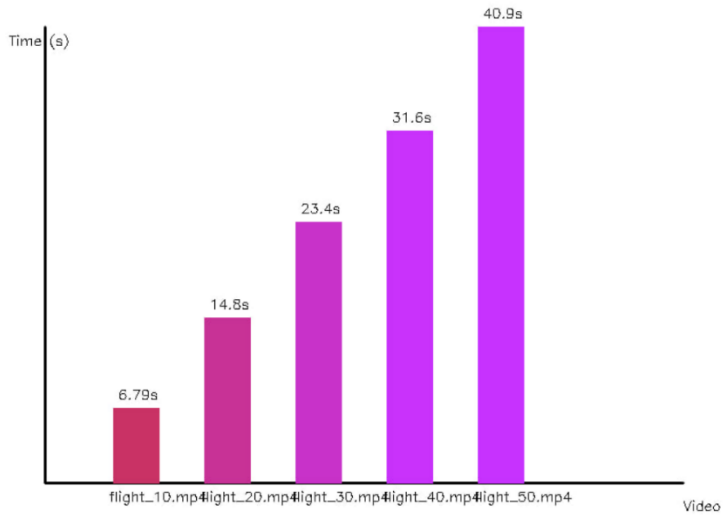


Рис. 3. Время выполнения обработки видео на C++ с Region of Interest

Для реализации алгоритма пригодного для промышленного применения был проведён анализ влияния количества дескрипторов и частоты обработки кадров на производительность. Рассматривались значения параметра nfeatures = {500, 5000, 10000, 15000, 30000} и частота обработки {1 из 5 кадров (6 fps), 1 из 15 кадров (2 fps), 1 из 30 кадров (1 fps)}. Полная сравнительная характеристика представлена на рисунке 4.

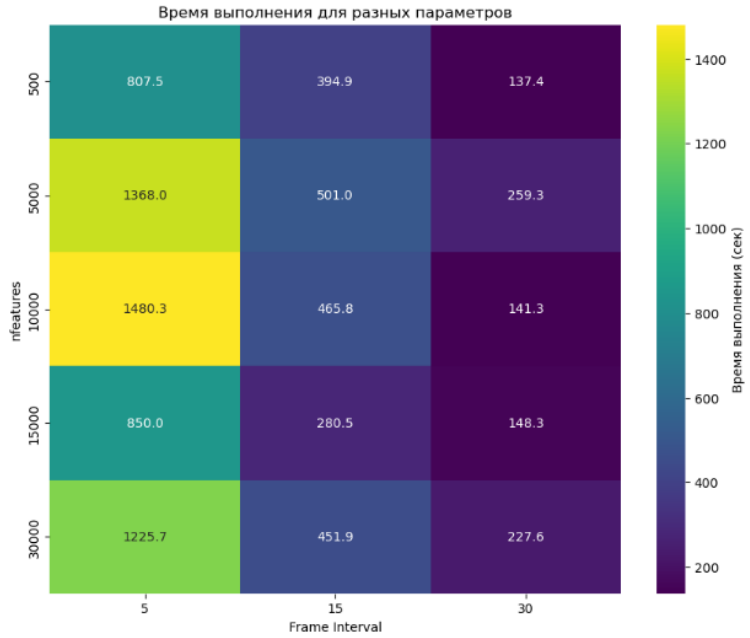


Рис. 4. Время выполнения алгоритма в зависимости от количества дескрипторов и частоте обработки фреймов

Анализ данных, приведённых на рисунке 4, показывает, что оптимальным с точки зрения соотношения скорости и качества является режим с обработкой 1 кадра в секунду и количеством дескрипторов 15000-20000. При данном количестве дескрипторов удаётся

надёжно сопоставлять кадры даже при значительных изменениях ландшафта, а время обработки одного кадра на C++ не превышает 0.8 секунды, что позволяет говорить о работе, близкой к реальному времени. Дальнейшее увеличение количества дескрипторов до 30000 не даёт существенного прироста качества, но приводит к росту вычислительной нагрузки на 30-40%.

Как видно из рисунка 5, предложенная модификация алгоритма позволяет получать целостное изображение сельскохозяйственного поля без видимых швов и геометрических искажений. Панорама сохраняет чёткость границ между участками поля и пригодна для дальнейшего анализа состояния посевов или контроля обработанной площади.

Рис. 5. Пример панорамы полученной алгоритмом на C++ с модификацией области интереса

Панорамы, полученные с использованием ROI, демонстрируют меньшие искажения по сравнению с базовой версией. В ходе экспериментов были выявлены следующие основные проблемы, влияющие на качество сшивания:

– наложение кадров при изменении высоты полёта;

– влияние облачности на контрастность изображений.

Для улучшения результатов в условиях промышленной эксплуатации рекомендовано:

– поддерживать постоянную высоту полёта (с допустимым отклонением не более 5 метров);

– избегать съёмки в условиях переменной облачности или резкой смены освещения.

Разработанный модуль на C++ с использованием области интереса демонстрирует высокую эффективность и пригоден для промышленного применения в системах мониторинга сельскохозяйственных угодий. Основные достижения работы:

1. достигнуто линейное время обработки видеоряда (O(n) вместо $O(n^2)$) в базовой реализации), что подтверждено экспериментальными данными.

2. обеспечена устойчивость алгоритма к шумам и изменениям масштаба благодаря использованию инвариантного дескриптора SIFT.

3. подтверждена возможность интеграции модуля в системы автономной.

Перспективы дальнейших исследований включают внедрение GPU-ускорения для обработки видеопотока в режиме 30 кадров в секунду, а также исследование возможности применения гибридных методов, сочетающих точность SIFT с быстротой более лёгких дескрипторов на этапе предварительного поиска соответствий.

Литература

1. Bai X., Cao Z., Wang Y., Yu Z., Zhang X., Li C. Crop Segmentation from Images by Morphology Modeling in the CIE L*a*b* Color Space // Computers and Electronics in Agriculture. 2013. Vol. 99. P. 21-34. https://doi.org/10.1016/j.compag.2013.08.022

2. Bendig J., Bolten A., Bennertz S., Broscheit J., Eichfuss S., Bareth G. Estimating Biomass of Barley Using Crop Surface Models (CSMs) Derived from UAV-Based RGB Imaging // Remote Sensing. 2014. Vol. 6. P. 10395-10412. https://doi.org/10.3390/rs61110395

3. Chen J., Li Z., Peng C., Wang Y., Gong W. UAV Image Stitching Based on Optimal Seam and Half-Projective Warp // Remote Sensing. 2022. Vol. 14, № 5. P. 1068. https://doi.org/10.3390/rs14051068

4. Hassanein M., Lari Z., El-Sheimy N. A New Vegetation Segmentation Approach for Cropped Fields Based on Threshold Detection from Hue Histograms // Sensors. 2018. Vol. 18. P. 1-25. https://doi.org/10.3390/s18041253

5. Khan Z. An Automatic Field Plot Extraction Method from Aerial Orthomosaic Images // Frontiers in Plant Science. 2019. Vol. 10. P. 683. https://doi.org/10.3389/fpls.2019.00683

6. Lelong C., Burger P., Jubelin G., Roux B., Labbe S., Baret F. Assessment of Unmanned Aerial Vehicles Imagery for Quantitative Monitoring of Wheat Crop in Small Plots // Sensors. 2008. Vol. 8. P. 3557-3585. https://doi.org/10.3390/s8053557

7. Panchal P. M., Panchal S. R., Shah S. K. A Comparison of SIFT and SURF // International Journal of Innovative Research in Computer and Communication Engineering. 2013. Vol. 1, № 2. P. 323-327.

8. Pham N. T., Park S., Park C. S. Fast and Efficient Method for Large-Scale Aerial Image Stitching // IEEE Access. 2021. Vol. 9. P. 127852-127865. https://doi.org/10.1109/ACCESS.2021.3111203

9. Rublee E., Rabaud V., Konolige K., Bradski G. ORB: An Efficient Alternative to SIFT or SURF // 2011 International Conference on Computer Vision. IEEE, 2011. P. 2564-2571.

10. Sankaran S., Khot L., Espinoza C., Jarolmasjed S., Sathuvalli V., Vandemark G. Low-Altitude, High-Resolution Aerial Imaging Systems for Row and Field Crop Phenotyping: A Review // European Journal of Agronomy. 2015. Vol. 70. P. 112-123. https://doi.org/10.1016/j.eja.2015.07.004

11. Shen C., Ji X., Miao C. Real-Time Image Stitching with Convolutional Neural Networks // 2019 IEEE International Conference on Real-time Computing and Robotics (RCAR). IEEE, 2019. P. 192-197. https://doi.org/10.1109/RCAR47638.2019.9044036

12. Yan N. et al. Deep Learning on Image Stitching with Multi-Viewpoint Images: A Survey // Neural Processing Letters. 2023. Vol. 55, № 4. P. 3863-3898.

Учителев М.В.
Нижневартовский государственный университет
г. Нижневартовск, Россия

ВСЕНАПРАВЛЕННАЯ МОБИЛЬНАЯ ПЛАТФОРМА И ДИСТАНЦИОННОЕ УПРАВЛЕНИЕ ПО BLUETOOTH

В данной статье рассматривается проект, в рамках которого разработаны два устройства – мобильная платформа и пульт управления, каждое из которых реализовано на базе отдельной отладочной платы Arduino Leonardo и использует беспроводную технологию Bluetooth для организации связи между устройствами.

Статью можно разделить на две основные части: аппаратную и программную часть проекта. В первой будет рассматриваться схема соединения устройств между собой и возможные варианты при создании проекта. В программной части будет предоставлена логика передвижения платформы и передачи данных между устройствами.

В связи с тем, что в проекте используются два устройства, различающиеся по назначению и архитектуре, аппаратная часть системы разделена на две функциональные группы.

Первая группа – мобильная платформа, аппаратная часть которой включает (рис. 1):

- конструктивные элементы: основание платформы, электродвигатели постоянного тока с редукторами (3 шт.), моторные кронштейны (3 шт.), монтажные ступицы (3 шт.), всенаправленные колёса (3 шт.);
- электронные компоненты: отладочную плату Arduino Leonardo, Bluetooth-модуль HC-06, четырёхканальный драйвер двигателей L9110S и источник питания с рабочим напряжением 5-12 В.

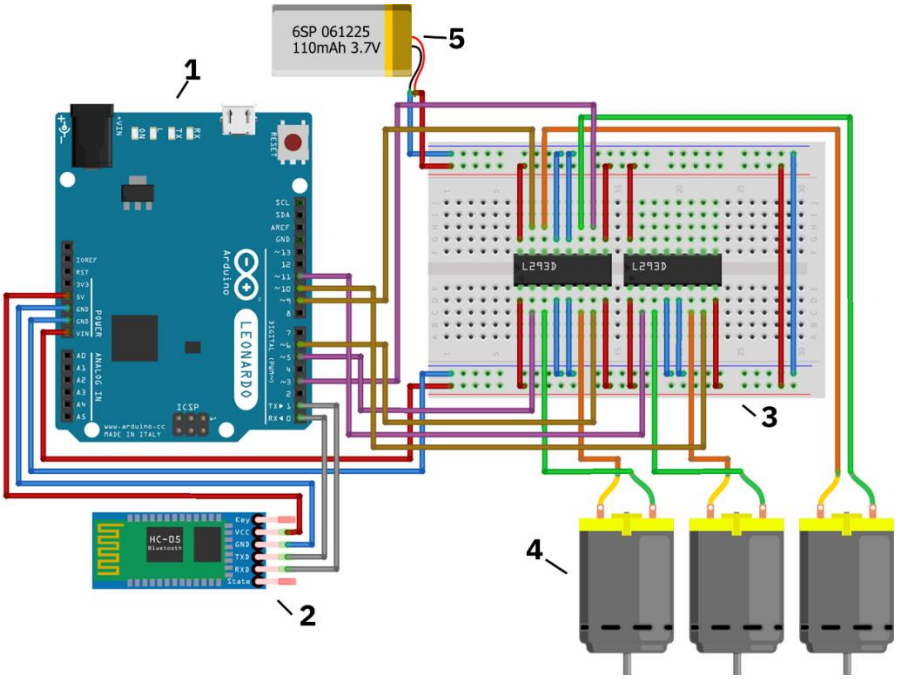


Рис. 1. Схема подключения электроники платформы: 1 – Плата Leonardo; 2 – Модуль Bluetooth HC-06; 3 – Аналог драйвера двигателей на основе двух микросхем L293D; 4 – Электродвигатели; 5 – Элемент питания всех устройств

Вторая группа – устройство управления мобильной платформой, в которую входят (рис. 2):

- отладочная плата Arduino Leonardo;
- плата расширения Power Shield;
- Bluetooth-модуль HC-05 (Troyka);
- модуль 3D-Joystick (Troyka).

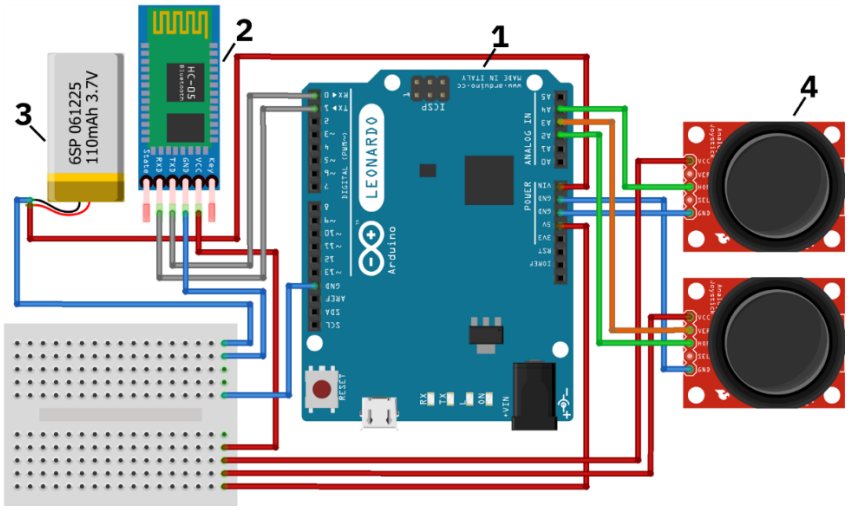


Рис. 2. Схема подключения электроники контроллера: 1 – Плата Leonardo; 2 – Модуль Bluetooth HC-05; 3 – Элемент питания всех устройств; 4 – Аналоговый джойстик

Рассмотрим каждый элемент подробнее.

Arduino Leonardo – это отладочная плата на базе микроконтроллера ATmega32u4. Обеспечивает выполнение управляющих алгоритмов, обработку входных данных и обмен данными с периферийными устройствами. Выбор платы обусловлен её доступностью и возможностью использования в рамках прототипирования, а также совместимостью с используемыми модулями и библиотеками платформы Arduino. Поддерживает необходимое для реализации проекта количество ШИМ-выходов. Наличие шести аппаратных ШИМ-портов обеспечивает возможность независимого управления скоростью двигателей [4].

В рамках проекта применяются две одинаковые микроконтроллерные платы Arduino Leonardo – в устройстве управления и в мобильной платформе. Использование идентичных аппаратных платформ позволило унифицировать архитектуру системы, исключить различия в реализации последовательных интерфейсов и сократить вероятность ошибок при передаче и обработке данных по Bluetooth-каналу.

Модуль 3D-Joystick Troyka – входное устройство управления, включающее двухосевой аналоговый джойстик. Обеспечивает формирование аналоговых сигналов по осям X и Y, что позволяет использовать модуль для задания направления и интенсивности управления.

Передаваемые в рамках проекта данные представляют собой структуру, содержащую нормализованные значения, полученные с модуля 3D-Joystick. Первичная обработка и нормализация входных данных выполняются на плате управляющего устройства, после чего сформированная структура передаётся по каналу Bluetooth. На принимающем устройстве

данные используются для формирования управляющих воздействий. Подробное описание формата структуры и алгоритмов обработки приведено в разделе, посвящённом программной реализации.

Bluetooth – это технология беспроводной связи малого радиуса действия, предназначенная для обмена данными между электронными устройствами. Передача данных осуществляется с использованием радиоканала. Обеспечивает простоту подключения, низкое энергопотребление и достаточную пропускную способность для систем дистанционного управления, что делает данную технологию удобной для использования в мобильных робототехнических и встраиваемых системах [5].

В проекте для организации беспроводного обмена данными между устройством управления и мобильной платформой используется технология Bluetooth 2.0 с применением модулей HC-05 и HC-06.

Для управления двигателями постоянного тока малой и средней мощности была применена микросхема драйвера L9110S. Она обеспечивает управление направлением вращения и скоростью двигателей за счёт подачи логических сигналов и ШИМ-управления. Четырёхканальная конфигурация позволяет независимо управлять несколькими двигателями.

Особенностью данного проекта является использование технологии омниколёс, обеспечивающих всенаправленное движение платформы без необходимости изменения её ориентации в пространстве. Конструктивная особенность омниколеса заключается в наличии дополнительных роликов, расположенных по периферии основного колеса и ориентированных перпендикулярно его плоскости вращения. Такая структура позволяет колесу свободно компенсировать боковые компоненты движения за счёт пассивного вращения роликов [3].

В ходе разработки отдельные элементы конструкции были спроектированы в среде трёхмерного моделирования и изготовлены с применением технологий аддитивного производства (3D-печати) [1].

Управление системой осуществляется с использованием двух аналоговых органов управления (стиков). Первый стик генерирует двухкоординатный сигнал, определяющий вектор линейного перемещения платформы. Второй стик задействует одну координату и формирует управляющее воздействие, соответствующее вращению платформы вокруг собственной оси (рис. 3).

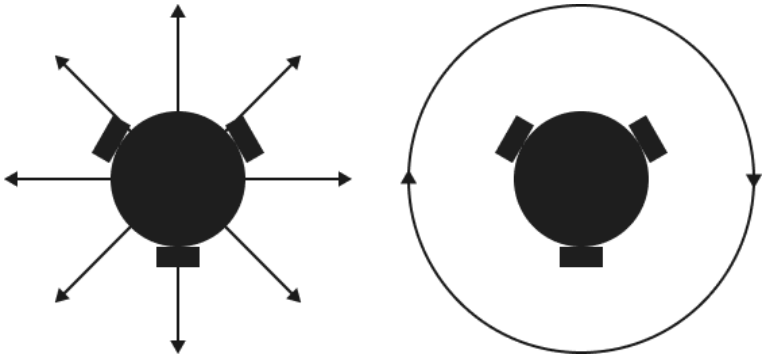


Рис. 3. Теоретическое управление

Первичная обработка данных осуществляется непосредственно на контроллере и включает считывание аналоговых сигналов с органов управления, их последующую нормализацию к заданному диапазону значений, формирование структурированного пакета фиксированного формата и передачу данного пакета по беспроводному каналу связи Bluetooth на принимающее устройство.

Обмен данными осуществляется посредством передачи структурированного пакета, содержащего значения координат стиков. Передача выполняется только при обнаружении изменения данных относительно предыдущего цикла считывания.

Принимающее устройство выполняет декодирование пакета данных, вычисление требуемых скоростей электродвигателей на основе кинематической модели платформы. На данном этапе происходит преобразование абстрактного вектора направления в конкретные значения скоростей каждого двигателя.

Расчёт индивидуальных скоростей электродвигателей производится в соответствии с выбранной кинематической моделью платформы происходит по следующей формуле:

$$\omega i = -sin(\theta i) * vx + cos(\theta i) * vy + L * \omega)$$

где vx и vy – линейные скорости платформы в плоскости, L – радиус платформы, ω – угловая скорость платформы вокруг вертикальной оси [2].

Расчёт основан на преобразовании входных нормализованных управляющих параметров в индивидуальные скорости каждого исполнительного электродвигателя.

В результате вычислений формируются предварительные значения скоростей электродвигателей, которые могут превышать допустимые пределы при сохранении заданного соотношения компонентов движения. В связи с этим выполняется взаимная нормализация полученных значений: максимальная по модулю скорость принимается за опорную, после чего остальные значения масштабируются пропорционально ей. Данный подход позволяет сохранить направление и характер движения при одновременном ограничении управляющих сигналов в пределах допустимого диапазона. После расчёта управляющие сигналы передаются на драйвер электродвигателей, обеспечивая реализацию требуемого движения платформы.

В итоге была создана система, состоящая из двух частей: управляющий модуль и сама платформа (рис. 4).

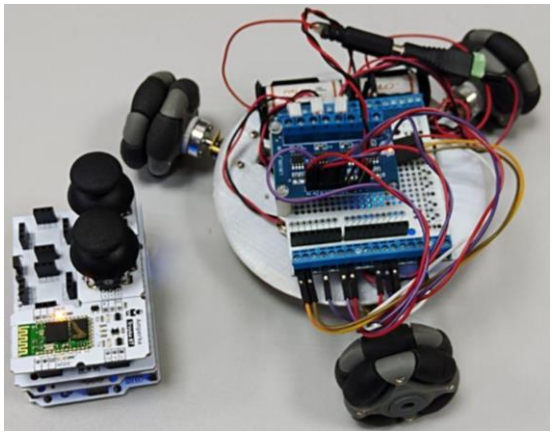


Рис. 4. Прототип разработки платформы и контроллера

Литература

1. Богаткин, М.А. Особенности FDM 3D-печати: от идеи до готового изделия // Постулат. 2022. № 12(86).

2. Брагина А.Э., Чикалов Н.В. Разработка робототехнической омниплатформы для маневренного перемещения в ограниченном пространстве // Актуальные проблемы инфотелекоммуникаций в науке и образовании (АПИНО 2025): Сборник научных статей XIV Международной научно-технической и научно-методической конференции. В 2-х томах (Санкт-Петербург, 25–27 июня 2025 года). Санкт-Петербург: Санкт-Петербургский государственный университет телекоммуникаций им. проф. М.А. Бонч-Бруевича, 2025. С. 272-276.

3. Кондаков И.С., Агафонов М.А. Сравнение трехколесной и четырехколесной компоновки мобильного робота на основе всенаправленных колес // Актуальные проблемы авиации и космонавтики: Сборник материалов XI Международной научно-практической конференции, посвященной Дню космонавтики и 10-летию науки и технологий. В 3-х томах (Красноярск, 07–11 апреля 2025 года). Красноярск: Сибирский государственный университет науки и технологий им. акад. М.Ф. Решетнева, 2025. С. 37-39.

4. Меньшиков С.В., Ващук Е.С. Осуществление программирования на платформе Arduino: способы и возможности использования платы Arduino UNO // Современные вопросы естествознания и экономики: Сборник трудов IV Международной научно-практической конференции (Прокопьевск, 17 марта 2022 года). Прокопьевск: Филиал ФГБОУ ВПО «Кузбасский государственный технический университет имени Т.Ф. Горбачева» в г. Прокопьевске, 2022. С. 256-262.

5. Садеков, Д. Решения Texas Instruments для беспроводных сетей Bluetooth / Д. Садеков // Электроника: Наука, технология, бизнес. 2021. № 7(208). С. 120-123. https://doi.org/10.22184/1992-4178.2021.208.7.120.123

Хаиров Т.А., Жажнева И.В., Еникеева А.И.
Казанский (Приволжский) федеральный университет
г. Казань, Россия

ПРИМЕНЕНИЕ МЕТОДОВ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА И МАШИННОГО ОБУЧЕНИЯ В ЦИФРОВОЙ ТРАНСФОРМАЦИИ СТРОИТЕЛЬНОГО ПРОЕКТИРОВАНИЯ

В работе исследуется роль технологий искусственного интеллекта (ИИ) в развитии систем автоматизированного проектирования (САПР) и информационного моделирования зданий (BIM). Рассматриваются современные подходы к интеграции алгоритмов машинного обучения (ML) для оптимизации архитектурно-строительных решений, прогнозирования рисков и анализа коллизий. Особое внимание уделяется архитектуре взаимодействия источников данных, BIM-моделей и алгоритмов искусственного интеллекта и машинного обучения. Выявлены основные технические и организационные барьеры внедрения интеллектуальных систем в строительную отрасль, а также определены перспективы развития технологий в контексте Индустрии 4.0.

Цифровые технологии на базе искусственного интеллекта (ИИ) оказывают значительное влияние на развитие строительного проектирования, автоматизируя процессы моделирования, расчёта, анализа и оптимизации проектных решений [1, с. 84]. ИИ активно применяется при создании BIM-моделей, прогнозировании рисков, управлении сроками и бюджетом строительства, а также при анализе инженерных данных. Использование нейронных сетей и методов машинного обучения позволяет повысить точность расчётов, сократить количество проектных ошибок и снизить финансовые издержки. Несмотря на значительный потенциал, внедрение ИИ в строительную отрасль сопровождается рядом проблем, включая необходимость стандартизации данных, высокие затраты на внедрение технологий и необходимость подготовки специалистов нового уровня. В целом искусственный интеллект становится важным инструментом цифровой трансформации строительной отрасли [5, с. 194].

Зарождение искусственного интеллекта как научного направления относится к середине XX века и связано с развитием вычислительной техники и теории алгоритмов. Первые исследования в области моделирования интеллектуальных процессов на основе математических формализаций заложили основу для последующего создания нейронных сетей и экспертных систем. В строительной сфере цифровая трансформация началась значительно позже, однако в XXI веке она получила мощный импульс благодаря внедрению технологий информационного моделирования зданий (BIM) [4, с. 51].

Современные системы проектирования позволяют создавать цифровые модели объектов, включающие архитектурные, конструктивные и инженерные решения. Сегодня искусственный интеллект интегрируется в программные комплексы для автоматизации проектирования. Например, программные решения компании Autodesk используют элементы машинного обучения для оптимизации архитектурных решений и анализа проектных параметров. В рамках концепции Generative Design алгоритмы способны автоматически генерировать десятки и сотни вариантов планировок с учётом заданных ограничений – площади, освещённости, нагрузки на конструкции и стоимости материалов. ИИ активно

применяется в расчёте строительных конструкций. Нейронные сети способны анализировать большие массивы данных о нагрузках, свойствах материалов и геометрических параметрах, выявляя оптимальные проектные решения. Это особенно актуально при проектировании высотных зданий, мостов и сложных инженерных сооружений, где требуется учёт множества факторов одновременно. Одним из ключевых направлений внедрения ИИ является прогнозирование рисков на этапе проектирования. Алгоритмы машинного обучения анализируют статистику прошлых проектов и позволяют заранее определить вероятность превышения бюджета, задержки сроков строительства или возникновения технических проблем. Такие решения используются крупными строительными корпорациями для повышения эффективности управления проектами.

Важную роль искусственный интеллект играет в анализе данных BIM-моделей (Building Information Modeling). Современные интеллектуальные системы способны автоматически обрабатывать огромные массивы информации, содержащиеся в цифровой модели здания, и выявлять коллизии между инженерными сетями, конструктивными элементами и архитектурными решениями. Это позволяет заранее обнаруживать потенциальные конфликты, например, пересечение вентиляционных каналов с несущими конструкциями или некорректное размещение трубопроводов и электрических кабелей. В результате значительно сокращается количество ошибок, которые ранее выявлялись только на стадии строительства, что снижает финансовые затраты и уменьшает сроки реализации проекта.

Кроме выявления коллизий, искусственный интеллект применяется для оптимизации проектных решений. Алгоритмы машинного обучения способны анализировать различные варианты планировок и выбирать наиболее рациональные с точки зрения прочности, стоимости и энергоэффективности. В области энергоэффективного проектирования ИИ используется для моделирования теплопотерь здания, расчёта оптимального расположения окон, подбора материалов и прогнозирования энергопотребления. Это позволяет создавать более экологичные и экономичные здания, соответствующие современным стандартам устойчивого развития.

Также интеллектуальные системы способны прогнозировать износ конструкций на основе анализа эксплуатационных данных, климатических условий, нагрузок и характеристик применяемых материалов. Используя методы машинного обучения и анализ больших данных, ИИ может оценивать скорость старения строительных элементов, выявлять потенциально уязвимые зоны и заранее предупреждать о необходимости проведения ремонта или технического обслуживания. Это особенно важно для объектов с высокой эксплуатационной нагрузкой – торговых центров, производственных зданий, жилых комплексов и инфраструктурных сооружений.

Кроме прогнозирования износа, искусственный интеллект активно применяется для анализа рисков. Системы способны учитывать множество факторов: ошибки проектирования, человеческий фактор, влияние внешней среды, экономические колебания и сроки поставок материалов. На основе этих данных формируются вероятностные модели, которые позволяют заранее определить возможные отклонения от графика строительства, перерасход бюджета

или технические проблемы. Такой подход помогает принимать более обоснованные управленческие решения и минимизировать негативные последствия.

Интеллектуальные технологии также играют ключевую роль в управлении жизненным циклом здания. Начиная с этапа концептуального проектирования и заканчивая эксплуатацией и модернизацией объекта, ИИ обеспечивает непрерывный анализ данных BIM-модели. В процессе эксплуатации система может интегрироваться с датчиками (IoT), отслеживать состояние инженерных систем, контролировать энергопотребление и автоматически формировать рекомендации по оптимизации работы оборудования. Это повышает эффективность управления объектом недвижимости и снижает эксплуатационные расходы.

Таким образом, внедрение искусственного интеллекта в BIM-технологии существенно повышает качество проектирования и строительства, снижает вероятность ошибок, ускоряет принятие решений и делает процессы более прозрачными. Более того, цифровизация строительной отрасли с использованием ИИ способствует переходу к концепции «умных» зданий и устойчивого развития, что является одним из ключевых направлений развития современной архитектуры и инженерии. Алгоритмы анализируют климатические данные, ориентацию здания, параметры теплоизоляции и системы вентиляции, предлагая решения, позволяющие снизить энергопотребление и эксплуатационные расходы. В области управления строительством ИИ используется для обработки данных с датчиков и камер наблюдения на строительных площадках. Компьютерное зрение позволяет контролировать соблюдение техники безопасности, отслеживать прогресс выполнения работ и выявлять отклонения от проектной документации.

Однако внедрение искусственного интеллекта в строительное проектирование сопровождается рядом сложностей. Во-первых, необходима стандартизация цифровых данных и совместимость различных программных платформ. Во-вторых, обучение нейросетевых моделей требует больших массивов корректных и структурированных данных, которые не всегда доступны. В-третьих, существует проблема подготовки специалистов, способных работать на стыке строительства и информационных технологий. Дополнительными вызовами являются вопросы ответственности за решения, принятые алгоритмами, а также кибербезопасность и защита проектной информации [3, с. 118]. Несмотря на существующие трудности, развитие искусственного интеллекта в строительном проектировании открывает значительные перспективы. Автоматизация расчётов, повышение точности прогнозирования, снижение количества ошибок и оптимизация ресурсов позволяют существенно повысить эффективность строительной отрасли.

В будущем ожидается дальнейшая интеграция ИИ с технологиями цифровых двойников, интернетом вещей (IoT) и роботизированными строительными системами [2, с. 259]. Это приведёт к формированию полностью цифровых экосистем проектирования и строительства, где большая часть рутинных операций будет выполняться автоматически, а специалисты смогут сосредоточиться на стратегических и творческих задачах.

Перспективным направлением дальнейшего развития является формирование интегрированных цифровых платформ, объединяющих BIM-модели, цифровые двойники,

системы мониторинга на базе IoT и алгоритмы искусственного интеллекта в единую управляемую среду данных (Common Data Environment). В такой архитектуре BIM-модель перестаёт быть статическим проектным артефактом и трансформируется в динамический цифровой актив, который постоянно обновляется за счёт потоков эксплуатационных данных. Это обеспечивает переход от реактивного управления объектом к предиктивному и адаптивному управлению.

Технология цифровых двойников предполагает создание виртуальной копии здания или инфраструктурного объекта, синхронизированной с физическим объектом в режиме реального времени. Интеграция ИИ позволяет анализировать телеметрические данные, выявлять аномалии функционирования инженерных систем, моделировать сценарии отказов и рассчитывать оптимальные режимы эксплуатации. В результате повышается надёжность конструкций, сокращаются затраты на техническое обслуживание и продлевается срок службы объекта.

Значительный потенциал имеет применение методов глубокого обучения и компьютерного зрения на строительных площадках. Использование беспилотных летательных аппаратов и стационарных камер в сочетании с алгоритмами распознавания образов позволяет автоматически сопоставлять фактическое состояние строительства с BIM-моделью, контролировать качество выполненных работ и фиксировать отклонения от проектных решений. Это формирует предпосылки для внедрения автономных систем строительного контроля и повышения прозрачности процессов.

Отдельного внимания заслуживает интеграция ИИ с роботизированными строительными системами и аддитивными технологиями. Роботы-манипуляторы и автоматизированные комплексы 3D-печати зданий способны выполнять строительные операции на основе данных BIM-модели, а алгоритмы машинного обучения оптимизируют траектории движения, расход материалов и параметры укладки. В перспективе это может привести к созданию частично или полностью автономных строительных площадок, функционирующих по принципам киберфизических систем.

С точки зрения управления проектами искусственный интеллект расширяет возможности календарно-сетевого планирования и ресурсного анализа. Предиктивные модели позволяют учитывать влияние внешних факторов – климатических условий, логистических ограничений, колебаний цен на материалы – и формировать адаптивные графики строительства. Это способствует повышению устойчивости проектов к неопределённости и снижению финансовых рисков.

Важным направлением является развитие интероперабельности и стандартизации данных. Использование открытых форматов информационного моделирования (например, IFC) в сочетании с онтологиями строительной предметной области создаёт основу для корректной интеграции алгоритмов ИИ в различные программные среды. Формирование единого информационного пространства повышает масштабируемость решений и снижает зависимость от конкретных вендоров.

Также следует учитывать этический и нормативный аспект внедрения интеллектуальных технологий. Возникает необходимость разработки регламентов ответственности за решения, принимаемые алгоритмами, а также методик валидации и сертификации ИИ-систем в строительной отрасли. Дополнительное значение приобретает защита информационных моделей от несанкционированного доступа и кибератак, поскольку BIM-данные содержат критически важную техническую информацию.

В контексте Индустрии 4.0 строительная отрасль постепенно трансформируется в высокотехнологичную цифровую экосистему, где ключевыми элементами становятся данные, автоматизация и интеллектуальная аналитика. Искусственный интеллект выступает связующим звеном между проектированием, строительством и эксплуатацией, обеспечивая непрерывность информационного потока на протяжении всего жизненного цикла объекта.

Таким образом, дальнейшее развитие ИИ в системах САПР и BIM связано с углублением интеграции алгоритмов машинного обучения, расширением применения цифровых двойников, роботизацией строительных процессов и формированием единых стандартов данных. Комплексное внедрение интеллектуальных технологий способно обеспечить качественный переход строительной отрасли к модели устойчивого, управляемого и технологически продвинутого развития.

Дополнительным вектором развития является внедрение методов генеративного проектирования на основе многокритериальной оптимизации. В рамках такого подхода алгоритмы ИИ формируют проектные решения с учётом одновременно действующих ограничений: нормативных требований, геотехнических условий, транспортной доступности, экологических параметров и экономической эффективности. Использование эволюционных алгоритмов, градиентной оптимизации и нейросетевых моделей позволяет находить нетривиальные архитектурно-конструктивные конфигурации, которые сложно получить традиционными методами инженерного расчёта.

Существенное значение приобретает применение графовых нейронных сетей для анализа структурных связей в BIM-моделях. Поскольку информационная модель здания представляет собой сложную иерархическую систему взаимосвязанных элементов (узлов, конструкций, инженерных сетей), графовые подходы позволяют учитывать топологические зависимости и прогнозировать влияние изменений одного элемента на другие компоненты системы. Это повышает точность анализа коллизий, устойчивости конструкций и взаимного влияния инженерных подсистем.

Развитие методов обработки естественного языка (NLP) открывает возможности автоматизации работы с проектной документацией. Алгоритмы способны анализировать текстовые спецификации, нормативные документы и технические задания, выявлять несоответствия требованиям, автоматически формировать отчёты и извлекать структурированные данные для последующей интеграции в BIM-среду. Это снижает нагрузку на специалистов и уменьшает вероятность человеческих ошибок при интерпретации нормативной базы.

Перспективным направлением является создание адаптивных моделей, обучающихся на данных конкретной строительной компании или региона. Такие модели учитывают локальные климатические условия, специфику применяемых материалов, логистические особенности и кадровые ресурсы. В результате формируются более точные прогнозы сроков реализации проектов, эксплуатационных характеристик зданий и экономических показателей.

Отдельно следует выделить развитие предиктивной аналитики в управлении строительной техникой и ресурсами. Интеллектуальные системы способны анализировать телеметрию строительных машин, прогнозировать вероятность отказов оборудования, оптимизировать графики технического обслуживания и снижать простои. Это особенно актуально для крупных инфраструктурных проектов с высокой стоимостью машино-часа.

В долгосрочной перспективе возможно формирование самообучающихся проектных экосистем, где каждый завершённый проект становится источником данных для совершенствования алгоритмов. Накопление отраслевых датасетов позволит создавать более универсальные и точные модели прогнозирования, а также переходить к частично автономному проектированию типовых объектов.

Таким образом, дальнейшее углубление интеграции искусственного интеллекта в САПР и BIM будет сопровождаться развитием генеративных методов, графовых моделей анализа, технологий обработки текстовых данных, предиктивной аналитики и адаптивных алгоритмов. Это формирует основу для перехода строительной отрасли к интеллектуальным, самооптимизирующимся цифровым системам, обеспечивающим высокий уровень эффективности, устойчивости и технологической зрелости.

Литература

1. Бойченко О.В., Фадина Ю.Ю. Цифровая трансформация экономики строительства: роль ИИ в прогнозировании рисков и ресурсов // Экономика строительства и природопользования. 2025. № 1(94). С. 84-91.

2. Каниева Б.А., Казагачева С.В. ИИ, машинное обучение и цифровые двойники: новые горизонты автоматизации производства // Передовые технологии и инновации в образовании и науке для улучшения качества жизни и стимулирования устойчивого экономического роста: сборник статей VIII Международной научно-технической конференции (г. Минск, 03–05 декабря 2025 года). Минск: Учреждение образования Белорусский государственный технологический университет, 2025. С. 259-261.

3. Копилец И.С., Сахаров Д.В. Роль машинного обучения и ИИ в обеспечении кибербезопасности // Технологии информационного общества: Сборник трудов XIX Международной отраслевой научно-технической конференции (г. Москва, 11–13 марта 2025). Москва: МТУСИ, 2025. С. 118-120.

4. Крутиков И.В. Повышение качества проектирования через внедрение технологий информационного моделирования зданий (BIM) // Молодежный научный форум: сборник статей по материалам CCC студенческой международной научно-практической конференции

(г. Москва, 05 июня 2025года). М.: Международный центр науки и образования, 2025. С. 51-57.

5. Письменова Ю.А. ИИ в планировании строительства: каковы преимущества? // Искусственный интеллект. Формирование будущего: Материалы I Международной научно-практической конференции (г. Краснодар, 29 апреля 2024). Краснодар: ИП Алзидан М., 2024. С. 194-197.

Шуляр К.Н.
Нижневартовский государственный университет
г. Нижневартовск, Россия

ПРОЕКТИРОВАНИЕ БАЗОВОГО ПРОГРАММНОГО МОДУЛЯ ДЛЯ УПРАВЛЕНИЯ РОБОТОТЕХНИЧЕСКОЙ ПЛАТФОРМОЙ С ИСПОЛЬЗОВАНИЕМ СРЕД РАЗРАБОТКИ

В последние годы робототехника всё активнее внедряется в повседневную жизнь, образование и сферу услуг. Интерес к созданию автономных систем и управляемых роботов стремительно растет как среди начинающих разработчиков, так и профессионалов. Но обилие готовых решений и конструкторов приводит многих начинающих разработчиков к трудности выбора подходящей программной среды под конкретные задачи, что зачастую отпугивает начинающих специалистов.

Создание базового программного модуля для управления робототехнической платформой является актуальной задачей, так как спрос на образовательные и бытовые роботизированные комплексы ежедневно растет. Разработка такого модуля позволит начинающим разработчикам не только понять принципы управления движением робота, но и выбрать понятное средство для разработки.

Для определения требований к проектируемому модулю рассмотрим несколько популярных платформ, используемых для управления робототехническими устройствами, а именно ROS (RobotOperating System) и Arduino IDE.

Arduino IDE – это широко известная среда разработки, базирующаяся на языке программирования C++ и предназначенная для программирования всех плат семейства Arduino.

Платформа Arduino построена на принципах открытости: её аппаратные и программные решения доступны для свободного использования, изучения и модификации. Это даёт возможность каждому создавать собственные проекты на ее основе, вносить изменения в существующие разработки и обмениваться опытом с сообществом. Такой подход ускоряет процесс создания прототипов с использованием готовых решений и открывает простор для творческих экспериментов [3, с. 134].

Принцип работы Arduino IDE довольно прост. Когда пользователь пишет код и компилирует его, среда IDE генерирует исполняющий файл, который отправляется на плату с помощью USB- кабеля [1, с. 63].

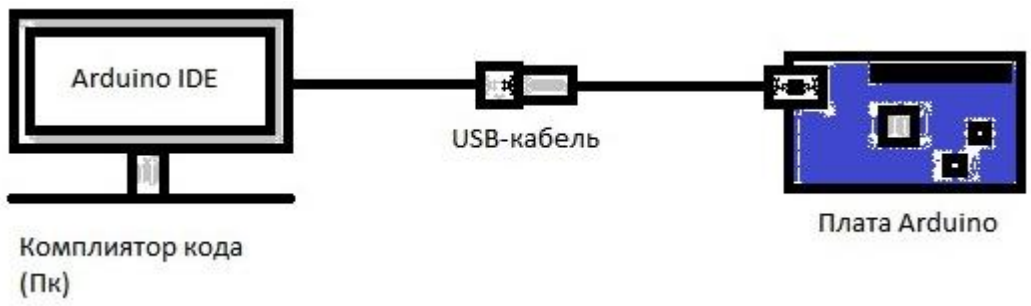


Рис. 1. Принцип работы Arduino IDE (Составлено автором)

Arduino IDE находит своё основное применение в сферах робототехники и создания автоматизированных систем. Кроме того, эта среда используется для программирования широкого спектра интерактивных, учебных, экспериментальных и развлекательных устройств и моделей.

Среди наиболее востребованных решений также выделяется ROS (Robot Operating System) – платформа, представляющая собой комплект программного обеспечения, вспомогательных инструментов и специализированных компонентов.

Приложения, разработанные в среде ROS, строятся из множества параллельно работающих процессов. Каждый такой процесс отвечает за решение конкретной, узконаправленной задачи. В терминологии они называются узлами (nodes). Основным механизмом их взаимодействия служат топики (topics) – виртуальные каналы, через которые узлы обмениваются информационными пакетами, или сообщениями (messages). Сообщения могут иметь произвольный тип и структуру. Узлы способны либо направлять данные в топик (выполнять публикацию), либо принимать их оттуда (осуществлять подписку), что создает основу для асинхронного обмена информацией между элементами системы [4, с. 48].

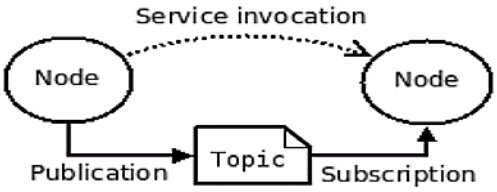


Рис. 2. Базовая схема ROS [4, с. 48]

Все эти компоненты функционируют в едином коммуникационном пространстве, которое координируется специальным сервисом, называемым **мастером** (master). Популярная утилита **Gazebo** позволяет наглядно отображать результаты работы алгоритмов в соответствии с разработанной математической моделью [5, с. 77].

Другим мощным инструментом визуализации является **rviz. Он** предназначен для трёхмерной визуализации сенсорных данных, поступающих с робота. Он отображает информацию с камер, лидаров, дальномеров и других датчиков [4, с. 48].

Сравним рассмотренные системы по нескольким критериям, таким как функциональность, удобство использования и масштабируемость:

Функциональность

Arduino:

Ориентирована на работу на низком уровне: чтение сигналов с кнопок, аналоговых датчиков, управление светодиодами, моторами и реле.

Работает «ближе к железу», но не имеет встроенных инструментов для сложных вычислений (компьютерное зрение, распознавание речи, навигация).

Не поддерживает многозадачность в привычном смысле (все инструкции выполняются последовательно в цикле).

Библиотеки простые и направлены на управление конкретными компонентами (драйвер двигателя, дисплей и т. д.).

ROS:

Предоставляет мощные встроенные библиотеки для компьютерного зрения (OpenCV), построения карт (SLAM), навигации (MoveIt для манипуляторов) и моделирования (Gazebo).

Позволяет обрабатывать данные с камер, лидаров, 3D-сенсоров.

Поддерживает распределенные вычисления: обработка может идти на нескольких компьютерах одновременно.

Использует современные стандарты связи (DDS в ROS 2) для надежной передачи данных между узлами.

Удобство использования

Arduino:

Чрезвычайно низкий порог входа.

Простая среда разработки (Arduino IDE), где код пишется на упрощенном C++.

Огромное количество готовых примеров и туториалов для начинающих. Достаточно подключить плату к USB и нажать «Загрузить».

Не требует знаний операционных систем или сетевых протоколов.

ROS:

Высокий порог входа. Требуется уверенное владение Linux (Ubuntu), знание C++ или Python, понимание архитектуры «узел-тема-сообщение».

Установка и настройка могут занять много времени из-за зависимостей и версий.

Отладка сложнее из-за распределенности системы (нужно мониторить много процессов сразу).

Требует понимания работы с командной строкой и файловыми системами Linux.

Масштабируемость

Arduino:

Крайне ограничена ресурсами микроконтроллера (память, тактовая частота).

Добавление новых функций часто упирается в физический предел платы.

При разрастании кода он превращается в трудночитаемую «спагетти-логику», если изначально не проектировался архитектурно.

Для добавления новых датчиков нужно переписывать код и разбираться в электрических схемах подключения.

ROS:

Система спроектирована для масштабирования с самого начала.

Модульная архитектура позволяет легко добавлять новые «узлы» (программы) с новым функционалом, не трогая старые.

Позволяет подключать к системе неограниченное количество датчиков (если хватает вычислительной мощности).

Легко масштабируется на несколько компьютеров или даже на группу роботов (работа в сети).

Из сравнения можно сделать вывод, что ROS является гораздо более сложной для начинающих системой с большим порогом входа, но и большим функционалом и

масштабируемостью, а Arduino IDE – более понятная новичку система с меньшими возможностями, но низким порогом входа.

Таким образом, для проектирования модуля для управления робототехнической платформой была выбрана среда Arduino, так как является более удобной для создания простого функционала.

Для проектирования модуля следует учитывать следующие требования:

Функциональные требования

1. Базовое управление приводами;
2. Сбор и первичная обработка данных с датчиков;
3. Обработка команд управления;
4. Обратная связь (телеметрия).

Нефункциональные требования

1. Модульность и расширяемость;
2. Простота конфигурации и использования;
3. Надежность и безопасность;
4. Эффективность использования ресурсов.

Проект программы состоит из нескольких файлов, реализующих модульную архитектуру (см рис. 3).

Рис. 3. Схема проекта программного модуля (составлено автором)

На начальном этапе разработки программного обеспечения для робототехнической платформы критически важно проанализировать и описать все вероятные сценарии перемещения устройства. Главная цель этого этапа – создание поведенческих алгоритмов, которые позволят роботу безопасно функционировать в динамической среде, своевременно

реагируя на препятствия и избегая аварийных ситуаций [2, с. 192]. Кроме того, перед написанием кода необходимо составить набросок того, как будет вести себя робот в той или иной ситуации, в виде схемы (см рис. 4).

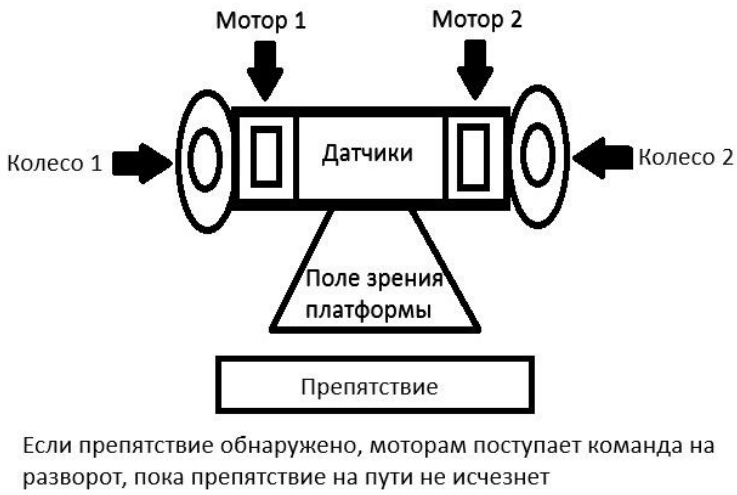


Рис. 4. Схема робототехнической платформы (составлено автором)

Основной файл программы выглядит следующим образом:

```
#include "config.h"
#include "MotorController.h"
#include "Sensors.h"
L298NMotorController motors(
   Создание объектов моторов
);
UltrasonicSensor distanceSensor(TRIG_PIN, ECHO_PIN);
LineSensor lineSensors(LINE_SENSOR_LEFT, LINE_SENSOR_RIGHT);
RobotMode currentMode = IDLE;
unsigned long lastSensorRead = 0;
const int sensorInterval = 50;
struct Telemetry {
Задаем структуру телеметрии
} telemetry;
void setup() {
Инициализация компонентов
}
void loop() {
Чтение команд, датчиков, отправка телеметрии
}
void processCommands() {
Обработка отправленных команд, принятие решение на их основе
}
```

```
void readSensors() {
Чтение датчиков
}
void runModeLogic() {
Логика режимов работы
}
void sendTelemetry() {
   Отправка телеметрии
}
void printConfig() {
   Вывод конфигурации и начального состояния
}
```

Такая система наглядно показывает базовый принцип работы робототехнической платформы, отправку телеметрии и управление в разных режимах работы.

Вывод:

В данной статье были рассмотрены основные среды для программирования модуля управления роботами и создана базовая схема их работы. Такая схема может быть полезной для начинающих в робототехнике, поскольку позволяет ознакомиться со структурой программирования и помочь с выбором средства разработки.

Литература

1. Гаврилова Т.В., Григорьева А.Н. Что такое среда программирования Arduino IDE? // Актуальные научные исследования в современном мире. 2021. № 5-2(73). С. 63-64.

2. Ефремов Д.И. Применение метода моделирования в современной робототехнике // Исторические, философские, методологические проблемы современной науки: Сборник статей 6-й Международной научной конференции молодых ученых (Курск, 19 мая 2023 года). Курск: Юго-Западный государственный университет, 2023. С. 191-195.

3. Овчинников В.В., Микаева С.А., Журавлева Ю.А. Принцип функционирования схемотехники микроконтроллера Arduino // Наукосфера. 2023. № 6-2. С. 132-136.

4. Пшеничный А.Д. Использование Robot Operating System (ROS) для изучения робототехники, автоматизации производства и эксплуатации сервисных роботов // Достижения вузовской науки 2022: сборник статей XXI Международного научно-исследовательского конкурса (Пенза, 05 июня 2022 года). Пенза: Наука и Просвещение (ИП Гуляев Г.Ю.), 2022. С. 47-50. EDN XAHASG.

5. Ходаковский Я.С., Бажутин Д.В. Разработка алгоритма управления мобильным роботом на Mecanum-колесах в среде ROS // Инновационные перспективы Донбасса: материалы 8-й Международной научно-практической конференции (Донецк, 24–26 мая 2022 года). Донецк: Донецкий национальный технический университет, 2022. Т. 2. С. 77-81.