

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
вычислительной техники

Кафедра
МОиПЭВМ

Направление подготовки 09.03.04 «Программная инженерия»

Направленность Программная инженерия

БАКАЛАВРСКАЯ РАБОТА

на тему

**Программные средства поиска оптимальных
решений строительного проекта**

Студент

(подпись, дата)

Данков Дмитрий Алексеевич

(ФИО полностью)

Руководитель

(подпись, дата)

Гурьянов Л.В.

(фамилия, инициалы)

Нормоконтролёр

(подпись, дата)

Такташкин Д.В.

(фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от 03.06.2026 № 12)

Заведующий кафедрой

(подпись)

Козлов А.Ю.

(фамилия, инициалы)

Работа защищена с отметкой _____ (протокол заседания ГЭК от _____ № _____)

Секретарь ГЭК

(подпись)

Попова Н.А.

(фамилия, инициалы)

Пенза, 2026

Федеральное государственное бюджетное образовательное учреждение
высшего образования
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
Вычислительной техники

Кафедра
МО и ПЭВМ

«Утверждаю»
Заведующий кафедрой
А.Ю. Козлов
«26» декабря 2025 г

**ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА**

1. Студент Данков Дмитрий Алексеевич гр. 22ВП2 факультета ВТ
<Фамилия, имя, отчество полностью>

2. Тема работы Программные средства поиска оптимальных решений строительного проекта

Тема утверждена приказом ПГУ №1583/о от " 30 " декабря 2025 г.

2. Руководитель работы к.т.н., доцент кафедры «МОиПЭВМ» Гурьянов Л.В

3. Задание на работу (назначение разработки, исходные данные и т.п.)

1) Требуется разработать сервис, обеспечивающий поиск оптимальных решений строительного проекта.

2) Языковые средства разработки: C#, SQL, TypeScript, UML

3) Инструментальные средства разработки: Jet Brains Rider, Jet Brains DataGrip, Visual Paradigm 17.1

4) Технологии разработки: Микросервисная архитектура, Docker

5) Основные функции приложения:

а) регистрация и авторизация;

б) создание проекта, заполнение информации о проекте

в) предварительная оценки стоимости и длительности проекта;

г) поиск оптимального пути проложения коммуникаций на заданном шаблоне;

д) коммуникация в чате проекта;

е) возможность скачать смету проекта в формате Excel и диаграмму Ганта в формате HTML.

4. Перечень подлежащих разработке вопросов _____

- 1) Постановка задачи и анализ предметной области _____
 - 2) Проектирование _____
 - 3) Реализация _____
 - 4) Контроль качества программного обеспечения _____
 - 5) План разработки и оценка бюджета _____
 - 6) Оформление пояснительной записки _____
- _____
- _____
- _____
- _____
- _____
- _____

5. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области и требований к системе	10%	07.04.2026 – 13.04.2026	
2	Проектирование программных средств	20%	14.04.2026 – 25.04.2026	
3	Реализация программных средств	30%	27.04.2026 – 11.05.2026	
4	Тестирование программных средств	20%	12.05.2026 – 21.05.2026	
5	Оформление пояснительной записки	20%	22.05.2026 – 02.06.2026	

Дата выдачи задания «26» декабря 2025г.

Руководитель бакалаврской работы _____ 26.12.2025
(подпись, дата)

Задание к исполнению принял студент _____ 26.12.2025
(подпись, дата)

Работу к защите допустить
Декан ФВТ _____ Л.Р. Фионова
(подпись, дата)

Реферат

Пояснительная записка содержит 112 листов, 27 рисунков, 9 таблиц, 2 листинга, 21 использованный источник, 4 приложения.

СТРОИТЕЛЬНЫЙ ПРОЕКТ, C#, DOCKER, POSTGRESQL, ОПТИМИЗАЦИЯ МАРШРУТА, МИКРОСЕРВИСНАЯ АРХИТЕКТУРА, REACT, UML, БАЗА ДАННЫХ.

Объект исследования – программные решения планирования, ведения и оптимизации строительного проекта.

Цель выпускной квалификационной работы – спроектировать и разработать программные средства поиска оптимальных решений строительного проекта.

Технологии разработки – разработка серверной части приложения проводилась на языке C# с использованием фреймворка ASP.NET, клиентская часть была разработана на языке TypeScript с использованием фреймворка React. В качестве СУБД использовалась PostgreSQL. В качестве среды разработки использовалась JetBrains Rider, а для управления базами данных DataGrip.

Приложение разворачивается в контейнеризированной среде с помощью Docker.

Результат работы – в ходе выполнения выпускной квалификационной работы были спроектированы и разработаны программные средства поиска оптимальных решений строительного проекта, проведено модульное и функциональное тестирование разработанного приложения. Все требования к разработке приложения выполнены.

					ПГУ 09.03.04 – 08BP222.08 ПЗ			
Изм	Лист	№ докум.	Подп.	Дата	Программные средства поиска оптимальных решений строительного проекта Пояснительная записка	Лит.	Лист	Листов
Разраб.	Данков Д.А.						4	112
Пров.	Гурьянов Л.В.					Группа 22ВП2		
Реценз.								
Н. контр.	Такташкин Д.В							
Утверд.	Козлов А.Ю.							

Содержание

Введение.....	7
1 Анализ предметной области и постановка задачи на разработку	9
1.1 Анализ предметной области	9
1.2 Анализ программного обеспечения аналогичного назначения	12
1.2.1 Web-приложение «Fieldwire»	12
1.2.2 Web-приложение «Procure»	13
1.2.3 Web-приложение «Gectaro»	14
1.2.4 Web-приложение «Remplanner»	15
1.2.5 Сравнение аналогов	16
1.3 Анализ функциональных требований.....	17
1.4 Анализ нефункциональных требований.....	21
1.5 Планирование разработки и оценка бюджета	22
2 Проектирование программных средств поиска оптимальных решений строительного проекта.....	24
2.1 Архитектура приложения.....	24
2.2 Проектирование модели базы данных	25
2.3 Проектирование серверной части приложения	29
2.2.1 Проектирование микросервисов AuthService и UserService.....	29
2.2.2 Проектирование микросервиса ProjectCalculationService.....	31
2.2.3 Проектирование математического обеспечения системы.....	32
2.4 Проектирование клиентской части приложения	34
3 Реализация программных средств поиска оптимальных решений строительного проекта.....	40
3.1 Реализация базы данных	40
3.2 Реализация клиентского приложения	42
3.3 Реализация серверной части приложения	43
4 Контроль качества	52
4.1 Выбор метода тестирования	52
4.2 Тестирование серверной части.....	53

4.3 Тестирование клиентского приложения	54
4.4 Метрики кода.....	56
Заключение.....	58
Список использованных источников.....	59
Приложение А (справочное) Глоссарий	61
Приложение Б (рекомендуемое) Результат тестирования клиентской части.....	64
Приложение В (обязательное) Листинг клиентского приложения.....	71
Приложение Г (обязательное) Листинг серверного приложения	92

Введение

В настоящее время проекты ремонта и небольшого строительства в жилых и коммерческих помещениях в значительной степени управляются на основе личного опыта, разрозненных рекомендаций и неполной информации. Пользователь вынужден самостоятельно определять состав работ, их последовательность, перечень материалов и потребность в специалистах, а также предварительно оценивать бюджет и сроки, опираясь на субъективные предположения и ограниченные справочные данные.

На рынке представлены средства фиксации задач, ведения документации и контроля исполнения, а также профессиональные решения для строительных организаций, ориентированные на корпоративные процессы. При этом для частного пользователя по-прежнему недостаточно инструментов, которые позволяют на этапе планирования получить предварительную оценку стоимости и длительности ремонта на основе исходных параметров объекта и выбранных решений. Кроме того, отсутствуют распространённые программные средства, обеспечивающие оптимизацию проектных решений, то есть сравнение вариантов и выбор более рационального сценария до начала работ, что приводит к ситуации, когда пользователи фактически начинают проект без проверяемого плана и вынуждены корректировать решения уже в процессе выполнения.

Целью выпускной квалификационной работы является разработка программного средства для планирования ремонтного проекта, формирования структуры работ и получения предварительной оценки стоимости и длительности, а также реализации механизмов поиска оптимальных решений, включая построение оптимального маршрута проведения различных коммуникаций. Разрабатываемое приложение направлено на повышение предсказуемости ремонта, снижение перерасхода ресурсов и улучшение управляемости проекта за счёт формализации данных и расчётной поддержки принятия решений.

В рамках работы выполняются анализ предметной области, обзор существующих программных решений аналогичного назначения, формирование и обоснование функциональных требований, формирование и обоснование нефункциональных требований, проектирование архитектуры и компонентов системы, реализация программного обеспечения и мероприятия по контролю качества разработанного решения.

Для разработки приложения используется язык программирования C#, который обеспечивает создание надёжного прикладного кода и серверной логики, удобной для тестирования и дальнейшего расширения. Для хранения данных применяется PostgreSQL, предназначенная для ведения структурированных данных проектов, справочников и результатов расчётов с обеспечением целостности, согласованности [1]. Для развертывания и сопровождения приложения используется Docker, обеспечивающий упаковку программных компонентов и их зависимостей в изолированные контейнеры [2]. Это позволяет зафиксировать единые версии окружения и настроек, добиться запуска приложения на любой платформе и упростить развертывание в тестовой и эксплуатационной средах. Дополнительно контейнеризация снижает риск конфликтов зависимостей между компонентами и ускоряет настройку рабочей среды разработки.

1 Анализ предметной области и постановка задачи на разработку

1.1 Анализ предметной области

Предметная область управления строительными проектами, в том числе проектами ремонта жилых и коммерческих помещений, характеризуется высокой сложностью и большим числом взаимосвязанных решений. Пользователь сталкивается с необходимостью одновременно определять состав работ, последовательность их выполнения, требования к качеству, перечень материалов, потребность в специалистах, календарный график, бюджет, логистику закупок и условия доступа на объект. Даже в небольших проектах просчёты, допущенные на старте, обычно приводят к цепочке проблем – сдвигу сроков, лишним расходам, повторному выполнению отдельных операций и разногласиям между заказчиком и исполнителями. При этом основная часть затрат и времени формируется не в процессе, а ещё до начала работ: на этапе выбора технологии, материалов, схемы снабжения и распределения обязанностей.

С позиции пользователя главная потребность заключается в том, чтобы получить контроль над проектом ещё до фактического начала ремонта. Для этого необходимо заранее зафиксировать исходные параметры: тип объекта, размеры и планировку помещений, состояние инженерных систем, перечень планируемых изменений, требования к отделочным материалам, а также ограничения по срокам, бюджету и уровню качества. Пользователю нужен инструмент, который на основе этих данных позволит собрать проект в понятную модель – с этапами, задачами, зависимостями, контрольными точками, расчётом материалов и трудозатрат. Не менее важно, чтобы система давала возможность оценить последствия любого изменения, например замены материала или переноса сроков. Кроме того, пользователю нужна прозрачная фиксация всех договорённостей, поскольку ремонт обычно выполняется не одним человеком, а несколькими исполнителями, и итог во многом зависит от правильного соблюдения последовательности и своевременных закупок.

Рынок программных средств в данной области можно условно разделить на две основные группы. Первая группа включает профессиональные системы для строительного бизнеса, ориентированные на управление объектами в организациях, контроль исполнения, работу с документацией, учёт замечаний, отчётность и взаимодействие участников проекта. Эти решения эффективны в среде подрядных организаций, технического надзора и застройщиков, поскольку поддерживают регламентированные процессы, роли и формальные процедуры контроля. Однако они обычно не предназначены для частного пользователя и не решают задачу первоначального проектирования ремонта как последовательности решений, поскольку их основная цель состоит в управлении уже запущенным проектом, а не в подготовке оптимального плана до начала работ. Вторая группа включает инструменты личного планирования и универсальные средства управления задачами, которые применяются для ведения списков работ, календарей, закупок и заметок. Такие средства удобны для организации информации, но, как правило, не содержат предметной модели ремонта, не обеспечивают автоматизированной проверки корректности плана и не обладают встроенными механизмами расчёта длительности и стоимости на основе параметров объекта и выбранных решений.

Одной из ключевых проблем предметной области является ограниченность средств предварительной оценки стоимости и длительности ремонта, доступных широкому кругу пользователей. На практике многие пользователи начинают проект без формализованного плана и без надёжной предварительной оценки бюджета и сроков, поскольку получение точной сметы и календарного графика требует либо профессиональной экспертизы, либо значительных временных затрат на сбор данных и ручные расчёты. Существующие калькуляторы и справочные материалы нередко дают усреднённые значения и не учитывают специфику конкретного объекта, сочетание работ, ограничения по доступности специалистов, особенности последовательности технологических операций и риски задержек поставок. В результате формируется ситуация, при которой пользователь вынужден принимать решения при недостатке информации и

действовать фактически вслепую, корректируя план уже по ходу выполнения работ. Такая модель поведения закономерно повышает вероятность перерасхода средств и увеличения сроков, поскольку исправление ошибок на поздних этапах является наиболее дорогим и трудоёмким.

Отдельной проблемой является отсутствие доступных пользователю механизмов оптимизации решений строительного проекта в форме, пригодной для личного применения. В реальном ремонте пользователь постоянно выбирает между альтернативами, которые различаются по стоимости, срокам, качеству, эксплуатационным характеристикам и рискам. Примером таких альтернатив являются выбор материалов и технологий, формат привлечения специалистов, способ организации закупок, параллельность выполнения работ, а также допустимая степень компромисса между скоростью и качеством. На рынке широко представлены средства фиксации фактов, то есть задачи, отчёты, замечания и документы, но существенно реже встречаются инструменты, которые помогают сравнивать альтернативы на этапе планирования, оценивать последствия выбора в денежном и временном выражении и предлагать более рациональные варианты. При отсутствии таких инструментов оптимизация фактически заменяется субъективными решениями пользователя или рекомендациями исполнителей, которые могут быть заинтересованы в определённом наборе работ и материалов и не всегда ориентированы на оптимальное соотношение цены и результата.

Следствием описанных ограничений является разрыв между потребностью пользователя в прогнозируемости и управляемости ремонта и фактическими возможностями большинства доступных решений. Пользователь получает либо профессиональные системы, ориентированные на организации и контроль исполнения, либо простые средства ведения списков и заметок без вычислительной модели проекта. Между этими полюсами часто отсутствуют инструменты, которые обеспечивают построение проекта как конструктора, дают предварительную оценку стоимости и длительности, позволяют учитывать зависимости между работами и обеспечивают поддержку выбора решения на

основе сравнимых критериев. В этих условиях повышение качества управления ремонтом требует внедрения подходов, в которых планирование рассматривается как формализация параметров объекта и целей пользователя, последующий расчёт стоимости и сроков строится на явной модели состава работ и ресурсов, а оптимизация достигается через анализ альтернатив и оценку их влияния на бюджет, сроки и риски.

Таким образом, предметная область строительных проектов в сегменте ремонта характеризуется системной нехваткой доступных инструментов предварительного расчёта и оптимизации, из-за чего пользователи часто принимают решения без достаточной аналитической поддержки. Наличие средств контроля исполнения и документооборота на рынке не компенсирует отсутствие средств планирования, поскольку контроль становится эффективным только при наличии реалистичного и согласованного плана. Для устранения проблемы требуется программное решение, которое связывает параметры объекта, модель работ, расчёт ресурсов и сравнение альтернатив, что позволяет пользователю переходить от интуитивного управления к управлению на основе предварительной оценки, сценарного анализа и оптимизации проектных решений.

1.2 Анализ программного обеспечения аналогичного назначения

1.2.1 Web-приложение «Fieldwire»

Web-приложение «Fieldwire» представляет собой программный сервис для организации работ на строительной площадке и контроля выполнения задач на объекте. Решение ориентировано на профессиональных участников строительного процесса, включая руководителей работ и исполнителей, и не предназначено для личного использования в бытовых целях.

Интерфейс предложения представлен на рисунке 1.

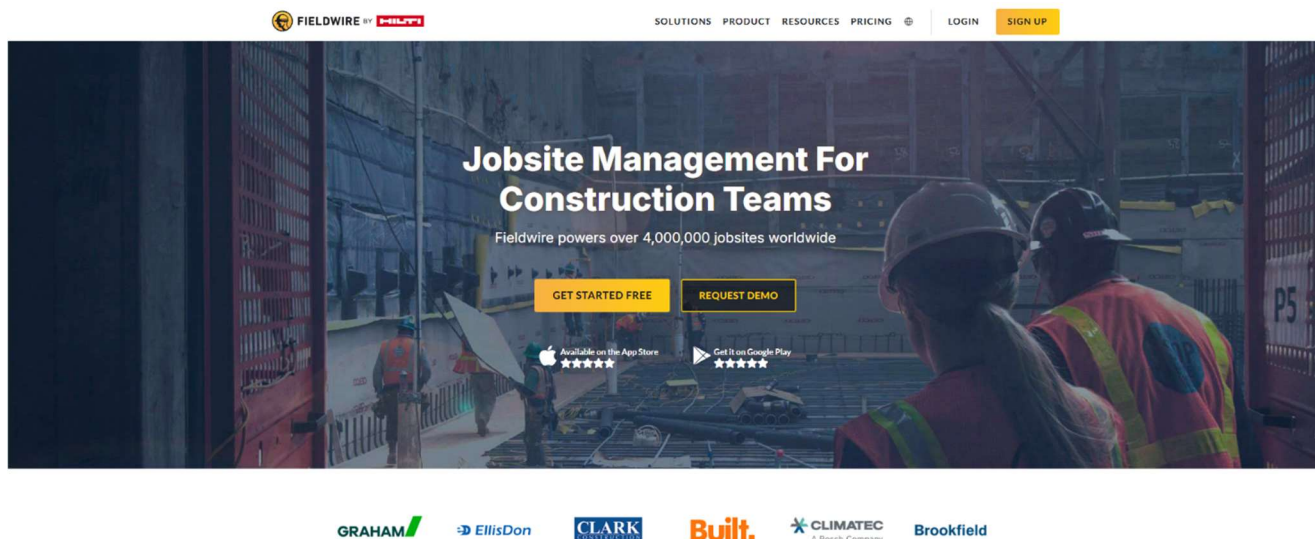


Рисунок 1 – Интерфейс приложения «Fieldwire»

Основной направленностью сервиса является постановка задач, фиксация замечаний и дефектов, ведение перечней работ, а также подготовка отчётности по ходу выполнения работ. Платформа применяется для ежедневной координации и контроля на объектах, где требуется оперативное распределение поручений и контроль сроков. При этом сервис не специализируется на оптимизации личного бюджета и календарного плана ремонта, так как его цель заключается в обеспечении дисциплины исполнения работ на площадке.

1.2.2 Web-приложение «Procore»

Web-приложение «Procore» представляет собой профессиональную платформу для управления строительными проектами и организации взаимодействия между участниками строительства. Приложение объединяет всех участников проекта в единой экосистеме и автоматизирует документооборот, управление бюджетом и ресурсами, а также коммуникацию на всех этапах.

Интерфейс предложения представлен на рисунке 2.

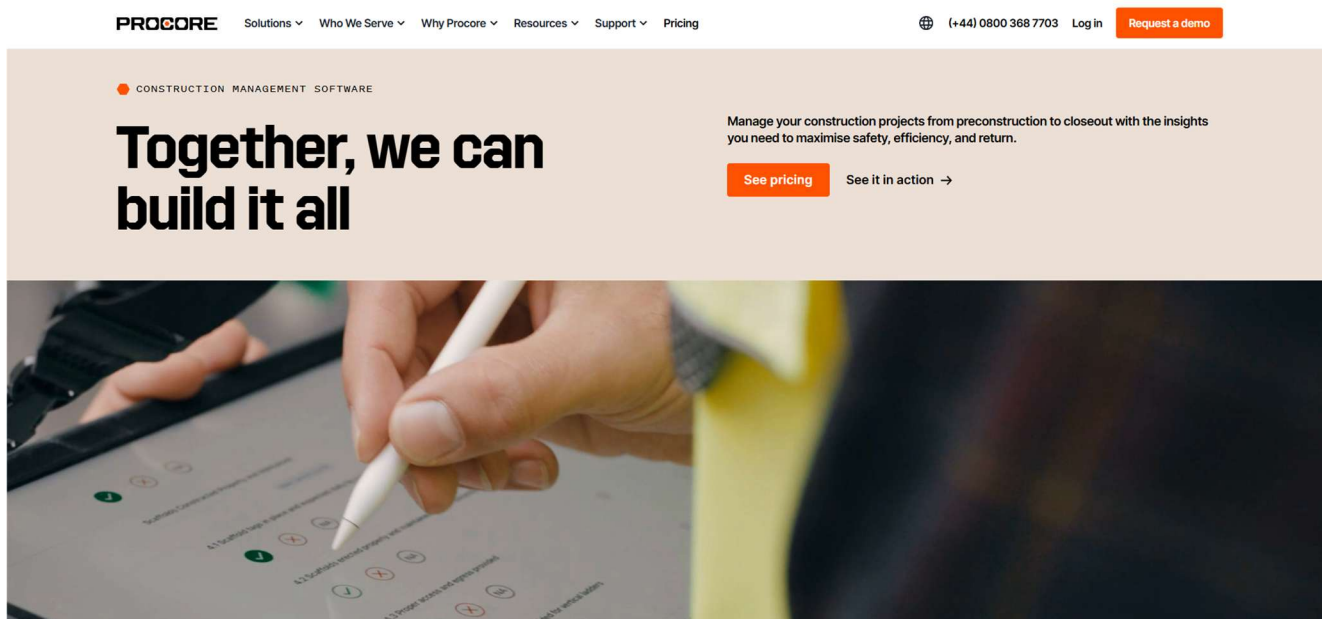


Рисунок 2 – Интерфейс приложения «Procore»

Сервис ориентирован на компании строительной отрасли, застройщиков, генеральных подрядчиков и подрядные организации, а не на личное использование. Основной направленностью платформы являются ведение проектной документации, постановка и контроль задач, фиксация замечаний, формирование отчётности и сопровождение согласований в рамках строительных объектов. Решение применяется преимущественно в среде среднего и крупного строительного бизнеса, где необходимы формализованные регламенты, роли и контроль исполнения.

1.2.3 Web-приложение «Gectaro»

Web-сайт «Gectaro» представляет собой российский программный сервис для управления строительными работами и контроля исполнения задач на объектах. Платформа ориентирована на профессиональное использование в строительных организациях, включая руководителей проектов, производителей работ и офисные подразделения.

Интерфейс предложения представлен на рисунке 3.

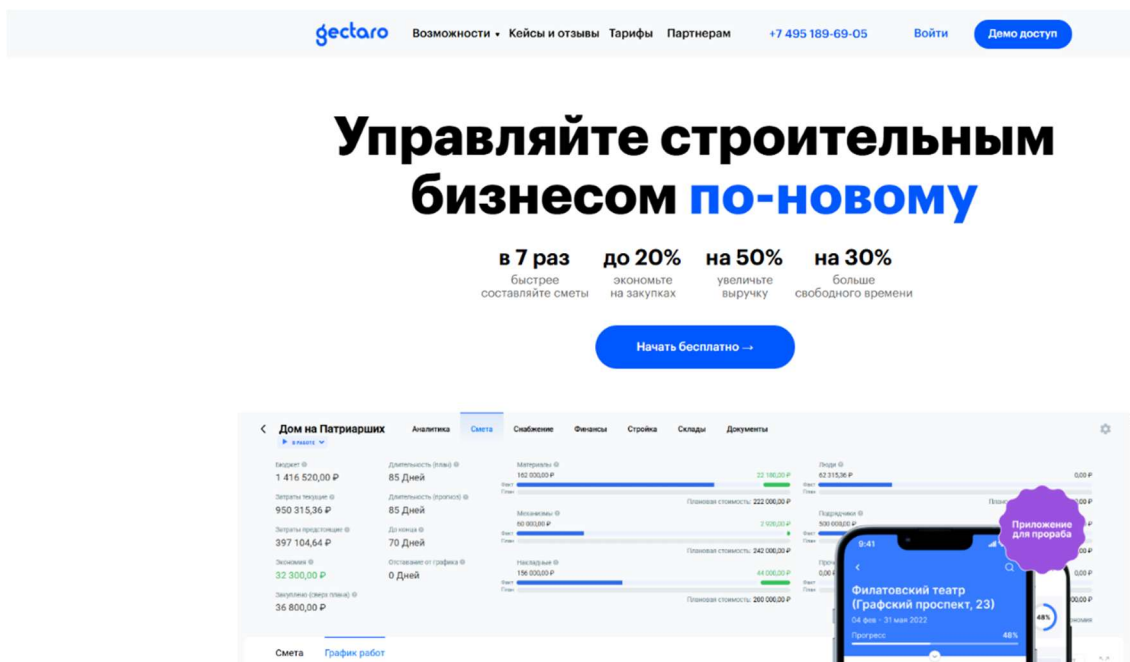


Рисунок 3 – Интерфейс приложения «Gectaro»

Основной направленностью сервиса является планирование и контроль выполнения работ, ведение задач, формирование отчетности и упорядочивание взаимодействия сотрудников по объектам. Решение применяется для повышения дисциплины исполнения и прозрачности статусов по работам, особенно при одновременном ведении нескольких объектов. Вместе с тем платформа ориентирована преимущественно на управление исполнением в компании, а не на личные сценарии планирования ремонта с оптимизацией расходов и сроков.

1.2.4 Web-приложение «Remplanner»

Web-сайт «Remplanner» представляет собой российский сервис для планирования ремонта, ориентированный на личное использование частными пользователями. Платформа предназначена для структурирования проекта ремонта, формирования перечня работ и материалов, а также фиксации последовательности этапов по помещениям

Интерфейс предложения представлен на рисунке 4.

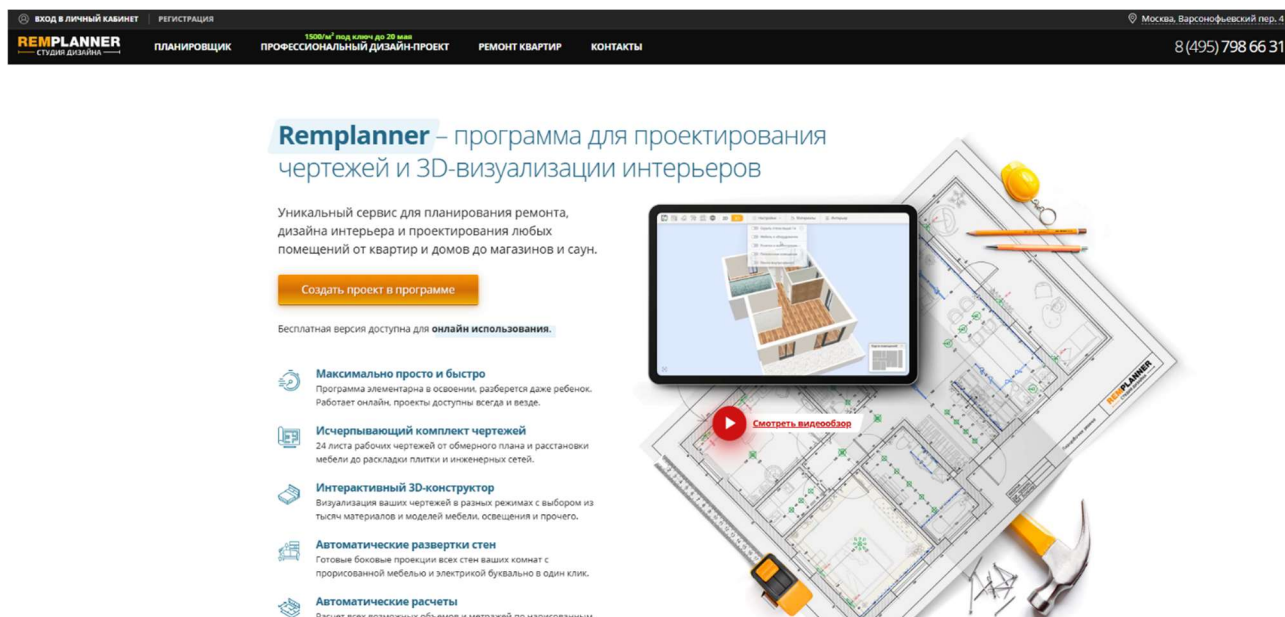


Рисунок 4 – Интерфейс приложения «Remplanner»

Основной направленностью сервиса является помощь пользователю в организации ремонта как проекта, включая подготовку планов и базовых расчётов, а также контроль выполнения по этапам. Решение применимо для самостоятельного управления ремонтом квартиры или дома, когда пользователю требуется наглядная структура работ и контроль прогресса. При этом уровень оптимизации расходов и сроков зависит от выбранного сценария использования и, как правило, носит ограниченный характер, поскольку многие решения остаются на стороне пользователя.

1.2.5 Сравнение аналогов

Для оценки существующих программных средств аналогичного назначения был проведен сравнительный анализ по наиболее значимым критериям. Оценка производилась по трехбалльной шкале, где 0 – полное отсутствие заданного функционала или возможности, 1 – частичная реализация рассматриваемой функции, 2 – полная реализация. Результат сравнения программных средств представлен на таблице 1.

Таблица 1 – Сравнение программных средств аналогичного назначения

Критерий	Procore	Fieldwire	Gectaro	Remplanner
Предварительная оценка стоимости	0	0	1	1
Предварительная оценка длительности	0	0	0	1
Оптимизация проектных решений	0	0	0	1
Возможность коммуникации внутри сервиса	2	1	1	0
Личное использование	0	0	0	2
Применимость в России	0	1	2	2
Доступность по стоимости	0	1	1	2
Итого	2	3	5	9

Из рассмотренных приложений наиболее подходящим для выполнения поставленной задачи является «Remplanner», так как он предназначен для личного использования и имеет базовый функционал оценки проектных решений, однако в нем полностью отсутствует возможность коммуникации внутри сервиса. Остальные приложения имеют итоговый бал ниже, поскольку не ориентированы на личное использования, а также не имеют механизмов оценки и оптимизации проектных решений. Подводя итог, можно сделать вывод, что ни одно из рассмотренных приложений не соответствует заявленным требованиям в полной мере.

1.3 Анализ функциональных требований

Функциональные требования (functional requirements) определяют, каким должно быть поведение продукта в тех или иных условиях. Они определяют, что разработчики должны создать, чтобы пользователи смогли выполнить свои задачи (пользовательские требования) в рамках бизнес-требований [3].

После анализа предметной области, а также существующего программного обеспечения аналогичного назначения были выявлены функциональные требования к разрабатываемой системе:

- регистрация в системе;
- вход в аккаунт;
- просмотр профиля;
- отправка обратной связи о работе приложения;
- просмотр обратной связи пользователей;
- просмотр списка добавленных проектов;
- создание проекта;
- переименование проекта;
- просмотр предварительной оценки стоимости и длительности ремонтных работ;
- скачивание сметы проекта;
- скачивание диаграммы Ганта проекта;
- заполнение паспорта проекта;
- просмотр паспорта проекта;
- заполнение конструктора проекта;
- просмотр конструктора проекта;
- создание комментария к проекту;
- просмотр комментария к проекту;
- создание кода доступа к проекту;
- выполнение предварительного расчета по оптимизации коммуникаций;
- создание оптимизации коммуникаций;
- просмотр оптимизаций коммуникаций;
- удаление оптимизации коммуникаций.

На основе выявленных функциональных требований была разработана диаграмма вариантов использования UML для каждой категории пользователей в системе, она изображена на рисунке 5.



Рисунок 5 – Диаграмма вариантов использования

На разработанной диаграмме представлены варианты использования для трех ролей в системе: администратора, создателя и наблюдателя. Функционал администратора ограничивается функциями контроля – он может только просматривать обратную связь, оставленную пользователями приложения.

Наблюдатель не может создавать проекты и менять их параметры, только просматривать различные параметры проекта, а также общаться в чате по проекту. Создатель в свою очередь имеет весь функционал наблюдателя, а также возможность создавать и вносить изменения в проект.

Диаграмма вариантов использования позволяет четко определить границы системы и понять потребности пользователей, фокусируясь на том, что система должна делать, а не на деталях ее внутренней реализации или архитектуры [4].

В таблице 2 приведена спецификация варианта использования «Войти в аккаунт».

Таблица 2 – Сценарий выполнения варианта использования «Войти в аккаунт»

Наименование	Войти в аккаунт
ID	1
Краткое описание	Пользователь аутентифицируется в системе
Действующие лица	Пользователь (любой из трех ролей)
Предусловие	Пользователь не был аутентифицирован
Основной поток	1. Пользователь переходит на страницу входа в приложении; 2. Пользователь вводит данные для входа в свой аккаунт (логин и пароль); 3. Пользователь нажимает на кнопку «Войти»; 4. Пользователь получает доступ к функционалу системы в соответствии со своей ролью.
Постусловие	Пользователь вошел в свой аккаунт

В таблице 3 приведена спецификация варианта использования «Оставить комментарий к проекту».

Таблица 3 – Сценарий выполнения варианта использования «Оставить комментарий к проекту»

Наименование	Оставить комментарий к проекту
ID	2
Краткое описание	Пользователь оставляет комментарий в чате проекта
Действующие лица	Пользователь (наблюдатель или создатель)
Предусловие	Пользователь авторизован
Основной поток	1. Пользователь переходит на страницу проекта; 2. Пользователь вводит сообщение в поле для ввода; 3. Пользователь нажимает на кнопку «Отправить»; 4. Пользователь видит отправленное им сообщение.
Постусловие	Сообщение отправлено, другие участники чата могут его просмотреть

В таблице 3 приведена спецификация варианта использования «Скачать смету проекта».

Таблица 4 – Сценарий выполнения варианта использования «Скачать смету проекта»

Наименование	Скачать смету проекта
ID	3
Краткое описание	Пользователь скачивает файл Excel, содержащий смету проекта
Действующие лица	Пользователь (наблюдатель или создатель)
Предусловие	Пользователь заполнил паспорт и конструктор проекта
Основной поток	1. Пользователь переходит на страницу проекта; 2. Пользователь находит раздел с краткой информацией о проекте; 3. Пользователь нажимает на кнопку «Скачать смету»; 4. Пользователь скачивает Excel-файл, содержащий смету проекта.
Постусловие	Смета скачана, доступна к просмотру.

Приведенная диаграмма вариантов использования и сценарии некоторых вариантов использования точно отражают функциональные требования, предъявленные к разрабатываемому программному обеспечению.

1.4 Анализ нефункциональных требований

Нефункциональные требования представляют собой требования к качественным и эксплуатационным характеристикам программного продукта, они определяют, насколько надёжно, быстро, безопасно и удобно приложение будет работать в реальных условиях [5]. По своему содержанию это архитектурные требования к разрабатываемому программному обеспечению, поскольку именно архитектура задаёт принципы масштабирования, устойчивости, защиты данных, сопровождаемости и интеграции.

Разрабатываемая система должна удовлетворять следующим нефункциональным требованиям:

- серверная часть и вычислительные модули должны обеспечивать быстрое выполнение расчётов стоимости, длительности и вариантов проектных решений при росте объёма данных по работам, материалам и сценариям, чтобы пользователь получал результат без заметных задержек (производительность);

- система должна сохранять работоспособность при росте числе пользователей (масштабируемость);
- программные средства должны использовать средства аутентификации для проверки подлинности личности пользователя (аутентифицируемость);
- система должна контролировать доступ к определенному функционалу на основе системы ролей пользователей (авторизируемость).

Сформулированные нефункциональные требования являются основой для проектирования архитектуры приложения и обеспечивают стабильную работу, безопасность и возможность развития функциональности. Их соблюдение делает сервис пригодным для масштабирования и практического применения.

1.5 Планирование разработки и оценка бюджета

Планирование проекта представляет собой процесс определения состава работ, их последовательности, сроков выполнения и необходимых ресурсов для достижения цели проекта. Это позволяет заранее оценить длительность работ, распределить нагрузку по этапам и контролировать выполнение проекта в соответствии с установленными сроками.

Для управления проектом использовалась программа Microsoft Project, программой для управления проектами [6], был составлен план разработки проекта, он представлен на рисунке 6.

		Режим задачи ▾	Название задачи ▾	Длительнс ▾	Начало ▾	Окончани ▾	Предшественники ▾
1			Подготовка ВКР	50 дней	Пн 06.04.26	Пт 12.06.26	
2			Анализ предметной области	6 дней	Пн 06.04.26	Пн 13.04.26	
3			Анализ требований	3 дней	Вт 14.04.26	Чт 16.04.26	2
4			Проектирование	7 дней	Пт 17.04.26	Пн 27.04.26	3
5			Разработка	20 дней	Вт 28.04.26	Пн 25.05.26	4
6			Тестирование	4 дней	Вт 26.05.26	Пт 29.05.26	5
7			Оформление ПЗ	10 дней	Пн 01.06.26	Пт 12.06.26	6

Рисунок 6 – План разработки проекта

Также для проекта была составлена диаграмма Ганта, она представлена на рисунке 7.

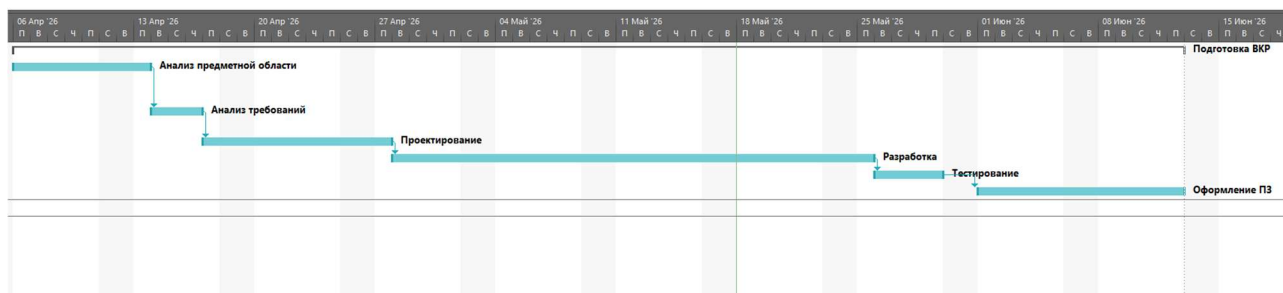


Рисунок 7 – Диаграмма Ганта

На основе составленного плана разработки проекта и зарплаты программиста-разработчика, взятую как средняя почасовая оплата программиста-разработчика в Пензенской области – 500 рублей, рассчитаем затраты на каждый из этапов разработки. Результаты расчетов представлены в таблице 5.

Таблица 5 – Затраты на реализацию проекта

Этап	Трудозатраты	Ставка	Стоимость
Анализ предметной области	48ч	500р/ч	24 000р
Анализ требований	24ч	500р/ч	12 000р
Проектирование	56ч	500р/ч	28 000р
Разработка	160ч	500р/ч	80 000р
Тестирование	32ч	500р/ч	16 000р
Оформление ПЗ	80ч	500р/ч	40 000р
Итого	400ч	500р/ч	200 000р

В результате анализа было выявлено, что разработка программных средств для оптимизации решений строительного проекта займет 400 часов и будет стоить 200 тысяч рублей.

2 Проектирование программных средств поиска оптимальных решений строительного проекта

2.1 Архитектура приложения

В соответствии с функциональными и нефункциональными требованиями, определенными на этапе анализа, для реализации программных средств была выбрана микросервисная архитектура. Микросервисная архитектура – это подход к построению программной системы, при котором она разделяется на несколько самостоятельных сервисов, каждый из которых отвечает за ограниченную область задач, развивается и разворачивается независимо, а обмен данными между сервисами выполняется по сети через четко определенные интерфейсы [7]. Крупноблочная схема архитектуры разрабатываемого приложения изображена на рисунке 8.

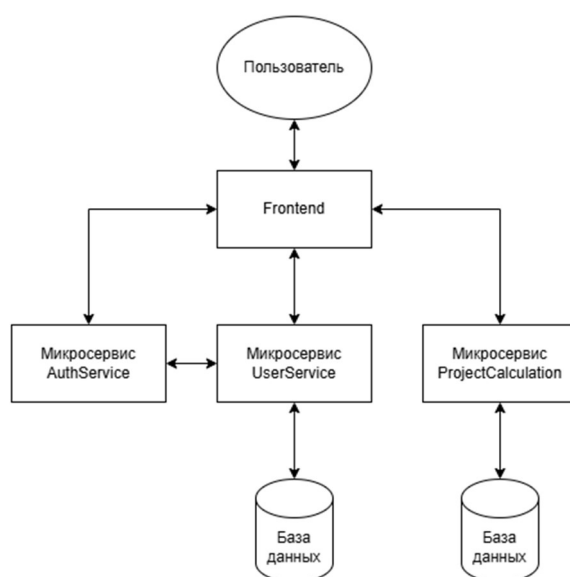


Рисунок 8 – Крупноблочная схема архитектуры

Для данного проекта микросервисная архитектура выбрана из-за удобства сопровождения и развития, так как функциональные части системы изолированы друг от друга, поэтому изменения в одной области в меньшей степени затрагивают остальные. Это также упрощает повышение надежности и производительности, поскольку отдельные сервисы можно масштабировать и обновлять без остановки всей системы [8].

В реализованной структуре пользователь взаимодействует с клиентской частью, которая принимает запросы и передает их в соответствующие сервисы. Система состоит из трех микросервисов: AuthService, UserService, ProjectCalculationService. Сервис аутентификации отвечает за идентификацию пользователей приложения, сервис работы с пользователями хранит и обрабатывает данные пользователей, а сервис расчета проекта выполняет вычисления и формирует результат по входным параметрам.

В соответствии с принципами проектирования микросервисной архитектуры данные хранятся в отдельных базах: сервис пользователей работает со своей базой, а сервис расчетов – со своей. Микросервис авторизации AuthService не имеет собственной базы данных, так как при идентификации пользователей использует данные, получаемые из UserService.

2.2 Проектирование модели базы данных

На основе анализа функциональных требований была составлена концептуальная модель базы данных в виде диаграммы «сущность-связь», она представлена на рисунке 9.

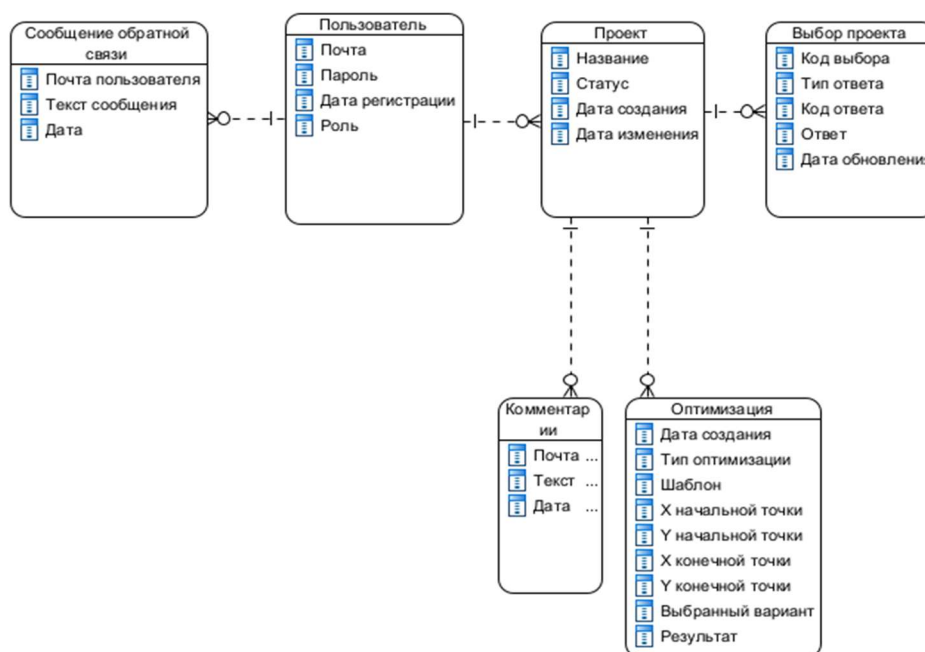


Рисунок 9 – Концептуальная модель базы данных

На основе концептуальной модели базы данных было составлено описание сущностей, приведенное в таблице 6.

Таблица 6 – Описание сущностей концептуальной модели

Название сущности	Идентификатор	Назначение
Пользователь	Users	Содержит информацию о пользователях приложения
Сообщение обратной связи	FeedbackMessages	Содержит информацию о сообщениях обратной связи
Проект	projects	Содержит основные сведения о проекте
Выбор проекта	project_step_answers	Содержит выбранные пользователем параметры проекта
Комментарий проекта	viewer_comments	Содержит комментарии пользователей к проекту
Оптимизация	project_optimization	Содержит информацию об оптимизациях, созданных пользователем

Построенная концептуальная модель описывает предметную область на уровне сущностей, их атрибутов и связей [9].

На основе разработанной концептуальной модели была построена логическая модель базы данных, более подробно описывающая все сущности, атрибуты, ключи и взаимосвязи. Логическая модель базы данных пользователей представлена на рисунке 10.

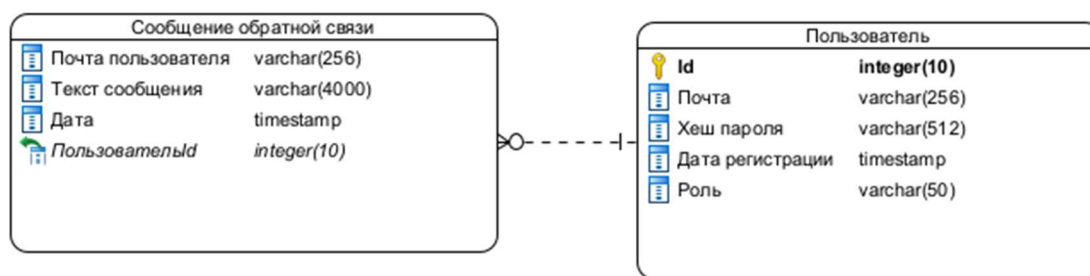


Рисунок 10 – Логическая модель базы данных пользователей

Логическая модель базы данных проекта представлена на рисунке 11.

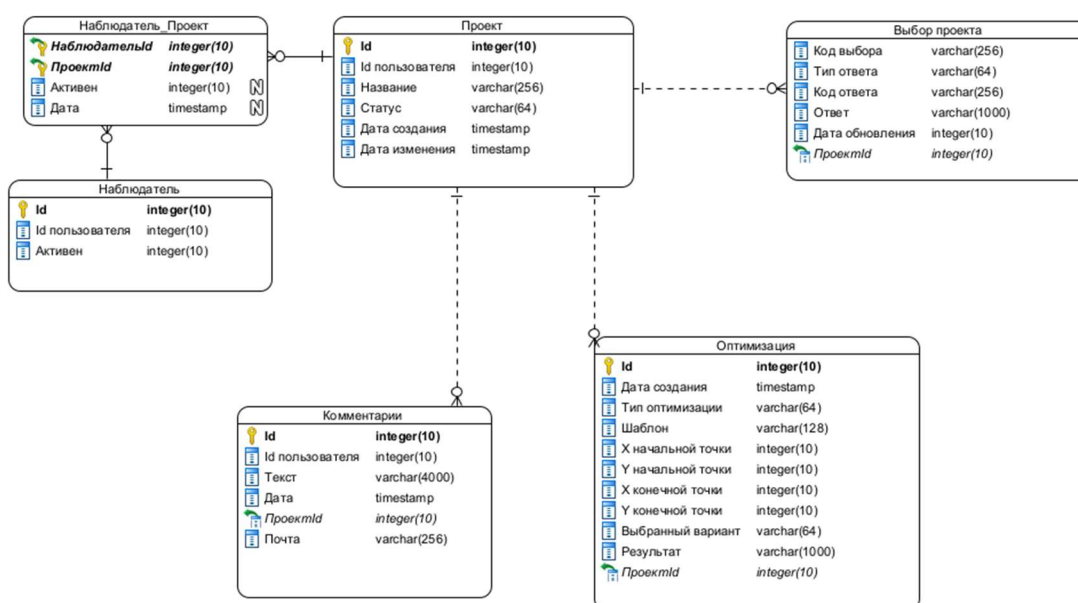


Рисунок 11 – Логическая модель базы данных проекта

Построенная логическая модель показывает независимое от конкретной СУБД схематическое представление данных и связей между ними [10]. Она является связующим звеном между абстрактной концептуальной моделью и физической реализацией, определяя бизнес-правила и структуру информации.

На основе логической модели базы данных была спроектирована диаграмма классов моделей предметной области, она изображена на рисунке 16.

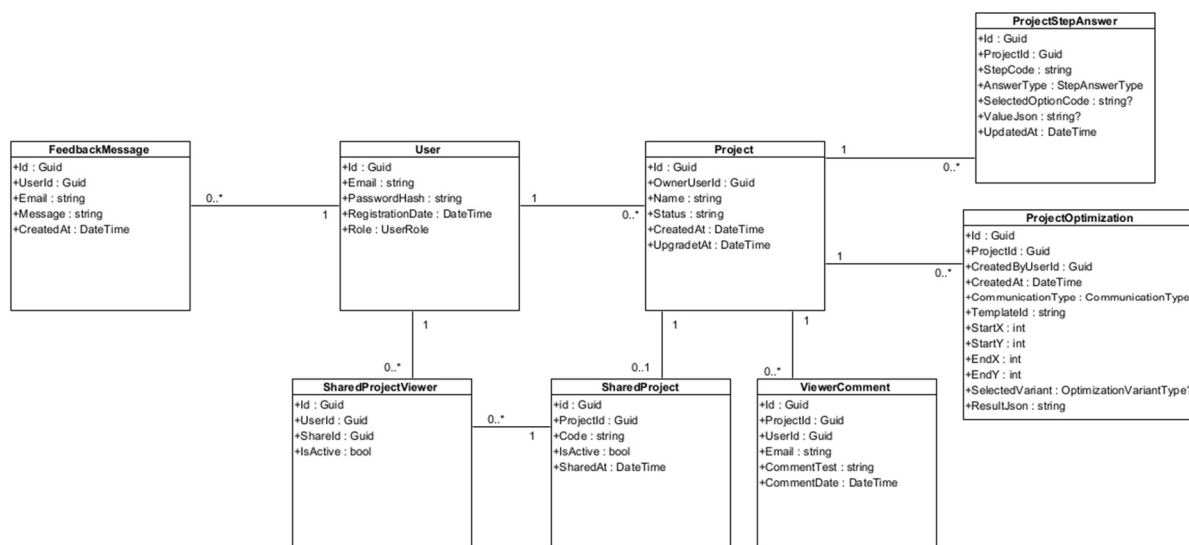


Рисунок 12 – Диаграмма классов моделей предметной области

Краткое описание назначения классов, представленных на диаграмме, приведено в таблице 7.

Таблица 7 – Назначение классов моделей предметной области

Класс	Назначение
User	Класс, описывающий модель пользователя
FeedbackMessage	Класс, описывающий модель данных сообщений обратной связи
Project	Класс, описывающий основные параметры проекта
ProjectStepAnswer	Класс, описывающий выборы пользователя на этапе заполнения информации о проекте
ProjectOptimization	Класс, описывающий оптимизации, созданные пользователем

Продолжение таблицы 7

Класс	Назначение
ViewerComment	Класс, описывающий комментарии пользователей к проекту
SharedProject	Класс, описывающий проект, открытый для просмотра наблюдателям
SharedProjectViewer	Класс, описывающий связь между проектом, открытым для просмотра, и наблюдателями

Таким образом, в процессе реализации будет создано 8 основных классов для работы с данными предметной области.

2.3 Проектирование серверной части приложения

2.3.1 Проектирование микросервисов AuthService и UserService

При проектировании серверной части приложения было принято решение разделить функциональность, связанную с аутентификацией и управлением пользователями, на два отдельных микросервиса. Такое разделение обусловлено различием их ответственности: AuthService отвечает за проверку учетных данных, выдачу JWT-токенов и обеспечение механизма авторизации, тогда как UserService занимается хранением и обработкой данных пользователей, а также сопутствующей пользовательской информацией. Подобный подход позволяет изолировать критически важную логику аутентификации от бизнес-логики работы с пользовательскими данными, повысить модульность системы и упростить дальнейшее сопровождение приложения.

Для демонстрации структуры проектируемых микросервисов была разработана диаграмма классов для каждого микросервиса, на который изображены основные классы, их методы, атрибуты и взаимосвязи. Диаграмма классов AuthService представлена на рисунке 13.

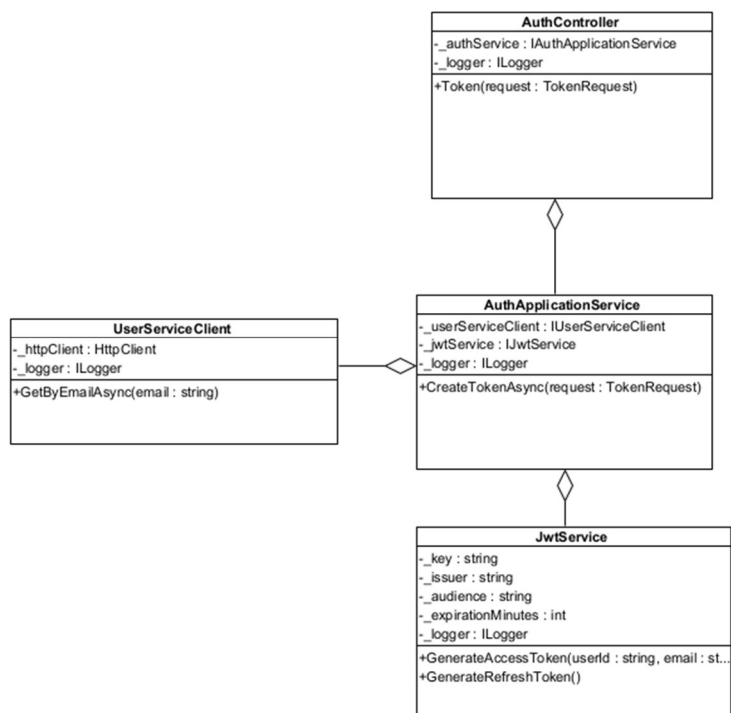


Рисунок 13 – Диаграмма классов AuthService

Диаграмма классов UserService представлена на рисунке 14.

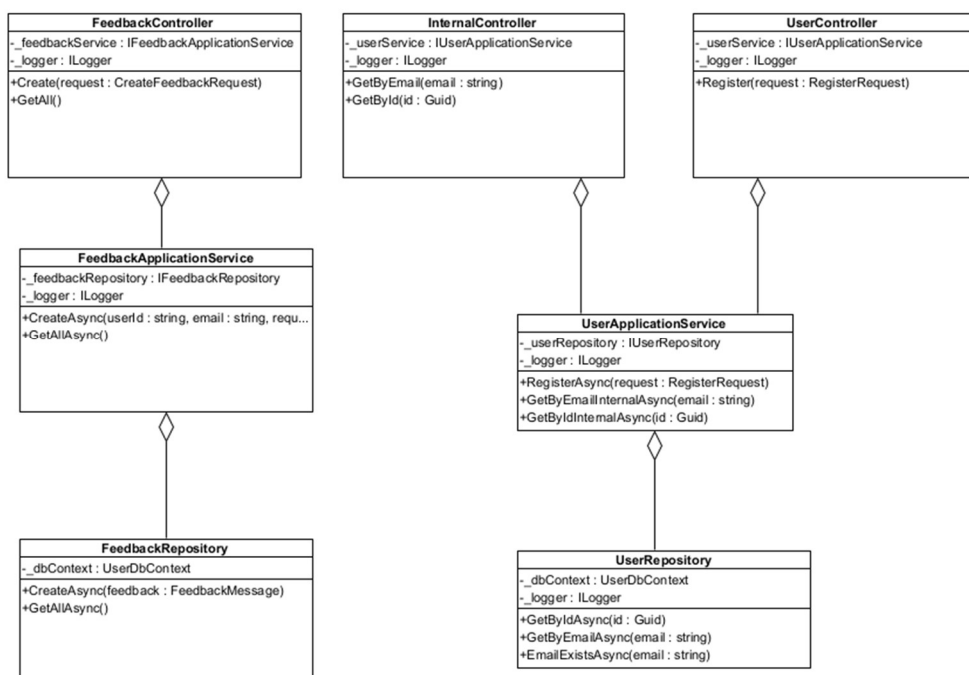


Рисунок 14 – Диаграмма классов UserService

В соответствии с принципом «разделения зависимости» все основные классы каждого микросервиса соответствуют определенной структуре. Их можно поделить на следующие группы:

- контроллеры – слой API (Application Programming Interface), принимает входящие запросы, например AuthController;
- сервисы – слой бизнес-логики, выполняет обработку запросов, полученных из контроллеров, например UserApplicationService;
- репозитории – слой взаимодействия с базой данных, например UserRepository.

Разделение классов по такому принципу позволяет четко разделить ответственность, чтобы систему было легко тестировать, развивать и поддерживать.

2.3.2 Проектирование микросервиса ProjectCalculationService

Для отображения структуры основных классов ProjectCalculationService была разработана диаграмма классов, она представлена на рисунке 15.

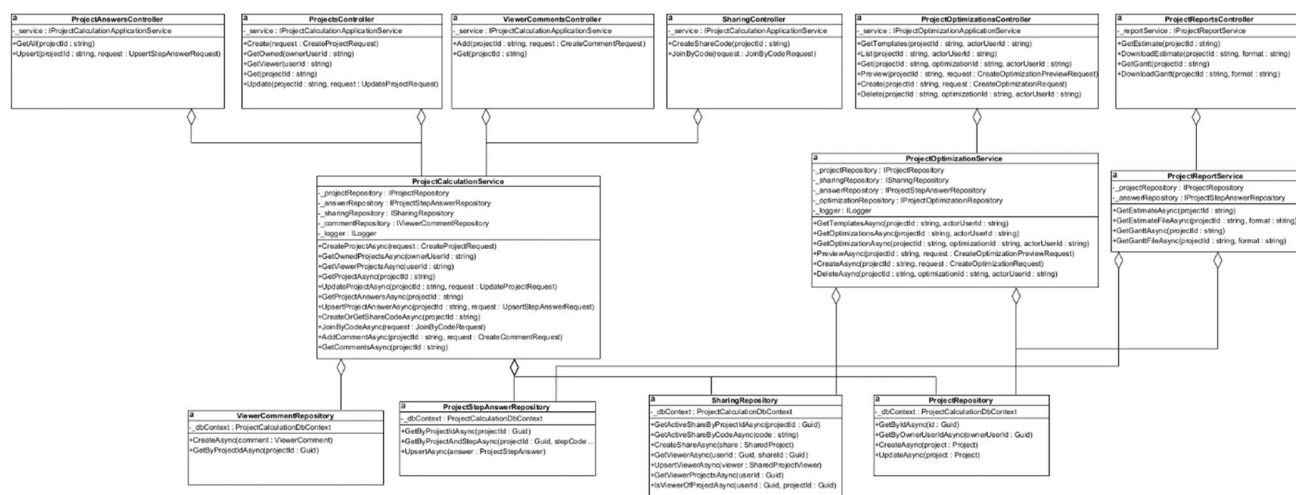


Рисунок 15 – Диаграмма классов ProjectCalculationService

Приведенная диаграмма классов отражает модульную структуру микросервиса и показывает разделение ответственности между уровнями. Контроллеры принимают и маршрутизируют запросы, сервисный слой выполняет обработку и координацию операций, а репозитории обеспечивают

взаимодействие с базой данных. Такая организация классов упрощает сопровождение и позволяет независимо развивать отдельные части микросервиса.

2.3.3 Проектирование математического обеспечения системы

В соответствии с функциональными требованиями необходимо реализовать автоматизированное построения рационального маршрута прокладки инженерных коммуникаций между заданными точками на планировочной схеме объекта. В рамках системы задача сводится к поиску оптимального пути на дискретной модели помещения с учетом конструктивных ограничений: стен, перегородок и запрещённых зон. Такой способ позволяет представить план помещения в формализованном виде и использовать для его анализа алгоритмы поиска на графах. В системе предусмотрены два типа маршрутов: маршрут минимальной длины, ориентированный на экономию материала и сокращение протяжённости трассы, и маршрут вдоль стен, который удобнее с точки зрения монтажа, последующего обслуживания и согласования. План помещения в модели задаётся прямоугольной сеткой, а каждая её ячейка соответствует отдельному участку пространства с собственными координатами. Все клетки делятся на две группы: проходимые, по которым допускается прокладка коммуникаций, и запрещённые – к ним относятся стены, перегородки и конструктивные области, пересечение которых недопустимо.

Для построения кратчайшего маршрута выбран алгоритм A^* . Он хорошо подходит для задач, где требуется минимизировать длину пути, поскольку использует не только фактическую стоимость пройденного участка, но и оценку расстояния до цели. За счёт этого поиск, как правило, выполняется быстрее, чем при полном переборе вариантов. Общая оценка пути задаётся выражением:

$$f(n) = g(n) + h(n),$$

где $g(n)$ – точная стоимость пути от начальной вершины до текущей;

$h(n)$ – эвристическая функция, представляющая собой приблизительную оценку оставшегося расстояния от текущей точки до цели.

Поскольку в модели допускается диагональное перемещение, применяется октильная эвристика. Для ее вычисления на каждом шаге считается расстояние от текущей позиции до целевой по горизонтали (dx) и вертикали (dy). Если dx больше dy вычисление происходит по следующей формуле:

$$h(n) = D_2 \cdot dy + D \cdot (dx - dy),$$

где D – стоимость шага по горизонтали или вертикали;

D_2 – стоимость шага по диагонали.

В случае, если dy больше dx , формула вычисления задается следующим образом:

$$h(n) = D_2 \cdot dx + D \cdot (dy - dx).$$

Для поиска маршрута вдоль стен используется алгоритм Дейкстры. В данном случае задача формулируется не только как поиск кратчайшего пути, но и как поиск пути с учетом дополнительных инженерных предпочтений: трасса должна проходить вдоль стен и содержать как можно меньше поворотов. Поэтому в качестве состояния рассматривается не только координата клетки, но и направление движения, то есть вершина графа задается тройкой x, y, d , где d – направление перехода в текущую клетку. Стоимость перехода в таком случае определяется следующей формулой:

$$c(i, j) = 1 + a \cdot I_{\text{не рядом со стеной}} + b \cdot I_{\text{поворот}},$$

где 1 – базовая стоимость шага;

a – штраф за движение по участкам, не прилегающим к стенам;

b – штраф за поворот;

I – индикаторная функция, принимающая значение 1 при выполнении условия и 0 в противном случае.

Суммарная стоимость маршрута вдоль стен определяется выражением:

$$F(P) = \sum_{k=1}^{n-1} c(v_k, v_{k+1})$$

где P – найденный путь;

v_k – его последовательные вершины.

Важным моментом является стоимость штрафов, алгоритм Дейкстры гарантирует нахождение оптимального решения в рамках заданной стоимости, только если штрафы неотрицательны [11].

Используя данные алгоритмы, в системе будет реализовано два подхода к поиску оптимального маршрута, алгоритм «А*» применяется для построения кратчайшей трассы по длине и обеспечивает минимальный расход материала. Алгоритм Дейкстры в свою очередь используется в случаях, когда важнее инженерная удобность маршрута: его прохождение вдоль стен и снижение количества поворотов. Пользователь может выбрать тот вариант, который лучше соответствует задаче проектирования – либо максимально короткий путь, либо более технологичное решение, соответствующее практике прокладки коммуникаций.

2.4 Проектирование клиентской части приложения

Клиентская часть приложения представляет собой веб-приложение, предоставляющее пользовательский интерфейс для взаимодействия с функционалом системы. С точки зрения архитектуры приложение является одностраничным, что позволяет переходить между разделами без полной перезагрузки страницы, а также производить обмен данными с серверной частью посредством HTTP-запросов к соответствующим сервисам.

Приложение разрабатывается на языке TypeScript с использованием фреймворка «React», также используется система сборки «Vite». React применяется для построения интерфейса из переиспользуемых компонентов и управления состоянием [12], а Vite обеспечивает быстрый старт проекта и высокую скорость обновления интерфейса за счет современной схемы сборки.

Преимущества такого подхода связаны с производительностью и сопровождаемостью: уменьшается время отклика интерфейса, так как большая часть логики отображения выполняется на стороне клиента, и сокращается объем повторной загрузки ресурсов. Компонентная структура упрощает развитие и тестирование интерфейса.

В соответствии с функциональными требованиями была разработана страница авторизации, она представлена на рисунке 16.

Рисунок 16 – Страница авторизации

К основным элементам данной страницы относятся:

- поле для ввода почты (1);
- поле для ввода пароля (2);
- кнопка входа (3);
- кнопка для перехода к регистрации (4).

На рисунке 17 представлена страница регистрации.

Рисунок 17 – Страница регистрации

Страница регистрации имеет аналогичные поля для заполнения, однако в соответствии с выявленными функциональными требованиями на ней присутствует возможность выбора роли пользователя при регистрации.

На рисунке 18 изображена страница проектов – основная страница, куда попадает пользователь при входе в аккаунт.

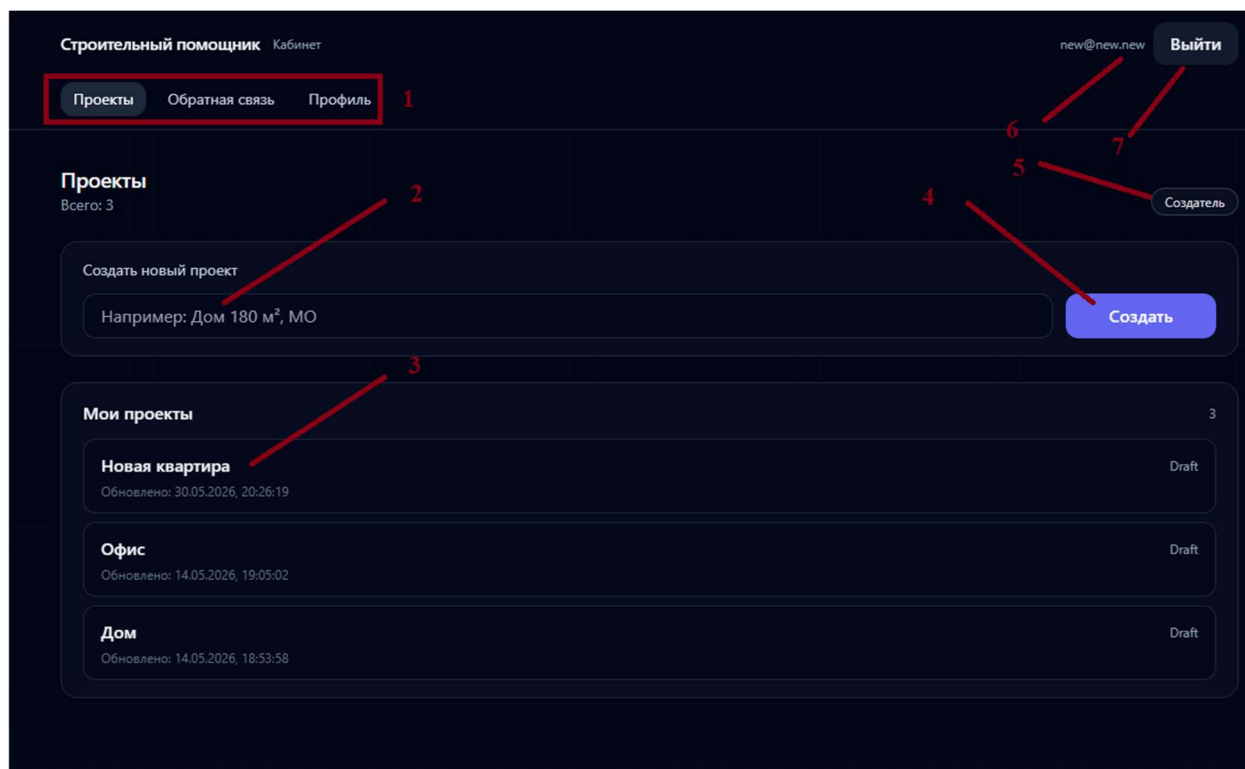


Рисунок 18 – Страница проектов

Рассмотрим основные элементы данной страницы относятся:

- навигация между разделами (1);
- поле для ввода имени нового проекта (2);
- созданные ранее проекты (3);
- кнопка для создания нового проекта (4);
- роль пользователя (5);
- почта пользователя (6);
- кнопка выхода из аккаунта (7).

Страница проекта представлена на рисунке 19.

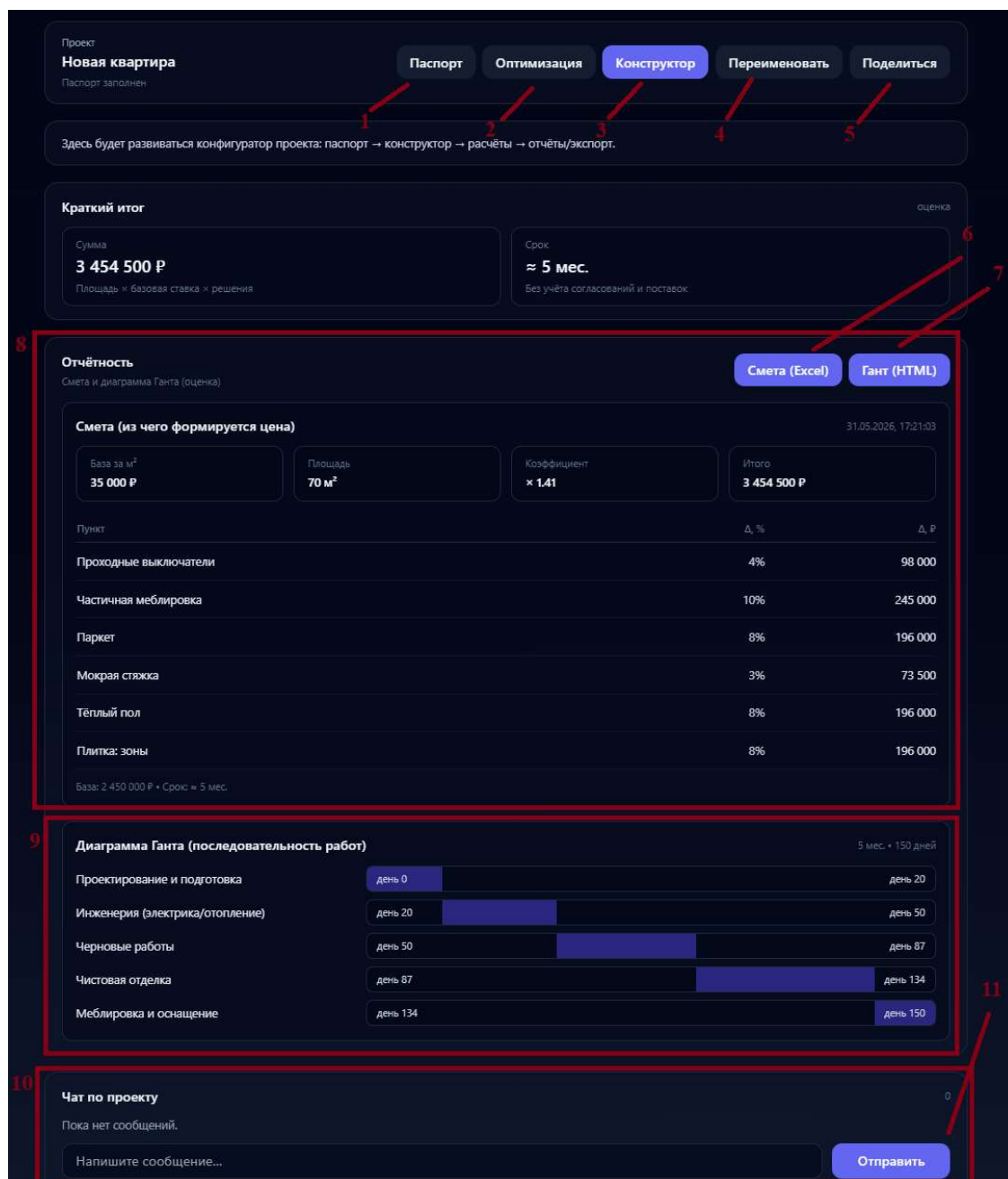


Рисунок 19 – Страница проекта

К основным элементам данной страницы относятся:

- кнопка перехода к паспорту проекта (1);
- кнопка перехода к странице оптимизации (2);
- кнопка перехода к конструктору проекта (3);
- кнопка для переименования проекта (4);
- кнопка для создания кода проекта (5);
- кнопка для скачивания сметы в формате Excel (6);

- кнопка для скачивания диаграммы Ганта проекта в формате HTML (7);
- раздел с предварительной оценкой стоимости работ (8);
- раздел с предварительной оценкой продолжительности работ (9);
- раздел с чатом по проекту (10);
- кнопка отправки сообщения в чат (11).

На рисунке 20 представлена страница оптимизации.

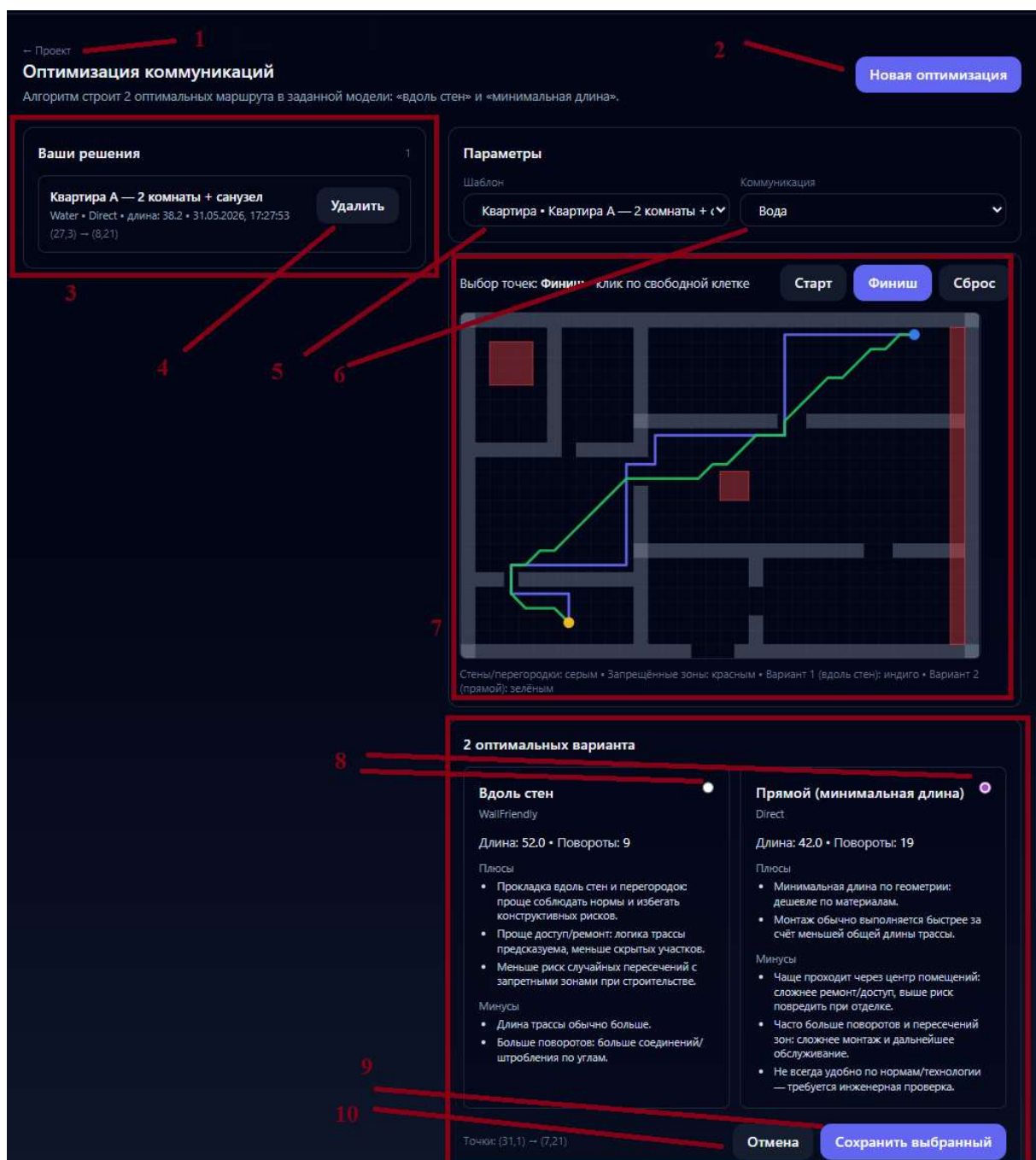


Рисунок 20 – Страница оптимизации

К основным элементам данной страницы относятся:

- переход на страницу проекта (1);
- кнопка создания новой оптимизации (2);
- раздел принятых решений (3);
- кнопка удаления решения (4);
- выбор шаблона для оптимизации (5);
- выбор типа коммуникаций (6);
- окно взаимодействия с оптимизацией (7);
- выбор оптимального решения (8);
- кнопка сохранения решения (9);
- кнопка отмены (10).

Таким образом, спроектированный пользовательский интерфейс соответствует определенным ранее функциональным требованиям к разрабатываемым программным средствам.

3 Реализация программных средств поиска оптимальных решений строительного проекта

3.1 Реализация базы данных

В процессе реализации на основе логической схемы базы данных была построена физическая модель базы данных, отображающая детализированное представление данных. Графическое представление физической модели было получено с помощью интегрированной среды разработки баз данных JetBrains DataGrip, физическая модель базы данных пользователей представлена на рисунке 21.

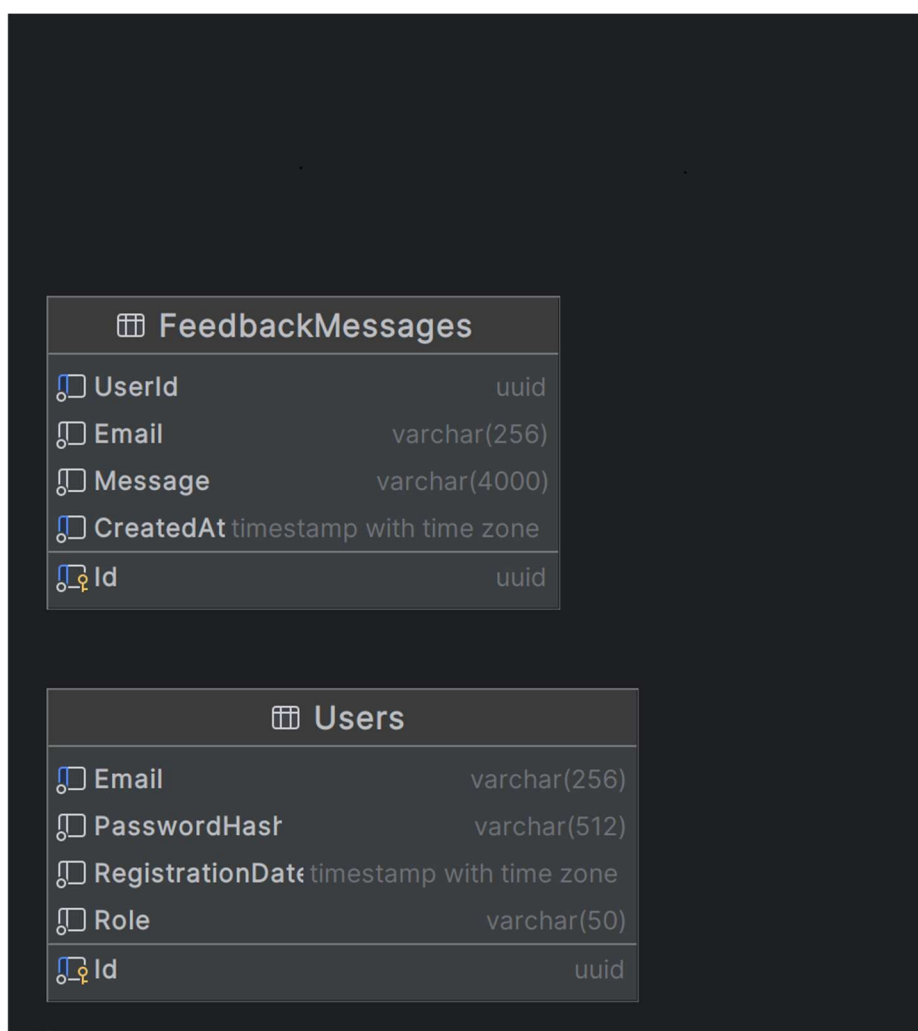


Рисунок 21 – Физическая модель базы данных пользователей

Физическая модель базы данных проектов представлена на рисунке 22.

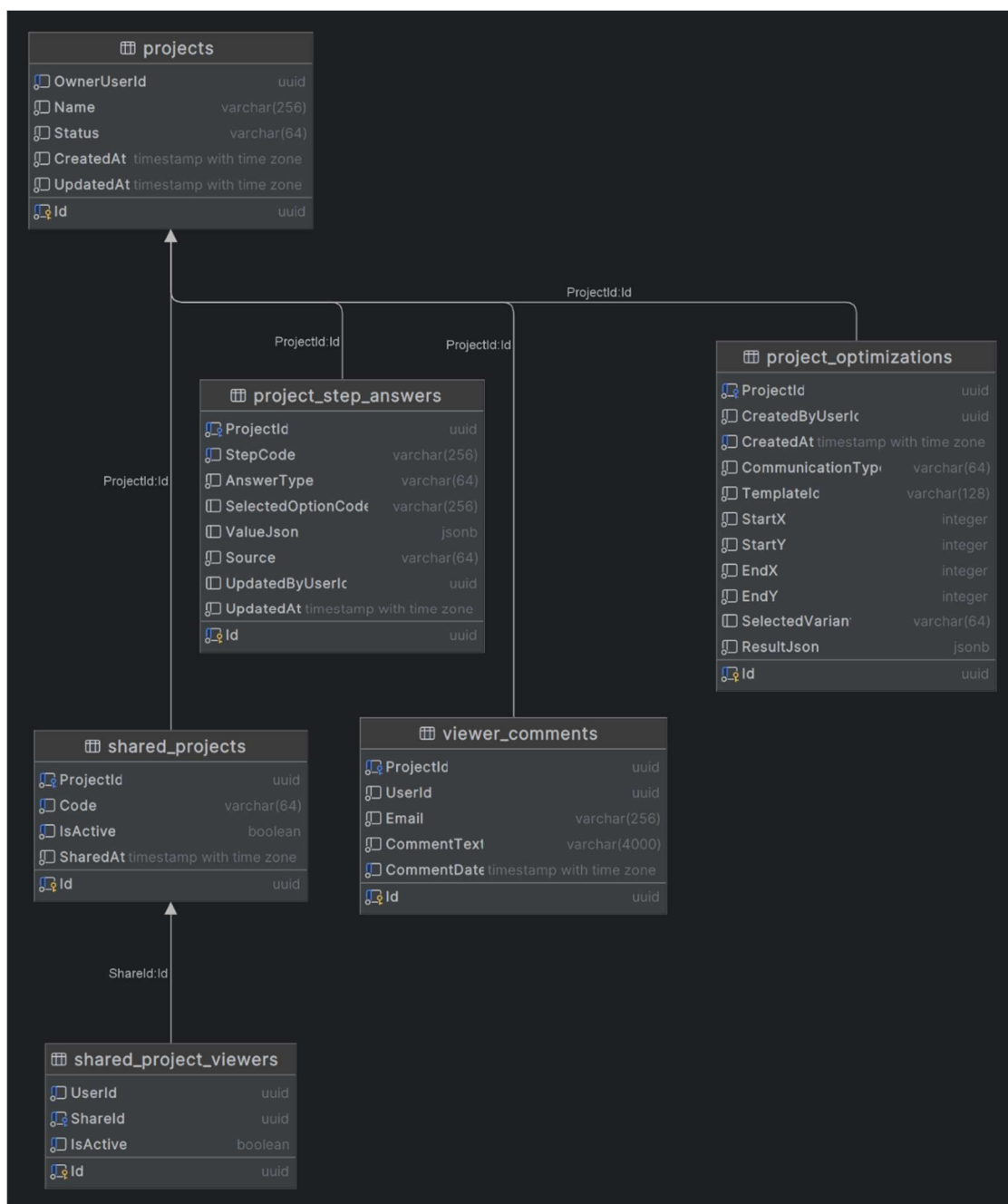


Рисунок 22 – Физическая модель базы данных проектов

Таким образом, физическая модель разработанных баз данных определяет описанные ранее сущности и их атрибуты с учетом специфики используемой системы управления базами данных PostgreSQL [13].

Миграции – это управление изменениями в схеме базы данных с помощью программных средств. Создание таблиц баз данных реализовано в программном коде серверного приложения путем миграций, реализованных с помощью библиотеки EntityFrameworkCore [14].

3.2 Реализация клиентского приложения

Клиентское приложение разрабатывалось на языке TypeScript с использованием фреймворка React. Для сборки и запуска в режиме разработки применялся Vite. В основе клиентского приложения лежит модульная архитектура, в которой логика интерфейса разделена на несколько самостоятельных слоёв, это позволяет отделить работу с сервером, обработку состояния и различные вспомогательные функции от основной логики.

На стороне пользователя приложение работает в веб-браузере: при открытии загружаются статические файлы, после чего интерфейс функционирует как одностраничное приложение без постоянных перезагрузок страницы. Действия пользователя (ввод данных, отправка форм, переходы по разделам) приводят к выполнению запросов к серверным сервисам, а полученные ответы обрабатываются и отображаются в интерфейсе. Для отправки запросов к микросервисам используется библиотека Axios. В проекте создано несколько заранее настроенных экземпляров HTTP-клиента: `authHttp`, `userHttp` и `projectHttp`. Каждый из них связан со своим микросервисом и получает базовый адрес из переменных окружения. Структура API-клиента разделена по предметным областям: отдельные модули отвечают за авторизацию, пользователей, проекты, комментарии, отчёты и оптимизацию. Это позволяет сохранять ясную связь между клиентской функцией и соответствующим серверным функционалом. Фрагмент кода клиентского приложения приведен в приложении В. Полный код расположен на сервисе GitHub [15].

Таким образом, клиентское приложение построено по принципу разделения ответственности: маршрутизация, состояние, работа с API, валидация и отображение интерфейса вынесены в отдельные слои. Использование React, TypeScript, Vite, Axios обеспечивает гибкую и масштабируемую реализацию клиентского приложения, а модульная структура облегчает его дальнейшее развитие и сопровождение.

3.3 Реализация серверной части приложения

Для серверной части приложения была разработана диаграмма компонентов, отображающая структуру системы, программные модули, а также их связи и зависимости, она представлена на рисунке 23.

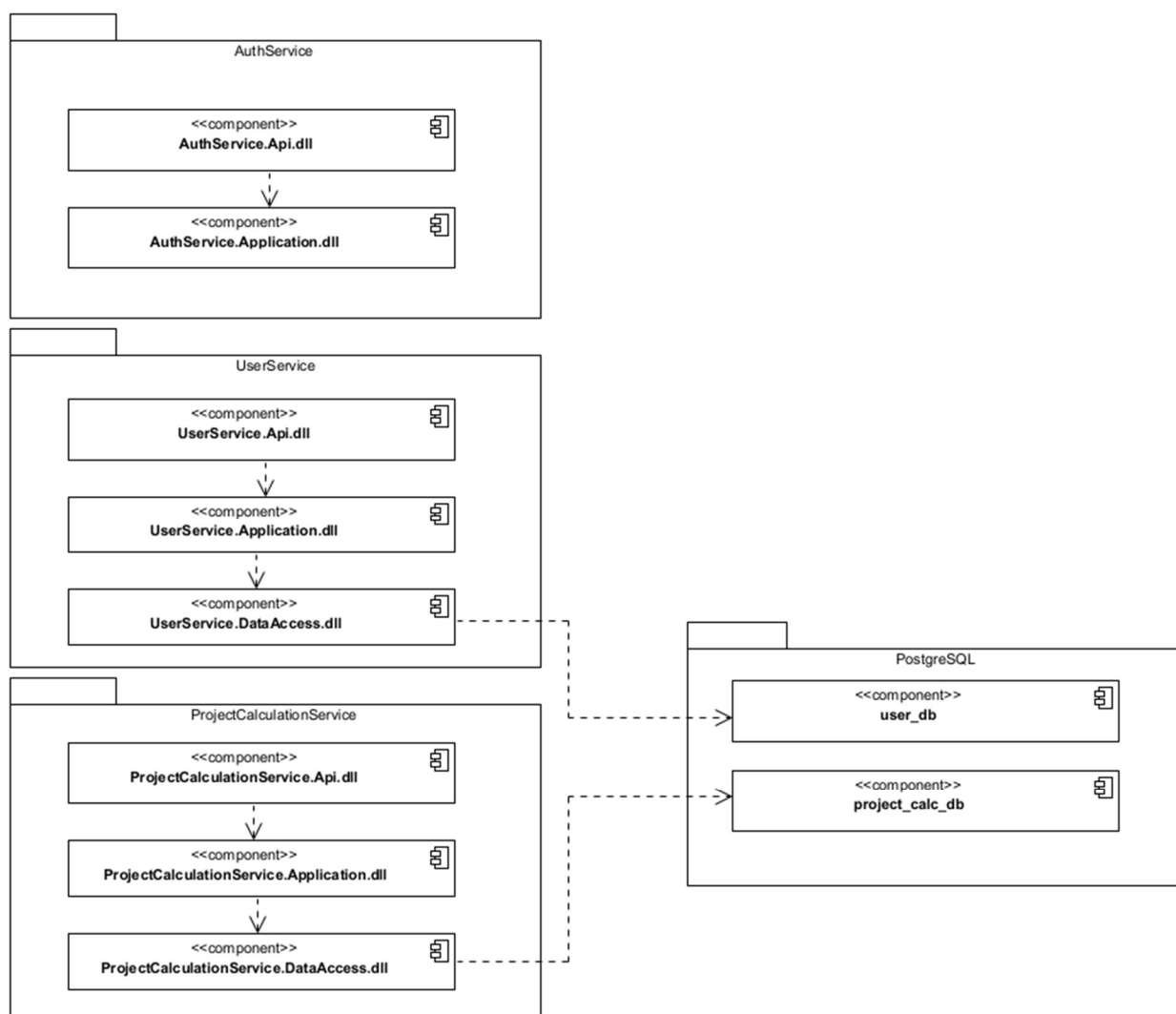


Рисунок 23 – Диаграмма компонентов

В процессе разработки микросервисов был использован подход «Чистая архитектура», главная цель которого отделить логику приложения от деталей реализации, чтобы систему было легко тестировать, развивать и поддерживать [16]. На диаграмме компонентов также прослеживается данный принцип, каждый микросервис состоит из проектов, компилируемых в DLL-

библиотеку. Каждый проект указывает на исполняемую им функцию, проекты именуемые «Api» содержат контроллеры, представляющие собой API (Application Programming Interface) приложения, проекты «Application» содержат сервисы с основной бизнес-логикой приложения, а проекты «DataAccess» содержат репозитории и взаимодействуют с базой данных [17]. У Микросервиса AuthService отсутствует слой взаимодействия с базой данных, поскольку он не имеет собственной базы данных, а получает информацию о пользователях из UserService посредством HTTP-запроса. Из диаграммы видно, что каждый микросервис имеет свою базу данных, что соответствует принципам микросервисной архитектуры и является необходимым для изоляции данных и параллельной разработки. Такой подход исключает ситуацию, при которой вся информация хранится в едином централизованном хранилище, и позволяет каждому сервису самостоятельно управлять структурой и содержимым своей БД.

Для обеспечения безопасного доступа к функционалу приложения в системе реализован механизм авторизации на основе JWT (JSON Web Token). JWT представляет собой компактный токен, содержащий набор утверждений о пользователе и подписанный сервером. Такой подход позволяет отказаться от хранения серверной сессии для каждого клиента и использовать аутентификацию без состояния, при которой сервер проверяет подлинность запроса по подписи токена. Это является преимуществом микросервисной архитектуры, так как разные сервисы могут независимо проверять токен и принимать решения о доступе. Функция получения токена при аутентификации представлена на листинге 1, фрагмент кода серверной части представлен в приложении Г. Полный код расположен на сервисе GitHub [15].

Листинг 1 – Функция получения JWT-токена

```
public async Task<TokenResponse> CreateTokenAsync(TokenRequest
request)
{
    var normalizedEmail = NormalizeEmail(request.Email);

    var user = await
_userServiceClient.GetByEmailAsync(normalizedEmail);
```

```

        if (user == null)
        {
            _logger.LogWarning("Token request failed: user not found - {Email}", normalizedEmail);
            throw new BadRequestException("Invalid credentials");
        }

        if (!VerifyPassword(request.Password, user.PasswordHash))
        {
            _logger.LogWarning("Token request failed: invalid password - {Email}", normalizedEmail);
            throw new BadRequestException("Invalid credentials");
        }

        var accessToken = _jwtService.GenerateAccessToken(user.Id,
user.Email, user.Role);
        var refreshToken = _jwtService.GenerateRefreshToken();

        return new TokenResponse
        {
            AccessToken = accessToken,
            RefreshToken = refreshToken,
            User = new UserResponse
            {
                Id = user.Id,
                Email = user.Email,
                Role = user.Role
            }
        };
    }
}

```

В рассматриваемой реализации используются два типа токенов: access token – короткоживущий токен, который передаётся в заголовке «Authorization» и используется для доступа к защищённым API-методам, и refresh token – более долговечный токен, предназначенный для получения нового access token без повторного ввода логина и пароля. Access token формируется на основе идентификатора пользователя, его электронной почты и роли. Включение роли в токен особенно важно, поскольку именно на её основании система определяет, какие действия разрешены пользователю.

Полная схема процесса аутентификации изображена на диаграмме последовательности, представленной на рисунке 24.

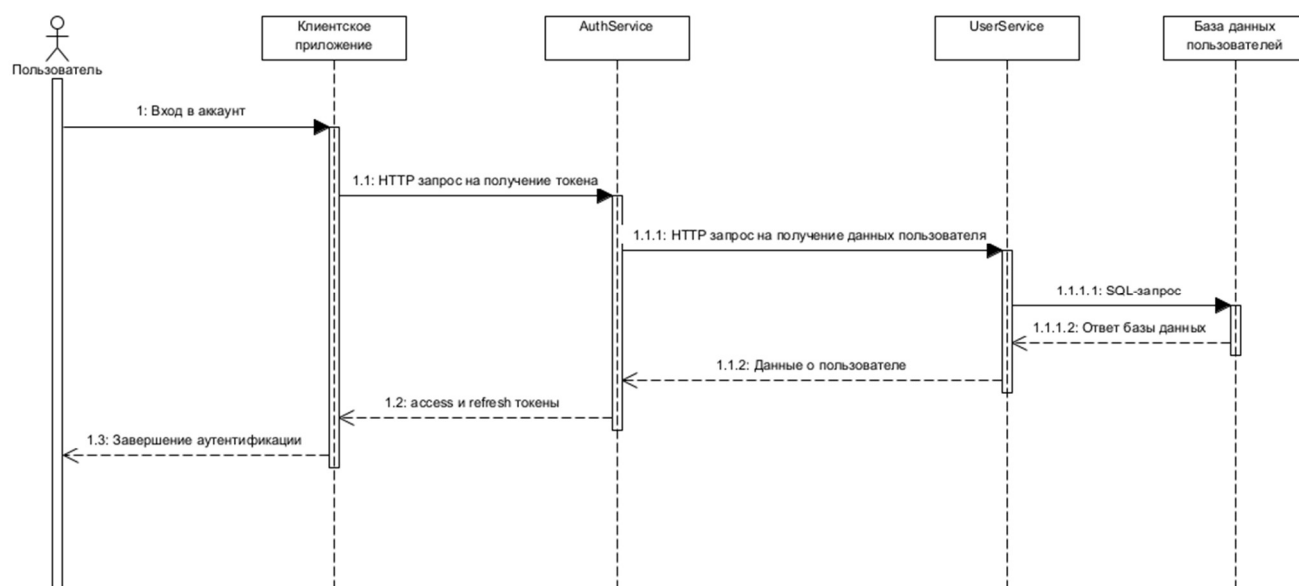


Рисунок 24 – Диаграмма последовательности процесса аутентификации

Полученные токены сохраняются клиентским приложением, затем используются при отправке запросов в микросервисы.

Взаимодействие серверной части приложения с базой данных реализовано с использованием Entity Framework Core и провайдера Npgsql, обеспечивающего работу с PostgreSQL. Данный подход позволяет использовать объектно-реляционное отображение, при котором данные базы представлены в виде сущностей C#, а операции чтения, добавления, изменения и удаления выполняются средствами объектной модели. Это упрощает сопровождение кода, повышает его читаемость и снижает объём ручного SQL-кода. Доступ к данным организован по принципу репозиторий. Такой подход отделяет логику хранения информации от прикладной бизнес-логики и обеспечивает более чёткое разделение ответственности между слоями системы. Все операции с базой данных выполняются асинхронно, что повышает производительность серверной части и позволяет эффективно обрабатывать несколько запросов одновременно. Таким образом, выбранный механизм взаимодействия с базой данных обеспечивает надёжное хранение информации, удобную работу с моделью данных и гибкость при дальнейшем развитии системы.

Для поиска кратчайшего пути была создана реализация описанного ранее алгоритма «А*», она представлена на листинге 2.

Листинг 2 – Реализация метода поиска кратчайшего пути

```
public static VariantResult
FindDirect(OptimizationTemplateResponse template, GridPointDto
start, GridPointDto end)
{
    if (IsBlocked(template, start.X, start.Y))
        return new VariantResult(false, "Стартовая точка находится
в стене/запрещённой зоне.", Array.Empty<GridPointDto>(), 0, 0, 0);
    if (IsBlocked(template, end.X, end.Y))
        return new VariantResult(false, "Конечная точка находится
в стене/запрещённой зоне.", Array.Empty<GridPointDto>(), 0, 0, 0);

    var width = template.Width;
    var height = template.Height;

    var dist = new double[width, height];
    var prev = new (int px, int py)[width, height];
    var hasPrev = new bool[width, height];
    var visited = new bool[width, height];

    for (var x = 0; x < width; x++)
        for (var y = 0; y < height; y++)
            dist[x, y] = double.PositiveInfinity;

    var pq = new PriorityQueue<(int x, int y), double>();
    dist[start.X, start.Y] = 0;
    pq.Enqueue((start.X, start.Y), HeuristicOctile(start.X,
start.Y, end.X, end.Y));

    var neighbors = new (int dx, int dy, double cost)[]
    {
        (1, 0, 1.0),
        (0, 1, 1.0),
        (-1, 0, 1.0),
        (0, -1, 1.0),
        (1, 1, Math.Sqrt(2)),
        (-1, 1, Math.Sqrt(2)),
        (-1, -1, Math.Sqrt(2)),
        (1, -1, Math.Sqrt(2))
    };

    while (pq.TryDequeue(out var node, out _))
    {
        if (visited[node.x, node.y]) continue;
        visited[node.x, node.y] = true;

        if (node.x == end.X && node.y == end.Y)
```

```

        {
            break;
        }

        var baseDist = dist[node.x, node.y];
        foreach (var (dx, dy, step) in neighbors)
        {
            var nx = node.x + dx;
            var ny = node.y + dy;
            if (nx < 0 || ny < 0 || nx >= width || ny >= height)
                continue;
            if (IsBlocked(template, nx, ny)) continue;

            if (dx != 0 && dy != 0)
            {
                if (IsBlocked(template, node.x + dx, node.y) ||
                    IsBlocked(template, node.x, node.y + dy)) continue;
            }

            var nd = baseDist + step;
            if (nd < dist[nx, ny])
            {
                dist[nx, ny] = nd;
                prev[nx, ny] = (node.x, node.y);
                hasPrev[nx, ny] = true;
                var priority = nd + HeuristicOctile(nx, ny, end.X,
                    end.Y);
                pq.Enqueue((nx, ny), priority);
            }
        }

        if (double.IsInfinity(dist[end.X, end.Y]))
        {
            return new VariantResult(false, "Путь не найден (все варианты перекрыты стенами/зонами).", Array.Empty<GridPointDto>(), 0, 0, 0);
        }

        var path = ReconstructPath(start, end, prev, hasPrev);
        var (len, turns) = ComputeMetrics(path);
        return new VariantResult(true, null, path, len, turns, dist[end.X, end.Y]);
    }

```

Рассматриваемая функция реализует поиск кратчайшего маршрута между двумя точками на дискретной планировочной схеме помещения. Основная задача алгоритма состоит в том, чтобы найти путь по свободным ячейкам сетки с учётом препятствий в виде стен и запрещённых зон. На каждом шаге выбирается та

вершина, для которой сумма уже пройденной стоимости и оценочной стоимости оставшегося пути является наименьшей.

В реализации применяются структуры данных, которые обеспечивают удобную и эффективную работу с двумерной сеткой:

- двумерный массив `dist` – хранит минимальную известную стоимость пути до каждой клетки, значение `double.PositiveInfinity` задаёт первоначально бесконечную стоимость, что позволяет корректно отличать ещё не посещённые клетки от уже обработанных;
- двумерный массив `prev` – хранит координаты предыдущей клетки для каждой позиции;
- двумерный массив `hasPrev` – фиксирует, была ли для клетки определена предыдущая вершина;
- двумерный массив `visited` – отмечает уже извлечённые из очереди вершины, предотвращает повторную обработку одной и той же клетки и исключает лишние вычисления;
- очередь с приоритетом `PriorityQueue` – хранит вершины, подлежащие обработке, в порядке возрастания их приоритета, именно эта структура обеспечивает эффективную реализацию алгоритма, так как следующей выбирается не ближайшая по координатам клетка, а клетка с наименьшей оценкой функции $f(n)$;
- массив соседей `neighbors` – задаёт возможные переходы из текущей клетки, в нём заранее перечислены все допустимые направления движения: четыре ортогональных и четыре диагональных, это упрощает цикл обхода соседей и делает логику алгоритма компактной и наглядной.

Для перемещения по сетке используется два типа шагов: горизонтальный или вертикальный с длиной один, диагональный с длиной равной корню из двух. Сначала проверяется корректность стартовой и конечной точек: они не должны попадать в стены или запрещённые зоны. После этого стартовая клетка помещается в очередь с приоритетом, а её стоимость считается равной нулю.

Далее алгоритм последовательно извлекает из очереди клетку с наименьшим значением $f(n)$, анализирует её соседей и обновляет для них минимально известную стоимость пути. Отдельно в реализации предусмотрена проверка, исключающая срезание угла при диагональном перемещении. Это означает, что переход по диагонали разрешается только тогда, когда обе ортогональные соседние клетки также свободны. Такое ограничение делает маршрут геометрически корректным и предотвращает прохождение через условно недоступные участки между препятствиями.

Разработанное приложение разворачивается с помощью Docker – средства контейнеризации, позволяющего упаковать каждый компонент приложения вместе со всеми зависимостями в изолированную среду выполнения. Благодаря этому приложение запускается одинаково на разных машинах и не зависит от особенностей настроек операционной системы и установленного программного окружения [18]. На рисунке 25 представлена диаграмма развертывания, на которой изображены основные компоненты приложения [19].

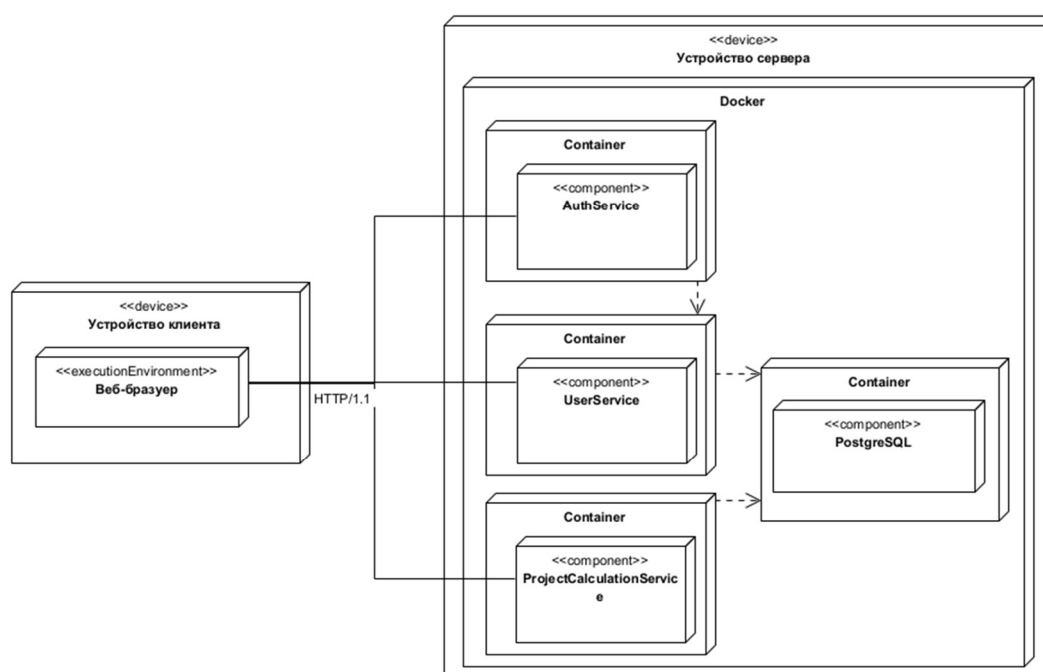


Рисунок 25 – Диаграмма развертывания

Из диаграммы видно, что на стороне клиента используется браузер, через который пользователь взаимодействует с графическим интерфейсом приложения и отправляет запросы к серверной части по протоколу HTTP/1.1. На серверной стороне развернуты контейнеры, содержащие микросервисы, а также контейнер с PostgreSQL. Контейнеры UserService и ProjectCalculationService взаимодействуют с базой данных для чтения и записи информации, а AuthService обращается к UserService для выполнения операций, связанных с учетными записями и проверкой данных пользователя.

4 Контроль качества

4.1 Выбор метода тестирования

В процессе разработки программных систем применяются разные способы тестирования: проверка отдельных частей кода, проверка взаимодействия компонентов и проверка работы системы с точки зрения пользователя. Выбор конкретного способа зависит от того, какую часть приложения требуется контролировать и какие риски важнее снизить: ошибки логики внутри модулей, ошибки интеграции или нарушения пользовательских сценариев.

Для серверной части в качестве метода тестирования было выбрано модульное тестирование, позволяющее проверить функционал отдельных классов и их функций независимо от внешних зависимостей. Модульное тестирование автоматизировано, что позволяет моментально находить ошибки в критическом функционале после внесения изменений. По этой причине модульными тестами покрывается основной функционал, в котором сосредоточена бизнес-логика приложения.

Клиентская часть приложения, в свою очередь, требует функциональной проверки, так как в данном случае важно проверить не отдельные программные модули, а поведение системы с точки зрения пользователя. В рамках функционального тестирования оценивается отображаются ли данные корректно, правильно ли отрабатывают основные сценарии [20]. Такая проверка позволяет убедиться, что приложение работает в точном соответствии с определенными функциональными требованиями.

Совместное использование этих методов тестирования позволяет убедиться в корректности работы приложения. Модульные тесты обеспечивают правильную работу серверной части приложения, помогают убедиться, что отдельные вычисления, обработка данных и логика сервисов работают безошибочно независимо от внешних факторов. Функциональное тестирование фокусируется на проверке пользовательского пути, то есть основных сценариев в интерфейсе приложения.

4.2 Тестирование серверной части

Модульное тестирование серверной части направлено на проверку отдельных методов и компонентов бизнес-логики в изоляции от внешних зависимостей, таких как базы данных, сети, файловой системы. Такой подход позволяет быстро выявлять ошибки в правилах обработки данных и контролировать регрессии после изменений. В проекте модульные тесты реализованы для прикладного слоя сервиса расчётов с использованием библиотеки xUnit, а внешние зависимости подменяются специальными объектами-заглушками, что обеспечивает воспроизводимость результатов [21]. План модульного тестирования приведен в таблице 8.

Таблица 8 – План модульного тестирования

Тест	Описание	Ожидаемый результат	Фактический результат
CreateProjectAsync_InvalidOwnerId	Передан некорректный идентификатор владельца проекта	Пройден	Пройден
CreateProjectAsync_EmptyName	Попытка создать проект с пустым названием	Пройден	Пройден
CreateProjectAsync_ValidRequest	Создание проекта с корректными данными и проверка возвращаемого ответа	Пройден	Пройден
GetProjectAsync_ProjectNotFound	Запрос проекта по несуществующему идентификатору	Пройден	Пройден
UpsertProjectAnswerAsync_NoExistingAnswer	Сохранение ответа шага при отсутствии предыдущего значения	Пройден	Пройден
UpsertProjectAnswerAsync_InvalidProjectId	Передан некорректный идентификатор проекта при сохранении ответа шага	Пройден	Пройден

Продолжение таблицы 8

Тест	Описание	Ожидаемый результат	Фактический результат
CreateOrGetShareCodeAsync_WhenShareExists	При наличии активной ссылки доступа повторная генерация не выполняется	Пройден	Пройден

Описанные тесты были проведены с помощью среды разработки JetBrains Rider, результат тестирования представлен на рисунке 26.

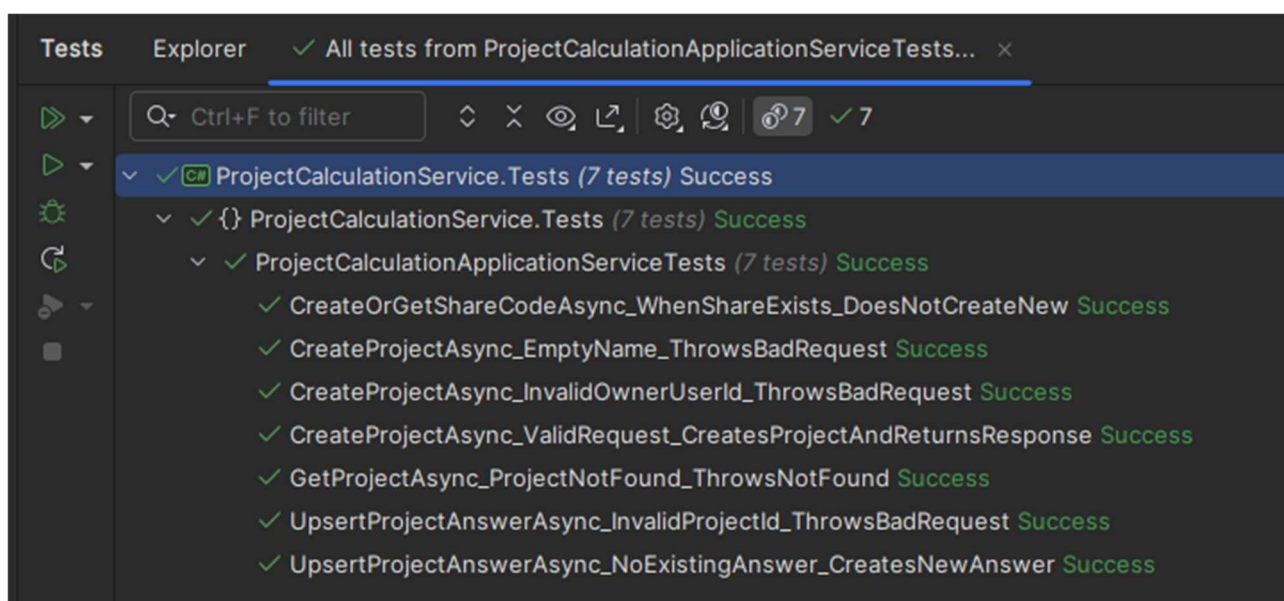


Рисунок 26 – Результат модульного тестирования

В ходе модульного тестирования ошибок выявлено не было, следовательно основной функционал серверной части приложения работает корректно.

4.3 Тестирование клиентского приложения

Для клиентского приложения было проведено функциональное тестирование, покрывающее основные варианты использования, результат тестирования приведен в таблице 9, соответствующие изображения с результатами тестирования приведены в приложении Б.

Таблица 9 – Результат функционального тестирования

№ п/п	Тест	Ожидаемый результат	Результат теста
1	Попытка входа с неверным паролем	Отображается сообщение об ошибке входа	Пройден (рисунок Б.1)
2	Регистрации с уже занятой почтой	Отображение сообщения о том, что пользователь с такой почтой уже существует	Пройден (рисунок Б.2)
3	Открытие страницы проекта	Отображаются основные данные на странице	Пройден (рисунок Б.3)
4	Создание проекта с пустым названием	Отображается сообщение о необходимости указать название	Пройден (рисунок Б.4)
5	Попытка оптимизации с точкой старта в замкнутом пространстве	Отображается сообщение о том, что путь перекрыт стенами	Пройден (рисунок Б.5)
6	Предпросмотр оптимизации при корректных точках старта/финиша	Отображаются два варианта маршрута на схеме и их параметры	Пройден (рисунок Б.6)
7	Создание оптимизации с выбором варианта	Оптимизация появляется в списке сохранённых решений	Пройден (рисунок Б.7)
8	Скачивание сметы проекта	Файл скачивается, смета корректно открывается и отображается	Пройден (рисунок Б.8)

Продолжение таблицы 9

№ п/п	Тест	Ожидаемый результат	Результат теста
9	Скачивание диаграммы Ганта	Файл скачивается, диаграмма корректно открывается и отображается	Пройден (рисунок Б.9)

Расхождения ожидаемого и фактического результата не выявлено, а значит клиентская часть приложения работает корректно.

4.4 Метрики кода

Для оценки кода приложения были использованы метрики устойчивости и абстрактности. Для вычисления позиционной устойчивости кода используется следующая формула:

$$I = \frac{Ce}{Ca + Ce},$$

где Ce – количество классов вне данного компонента, зависящих от классов внутри компонента;

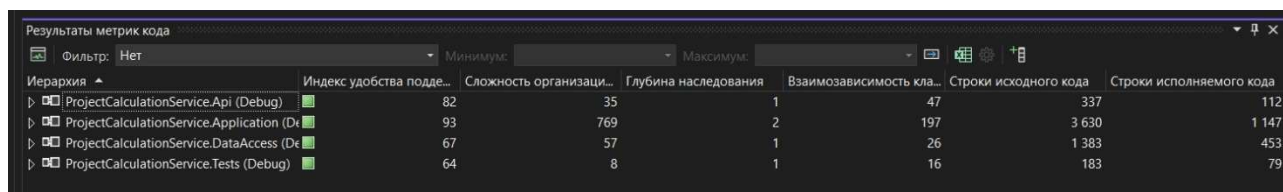
Ca – количество классов внутри данного компонента, зависящих от классов вне компонента.

Полученное значение оценивается в диапазоне от нуля до одного, где 0 – компонент максимально устойчив, 1 – неустойчив. Данная метрика была рассчитана для компонента `UserApplicationService`, имеющего 3 зависящих класса от данного компонента, а также 3 внешних зависимости. Полученный результат – 0.5, что означает сбалансированное соотношение входящих и исходящих связей.

Также была рассчитана метрика абстрактности для микросервиса `UserService`, показывающая долю абстрактных классов и интерфейсов в данном компоненте. Она определяется как отношение интерфейсов и абстрактных классов к общему числу классов. Для микросервиса `UserService` было получено значение 12,5%, это означает, что микросервис в основном состоит из

конкретных реализаций (контроллеры, модели, сервисы, репозитории), а абстракции используются точно – прежде всего для отделения прикладной логики от доступа к данным и для упрощения подмены зависимостей при тестировании.

Также с помощью Visual Studio были получены метрики кода микросервиса ProjectCalculationService, результат представлен на рисунке 27.



Иерархия	Индекс удобства поддержке	Сложность организации	Глубина наследования	Взаимозависимость классов	Строки исходного кода	Строки исполняемого кода
ProjectCalculationService.Api (Debug)	82	35	1	47	337	112
ProjectCalculationService.Application (Debug)	93	769	2	197	3 630	1 147
ProjectCalculationService.DataAccess (Debug)	67	57	1	26	1 383	453
ProjectCalculationService.Tests (Debug)	64	8	1	16	183	79

Рисунок 27 – Метрики кода

Индекс удобства поддержки показывает, насколько легко и удобно поддерживать существующий код, в данном случае значения не превышают 100, что является нормальным значением и означает, что код удобен в поддержке.

Заключение

В ходе выполнения выпускной квалификационной работы было разработано приложение для поиска оптимальных решений строительного проекта.

Был проведен анализ предметной области, рассмотрены существующие программные решения аналогичного назначения, после чего проведен их сравнительный анализ, показавший преимущества и недостатки данных этих программных средств. Затем был проведен анализ функциональных и нефункциональных требований к разрабатываемому программному обеспечению, в результате чего была получена диаграмма вариантов использования, отображающая основные функции приложения для поиска оптимальных решений строительного проекта. Проведено планирование и расчет бюджета разрабатываемых программных средств.

На этапе проектирование было произведено проектирование архитектуры, а также структур данных, в результате чего были получены концептуальная и логическая модели базы данных. Также был спроектирован графический интерфейс и диаграмма классов моделей предметной области.

В процессе реализации была разработана физическая модель базы данных приложения, а также диаграммы классов, компонентов и развертывания для разработанных программных компонентов.

Произведено модульное тестирование серверной части приложения, а также функциональное тестирование клиентской части, показавшее корректность работы разработанных программных средств.

Список использованных источников

1. Ткачев, О. А. PostgreSQL: SQL + PL/pgSQL для тех, кто хочет стать профессионалом / О. А. Ткачев – СПб.: Наука и Техника, 2024. – 480 с.
2. Гош, С. Docker без секретов: разработка и развертывание приложений с помощью Docker / С. Гош, пер. с англ. М. Попович – СПб.: БХВ-Петербург, 2023. – 221 с.
3. Вигерс, К. Разработка требований к программному обеспечению / К. Вигерс, Д. Битти – СПб.: БХВ-Петербург, 2018. – 736 с.
4. Арлоу, Д. UML 2 и Унифицированный процесс. Практический объектно-ориентированный анализ и проектирование / Д. Арлоу – СПб.: Символ-Плюс, 2007. – 624 с.
5. Ричардс, М. Фундаментальный подход к программной архитектуре: паттерны, свойства, проверенные методы / М. Ричардс, Н. Форд – СПб.: Питер, 2023. – 443 с.
6. Богданов, В. В. Управление проектами в Microsoft Project 2007: учебный курс, с компакт диском / В. В. Богданов – СПб.: Питер, 2008. – 592 с.
7. Ньюмен, С. Создание микросервисов. 2-е издание / С. Ньюмен – СПб.: Питер, 2026. – 624 с.
8. Ричардсон, К. Микросервисы: паттерны разработки и рефакторинга / К. Ричардсон – СПб.: Питер, 2020. – 542 с.
9. Коннолли, Т. Базы данных: проектирование, реализация и сопровождение: теория и практика / Т. Коннолли, К. Бегг – 3-е изд. – М.: Вильямс, 2017. – 1439 с.
10. Дейт, К. Дж. Введение в системы баз данных / К. Дж. Дейт – 8-е изд. – М.: Вильямс, 2005. – 1327 с.
11. Кормен, Т. Алгоритмы: построение и анализ / Т. Кормен, Ч. Лейзерсон, Р. Ривест, К. Штайн – 2-е изд. – М.: Вильямс, 2005. – 1290 с.

12. Чиннатамби, К. Изучаем React: практическое руководство по созданию веб-приложений при помощи React и Redux / К. Чиннатамби – 2-е изд. – М.: Эксмо, 2019. – 368 с.
13. Моргунов, Е. П. PostgreSQL: основы языка SQL: учебное пособие / Е. П. Моргунов – СПб.: БХВ-Петербург, 2018. – 336 с.
14. Смит, Дж. П. Entity Framework Core в действии / Дж. П. Смит – М.: ДМК Пресс, 2023. – 690 с.
15. Данков, Д.А. Программные средства поиска оптимальных решений строительного проекта // GitHub [Электронный ресурс]. URL: <https://github.com/Apizlon/construction-assistant> (дата обращения: 12.05.2026).
16. Мартин, Р. Чистая архитектура: искусство разработки программного обеспечения / Р. Мартин – СПб.: Питер, 2026. – 352 с.
17. Мартин, Р. Принципы, паттерны и методики гибкой разработки на языке C# / Р. Мартин, М. Мартин – СПб.: Символ-Плюс, 2011.
18. Гош, С. Docker без секретов: разработка и развертывание приложений с помощью Docker / С. Гош – СПб.: БХВ-Петербург, 2023. – 221 с.
19. Фаулер, М. UML. Основы: краткое руководство по унифицированному языку моделирования / М. Фаулер, К. Скотт – СПб.: Символ, 2002. – 185 с.
20. Канер, С. Тестирование программного обеспечения / С. Канер, Д. Фолк, Е. К. Нгуен – 2-е изд. – Киев: DiaSoft, 2000. – 544 с.
21. Ошероув, Р. Искусство автономного тестирования с примерами на C# / Р. Ошероув – М.: ДМК Пресс, 2014. – 360 с.

Приложение А
(справочное)
Глоссарий

Микросервисная архитектура – подход, при котором система состоит из набора самостоятельных сервисов, взаимодействующих по сети через интерфейсы и разворачиваемых независимо.

Клиентское приложение – часть системы, выполняемая на устройстве пользователя (в браузере) и обеспечивающая графический интерфейс и взаимодействие с сервером.

React – библиотека для построения пользовательского интерфейса на основе компонентов и управления состоянием.

Vite – инструмент для разработки и сборки веб-приложений, обеспечивающий быстрый запуск и сборку проекта.

TypeScript – язык программирования, расширяющий JavaScript статической типизацией для повышения надежности кода.

Одностраничное приложение (SPA) – веб-приложение, в котором переходы между разделами выполняются без полной перезагрузки страницы.

Компонент – самостоятельный фрагмент интерфейса с собственными данными и логикой, из которых собирается экран приложения.

Состояние (state) – набор данных, определяющих текущее поведение и внешний вид интерфейса.

HTTP – сетевой протокол передачи данных, используемый для обмена запросами и ответами между клиентом и сервером.

API – интерфейс взаимодействия, определяющий набор доступных операций и формат обмена данными между компонентами системы.

Контейнеризация – способ запуска приложений в изолированной среде с зафиксированными зависимостями и настройками.

Docker – платформа контейнеризации, позволяющая собирать и запускать приложение в контейнерах.

Образ контейнера – шаблон, содержащий приложение и его зависимости, на основе которого создаются контейнеры.

Контейнер – запущенный экземпляр образа, изолированная среда выполнения приложения.

PostgreSQL – реляционная система управления базами данных для хранения и обработки структурированных данных.

Реляционная база данных – база данных, где информация хранится в таблицах со связями между ними.

Модульное тестирование (unit-тестирование) – проверка отдельных функций/классов в изоляции с заранее заданными входными данными и ожидаемым результатом.

Функциональное тестирование – проверка выполнения пользовательских функций и сценариев приложения на уровне интерфейса и результата работы.

Mock (подмена зависимости) – тестовый объект, имитирующий поведение внешней зависимости для изоляции тестируемого кода.

Алгоритм Дейкстры – алгоритм поиска кратчайшего пути в графе с неотрицательными весами ребер.

Алгоритм A^* – алгоритм поиска кратчайшего пути, использующий эвристику для ускорения поиска при сохранении оптимальности.

Запрещённая зона – область на плане, по которой прокладка маршрута недопустима и которая исключается из поиска.

Целевая функция – правило вычисления итоговой стоимости решения, по которому выбирается оптимальный вариант.

Приложение Б
(рекомендуемое)
Результат тестирования клиентской части

Строительный помощник

Вход

Войдите, чтобы управлять проектами и расчетами.

Email

Пароль

Неверный логин или пароль

Войти

[Нет аккаунта? Регистрация](#)

Рисунок Б.1 – Попытка входа с неверным паролем

Строительный помощник

Регистрация

Войдите, чтобы управлять проектами и расчетами.

Email

Пароль

Роль

Создатель

Наблюдатель

Наблюдатель не может создавать проекты — только добавлять по коду и оставлять комментарии.

Этот email уже используется

Создать аккаунт

[Уже есть аккаунт? Войти](#)

Рисунок Б.2 – Регистрации с уже занятой почтой

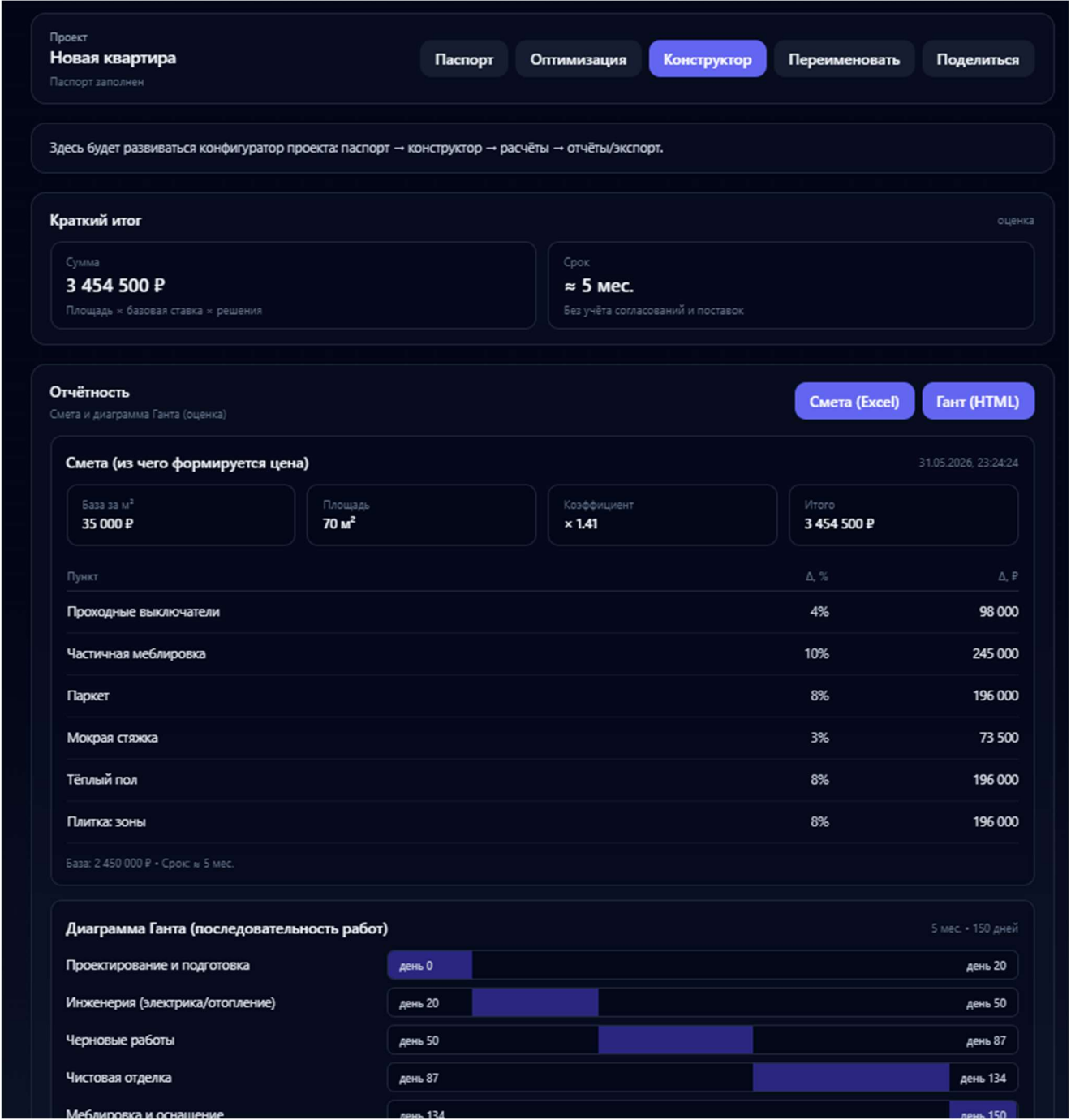


Рисунок Б.3 – Открытие страницы проекта

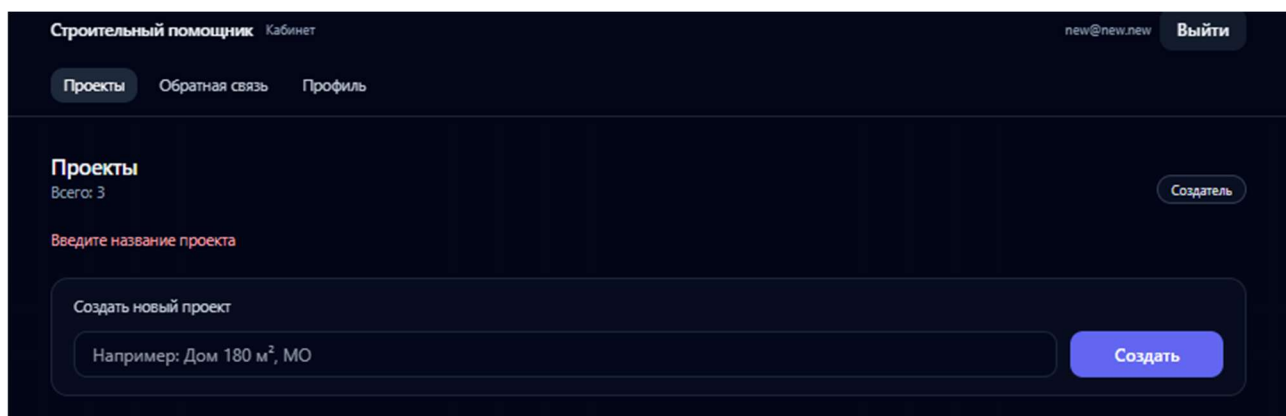


Рисунок Б.4 – Создание проекта с пустым названием

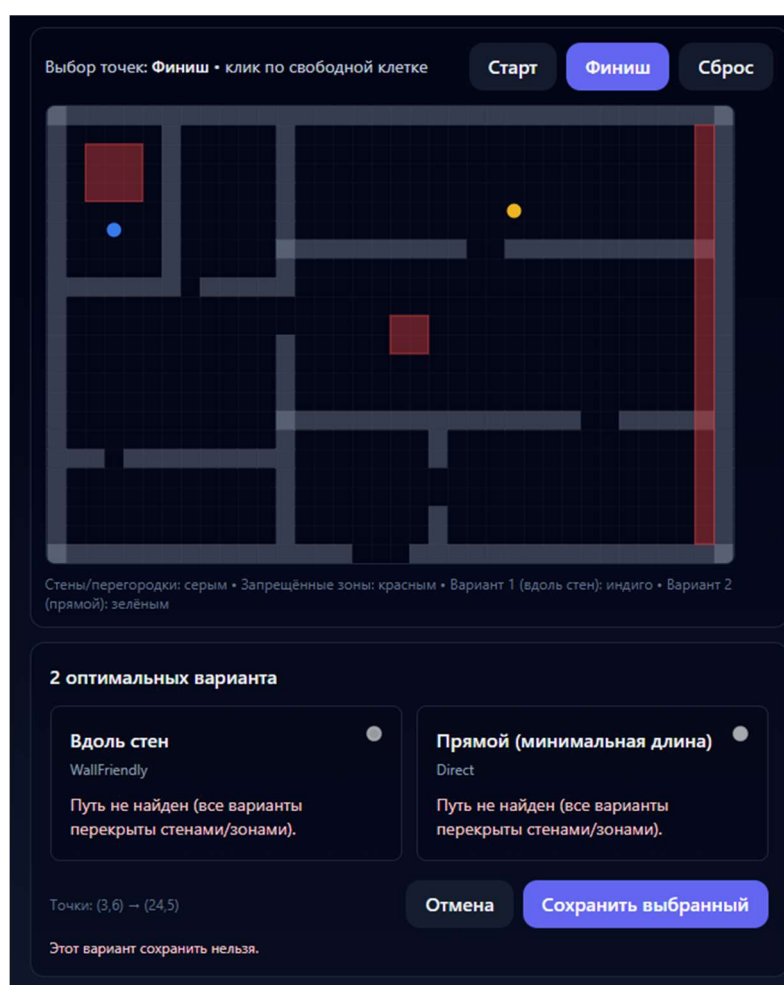


Рисунок Б.5 – Попытка оптимизации с точкой старта в замкнутом пространстве



Рисунок Б.6 – Предпросмотр оптимизации при корректных точках старта/финиша

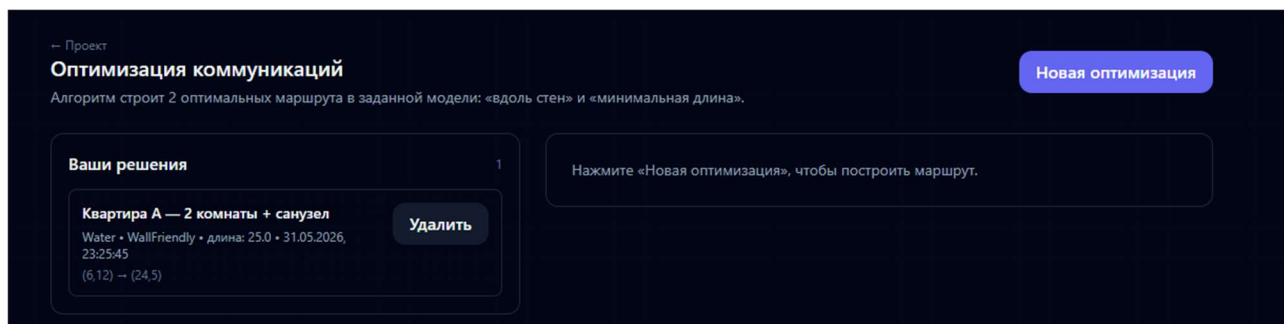


Рисунок Б.7 – Создание оптимизации с выбором варианта

Рисунок Б.8 – Скачивание сметы проекта

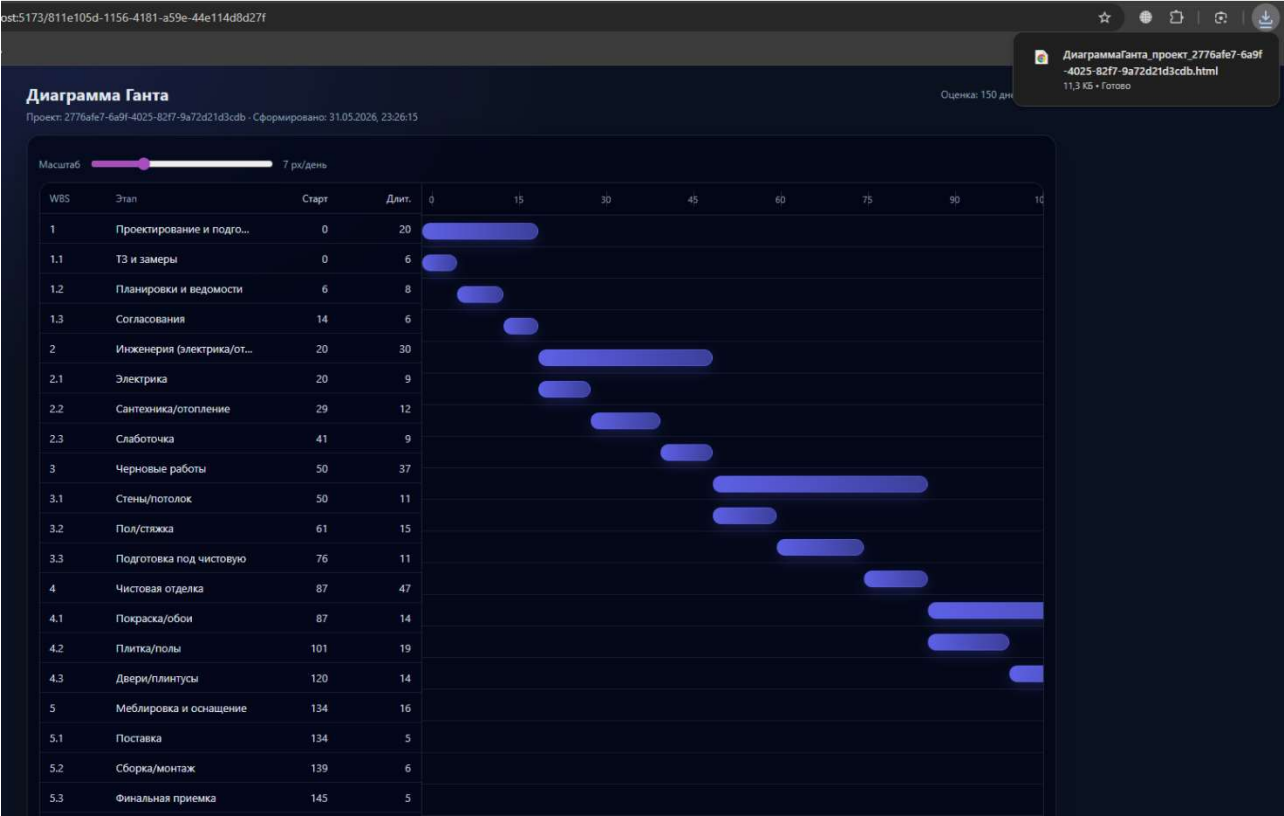


Рисунок Б.9 – Скачивание диаграммы Ганта

Приложение В
(обязательное)
Листинг клиентского приложения

Листинг В.1 – Файл main.tsx

```
import React from "react";
import ReactDOM from "react-dom/client";
import { BrowserRouter } from "react-router-dom";
import { App } from "../routes/App";
import { AuthProvider } from "../state/auth";
import "../styles.css";

ReactDOM.createRoot(document.getElementById("root")!).render(
  <BrowserRouter>
    <AuthProvider>
      <App />
    </AuthProvider>
  </BrowserRouter>
);
```

Листинг В.2 – Файл auth.tsx

```
import React, { createContext, useContext, useMemo, useState }
from "react";

export type UserRole = "Creator" | "Viewer" | "Admin";

export type AuthUser = {
  id: string;
  email: string;
  role: UserRole;
};

type AuthState = {
  accessToken: string | null;
  user: AuthUser | null;
  isAuthenticated: boolean;
  setAuth: (token: string, user: AuthUser) => void;
  logout: () => void;
};

const AuthContext = createContext<AuthState | null>(null);

const STORAGE_KEY = "ca_auth_v1";

type Stored = { accessToken: string; user: AuthUser };

function loadStored(): Stored | null {
  try {
    const raw = localStorage.getItem(STORAGE_KEY);
    if (!raw) return null;
    return JSON.parse(raw) as Stored;
  } catch {
    return null;
  }
}
```

```

export function AuthProvider({ children }: { children:
React.ReactNode }) {
  const stored = loadStored();
  const [accessToken, setAccessToken] = useState<string |
null>(stored?.accessToken ?? null);
  const [user, setUser] = useState<AuthUser | null>(stored?.user
?? null);

  const value = useMemo<AuthState>(
    () => ({
      accessToken,
      user,
      isAuthenticated: Boolean(accessToken && user),
      setAuth: (token, u) => {
        setAccessToken(token);
        setUser(u);
        localStorage.setItem(STORAGE_KEY, JSON.stringify({
accessToken: token, user: u } satisfies Stored));
      },
      logout: () => {
        setAccessToken(null);
        setUser(null);
        localStorage.removeItem(STORAGE_KEY);
      }
    })),
    [accessToken, user]
  );

  return <AuthContext.Provider
value={value}>{children}</AuthContext.Provider>;
}

export function useAuth() {
  const ctx = useContext(AuthContext);
  if (!ctx) throw new Error("AuthProvider missing");
  return ctx;
}

```

Листинг В.3 – Файл project.tsx

```

import React, { createContext, useContext, useMemo, useRef,
useState } from "react";
import { getOwnedProjects, getViewerProjects, ProjectListItem }
from "../api/projectApi";
import { AuthUser } from "../auth";

type ProjectsState = {
  owned: ProjectListItem[];
  viewer: ProjectListItem[];
  loading: boolean;
  error: string | null;
  refresh: () => Promise<void>;
}

```

```

};

const ProjectsContext = createContext<ProjectsState | null>(null);

export function ProjectsProvider({ user, children }: { user:
AuthUser | null; children: React.ReactNode }) {
  const [owned, setOwned] = useState<ProjectListItem[]>([]);
  const [viewer, setViewer] = useState<ProjectListItem[]>([]);
  const [loading, setLoading] = useState(false);
  const [error, setError] = useState<string | null>(null);

  const inFlight = useRef<Promise<void> | null>(null);

  const refresh = useMemo(() => {
    return async () => {
      if (!user) return;
      if (inFlight.current) return inFlight.current;

      setLoading(true);
      setError(null);

      const p = (async () => {
        try {
          if (user.role === "Creator") {
            const o = await getOwnedProjects(user.id);
            setOwned(o);
            setViewer([]);
            return;
          }

          if (user.role === "Viewer") {
            const v = await getViewerProjects(user.id);
            setOwned([]);
            setViewer(v);
            return;
          }

          // Admin: can see both sets (useful for testing)
          const [o, v] = await
Promise.all([getOwnedProjects(user.id).catch(() => []),
getViewerProjects(user.id).catch(() => [])]);
          setOwned(o);
          setViewer(v);
        } catch (e: any) {
          setError(e?.response?.data?.message ?? "Не удалось
загрузить проекты");
        } finally {
          setLoading(false);
          inFlight.current = null;
        }
      })();
    }
  })();
}

```

```

        inFlight.current = p;
        return p;
    };
}, [user]));

const value = useMemo(
    () => ({
        owned,
        viewer,
        loading,
        error,
        refresh
    })),
    [owned, viewer, loading, error, refresh]
);

return <ProjectsContext.Provider
value={value}>{children}</ProjectsContext.Provider>;
}

export function useProjects() {
    const ctx = useContext(ProjectsContext);
    if (!ctx) throw new Error("ProjectsProvider missing");
    return ctx;
}

```

Листинг В.4 – Файл App.tsx

```

import { Navigate, Route, Routes } from "react-router-dom";
import { LoginPage } from "../LoginPage";
import { RegisterPage } from "../RegisterPage";
import { DashboardLayout } from "../DashboardLayout";
import { ProjectsPage } from "../ProjectsPage";
import { FeedbackPage } from "../FeedbackPage";
import { ProfilePage } from "../ProfilePage";
import { ProjectPage } from "../ProjectPage";
import { PassportPage } from "../PassportPage";
import { ProjectBuilderPage } from "../ProjectBuilderPage";
import { ProjectOptimizationPage } from
"./ProjectOptimizationPage";
import { useAuth } from "../state/auth";

function NonAdminOnly({ children }: { children: React.ReactNode })
{
    const { user } = useAuth();
    if (user?.role === "Admin") return <Navigate to="/app/feedback"
replace />;
    return <>{children}</>;
}

export function App() {
    const { isAuthenticated, user } = useAuth();

```

```

return (
  <Routes>
    <Route
      path="/"
      element={<Navigate to={isAuthenticated ? (user?.role ===
"Admin" ? "/app/feedback" : "/app/projects") : "/login"} replace
/>}
    />
    <Route path="/login" element={<LoginPage />} />
    <Route path="/register" element={<RegisterPage />} />

    <Route path="/app" element={isAuthenticated ?
<DashboardLayout /> : <Navigate to="/login" replace />}>
      <Route
        path="projects"
        element={
          <NonAdminOnly>
            <ProjectsPage />
          </NonAdminOnly>
        }
      />
      <Route
        path="projects/:projectId"
        element={
          <NonAdminOnly>
            <ProjectPage />
          </NonAdminOnly>
        }
      />
      <Route
        path="projects/:projectId/passport"
        element={
          <NonAdminOnly>
            <PassportPage />
          </NonAdminOnly>
        }
      />
      <Route
        path="projects/:projectId/builder"
        element={
          <NonAdminOnly>
            <ProjectBuilderPage />
          </NonAdminOnly>
        }
      />
      <Route
        path="projects/:projectId/optimization"
        element={
          <NonAdminOnly>
            <ProjectOptimizationPage />
          </NonAdminOnly>
        }
      />
    </Route>
  </Routes>
)

```

```

    />
    <Route
      path="projects/:projectId/optimization/:optimizationId"
      element={
        <NonAdminOnly>
          <ProjectOptimizationPage />
        </NonAdminOnly>
      }
    />
    <Route path="feedback" element={<FeedbackPage />} />
    <Route path="profile" element={<ProfilePage />} />
  </Route>

  <Route path="*" element={<Navigate to="/" replace />} />
</Routes>
);
}

```

Листинг В.5 – Файл LoginPage.tsx

```

import React, { useState } from "react";
import { Link, useNavigate } from "react-router-dom";
import { login } from "../api/authApi";
import { useAuth } from "../state/auth";
import { AuthShell } from "../ui";
import { getRuErrorMessage } from "../utils/errors";
import { isValidEmail, isValidPassword } from
"../utils/validation";

export function LoginPage() {
  const navigate = useNavigate();
  const { setAuth } = useAuth();
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  const [error, setError] = useState<string | null>(null);
  const [loading, setLoading] = useState(false);

  const isAdminLogin = email.trim().toLowerCase() === "admin";

  const validationError =
    !email.trim()
      ? "Введите email"
      : !isAdminLogin && !isValidEmail(email)
        ? "Введите корректный email"
        : !password
          ? "Введите пароль"
          : !isAdminLogin && !isValidPassword(password)
            ? "Пароль должен быть минимум 6 символов"
            : null;

  async function onSubmit(e: React.FormEvent) {
    e.preventDefault();
    setError(null);
  }

```

```

    if (validationError) {
      setError(validationError);
      return;
    }
    setLoading(true);
    try {
      const token = await login({ email, password });
      setAuth(token.accessToken, token.user);
      navigate("/app/projects", { replace: true });
    } catch (err: any) {
      setError(getRuErrorMessage(err, "Не удалось войти"));
    } finally {
      setLoading(false);
    }
  }
}

return (
  <AuthShell title="Вход">
    <div className="card">
      <form onSubmit={onSubmit} className="space-y-4">
        <div>
          <div className="label mb-1">Email</div>
          <input className="input" value={email} onChange={ (e)
=> setEmail(e.target.value)} placeholder="you@company.com" />
        </div>
        <div>
          <div className="label mb-1">Пароль</div>
          <input className="input" value={password}
onChange={ (e) => setPassword(e.target.value)} type="password"
placeholder="....." />
        </div>
        {error && <div className="text-sm text-rose-
300">{error}</div>}
        <button className="btn-primary w-full"
disabled={loading}>
          {loading ? "Входим..." : "Войти"}
        </button>
        <div className="text-sm text-slate-400">
          Нет аккаунта?{" "}
          <Link to="/register" className="text-indigo-300
hover:text-indigo-200">
            Регистрация
          </Link>
        </div>
      </form>
    </div>
  </AuthShell>
);
}

```

Листинг В.6 – Файл ProjectPage.tsx

```
import React, { useEffect, useMemo, useState } from "react";
```

```

import { Link, useParams } from "react-router-dom";
import { getAnswers } from "../api/answersApi";
import { addComment, getComments, ViewerComment } from
"../api/commentsApi";
import { createShareCode, getProject, updateProject } from
"../api/projectApi";
import {
  downloadEstimateReportXlsx,
  downloadGanttReportHtml,
  getEstimateReport,
  getGanttReport,
  ProjectEstimateBreakdown,
  ProjectGantt
} from "../api/reportsApi";
import { useAuth } from "../state/auth";
import { getRuErrorMessage } from "../utils/errors";
import { getPassportFromAnswers, isPassportComplete } from
"../utils/passport";
import { builderMultiStepCodes, builderStepCodes, estimateProject,
parseMultiValues } from "../utils/builder";

export function ProjectPage() {
  const { projectId } = useParams();
  const { user } = useAuth();

  const [shareCode, setShareCode] = useState<string | null>(null);
  const [error, setError] = useState<string | null>(null);
  const [passportComplete, setPassportComplete] = useState<boolean
| null>(null);
  const [projectName, setProjectName] = useState<string>(projectId
?? "");
  const [renaming, setRenaming] = useState(false);
  const [nameDraft, setNameDraft] = useState("");
  const [savingName, setSavingName] = useState(false);
  const [comments, setComments] = useState<ViewerComment[]>([]);
  const [loadingComments, setLoadingComments] = useState(false);
  const [commentDraft, setCommentDraft] = useState("");
  const [estimate, setEstimate] = useState<{ totalRub: number;
months: number } | null>(null);

  const [reportsLoading, setReportsLoading] = useState(false);
  const [reportsError, setReportsError] = useState<string |
null>(null);
  const [estimateReport, setEstimateReport] =
useState<ProjectEstimateBreakdown | null>(null);
  const [ganttReport, setGanttReport] = useState<ProjectGantt |
null>(null);
  const [downloading, setDownloading] = useState<"estimate" |
"ganttHtml" | null>(null);

  useEffect(() => {
    setShareCode(null);
  
```

```

        setError(null);
        setRenaming(false);
        setNameDraft("");
    }, [projectId]);

    useEffect(() => {
        if (!projectId) return;
        let cancelled = false;

        (async () => {
            try {
                const [p, answers] = await
Promise.all([getProject(projectId), getAnswers(projectId)]);
                if (cancelled) return;
                setProjectName(p.name);
                const passport = getPassportFromAnswers(answers);
                const complete = isPassportComplete(passport);
                setPassportComplete(complete);

                const selectedSingle: Record<string, string | null> = {};
                for (const code of Object.values(builderStepCodes)) {
                    const a = answers.find((x) => x.stepCode === code);
                    selectedSingle[code] = a?.selectedOptionCode ?? null;
                }
                const selectedMulti: Record<string, string[]> = {};
                for (const code of Object.values(builderMultiStepCodes)) {
                    const a = answers.find((x) => x.stepCode === code);
                    selectedMulti[code] = parseMultiValues(a?.valueJson);
                }
                const est = estimateProject(passport.buildingType,
passport.areaM2, selectedSingle, selectedMulti);
                setEstimate(est.totalRub > 0 ? est : null);

                setReportsError(null);
                setReportsLoading(true);
                try {
                    const [estimateData, ganttData] = await
Promise.all([getEstimateReport(projectId),
getGanttReport(projectId)]);
                    if (!cancelled) {
                        setEstimateReport(estimateData);
                        setGanttReport(ganttData);
                    }
                } catch (e: any) {
                    if (!cancelled) {
                        setEstimateReport(null);
                        setGanttReport(null);
                        setReportsError(getRuErrorMessage(e, "Не удалось
загрузить отчёты"));
                    }
                }
            } finally {
                if (!cancelled) setReportsLoading(false);
            }
        })();
    }, [projectId]);

```

```

    }
  } catch {
    if (!cancelled) setPassportComplete(null);
  }
})();

return () => {
  cancelled = true;
};
}, [projectId]);

useEffect(() => {
  if (!projectId) return;
  let cancelled = false;

  (async () => {
    setLoadingComments(true);
    try {
      const data = await getComments(projectId);
      if (!cancelled) setComments(data.slice().sort((a, b) =>
new Date(a.commentDate).getTime() - new
Date(b.commentDate).getTime()));
    } catch {
      // ignore
    } finally {
      if (!cancelled) setLoadingComments(false);
    }
  })();

  return () => {
    cancelled = true;
  };
}, [projectId]);

const passportHint = useMemo(() => {
  if (passportComplete === true) return "Паспорт заполнен";
  if (passportComplete === false) return "Заполните паспорт,
чтобы начать конструктор";
  return null;
}, [passportComplete]);

function toRuTitle(title: string) {
  if (title.trim().toLowerCase() === "lvt") return "LVT
(кварцвинил)";
  return title;
}

async function saveBlob(bytes: ArrayBuffer, fileName: string,
contentType?: string) {
  const blob = new Blob([bytes], {
    type: contentType || "application/vnd.openxmlformats-
officedocument.spreadsheetml.sheet"
  });

```

```

    });
    const url = URL.createObjectURL(blob);
    try {
        const a = document.createElement("a");
        a.href = url;
        a.download = fileName;
        document.body.appendChild(a);
        a.click();
        a.remove();
    } finally {
        URL.revokeObjectURL(url);
    }
}

function openBlobInNewTab(bytes: ArrayBuffer, contentType:
string | undefined) {
    const blob = new Blob([bytes], { type: contentType ||
"text/html; charset=utf-8" });
    const url = URL.createObjectURL(blob);
    const win = window.open(url, "_blank", "noopener,noreferrer");
    // On some browsers window.open can be blocked; keep url so
user still downloaded the file.
    if (!win) {
        setTimeout(() => URL.revokeObjectURL(url), 60_000);
        return;
    }
    setTimeout(() => URL.revokeObjectURL(url), 60_000);
}

async function onDownloadEstimate() {
    if (!projectId) return;
    setReportsError(null);
    setDownloading("estimate");
    try {
        const { bytes, contentType } = await
downloadEstimateReportXlsx(projectId);
        await saveBlob(bytes, `Смета_проект_${projectId}.xlsx`,
contentType);
    } catch (e: any) {
        setReportsError(getRuErrorMessage(e, "Не удалось скачать
смету"));
    } finally {
        setDownloading(null);
    }
}

async function onDownloadGanttHtml() {
    if (!projectId) return;
    setReportsError(null);
    setDownloading("ganttHtml");
    try {

```

```

        const { bytes, contentType } = await
downloadGanttReportHtml(projectId);
        // 1) Скачивание файла
        await saveBlob(bytes,
`Диаграмма Ганта_проект_${projectId}.html`, contentType);
        // 2) Быстрый просмотр в новой вкладке
        openBlobInNewTab(bytes, contentType);
    } catch (e: any) {
        setReportsError(getRuErrorMessage(e, "Не удалось скачать
диаграмму Ганта (HTML)"));
    } finally {
        setDownloading(null);
    }
}

async function onShare() {
    if (!projectId) return;
    setError(null);
    try {
        const data = await createShareCode(projectId);
        setShareCode(data.code);
    } catch (e: any) {
        setError(getRuErrorMessage(e, "Не удалось сгенерировать
код"));
    }
}

async function onSaveName() {
    if (!projectId) return;
    const next = nameDraft.trim();
    if (!next) {
        setError("Введите название проекта");
        return;
    }
    setError(null);
    setSavingName(true);
    try {
        const updated = await updateProject(projectId, { name: next
});
        setProjectName(updated.name);
        setRenaming(false);
        setNameDraft("");
    } catch (e: any) {
        setError(getRuErrorMessage(e, "Не удалось переименовать
проект"));
    } finally {
        setSavingName(false);
    }
}

async function onSendComment() {
    if (!projectId) return;

```

```

    if (!user) return;
    const text = commentDraft.trim();
    if (!text) return;
    setError(null);
    try {
      await addComment(projectId, { userId: user.id, email:
user.email, commentText: text });
      setCommentDraft("");
      const data = await getComments(projectId);
      setComments(data.slice().sort((a, b) => new
Date(a.commentDate).getTime() - new
Date(b.commentDate).getTime()));
    } catch (e: any) {
      setError(getRuErrorMessage(e, "Не удалось отправить
сообщение"));
    }
  }

  return (
    <div className="space-y-5">
      <div className="card">
        <div className="flex items-center justify-between gap-3">
          <div>
            <div className="text-xs text-slate-400">Проект</div>
            <div className="text-lg font-semibold">{projectName ||
projectId}</div>
            {passportHint && <div className="mt-1 text-xs text-
slate-500">{passportHint}</div>}
          </div>
          <div className="flex items-center gap-2">
            <Link className="btn-ghost"
to={` /app/projects/${projectId}/passport`} >
              Паспорт
            </Link>
            <Link className="btn-ghost"
to={` /app/projects/${projectId}/optimization`} >
              Оптимизация
            </Link>
            <Link className="btn-primary"
to={` /app/projects/${projectId}/builder`} >
              Конструктор
            </Link>
            {(user?.role === "Creator" || user?.role === "Admin")
&& (
              <button
                className="btn-ghost"
                onClick={() => {
                  setError(null);
                  setRenaming((v) => {
                    const next = !v;
                    if (next) setNameDraft(projectName);
                    return next;

```

```

        });
    }}
    disabled={savingName}
  >
    Переименовать
  </button>
)}
{user?.role === "Creator" && (
  <button className="btn-ghost" onClick={onShare}>
    Поделиться
  </button>
)}
</div>
</div>
{error && <div className="mt-3 text-sm text-rose-300">{error}</div>}
{renaming && (
  <div className="mt-4 rounded-xl border border-slate-800 bg-slate-950/60 px-4 py-3">
    <div className="text-xs text-slate-400">Название проекта</div>
    <div className="mt-2 flex flex-col sm:flex-row gap-3">
      <input
        className="input"
        value={nameDraft}
        onChange={(e) => setNameDraft(e.target.value)}
        placeholder="Например: Дом 180 м², МО"
        onKeyDown={(e) => {
          if (e.key === "Enter") onSaveName();
        }}
        disabled={savingName}
      />
      <div className="flex gap-2 sm:w-56">
        <button className="btn-primary flex-1"
          onClick={onSaveName} disabled={savingName}>
          Сохранить
        </button>
        <button
          className="btn-ghost flex-1"
          onClick={() => {
            setRenaming(false);
            setNameDraft("");
            setError(null);
          }}
          disabled={savingName}
        >
          Отмена
        </button>
      </div>
    </div>
  </div>
)}
) }

```

```

        {shareCode && (
            <div className="mt-4 rounded-xl border border-slate-800
bg-slate-950/60 px-4 py-3">
                <div className="text-xs text-slate-400">Код
доступа</div>
                <div className="mt-1 font-mono text-lg tracking-
widest">{shareCode}</div>
                <div className="mt-2 text-xs text-slate-500">Передайте
этот код наблюдателю, чтобы он добавил проект в свой
кабинет.</div>
            </div>
        )}
    </div>

    <div className="card">
        <div className="text-sm text-slate-300">Здесь будет
развиваться конфигуратор проекта: паспорт → конструктор → расчёты
→ отчёты/экспорт.</div>
    </div>

    {estimate && (
        <div className="card">
            <div className="flex items-center justify-between">
                <div className="font-medium">Краткий итог</div>
                <div className="text-xs text-slate-500">оценка</div>
            </div>
            <div className="mt-3 grid grid-cols-1 md:grid-cols-2
gap-3">
                <div className="rounded-xl border border-slate-800 bg-
slate-950/50 px-4 py-3">
                    <div className="text-xs text-slate-500">Сумма</div>
                    <div className="mt-1 text-xl font-
semibold">{estimate.totalRub.toLocaleString("ru-RU")} ₽</div>
                    <div className="mt-1 text-xs text-slate-500">Площадь
× базовая ставка × решения</div>
                </div>
                <div className="rounded-xl border border-slate-800 bg-
slate-950/50 px-4 py-3">
                    <div className="text-xs text-slate-500">Срок</div>
                    <div className="mt-1 text-xl font-semibold">≈
{estimate.months} мес.</div>
                    <div className="mt-1 text-xs text-slate-500">Без
учёта согласований и поставок</div>
                </div>
            </div>
        </div>
    )}

    <div className="card">
        <div className="flex items-center justify-between gap-3">
            <div>
                <div className="font-medium">Отчётность</div>

```

```

        <div className="text-xs text-slate-500 mt-1">Смета и
диаграмма Ганта (оценка)</div>
    </div>
    <div className="flex items-center gap-2">
        <button className="btn-primary"
onClick={onDownloadEstimate} disabled={!estimateReport ||
downloading !== null}>
            {downloading === "estimate" ? "Скачивание..." : "Смета
(Excel)"}
        </button>
        <button className="btn-primary"
onClick={onDownloadGanttHtml} disabled={!ganttReport ||
downloading !== null}>
            {downloading === "ganttHtml" ? "Скачивание..." : "Гант
(HTML)"}
        </button>
    </div>
    </div>

    {reportsLoading && <div className="mt-3 text-sm text-
slate-400">Загрузка отчётов...</div>}
    {reportsError && <div className="mt-3 text-sm text-rose-
300">{reportsError}</div>}

    {estimateReport && (
        <div className="mt-4 rounded-xl border border-slate-800
bg-slate-950/50 p-4">
            <div className="flex items-center justify-between">
                <div className="font-medium">Смета (из чего
формируется цена)</div>
                <div className="text-xs text-slate-500">{new
Date(estimateReport.generatedAt).toLocaleString()}</div>
            </div>
            <div className="mt-3 grid grid-cols-1 md:grid-cols-4
gap-3 text-sm">
                <div className="rounded-xl border border-slate-800
bg-slate-950/60 px-4 py-3">
                    <div className="text-xs text-slate-500">База за
м²</div>
                    <div className="mt-1 font-
semibold">{estimateReport.basePerM2Rub.toLocaleString("ru-RU")}
₽</div>
                </div>
                <div className="rounded-xl border border-slate-800
bg-slate-950/60 px-4 py-3">
                    <div className="text-xs text-slate-
500">Площадь</div>
                    <div className="mt-1 font-
semibold">{estimateReport.areaM2 ? `${estimateReport.areaM2} м²` :
""}</div>
                </div>
            </div>
        </div>
    )

```

```

        <div className="rounded-xl border border-slate-800
bg-slate-950/60 px-4 py-3">
            <div className="text-xs text-slate-
500">Коэффициент</div>
            <div className="mt-1 font-semibold">×
{estimateReport.factor.toFixed(2)}</div>
        </div>
        <div className="rounded-xl border border-slate-800
bg-slate-950/60 px-4 py-3">
            <div className="text-xs text-slate-
500">Итого</div>
            <div className="mt-1 font-
semibold">{estimateReport.totalRub.toLocaleString("ru-RU")}
₽</div>
        </div>
    </div>

    <div className="mt-4 overflow-auto">
        <table className="min-w-[720px] w-full text-sm">
            <thead className="text-xs text-slate-500">
                <tr className="border-b border-slate-800">
                    <th className="py-2 pr-3 text-left font-
medium">Пункт</th>
                    <th className="py-2 pr-3 text-right font-
medium">Δ, %</th>
                    <th className="py-2 pr-0 text-right font-
medium">Δ, ₽</th>
                </tr>
            </thead>
            <tbody>
                {estimateReport.items.length === 0 ? (
                    <tr className="border-b border-slate-900">
                        <td className="py-3 text-slate-400"
colSpan={3}>
                            Нет влияющих факторов (используется
базовая ставка).
                        </td>
                    </tr>
                ) : (
                    estimateReport.items.map((it) => (
                        <tr key={it.code} className="border-b
border-slate-900">
                            <td className="py-3 pr-3 text-slate-
200">{toRuTitle(it.title)}</td>
                            <td className="py-3 pr-3 text-right text-
slate-300">{(it.percentDelta * 100).toFixed(0)}%</td>
                            <td className="py-3 pr-0 text-right text-
slate-200">{it.deltaRub.toLocaleString("ru-RU")}</td>
                        </tr>
                    ))
                )}
            </tbody>
        </table>
    </div>

```

```

        </table>
      </div>

      <div className="mt-3 text-xs text-slate-500">
        База:
        {estimateReport.baseCostRub.toLocaleString("ru-RU")} ₽ • Срок: ≈
        {estimateReport.months} мес.
      </div>
    </div>
  )}

  {gantttReport && (
    <div className="mt-4 rounded-xl border border-slate-800
bg-slate-950/50 p-4">
      <div className="flex items-center justify-between">
        <div className="font-medium">Диаграмма Ганта
(последовательность работ)</div>
        <div className="text-xs text-slate-500">
          {gantttReport.totalMonths} мес. •
          {gantttReport.totalDays} дней
        </div>
      </div>
      <div className="mt-3 space-y-2">
        {gantttReport.tasks.map((t) => {
          const total = gantttReport.totalDays || 1;
          const left = Math.max(0, Math.min(100, (t.startDay
/ total) * 100));
          const width = Math.max(1, Math.min(100 - left,
((t.endDay - t.startDay) / total) * 100));
          return (
            <div key={t.id} className="grid grid-cols-12
gap-3 items-center">
              <div className="col-span-12 md:col-span-4
text-sm text-slate-200">{t.title}</div>
              <div className="col-span-12 md:col-span-8">
                <div className="h-8 rounded-lg border
border-slate-800 bg-slate-950/60 relative overflow-hidden">
                  <div className="absolute inset-y-0 bg-
indigo-600/50" style={{ left: `${left}%`, width: `${width}%` }} />
                  <div className="absolute inset-0 flex
items-center justify-between px-3 text-xs text-slate-300">
                    <span>день {t.startDay}</span>
                    <span>день {t.endDay}</span>
                  </div>
                </div>
              </div>
            </div>
          );
        })}
      </div>
    </div>
  )}

```

```

</div>

<div className="card">
  <div className="flex items-center justify-between">
    <div className="font-medium">Чат по проекту</div>
    <div className="text-xs text-slate-500">{loadingComments
? "Загрузка..." : `${comments.length}`}</div>
  </div>
  <div className="mt-3 space-y-2 max-h-72 overflow-auto pr-
1">
    {comments.length === 0 ? (
      <div className="text-sm text-slate-400">Пока нет
сообщений.</div>
    ) : (
      comments.map((c) => {
        const mine = c.userId === user?.id;
        return (
          <div key={c.id} className={mine ? "flex justify-
end" : "flex justify-start"}>
            <div className="max-w-[80%] rounded-xl border
border-slate-800 bg-slate-950/50 px-4 py-3">
              <div className="text-xs text-slate-500 flex
items-center justify-between gap-3">
                <span>{mine ? "Вы" : c.email}</span>
                <span>{new
Date(c.commentDate).toLocaleString()}</span>
              </div>
              <div className="mt-1 text-sm text-slate-200
whitespace-pre-wrap">{c.commentText}</div>
            </div>
          </div>
        );
      })
    )}
  </div>
  <div className="mt-3 flex flex-col sm:flex-row gap-3">
    <input
      className="input"
      value={commentDraft}
      onChange={(e) => setCommentDraft(e.target.value)}
      placeholder="Напишите сообщение..."
      onKeyDown={(e) => {
        if (e.key === "Enter") onSendComment();
      }}
    />
    <button className="btn-primary sm:w-40"
onClick={onSendComment} disabled={!commentDraft.trim()}>
      Отправить
    </button>
  </div>
</div>
</div>

```

```
    );  
}
```

Приложение Г
(обязательное)
Листинг серверного приложения

Листинг Г.1 – Файл AuthApplicationService.cs

```

using System.Security.Cryptography;
using System.Text;
using AuthService.Application.Contracts;
using AuthService.Application.Exceptions;
using AuthService.Application.Interfaces;
using Microsoft.Extensions.Logging;

namespace AuthService.Application.Services;

public class AuthApplicationService : IAuthApplicationService
{
    private readonly IUserServiceClient _userServiceClient;
    private readonly IJwtService _jwtService;
    private readonly ILogger<AuthApplicationService> _logger;

    public AuthApplicationService(
        IUserServiceClient userServiceClient,
        IJwtService jwtService,
        ILogger<AuthApplicationService> logger)
    {
        _userServiceClient = userServiceClient;
        _jwtService = jwtService;
        _logger = logger;
    }

    public async Task<TokenResponse> CreateTokenAsync(TokenRequest request)
    {
        var normalizedEmail = NormalizeEmail(request.Email);

        var user = await
        _userServiceClient.GetByEmailAsync(normalizedEmail);
        if (user == null)
        {
            _logger.LogWarning("Token request failed: user not
found - {Email}", normalizedEmail);
            throw new BadRequestException("Invalid credentials");
        }

        if (!VerifyPassword(request.Password, user.PasswordHash))
        {
            _logger.LogWarning("Token request failed: invalid
password - {Email}", normalizedEmail);
            throw new BadRequestException("Invalid credentials");
        }

        var accessToken = _jwtService.GenerateAccessToken(user.Id,
user.Email, user.Role);
        var refreshToken = _jwtService.GenerateRefreshToken();

        return new TokenResponse

```

```

        {
            AccessToken = accessToken,
            RefreshToken = refreshToken,
            User = new UserResponse
            {
                Id = user.Id,
                Email = user.Email,
                Role = user.Role
            }
        };
    }

    private static string NormalizeEmail(string email) =>
        email.Trim().ToLowerInvariant();

    private static bool VerifyPassword(string password, string
        hash) => HashPassword(password) == hash;

    private static string HashPassword(string password)
    {
        using var sha256 = SHA256.Create();
        var hashedBytes =
            sha256.ComputeHash(Encoding.UTF8.GetBytes(password));
        return Convert.ToBase64String(hashedBytes);
    }
}

```

Листинг Г.2 – Файл UserService.cs

```

using System.Security.Cryptography;
using System.Text;
using Microsoft.Extensions.Logging;
using UserService.Application.Contracts;
using UserService.Application.Exceptions;
using UserService.Application.Interfaces;
using UserService.Application.Models;

namespace UserService.Application.Services;

public class UserApplicationService : IUserApplicationService
{
    private readonly IUserRepository _userRepository;
    private readonly ILogger<UserApplicationService> _logger;

    public UserApplicationService(IUserRepository userRepository,
        ILogger<UserApplicationService> logger)
    {
        _userRepository = userRepository;
        _logger = logger;
    }

    public async Task<UserResponse> RegisterAsync(RegisterRequest
        request)
    {

```

```

    {
        if (request.Role == UserRole.Admin)
        {
            throw new
BadRequestException(RegistrationError.ADMIN_REGISTRATION_ATTEMPT.T
oString());
        }

        var normalizedEmail = NormalizeEmail(request.Email);
        if (await
_userRepository.EmailExistsAsync(normalizedEmail))
        {
            throw new
BadRequestException(RegistrationError.EMAIL_ALREADY_USED.ToString(
));
        }

        var user = new User
        {
            Id = Guid.NewGuid(),
            Email = normalizedEmail,
            PasswordHash = HashPassword(request.Password),
            RegistrationDate = DateTime.UtcNow,
            Role = request.Role
        };

        await _userRepository.CreateAsync(user);
        _logger.LogInformation("User registered - {UserId}
{Email}", user.Id, user.Email);

        return new UserResponse
        {
            Id = user.Id.ToString(),
            Email = user.Email,
            Role = user.Role
        };
    }

    public async Task<InternalUserByEmailResponse?>
GetByEmailInternalAsync(string email)
    {
        var normalizedEmail = NormalizeEmail(email);
        var user = await
_userRepository.GetByEmailAsync(normalizedEmail);
        if (user == null)
        {
            return null;
        }

        return new InternalUserByEmailResponse
        {
            Id = user.Id.ToString(),

```

```

        Email = user.Email,
        PasswordHash = user.PasswordHash,
        Role = user.Role
    };
}

public async Task<InternalUserResponse?>
GetByIdInternalAsync(Guid id)
{
    var user = await _userRepository.GetByIdAsync(id);
    if (user == null)
    {
        return null;
    }

    return new InternalUserResponse
    {
        Id = user.Id.ToString(),
        Email = user.Email,
        RegistrationDate =
DateTime.SpecifyKind(user.RegistrationDate.AddHours(3),
DateTimeKind.Utc),
        Role = user.Role
    };
}

private static string NormalizeEmail(string email) =>
email.Trim().ToLowerInvariant();

private static string HashPassword(string password)
{
    using var sha256 = SHA256.Create();
    var hashedBytes =
sha256.ComputeHash(Encoding.UTF8.GetBytes(password));
    return Convert.ToBase64String(hashedBytes);
}
}

```

Листинг Г.3 – Файл ProjectApplicationService.cs

```

using Microsoft.Extensions.Logging;
using ProjectCalculationService.Application.Contracts;
using ProjectCalculationService.Application.Exceptions;
using ProjectCalculationService.Application.Interfaces;
using ProjectCalculationService.Application.Models;

namespace ProjectCalculationService.Application.Services;

public class ProjectCalculationApplicationService :
IProjectCalculationApplicationService
{
    private readonly IProjectRepository _projectRepository;

```

```

        private readonly IProjectStepAnswerRepository
        _answerRepository;
        private readonly ISharingRepository _sharingRepository;
        private readonly IViewerCommentRepository _commentRepository;
        private readonly ILogger<ProjectCalculationApplicationService>
        _logger;

        public ProjectCalculationApplicationService(
            IProjectRepository projectRepository,
            IProjectStepAnswerRepository answerRepository,
            ISharingRepository sharingRepository,
            IViewerCommentRepository commentRepository,
            ILogger<ProjectCalculationApplicationService> logger)
        {
            _projectRepository = projectRepository;
            _answerRepository = answerRepository;
            _sharingRepository = sharingRepository;
            _commentRepository = commentRepository;
            _logger = logger;
        }

        public async Task<ProjectResponse>
        CreateProjectAsync(CreateProjectRequest request)
        {
            if (!Guid.TryParse(request.OwnerUserId, out var
            ownerUserId))
            {
                throw new BadRequestException("Invalid OwnerUserId");
            }

            if (string.IsNullOrEmpty(request.Name))
            {
                throw new BadRequestException("Project name is
            required");
            }

            var now = DateTime.UtcNow;
            var project = new Project
            {
                Id = Guid.NewGuid(),
                OwnerUserId = ownerUserId,
                Name = request.Name.Trim(),
                Status = "Draft",
                CreatedAt = now,
                UpdatedAt = now
            };

            await _projectRepository.CreateAsync(project);
            _logger.LogInformation("Project created - {ProjectId}
            Owner {OwnerUserId}", project.Id, project.OwnerUserId);

            return MapProject(project);
        }

```

```

    }

    public async Task<IReadOnlyList<ProjectListItemResponse>>
    GetOwnedProjectsAsync(string ownerId)
    {
        if (!Guid.TryParse(ownerId, out var ownerId))
        {
            throw new BadRequestException("Invalid ownerId");
        }

        var projects = await
        _projectRepository.GetByOwnerIdAsync(ownerId);
        return projects
            .OrderByDescending(p => p.UpdatedAt)
            .Select(MapProjectListItem)
            .ToList();
    }

    public async Task<IReadOnlyList<ProjectListItemResponse>>
    GetViewerProjectsAsync(string userId)
    {
        if (!Guid.TryParse(userId, out var viewerUserId))
        {
            throw new BadRequestException("Invalid userId");
        }

        var projects = await
        _sharingRepository.GetViewerProjectsAsync(viewerUserId);
        return projects
            .OrderByDescending(p => p.UpdatedAt)
            .Select(MapProjectListItem)
            .ToList();
    }

    public async Task<ProjectResponse> GetProjectAsync(string
    projectId)
    {
        if (!Guid.TryParse(projectId, out var id))
        {
            throw new BadRequestException("Invalid projectId");
        }

        var project = await _projectRepository.GetByIdAsync(id);
        if (project == null)
        {
            throw new NotFoundException("Project not found");
        }

        return MapProject(project);
    }

```

```

    public async Task<ProjectResponse> UpdateProjectAsync(string
projectId, UpdateProjectRequest request)
    {
        if (!Guid.TryParse(projectId, out var id))
        {
            throw new BadRequestException("Invalid projectId");
        }

        if (string.IsNullOrEmpty(request.Name))
        {
            throw new BadRequestException("Project name is
required");
        }

        var project = await _projectRepository.GetByIdAsync(id);
        if (project == null)
        {
            throw new NotFoundException("Project not found");
        }

        project.Name = request.Name.Trim();
        project.UpdatedAt = DateTime.UtcNow;

        await _projectRepository.UpdateAsync(project);
        _logger.LogInformation("Project updated - {ProjectId}",
project.Id);

        return MapProject(project);
    }

    public async Task<IReadOnlyList<StepAnswerResponse>>
GetProjectAnswersAsync(string projectId)
    {
        if (!Guid.TryParse(projectId, out var id))
        {
            throw new BadRequestException("Invalid projectId");
        }

        var answers = await
_answerRepository.GetByProjectIdAsync(id);
        return answers
            .OrderBy(a => a.StepCode)
            .Select(MapAnswer)
            .ToList();
    }

    public async Task<StepAnswerResponse>
UpsertProjectAnswerAsync(string projectId, UpsertStepAnswerRequest
request)
    {
        if (!Guid.TryParse(projectId, out var id))
        {

```

```

        throw new BadRequestException("Invalid projectId");
    }

    if (string.IsNullOrEmpty(request.StepCode))
    {
        throw new BadRequestException("StepCode is required");
    }

    Guid? updatedBy = null;
    if (!string.IsNullOrEmpty(request.UpdatedByUserId))
    {
        if (!Guid.TryParse(request.UpdatedByUserId, out var
parsed))
        {
            throw new BadRequestException("Invalid
UpdatedByUserId");
        }
        updatedBy = parsed;
    }

    var existing = await
_answerRepository.GetByProjectAndStepAsync(id,
request.StepCode.Trim());

    var answer = existing ?? new ProjectStepAnswer
    {
        Id = Guid.NewGuid(),
        ProjectId = id,
        StepCode = request.StepCode.Trim()
    };

    answer.AnswerType = request.AnswerType;
    answer.SelectedOptionCode =
request.SelectedOptionCode?.Trim();
    answer.ValueJson = request.ValueJson;
    answer.Source = request.Source;
    answer.UpdatedByUserId = updatedBy;
    answer.UpdatedAt = DateTime.UtcNow;

    await _answerRepository.UpsertAsync(answer);
    return MapAnswer(answer);
}

public async Task<CreateShareCodeResponse>
CreateOrGetShareCodeAsync(string projectId)
{
    if (!Guid.TryParse(projectId, out var id))
    {
        throw new BadRequestException("Invalid projectId");
    }

    var project = await _projectRepository.GetByIdAsync(id);

```

```

        if (project == null)
        {
            throw new NotFoundException("Project not found");
        }

        var existingShare = await
        _sharingRepository.GetActiveShareByProjectIdAsync(id);
        if (existingShare != null)
        {
            return MapShare(existingShare);
        }

        var share = new SharedProject
        {
            Id = Guid.NewGuid(),
            ProjectId = id,
            Code = GenerateShareCode(),
            IsActive = true,
            SharedAt = DateTime.UtcNow
        };

        await _sharingRepository.CreateShareAsync(share);
        return MapShare(share);
    }

    public async Task<JoinByCodeResponse>
    JoinByCodeAsync(JoinByCodeRequest request)
    {
        if (!Guid.TryParse(request.UserId, out var userId))
        {
            throw new BadRequestException("Invalid userId");
        }

        if (string.IsNullOrEmpty(request.Code))
        {
            throw new BadRequestException("Code is required");
        }

        var share = await
        _sharingRepository.GetActiveShareByCodeAsync(request.Code.Trim());
        if (share == null)
        {
            throw new NotFoundException("Share code not found or
inactive");
        }

        var existingViewer = await
        _sharingRepository.GetViewerAsync(userId, share.Id);
        var viewer = existingViewer ?? new SharedProjectViewer
        {
            Id = Guid.NewGuid(),
            UserId = userId,

```

```

        ShareId = share.Id
    };
    viewer.IsActive = true;

    await _sharingRepository.UpsertViewerAsync(viewer);

    return new JoinByCodeResponse
    {
        ShareId = share.Id.ToString(),
        ProjectId = share.ProjectId.ToString(),
        ViewerId = viewer.Id.ToString(),
        IsActive = viewer.IsActive
    };
}

public async Task<ViewerCommentResponse>
AddCommentAsync(string projectId, CreateCommentRequest request)
{
    if (!Guid.TryParse(projectId, out var id))
    {
        throw new BadRequestException("Invalid projectId");
    }

    if (!Guid.TryParse(request.UserId, out var userId))
    {
        throw new BadRequestException("Invalid userId");
    }

    if (string.IsNullOrEmpty(request.Email))
    {
        throw new BadRequestException("Email is required");
    }

    if (string.IsNullOrEmpty(request.CommentText))
    {
        throw new BadRequestException("CommentText is
required");
    }

    var project = await _projectRepository.GetByIdAsync(id);
    if (project == null)
    {
        throw new NotFoundException("Project not found");
    }

    var comment = new ViewerComment
    {
        Id = Guid.NewGuid(),
        ProjectId = id,
        UserId = userId,
        Email = request.Email.Trim(),
        CommentText = request.CommentText.Trim(),
    }
}

```

```

        CommentDate = DateTime.UtcNow
    };

    await _commentRepository.CreateAsync(comment);

    return MapComment(comment);
}

public async Task<IReadOnlyList<ViewerCommentResponse>>
GetCommentsAsync(string projectId)
{
    if (!Guid.TryParse(projectId, out var id))
    {
        throw new BadRequestException("Invalid projectId");
    }

    var comments = await
        _commentRepository.GetByProjectIdAsync(id);
    return comments
        .OrderByDescending(c => c.CommentDate)
        .Select(MapComment)
        .ToList();
}
}

```

Листинг Г.4 – Файл GridPathfinding.cs

```

using System.Text.Json;
using
ProjectCalculationService.Application.Contracts.Optimizations;

namespace
ProjectCalculationService.Application.Services.Optimizations;

internal static class GridPathfinding
{
    internal sealed record VariantResult(
        bool IsFound,
        string? Message,
        IReadOnlyList<GridPointDto> Path,
        double LengthUnits,
        int Turns,
        double WeightedCost);

    public static bool IsBlocked(OptimizationTemplateResponse
template, int x, int y)
    {
        if (x < 0 || y < 0 || x >= template.Width || y >=
template.Height) return true;
        foreach (var w in template.Walls)
        {
            if (InRect(x, y, w)) return true;
        }
    }
}

```

```

        foreach (var z in template.ForbiddenZones)
        {
            if (InRect(x, y, z)) return true;
        }

        return false;
    }

    public static VariantResult
    FindWallFriendly(OptimizationTemplateResponse template,
    GridPointDto start, GridPointDto end)
    {
        if (IsBlocked(template, start.X, start.Y))
            return new VariantResult(false, "Стартовая точка
находится в стене/запрещённой зоне.", Array.Empty<GridPointDto>(),
0, 0, 0);
        if (IsBlocked(template, end.X, end.Y))
            return new VariantResult(false, "Конечная точка
находится в стене/запрещённой зоне.", Array.Empty<GridPointDto>(),
0, 0, 0);

        // Dijkstra on (x,y,dir) with wall-preference + turn
penalty.
        // This is mathematically optimal for the defined cost
function.
        // dir: 0..3 = (E,S,W,N), 4 = none (start)
        var width = template.Width;
        var height = template.Height;
        const int dirNone = 4;
        const double stepCost = 1.0;
        const double turnPenalty = 0.35;
        const double awayFromWallPenalty = 0.25;

        var dist = new double[width, height, 5];
        var prev = new (int px, int py, int pdir)[width, height,
5];
        var hasPrev = new bool[width, height, 5];

        for (var x = 0; x < width; x++)
            for (var y = 0; y < height; y++)
                for (var d = 0; d < 5; d++)
                    dist[x, y, d] = double.PositiveInfinity;

        var pq = new PriorityQueue<(int x, int y, int dir),
double>();
        dist[start.X, start.Y, dirNone] = 0;
        pq.Enqueue((start.X, start.Y, dirNone), 0);

        var dirs = new (int dx, int dy, int dir)[]
        {
            (1, 0, 0),

```

```

        (0, 1, 1),
        (-1, 0, 2),
        (0, -1, 3)
    };

    var visited = new bool[width, height, 5];
    while (pq.TryDequeue(out var node, out var nodeDist))
    {
        if (visited[node.x, node.y, node.dir]) continue;
        visited[node.x, node.y, node.dir] = true;
        if (nodeDist > dist[node.x, node.y, node.dir])
            continue;

        foreach (var (dx, dy, ndir) in dirs)
        {
            var nx = node.x + dx;
            var ny = node.y + dy;
            if (nx < 0 || ny < 0 || nx >= width || ny >=
height) continue;
            if (IsBlocked(template, nx, ny)) continue;

            var cost = stepCost;

            // prefer "along walls/partitions": penalize cells
not adjacent to walls.
            if (!IsAdjacentToWall(template, nx, ny))
            {
                cost += awayFromWallPenalty;
            }

            // penalize turns for easier maintenance.
            if (node.dir != dirNone && node.dir != ndir)
            {
                cost += turnPenalty;
            }

            var nd = nodeDist + cost;
            if (nd < dist[nx, ny, ndir])
            {
                dist[nx, ny, ndir] = nd;
                prev[nx, ny, ndir] = (node.x, node.y,
node.dir);

                hasPrev[nx, ny, ndir] = true;
                pq.Enqueue((nx, ny, ndir), nd);
            }
        }
    }

    // choose best dir at end.
    var bestDir = -1;
    var best = double.PositiveInfinity;
    for (var d = 0; d < 5; d++)

```

```

    {
        if (dist[end.X, end.Y, d] < best)
        {
            best = dist[end.X, end.Y, d];
            bestDir = d;
        }
    }

    if (double.IsInfinity(best) || bestDir == -1)
    {
        return new VariantResult(false, "Путь не найден (все варианты перекрыты стенами/зонами).", Array.Empty<GridPointDto>(), 0, 0, 0);
    }

    var path = ReconstructPathWithDir(start, end, bestDir, prev, hasPrev);
    var (len, turns) = ComputeMetrics(path);
    return new VariantResult(true, null, path, len, turns, best);
}

public static VariantResult
FindDirect(OptimizationTemplateResponse template, GridPointDto start, GridPointDto end)
{
    if (IsBlocked(template, start.X, start.Y))
        return new VariantResult(false, "Стартовая точка находится в стене/запрещённой зоне.", Array.Empty<GridPointDto>(), 0, 0, 0);
    if (IsBlocked(template, end.X, end.Y))
        return new VariantResult(false, "Конечная точка находится в стене/запрещённой зоне.", Array.Empty<GridPointDto>(), 0, 0, 0);

    // A* on (x,y) with 8-neighborhood to get an optimal shortest path under the chosen metric.
    var width = template.Width;
    var height = template.Height;

    var dist = new double[width, height];
    var prev = new (int px, int py)[width, height];
    var hasPrev = new bool[width, height];
    var visited = new bool[width, height];

    for (var x = 0; x < width; x++)
        for (var y = 0; y < height; y++)
            dist[x, y] = double.PositiveInfinity;

    var pq = new PriorityQueue<(int x, int y), double>();
    dist[start.X, start.Y] = 0;

```

```

    pq.Enqueue((start.X, start.Y), HeuristicOctile(start.X,
start.Y, end.X, end.Y));

    var neighbors = new (int dx, int dy, double cost)[]
    {
        (1, 0, 1.0),
        (0, 1, 1.0),
        (-1, 0, 1.0),
        (0, -1, 1.0),
        (1, 1, Math.Sqrt(2)),
        (-1, 1, Math.Sqrt(2)),
        (-1, -1, Math.Sqrt(2)),
        (1, -1, Math.Sqrt(2))
    };

    while (pq.TryDequeue(out var node, out _))
    {
        if (visited[node.x, node.y]) continue;
        visited[node.x, node.y] = true;

        if (node.x == end.X && node.y == end.Y)
        {
            break;
        }

        var baseDist = dist[node.x, node.y];
        foreach (var (dx, dy, step) in neighbors)
        {
            var nx = node.x + dx;
            var ny = node.y + dy;
            if (nx < 0 || ny < 0 || nx >= width || ny >=
height) continue;
            if (IsBlocked(template, nx, ny)) continue;

            // prevent "corner cutting": if moving diagonally,
both adjacent orth cells must be free.
            if (dx != 0 && dy != 0)
            {
                if (IsBlocked(template, node.x + dx, node.y)
|| IsBlocked(template, node.x, node.y + dy)) continue;
            }

            var nd = baseDist + step;
            if (nd < dist[nx, ny])
            {
                dist[nx, ny] = nd;
                prev[nx, ny] = (node.x, node.y);
                hasPrev[nx, ny] = true;
                var priority = nd + HeuristicOctile(nx, ny,
end.X, end.Y);
                pq.Enqueue((nx, ny), priority);
            }
        }
    }

```

```

        }
    }

    if (double.IsInfinity(dist[end.X, end.Y]))
    {
        return new VariantResult(false, "Путь не найден (все варианты перекрыты стенами/зонами).", Array.Empty<GridPointDto>(), 0, 0, 0);
    }

    var path = ReconstructPath(start, end, prev, hasPrev);
    var (len, turns) = ComputeMetrics(path);
    return new VariantResult(true, null, path, len, turns, dist[end.X, end.Y]);
}

public static string
ToResultJson(ProjectOptimizationPreviewResponse preview)
{
    return JsonSerializer.Serialize(preview, new
        JsonSerializerOptions { PropertyNamingPolicy =
            JsonNamingPolicy.CamelCase });
}

private static bool InRect(int x, int y, RectDto r)
{
    return x >= r.X && y >= r.Y && x < r.X + r.W && y < r.Y +
        r.H;
}

private static bool
IsAdjacentToWall(OptimizationTemplateResponse template, int x, int
y)
{
    // Adjacent to a wall/partition (not forbidden zone).
    static bool IsWall(OptimizationTemplateResponse t, int cx,
int cy)
    {
        if (cx < 0 || cy < 0 || cx >= t.Width || cy >=
t.Height) return true;
        foreach (var w in t.Walls)
        {
            if (InRect(cx, cy, w)) return true;
        }
        return false;
    }

    return IsWall(template, x + 1, y) || IsWall(template, x -
1, y) || IsWall(template, x, y + 1) || IsWall(template, x, y - 1);
}

```

```

    private static double HeuristicOctile(int x, int y, int tx,
int ty)
    {
        var dx = Math.Abs(tx - x);
        var dy = Math.Abs(ty - y);
        var f = Math.Sqrt(2) - 1;
        return (dx < dy) ? f * dx + dy : f * dy + dx;
    }

    private static IReadOnlyList<GridPointDto>
ReconstructPath(GridPointDto start, GridPointDto end, (int px, int
py)[,] prev, bool[,] hasPrev)
    {
        var stack = new Stack<GridPointDto>();
        var x = end.X;
        var y = end.Y;
        stack.Push(new GridPointDto(x, y));
        while (!(x == start.X && y == start.Y))
        {
            if (!hasPrev[x, y]) break;
            var p = prev[x, y];
            x = p.px;
            y = p.py;
            stack.Push(new GridPointDto(x, y));
        }

        return stack.ToList();
    }

    private static IReadOnlyList<GridPointDto>
ReconstructPathWithDir(
        GridPointDto start,
        GridPointDto end,
        int endDir,
        (int px, int py, int pdir)[,,] prev,
        bool[,,] hasPrev)
    {
        var stack = new Stack<GridPointDto>();
        var x = end.X;
        var y = end.Y;
        var d = endDir;
        stack.Push(new GridPointDto(x, y));
        while (!(x == start.X && y == start.Y))
        {
            if (!hasPrev[x, y, d]) break;
            var p = prev[x, y, d];
            x = p.px;
            y = p.py;
            d = p.pdir;
            stack.Push(new GridPointDto(x, y));
        }
    }

```

```

        return stack.ToList();
    }

    private static (double lengthUnits, int turns)
    ComputeMetrics(IReadOnlyList<GridPointDto> path)
    {
        if (path.Count <= 1) return (0, 0);

        double length = 0;
        var turns = 0;
        var prevDx = 0;
        var prevDy = 0;

        for (var i = 1; i < path.Count; i++)
        {
            var dx = path[i].X - path[i - 1].X;
            var dy = path[i].Y - path[i - 1].Y;
            length += (dx != 0 && dy != 0) ? Math.Sqrt(2) : 1.0;

            if (i >= 2 && (dx != prevDx || dy != prevDy))
            {
                turns++;
            }

            prevDx = dx;
            prevDy = dy;
        }

        return (length, turns);
    }
}

```

Листинг Г.5 – Файл SharingRepository.cs

```

using Microsoft.EntityFrameworkCore;
using ProjectCalculationService.Application.Interfaces;
using ProjectCalculationService.Application.Models;

namespace ProjectCalculationService.DataAccess.Repositories;

public class SharingRepository : ISharingRepository
{
    private readonly ProjectCalculationDbContext _dbContext;

    public SharingRepository(ProjectCalculationDbContext
dbContext)
    {
        _dbContext = dbContext;
    }

    public Task<SharedProject?>
GetActiveShareByProjectIdAsync(Guid projectId)
    {

```

```

        return _dbContext.SharedProjects
            .FirstOrDefaultAsync(s => s.ProjectId == projectId &&
s.IsActive);
    }

    public Task<SharedProject?> GetActiveShareByCodeAsync(string
code)
    {
        return _dbContext.SharedProjects
            .FirstOrDefaultAsync(s => s.Code == code &&
s.IsActive);
    }

    public async Task CreateShareAsync(SharedProject share)
    {
        _dbContext.SharedProjects.Add(share);
        await _dbContext.SaveChangesAsync();
    }

    public Task<SharedProjectViewer?> GetViewerAsync(Guid userId,
Guid shareId)
    {
        return _dbContext.SharedProjectViewers
            .FirstOrDefaultAsync(v => v.UserId == userId &&
v.ShareId == shareId);
    }

    public async Task UpsertViewerAsync(SharedProjectViewer
viewer)
    {
        var exists = await
_dbContext.SharedProjectViewers.AnyAsync(v => v.Id == viewer.Id);
        if (exists)
        {
            _dbContext.SharedProjectViewers.Update(viewer);
        }
        else
        {
            _dbContext.SharedProjectViewers.Add(viewer);
        }

        await _dbContext.SaveChangesAsync();
    }

    public async Task<IReadOnlyList<Project>>
GetViewerProjectsAsync(Guid userId)
    {
        var query =
            from viewer in _dbContext.SharedProjectViewers
            join share in _dbContext.SharedProjects on
viewer.ShareId equals share.Id

```

```

        join project in _dbContext.Projects on share.ProjectId
equals project.Id
        where viewer.UserId == userId && viewer.IsActive &&
share.IsActive
        select project;

    return await query.Distinct().ToListAsync();
}

public async Task<bool> IsViewerOfProjectAsync(Guid userId,
Guid projectId)
{
    var query =
        from viewer in _dbContext.SharedProjectViewers
        join share in _dbContext.SharedProjects on
viewer.ShareId equals share.Id
        where viewer.UserId == userId
            && viewer.IsActive
            && share.IsActive
            && share.ProjectId == projectId
        select viewer.Id;

    return await query.AnyAsync();
}
}

```