

Специальность 09.02.07 Информационные системы и программирование

Курсовой проект

по МДК 05.01 Проектирование и дизайн информационных систем,   
МДК.05.02 Разработка кода информационных систем

Тема: Проектирование, дизайн и программная реализация модуля управления проектами для строительной компании.

Студент: Баграш Ф.М.

(Фамилия, И.О.)

Курс 3 группа 33ИСП

Защищен с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Преподаватели: Бурнин Д.А.

(Фамилия, И.О.)

Игнатьева Т.А.

(Фамилия, И.О.)

Москва 2025 г.

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 4](#_Toc191020867)

[1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ 5](#_Toc191020868)

[2. ПРАКТИЧЕСКАЯ ЧАСТЬ 6](#_Toc191020869)

[3. ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ 7](#_Toc191020870)

[ЗАКЛЮЧЕНИЕ 8](#_Toc191020871)

[СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ 9](#_Toc191020872)

# Введение

Актуальность исследования обусловлена возрастающей потребностью строительных компаний в эффективных инструментах управления проектами, которые позволяют оптимизировать процессы планирования, контроля и координации работ. В условиях высокой конкуренции и сложности строительных проектов внедрение специализированного модуля управления проектами становится необходимым для повышения прозрачности процессов, снижения рисков и улучшения качества выполнения задач. Несмотря на наличие готовых решений на рынке, их адаптация под специфику строительной отрасли часто оказывается недостаточной, что требует разработки индивидуального подхода. Данное исследование направлено на устранение этого пробела путем проектирования, дизайна и программной реализации модуля, учитывающего особенности строительной компании, что делает его актуальным и востребованным.

Цель исследования – разработать и реализовать модуль управления проектами, адаптированный под специфику строительной компании, который позволит оптимизировать процессы планирования, контроля и координации строительных проектов, повысить их прозрачность и эффективность управления ресурсами.

Задачи исследования:

1. Проанализировать существующие решения и подходы к управлению проектами в строительной отрасли.
2. Исследовать особенности и потребности строительных компаний в области управления проектами.
3. Разработать требования к функционалу и интерфейсу модуля управления проектами.
4. Спроектировать архитектуру модуля и разработать его программную реализацию.

Объектом исследования является процесс управления проектами в строительной компании, включающий планирование, контроль выполнения задач, распределение ресурсов и координацию взаимодействия между участниками проектов.

Предметом исследования является разработка модуля управления проектами для строительной компании, включающего проектирование, дизайн интерфейса и программную реализацию, направленного на оптимизацию процессов планирования, контроля и координации строительных проектов.

# Глава 1. Теоретические аспекты проектирования модуля информационной системы

Разработка веб-приложения требует тщательного планирования и соблюдения ключевых принципов, чтобы проект получился функциональным, безопасным и удобным в поддержке. Рассмотрим основные этапы и подходы к созданию веб-приложения.

## 1.1 Основные принципы разработки веб-приложений

1. Пользовательский опыт (UX) и интерфейс (UI)

Разработка качественного пользовательского опыта начинается с глубокого понимания целевой аудитории. Проводится серия исследований, включающая анализ демографических данных, поведенческих паттернов и конкретных потребностей потенциальных пользователей. На основе собранной информации создаются детализированные user personas - профили типичных пользователей, которые помогают принимать дизайн-решения.

Следующим этапом становится проектирование взаимодействия, где особое внимание уделяется созданию интуитивно понятной информационной архитектуры. Дизайнеры разрабатывают wireframes - схематичные макеты будущего интерфейса, которые постепенно превращаются в интерактивные прототипы. Эти прототипы проходят серию тестирований на юзабилити, позволяя выявить и исправить потенциальные проблемы взаимодействия до начала разработки.

Визуальная составляющая разрабатывается с учетом современных тенденций и принципов дизайна. Создается комплексная дизайн-система, включающая единые стили для всех элементов интерфейса, гармоничные цветовые схемы и продуманную типографику. Особое внимание уделяется адаптивности - все визуальные решения должны одинаково хорошо работать на устройствах с разными размерами экранов.

2. Клиент-серверная архитектура

Современные веб-приложения строятся по принципу разделения клиентской и серверной частей. Frontend-разработка сегодня практически всегда ведется с использованием прогрессивных фреймворков, таких как React, Angular или Vue.js. Эти технологии позволяют создавать сложные интерактивные интерфейсы, основанные на компонентном подходе. Управление состоянием приложения осуществляется через специализированные инструменты вроде Redux или Vuex, что особенно важно для крупных проектов с множеством взаимосвязанных элементов.

Backend-разработка сосредоточена на создании надежной и производительной серверной логики. Современные подходы предполагают проектирование RESTful API (с использованием таких фреймворков как Express.js для Node.js, Django REST Framework для Python или Spring Boot для Java) или использование GraphQL (с Apollo Server или Graphene) для более гибкого взаимодействия между клиентом и сервером. Серверная часть, реализуемая через популярные фреймворки типа Laravel (PHP), Ruby on Rails или ASP.NET Core, отвечает за обработку бизнес-логики, работу с базами данных, аутентификацию и авторизацию пользователей (часто с использованием специализированных решений вроде Passport.js или Spring Security), а также интеграцию со сторонними сервисами. Для высоконагруженных систем часто выбирают производительные фреймворки типа Gin для Go или Actix-web для Rust, обеспечивающие максимальную скорость обработки запросов.

3. Безопасность веб-приложений

Обеспечение безопасности является критически важным аспектом разработки. Современные угрозы включают не только традиционные SQL-инъекции и XSS-атаки, но и более изощренные методы взлома. При реализации системы аутентификации рекомендуется использовать современные протоколы OAuth 2.0 и OpenID Connect, которые обеспечивают безопасный механизм авторизации. Для защиты передаваемых данных обязательно применение HTTPS с использованием актуальных версий TLS. Особое внимание следует уделять валидации и санации всех входящих данных, как на клиентской, так и на серверной стороне. Регулярное проведение аудитов безопасности и тестирование на проникновение помогают выявлять уязвимости на ранних этапах.

4. Производительность и оптимизация

Эффективная работа приложения напрямую влияет на пользовательский опыт. Оптимизация начинается с анализа критического пути рендеринга и минимизации времени первой загрузки. Современные подходы включают использование lazy loading для ресурсов, стратегическое кэширование через Service Workers и оптимизацию изображений с помощью современных форматов типа WebP. На серверной стороне важно правильно настроить кэширование ответов API и реализовать эффективную работу с базой данных, включая создание оптимальных индексов и денормализацию там, где это необходимо. Мониторинг производительности в production-среде помогает оперативно выявлять и устранять узкие места.

5. Масштабируемость и поддержка

Архитектура приложения должна позволять легко наращивать функциональность и обрабатывать увеличивающуюся нагрузку. Микросервисный подход помогает разделить систему на независимые компоненты, которые можно масштабировать отдельно. Контейнеризация с использованием Docker упрощает развертывание и обеспечивает согласованность сред разработки, тестирования и production. Реализация CI/CD-процессов позволяет автоматизировать тестирование и deployment, сокращая время вывода новых версий. Важным аспектом является настройка комплексного мониторинга, включающего не только отслеживание ошибок, но и метрики производительности, что позволяет решать потенциальные проблемы.

## 1.2. Анализ современной строительной отрасли и её информационных систем

Современная строительная отрасль переживает период цифровой трансформации, что существенно меняет подходы к управлению проектами. Анализ эволюции систем управления строительными проектами показывает последовательный переход от простых планировщиков задач к комплексным платформам, интегрирующим все аспекты строительного процесса.

Мировой опыт демонстрирует несколько ключевых направлений развития подобных систем. В европейской практике преобладает подход, ориентированный на BIM-технологии (Building Information Modeling), где основное внимание уделяется информационному моделированию зданий. Американские разработки, такие как Procore, делают акцент на collaboration-функциях и облачных технологиях. Азиатский рынок предлагает решения с упором на мобильность и искусственный интеллект.

Анализ научных публикаций последних лет выявляет несколько проблемных зон в существующих системах:

1. Проблема интеграции - большинство систем слабо взаимодействуют с CAD-программами и бухгалтерскими системами, создавая "информационные острова".
2. Вопрос адаптивности - жесткие архитектурные решения не позволяют гибко подстраиваться под специфику конкретных строительных проектов.
3. Сложность внедрения - по данным исследований, до 40% внедрений строительных ERP-систем завершаются неудачно из-за сопротивления персонала.

Теоретические исследования в области управления строительными проектами выделяют несколько ключевых концепций:

Концепция "цифрового двойника" (Digital Twin) предполагает создание виртуальной копии строительного объекта, синхронизированной в реальном времени. Эта технология, активно развиваемая в Европе, позволяет прогнозировать сроки и выявлять потенциальные конфликты на ранних стадиях.

Теория "бережливого строительства" (Lean Construction) предлагает принципы минимизации потерь через оптимизацию потоков информации и ресурсов. Реализация этих принципов в программных системах остается сложной задачей.

Особого внимания заслуживает анализ нормативных требований к системам управления в строительстве. Российские нормативные документы (СНиПы, СП, ГОСТы) предъявляют строгие требования к документообороту и отчетности, что должно учитываться при проектировании любых информационных систем для этой отрасли.

Сравнительный анализ методологий разработки показывает, что для строительных систем наиболее эффективен гибридный подход. Каскадная модель (Waterfall) подходит для нормативно-регламентированных компонентов, в то время как Agile-методы лучше работают для пользовательских интерфейсов и мобильных приложений.

Перспективным направлением исследований является применение искусственного интеллекта в управлении строительными проектами. Современные алгоритмы машинного обучения уже сейчас способны:

* прогнозировать сроки выполнения работ
* выявлять потенциальные риски
* оптимизировать распределение ресурсов
* автоматизировать контроль качества

Однако практическое внедрение этих технологий сталкивается с проблемами качества входных данных и недоверием со стороны традиционных строительных компаний.

Анализ российского рынка строительных информационных систем выявляет значительный разрыв между международными и отечественными решениями. Если зарубежные продукты предлагают современные технологии, но слабо адаптированы к российским нормам, то отечественные разработки часто отстают в технологическом плане, хотя лучше учитывают местные требования.

Теоретические исследования подчеркивают необходимость создания систем нового поколения, которые бы сочетали:

* современные технологии (BIM, AI, IoT)
* гибкую архитектуру
* глубокую адаптацию к местным нормам
* простоту внедрения и использования

## 1.3. Архитектурные решения и технологический выбор

**Функциональные требования к системе**

Разрабатываемая система управления строительными проектами требует реализации сложного функционала по работе с различными типами ресурсов. Ядро системы составляет механизм создания и редактирования проектов, где каждый проект может включать произвольную комбинацию строительных материалов, единиц техники и рабочих бригад. Особое внимание уделяется гибкости системы - пользователи должны иметь возможность настраивать структуру проектов под конкретные задачи, добавляя элементы или редактируя их.

Система ресурсов представляет собой базу данных, где каждый тип ресурса обладает собственным описанием характеристик. Для материалов это могут быть технические спецификации и данные о поставщиках, для техники - время эксплуатации и технического обслуживания, для бригад - информация о квалификации. Все эти данные должны быть взаимосвязаны и доступны для анализа в рамках конкретного проекта.

Пользовательский интерфейс требует реализации многоуровневой системы доступа с двумя основными ролями: администраторы и управляющие проектами. Каждая роль получает специализированный набор инструментов и уровень доступа к информации. Персонализация рабочих пространств позволяет адаптировать систему под конкретные рабочие процессы.

**Нефункциональные требования**

Производительность системы является критически важным аспектом, особенно при работе с большими объемами данных. Интерфейс должен оставаться отзывчивым даже при обработке сложных запросов и работе с обширными списками ресурсов. Надежность системы обеспечивается комплексом мер по защите данных.

Требования безопасности включают не только стандартные меры защиты данных, но и реализацию сложной системы разграничения прав доступа. Особое внимание уделяется защите от случайных или преднамеренных изменений критически важной информации.

**Концепция единого приложения**

Проект реализован с использованием прогрессивного подхода, объединяющего серверную и клиентскую части в единую архитектуру. В отличие от традиционного разделения на независимые модули, система построена на принципах тесной интеграции Laravel и Vue.js через технологию Inertia.js. Такой подход позволяет сохранить преимущества монолитной архитектуры, обеспечивая при этом современный пользовательский опыт, характерный для SPA-приложений.

**Обоснование технологического стека**

Выбор Laravel в качестве базового фреймворка обусловлен его комплексным набором инструментов для быстрой разработки бизнес-логики. Встроенные механизмы аутентификации, работы с базой данных через Eloquent ORM и система шаблонов Blade создают прочный фундамент для приложения. Inertia.js выступает в качестве связующего звена, позволяя использовать мощь Vue.js для построения интерактивных интерфейсов без необходимости создания отдельного API.

Фронтенд-часть реализована на Vue 3 с использованием Composition API, что обеспечивает гибкость в управлении состоянием приложения. Tailwind CSS выбран в качестве CSS-фреймворка благодаря его утилитарному подходу, который идеально сочетается с компонентной архитектурой Vue и ускоряет процесс создания адаптивных интерфейсов.

**Ключевые архитектурные особенности**

Система построена по принципу единого приложения, где все запросы обрабатываются Laravel-роутами, а Inertia.js преобразует эти ответы в динамические Vue-компоненты. Такой подход устраняет необходимость создания отдельного REST API, сокращая время разработки и уменьшая сложность системы. Состояние приложения сохраняется в Laravel-сессии, обеспечивая безопасность и предсказуемость работы.

Модель данных реализована с использованием реляционной базы MySQL, с особым вниманием к полиморфным связям для работы с различными типами ресурсов. Система предусматривает мягкое удаление элементов, что позволяет восстанавливать случайно удаленные данные. Для сложных выборок данных активно используются Eloquent-скопы.

Система безопасности включает многоуровневую защиту: стандартную Laravel-аутентификацию, ролевую модель доступа через Gates и Policies, а также middleware для защиты маршрутов. Это обеспечивает гибкое управление правами пользователей при сохранении простоты реализации.

**Процесс разработки и инструменты**

Разработка ведется в локальной среде. Сборка фронтенд-части осуществляется через Vite, обеспечивающий мгновенную горячую перезагрузку компонентов во время разработки. Для контроля версий используется Git с адаптированным workflow, учитывающим особенности монолитной архитектуры с Inertia.js.

Тестирование реализовано на нескольких уровнях: модульные тесты для бизнес-логики на PHPUnit, компонентные тесты для Vue-компонентов, а также end-to-end тесты критических сценариев использования. Непрерывная интеграция настроена через GitHub Actions, автоматизируя процессы проверки кода и развертывания.

**Перспективы развития архитектуры**

Выбранная архитектура предусматривает несколько направлений для будущего расширения. В первую очередь это возможность постепенного перехода к гибридной модели с выделением отдельных API-эндпоинтов для сложных функций. Система также готова к интеграции с внешними сервисами через механизмы очередей Laravel и может быть дополнена real-time функциональностью через Laravel Echo и WebSockets.

Адаптивная структура проекта позволяет рассматривать различные варианты масштабирования - от вертикального увеличения мощности сервера до возможного разделения на микросервисы в будущем. Особое внимание в архитектуре уделено возможности постепенного внедрения новых технологий, таких как Serverless-функции для отдельных задач или GraphQL для сложных запросов данных.

# 2. ПРАКТИЧЕСКАЯ ЧАСТЬ

# 3. ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ

# ЗАКЛЮЧЕНИЕ

# СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ