

Минобрнауки России

Юго-Западный государственный университет

Кафедра программной инженерии

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ПО ПРОГРАММЕ БАКАЛАВРИАТА**

09.03.04 Программная инженерия

(код, наименование ОПОП ВО: направление подготовки, направленность (профиль))

«Разработка программно-информационных систем»

Программная реализация системы управления проектами

(название темы)

Дипломный проект

(вид ВКР: дипломная работа или дипломный проект)

Автор ВКР

(подпись, дата)

А. Ю. Геворкян

(инициалы, фамилия)

Группа ПО-116

Руководитель ВКР

(подпись, дата)

Р. А. Томакова

(инициалы, фамилия)

Нормоконтроль

(подпись, дата)

А. А. Чаплыгин

(инициалы, фамилия)

ВКР допущена к защите:

Заведующий кафедрой

(подпись, дата)

А. В. Малышев

(инициалы, фамилия)

Курск 2025 г.

Минобрнауки России

Юго-Западный государственный университет

Кафедра программной инженерии

УТВЕРЖДАЮ:

Заведующий кафедрой

(подпись, инициалы, фамилия)

« ____ » _____ 20 ____ г.

ЗАДАНИЕ НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ ПО ПРОГРАММЕ БАКАЛАВРИАТА

Студента Геворкян А. Ю., шифр 21-06-0040, группа ПО-116

1. Тема «Программная реализация системы управления проектами» утверждена приказом ректора ЮЗГУ от «04» апреля 2025 г. № 1696-с.

2. Срок предоставления работы к защите «09» июня 2025 г.

3. Исходные данные для создания программной системы:

3.1. Перечень решаемых задач:

- провести анализ предметной области;
- разработать концептуальную модель системы управления проектами на основе Kanban-методологии;
- спроектировать модель системы и веб-приложения;
- реализовать и протестировать программную систему.

3.2. Входные данные и требуемые результаты для программы:

1. Входными данными для системы являются: учетные данные пользователя; параметры задач, проектов, досок; комментарии к задачам.

2. Выходными данными для системы являются: полная и детализированная информация о пользователях, задачах, проектах, досках; системные уведомления и информационные сообщения.

4. Содержание работы (по разделам):

4.1. Введение.

4.1. Анализ предметной области.

4.2. Техническое задание: основание для разработки, назначение разработки, требования к программной системе, требования к оформлению документации.

4.3. Технический проект: общие сведения о программной системе, проект данных программной системы, проектирование архитектуры программной системы, проектирование пользовательского интерфейса программной системы.

4.4. Рабочий проект: спецификация компонентов и классов программной системы, тестирование программной системы, сборка компонентов программной системы.

4.5. Заключение.

4.6. Список использованных источников.

5. Перечень графического материала:

Лист 1. Сведения о ВКРБ.

Лист 2. Цель и задачи разработки.

Лист 3. Диаграмма прецедентов.

Лист 4. Диаграмма компонентов.

Лист 5. ER – диаграмма.

Лист 6. Структура модулей.

Руководитель ВКР

(подпись, дата)

Р. А. Томакова

(инициалы, фамилия)

Задание принял к исполнению

(подпись, дата)

А. Ю. Геворкян

(инициалы, фамилия)

РЕФЕРАТ

Объем работы равен 118 страницам. Работа содержит 48 иллюстраций, 36 таблиц, 51 библиографических источников и 6 листов графического материала. Количество приложений – 2. Графический материал представлен в приложении А. Фрагменты исходного кода представлены в приложении Б.

Перечень ключевых слов: программа, система, компьютер, веб-приложение, Agile, Kanban, доска, проект, задача, пользователь, пользовательский интерфейс, база данных, управление, компонент, модуль, сущность, React, Javascript, Node.js, Zustand, PostgreSQL.

Объектом разработки является программная система для управления проектами с использованием методологии Kanban.

Целью выпускной квалификационной работы является повышение эффективности управления проектами в IT-сфере и улучшения качества рабочего процесса путем разделения жизненного цикла проекта на подзадачи.

В процессе разработки веб-приложения были выделены основные сущности путем создания информационных блоков, использованы методы модулей, обеспечивающие работу с сущностями предметной области, а также корректную работу веб-приложения, разработан пользовательский интерфейс на основе React/CSS и фреймворка Node.js.

При разработке системы использовались библиотека React для клиентской части, фреймворк Node.js для серверной части, СУБД PostgreSQL для хранения данных о всех сущностях, а также язык Javascript и связанные с ним библиотеки.

Разработанная система была успешно протестирована.

ABSTRACT

The scope of work is 118 pages. The work contains 48 illustrations, 36 tables, 51 bibliographic sources, and 6 sheets of graphic material. The number of appendices is 2. Graphic material is presented in Appendix A. Source code fragments are presented in Appendix B.

List of keywords: program, system, computer, web application, Agile, Kanban, board, project, task, user, user interface, database, management, component, module, entity, React, Javascript, Node.js, Zustand, PostgreSQL.

The object of development is a software system for project management using the Kanban methodology.

The aim of the final qualifying work is to increase the efficiency of project management in the IT sphere and improve the quality of the work process by dividing the project life cycle into subtasks.

In the process of developing the web application, the main entities were identified by creating information blocks, module methods were used to ensure work with domain entities, as well as the correct operation of the web application, and a user interface was developed based on React/CSS and the Node.js framework.

During the system development, the React library was used for the client-side, the Node.js framework for the server-side, PostgreSQL DBMS for storing data about all entities, as well as the Javascript language and related libraries.

The developed system was successfully tested.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	11
1 Анализ предметной области	14
1.1 Описание предметной области	14
1.2 Методология Agile	15
1.3 Методология Kanban	17
1.4 IT в России	19
1.5 Динамика и перспективы развития IT-сферы	21
2 Техническое задание	23
2.1 Основание для разработки	23
2.2 Цель и назначение разработки	23
2.3 Актуальность темы разработки	24
2.4 Этапы разработки	25
2.5 Требования к программной системе	25
2.5.1 Требования к данным	25
2.5.2 Функциональные требования	26
2.5.3 Сценарий прецедента «Регистрация»	27
2.5.4 Сценарий прецедента «Аутентификация»	28
2.5.5 Сценарий прецедента «Выход из системы»	28
2.5.6 Сценарий прецедента «Просмотр профиля»	29
2.5.7 Сценарий прецедента «Редактирование профиля»	29
2.5.8 Сценарий прецедента «Удаление профиля»	29
2.5.9 Сценарий прецедента «Создание нового проекта»	30
2.5.10 Сценарий прецедента «Редактирование проекта»	30
2.5.11 Сценарий прецедента «Удаление проекта»	31
2.5.12 Сценарий прецедента «Управление участниками проекта»	31
2.5.13 Сценарий прецедента «Создание доски»	32
2.5.14 Сценарий прецедента «Редактирование доски»	33
2.5.15 Сценарий прецедента «Удаление доски»	33
2.5.16 Сценарий прецедента «Изменение порядка досок»	33

2.5.17 Сценарий прецедента «Создание задачи»	34
2.5.18 Сценарий прецедента «Просмотр задачи»	34
2.5.19 Сценарий прецедента «Редактирование задачи»	35
2.5.20 Сценарий прецедента «Удаление задачи»	35
2.5.21 Сценарий прецедента «Перемещение задачи»	36
2.6 Требования пользователя к интерфейсу web-интерфейса	36
2.7 Нефункциональные требования	37
2.7.1 Требования к программному обеспечению	37
2.8 Ограничения	37
2.8.1 Требования к аппаратному обеспечению	38
2.9 Требования к оформлению документации	38
3 Технический проект	39
3.1 Общая характеристика организации решения задачи	39
3.2 Обоснование выбора технологии проектирования	39
3.2.1 Язык программирования JavaScript	39
3.2.2 Среда выполнения Node.js	40
3.2.3 Библиотека React	40
3.2.4 PostgreSQL	40
3.3 Диаграмма компонентов	40
3.4 Структура программы	42
3.4.1 Клиентская часть	42
3.4.2 Серверная часть	45
3.4.3 Клиентские сервисы	45
3.4.4 Управление состоянием и стилизация	46
3.4.5 Точка входа и конфигурация БД	46
3.5 Структура базы данных	48
4 Рабочий проект	55
4.1 Спецификация системы	55
4.1.1 Модуль App.jsx	55
4.1.2 Модуль index.css	56
4.1.3 Модуль main.jsx	56

4.1.4	Модуль store.js	57
4.1.5	Модуль projectsPanel.jsx	58
4.1.6	Модуль settingsPanel.jsx	59
4.1.7	Модуль authPage.jsx	59
4.1.8	Модуль taskPage.jsx	61
4.1.9	Модуль userProfilePage.jsx	61
4.1.10	Модуль layoutObject.jsx	62
4.1.11	Модуль taskObject.jsx	63
4.1.12	Модуль tooltipObject.jsx	64
4.1.13	Модуль boardObject.jsx	64
4.1.14	Модуль editBoardModal.jsx	65
4.1.15	Модуль editProfileModal.jsx	66
4.1.16	Модуль editProjectModal.jsx	66
4.1.17	Модуль editTaskModal.jsx	67
4.1.18	Модуль manageMembersModal.jsx	68
4.1.19	Модуль modalObject.jsx	69
4.1.20	Модуль addBoardModal.jsx	70
4.1.21	Модуль addProjectModal.jsx	70
4.1.22	Модуль alertModal.jsx	71
4.1.23	Модуль confirmModal.jsx	72
4.1.24	Модуль server.js	72
4.1.25	Модуль tokenService.js	73
4.1.26	Модуль apiService.js	73
4.2	Функциональное тестирование веб-приложения	74
	ЗАКЛЮЧЕНИЕ	96
	СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	96
	ПРИЛОЖЕНИЕ А Представление графического материала	103
	ПРИЛОЖЕНИЕ Б Фрагменты исходного кода программы	110
	На отдельных листах (CD-RW в прикрепленном конверте)	118
	Сведения о ВКРБ (Графический материал / Сведения о ВКРБ.png)	Лист 1
	Цель и задачи разработки (Графический материал / Цель и задачи разработ-	

ки.png)	Лист 2
Диаграмма прецедентов (Графический материал / Диаграмма прецедентов.png)	Лист 3
Диаграмма компонентов (Графический материал / Диаграмма компонентов.png)	Лист 4
ER – диаграмма (Графический материал / ER – диаграмма.png)	Лист 5
Структура модулей (Графический материал / Структура модулей.png)	Лист 6

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

БД – база данных.

ИТ (ИТ) – информационные технологии.

ПО – программное обеспечение.

СУБД – система управления базами данных.

ВВЕДЕНИЕ

Современное управление проектами и командная работа претерпевают значительные изменения, обусловленные стремительным развитием цифровых технологий и необходимостью повышения эффективности рабочих процессов. Увеличение продуктивности команд и качества выполнения задач требует внедрения инновационных решений, способных минимизировать временные затраты, улучшить коммуникацию и оптимизировать распределение ресурсов. Одной из ключевых задач в этой области остается обеспечение прозрачности процессов, своевременное отслеживание прогресса и эффективное управление задачами в рамках проектов, особенно в условиях распределенных команд и многозадачности.

Традиционные методы управления проектами, зачастую основанные на ручном отслеживании, электронной почте или разрозненных инструментах, часто оказываются громоздкими, приводят к потере информации и затрудняют оперативное принятие решений. Современные подходы, такие как методология Kanban, в сочетании с веб-технологиями и интерактивными интерфейсами, предоставляют новые возможности для автоматизации и визуализации рабочих процессов, обеспечивая высокую наглядность и гибкость управления.

Создание удобных и интуитивно понятных веб-приложений для управления проектами на базе Kanban-досок открывает новые возможности для командной работы. Такие решения позволяют эффективно создавать, назначать и отслеживать задачи, визуализировать этапы их выполнения, управлять приоритетами и сроками, а также обеспечивать централизованное взаимодействие между участниками проекта. Интуитивно понятный интерфейс и автоматизация рутинных операций делают данный подход доступным и эффективным для широкого круга пользователей и типов проектов.

Цель данной работы – разработка веб-приложения для управления проектами по методологии Kanban, предоставляющего пользователям инструменты для эффективной организации, отслеживания и совместной работы

над задачами. Для достижения этой цели необходимо решить следующие задачи:

1. Определить требования к системе управления проектами и описать её функциональные возможности, включая управление пользователями, проектами, досками и задачами.
2. Разработать архитектуру программного комплекса, включающую клиентскую часть на React и серверную часть на Node.js с базой данных PostgreSQL.
3. Реализовать основной функционал системы, включая аутентификацию пользователей, создание и управление проектами, досками и задачами, а также механизм перетаскивания для изменения статуса и порядка задач.
4. Разработать пользовательский интерфейс, обеспечивающий удобное и интуитивно понятное взаимодействие с системой.
5. Провести комплексное тестирование системы для оценки её надежности, функциональности и удобства использования.

Структура и объем работы. Отчет состоит из введения, 4 разделов основной части, заключения, списка использованных источников, 2 приложений. Текст выпускной квалификационной работы равен 118 страницам.

Во введении сформулирована цель работы, поставлены задачи разработки, описана структура работы, приведено краткое содержание каждого из разделов.

В первом разделе проводится анализ предметной области, изучение актуальных методологий управления проектами и перспективах их использования.

Во втором разделе описываются требования к разрабатываемой системе управления проектами.

В третьем разделе представлены проектные решения для программной системы и архитектура веб-приложения.

В четвертом разделе приведены программные компоненты системы, их описание и результаты тестирования.

В заключении излагаются основные результаты работы, полученные в ходе разработки.

В приложении А представлен графический материал.

В приложении Б представлены фрагменты исходного кода.

1 Анализ предметной области

1.1 Описание предметной области

В современных условиях быстро меняющегося рынка и возрастающей сложности проектов, эффективное управление проектами становится ключевым фактором успеха для многих организаций [1]. Компании постоянно ищут способы оптимизации процессов, сокращения сроков выполнения задач и повышения качества конечного продукта или услуги.

На сегодняшний день проблема неэффективного управления проектами и срыва сроков остается крайне актуальной. По данным различных исследований, значительная доля проектов не укладывается в первоначальные бюджеты, превышает запланированные сроки или не достигает поставленных целей в полном объеме [2]. В России, как и во всем мире, компании сталкиваются с потерями из-за недостаточной гибкости управления, слабой координации команд и отсутствия прозрачности в ходе выполнения работ. Неспособность быстро адаптироваться к изменениям требований, неэффективное распределение ресурсов [3] и трудности в отслеживании прогресса являются частыми причинами неудач.

Традиционные подходы к управлению проектами включают в себя несколько методологий:

1. Каскадная модель: этот подход характеризуется строгой последовательностью этапов выполнения проекта (анализ требований, проектирование, реализация, тестирование, внедрение). Планирование осуществляется детально на начальном этапе, и изменения в ходе проекта не приветствуются [2].

2. Методы, основанные на стандартах PMBOK или PRINCE2: данные подходы предлагают структурированные фреймворки, наборы процессов и областей знаний для управления проектами. Они ориентированы на формализацию и контроль всех аспектов проекта [1].

3. Управление на основе личного опыта и неформальных договоренностей: в некоторых, особенно небольших, командах или проектах управление

может осуществляться без строгой методологии, опираясь на опыт руководителя и устные договоренности.

Однако, традиционные и строго регламентированные методы управления проектами часто демонстрируют недостаточную гибкость в условиях высокой неопределенности и частых изменений требований [6], характерных для современной разработки программного обеспечения и других инновационных сфер. Они могут приводить к затягиванию сроков, увеличению бюрократии и снижению мотивации команды. Отсутствие же формализованных подходов ведет к хаосу и потере контроля над проектом. Поэтому использование гибких методологий, таких как Agile [14], и инструментов визуализации и управления потоком работ, таких как Kanban-доски [16], может значительно повысить адаптивность, прозрачность и эффективность управления проектами.

1.2 Методология Agile

Методология Agile представляет собой итеративный и гибкий подход к управлению проектами, в первую очередь к разработке программного обеспечения [11]. В отличие от традиционных каскадных моделей, где каждый этап выполняется последовательно и полностью перед началом следующего, Agile делает упор на постепенное развитие продукта через короткие циклы и постоянную обратную связь [15]. Основная цель Agile – быстро адаптироваться к изменениям требований, обеспечивать высокое качество продукта и удовлетворенность заказчика. Ключевые принципы Agile включают [43]:

1. Люди и взаимодействие важнее процессов и инструментов: подчеркивается важность командной работы, общения и сотрудничества [15].

2. Работающий продукт важнее исчерпывающей документации: акцент делается на создании функционального продукта, а не на чрезмерном документировании.

3. Сотрудничество с заказчиком важнее согласования условий контракта: подразумевается тесное взаимодействие с заказчиком на протяжении всего проекта для уточнения требований и получения обратной связи [12].

4. Готовность к изменениям важнее следования первоначальному плану: Agile-команды готовы адаптироваться к новым требованиям и изменениям в приоритетах.

Центральным элементом многих Agile-фреймворков (таких как Scrum) является понятие спринта [11]. Спринт – это короткий, ограниченный по времени период (обычно от одной до четырех недель), в течение которого команда разработчиков создает определенный, заранее согласованный объем функциональности продукта. Каждый спринт имеет четко определенную цель и набор задач, которые должны быть выполнены к его завершению.

Основные характеристики спринта:

1. Фиксированная длительность: продолжительность спринта устанавливается в начале проекта и остается неизменной для всех последующих спринтов. Это помогает команде выработать ритм и предсказуемость.

2. Цель спринта: каждый спринт направлен на достижение конкретной, измеримой цели, которая вносит вклад в общую цель продукта.

3. Бэклог спринта: в начале каждого спринта команда выбирает задачи из общего бэклога продукта – списка всех требуемых функций и улучшений – и формирует бэклог спринта. Это список задач, которые команда обязуется выполнить в течение текущего спринта.

4. Ежедневные встречи: короткие ежедневные совещания, на которых команда синхронизируется, обсуждает прогресс, выявляет препятствия и планирует работу на ближайшие 24 часа.

5. Обзор спринта: в конце спринта команда демонстрирует заказчику и другим заинтересованным сторонам созданный инкремент продукта (работающую функциональность). Цель – получить обратную связь.

6. Ретроспектива спринта: после обзора спринта команда проводит внутреннее обсуждение, чтобы проанализировать, что прошло хорошо, что можно улучшить в процессах работы, и планирует конкретные действия по улучшению для следующего спринта [13].

Спринты обеспечивают структуру для итеративной разработки [15], позволяя команде регулярно поставлять работающие части продукта, полу-

чать обратную связь и адаптироваться к изменениям, что является основой гибкости Agile-подхода.

1.3 Методология Kanban

Kanban (с японского ”сигнальная доска”или ”визуальный сигнал”) – это гибкая методология управления рабочими процессами, нацеленная на постепенное улучшение существующей системы работы и повышение ее эффективности [16]. Изначально разработанная для оптимизации производства в компании Toyota, сегодня Kanban успешно применяется в самых разных сферах, особенно в разработке программного обеспечения, IT-операциях и управлении проектами [19]. Главная цель Kanban – обеспечить плавный, предсказуемый и эффективный поток выполнения задач, минимизируя при этом потери и перегрузку команды.

Методология Kanban базируется на нескольких ключевых принципах [44]:

1. Начать с того, что есть сейчас: Kanban не требует немедленных кардинальных изменений существующих процессов, ролей или обязанностей. Он применяется ”поверх”текущей системы.

2. Стремиться к постепенным, эволюционным изменениям: Kanban поощряет небольшие, но постоянные улучшения, которые легче принимаются командой и менее рискованны.

3. Уважать текущие процессы, роли и обязанности: Kanban признает ценность существующих структур и не стремится их разрушить без необходимости.

4. Поощрять лидерство на всех уровнях: Каждый член команды может и должен вносить вклад в улучшение процессов.

Центральным инструментом и наиболее узнаваемым элементом методологии Kanban является Kanban-доска [17]. Это визуальное представление всего рабочего процесса, которое делает работу и ее статус видимыми для всей команды и заинтересованных сторон. Как устроена и работает Kanban-доска [18]:

1. Колонки: доска разделена на вертикальные колонки, каждая из которых представляет определенный этап рабочего процесса. Набор колонок адаптируется под конкретный процесс команды.

2. Карточки: каждая рабочая задача или элемент представляется отдельной карточкой. Карточка содержит информацию о задаче и перемещается по доске слева направо по мере прохождения этапов.

3. WIP-лимиты: одна из ключевых практик Kanban – это установление максимального количества задач, которые могут одновременно находиться в определенной колонке [16]. Эти лимиты явно указываются на доске. WIP-лимиты помогают предотвратить перегрузку команды, выявлять ”узкие места” и фокусироваться на завершении начатой работы, а не на старте новой. Это улучшает поток и сокращает время выполнения задач.

4. Поток: движение карточек по доске визуализирует поток работы. Цель – сделать этот поток как можно более плавным, быстрым и предсказуемым. Команда отслеживает, как задачи проходят через систему, и ищет способы устранения задержек и блокировок.

5. Явные политики: правила работы часто делаются явными и размещаются на доске или обсуждаются командой.

Преимущества использования Kanban:

1. Прозрачность: все видят текущее состояние дел, кто над чем работает и где возникают проблемы.

2. Улучшение коммуникации и сотрудничества: доска служит общим центром информации и обсуждений.

3. Выявление ”узких мест”: WIP-лимиты помогают быстро обнаружить этапы, замедляющие общий процесс.

4. Сокращение времени цикла: фокус на потоке и ограничении WIP ускоряет выполнение задач.

5. Гибкость: Kanban легко адаптируется к различным процессам и может использоваться совместно с другими методологиями.

6. Снижение стресса: ограничение многозадачности помогает команде работать более сосредоточенно.

7. Стимулирование непрерывного улучшения: визуализация и регулярный анализ потока побуждают команду постоянно искать способы оптимизации своей работы.

1.4 IT в России

Российская сфера информационных технологий прошла сложный и динамичный путь развития, отражающий как глобальные технологические тренды, так и уникальные национальные особенности. История IT-проектов в России – это повесть о переходе от централизованных государственных инициатив советской эпохи к формированию конкурентного рынка и активному стремлению к цифровому суверенитету в наши дни.

На ранних этапах, во времена СССР, IT-проекты были преимущественно сосредоточены в оборонной промышленности, науке и государственном управлении. Разработка автоматизированных систем управления (АСУ) для предприятий, создание вычислительных центров и специализированного программного обеспечения велись в рамках плановой экономики. Эти проекты отличались масштабностью, но зачастую инертностью внедрения и ограниченной гибкостью [3]. Недостаток конкуренции и изолированность от мирового IT-рынка сдерживали темпы инноваций в гражданском секторе.

Переход к рыночной экономике в 1990-е годы открыл новую страницу. Появились первые частные IT-компании, ориентированные на коммерческие заказы. Началась активная компьютеризация предприятий, банковского сектора и торговли. В этот период IT-проекты часто были связаны с внедрением зарубежных программных и аппаратных решений, адаптацией их к российским реалиям. Это было время накопления опыта, формирования кадрового потенциала и становления основ отечественного IT-рынка. Возникли компании, ставшие впоследствии лидерами индустрии, например, в области разработки антивирусного ПО, поисковых систем и системной интеграции.

Начало 2000-х годов ознаменовалось ростом российской экономики и увеличением инвестиций в информационные технологии. Государство стало проявлять все больший интерес к цифровизации, инициируя крупные наци-

ональные проекты. Одним из знаковых направлений стало создание "Электронного правительства нацеленного на повышение доступности и качества государственных услуг для граждан и бизнеса. Развивались системы межведомственного электронного взаимодействия, порталы госуслуг, электронный документооборот. Параллельно активно росли коммерческие IT-проекты, особенно в сферах телекоммуникаций, интернет-сервисов, электронной коммерции и разработки программного обеспечения на заказ [5]. Российские разработчики завоевали признание на международном уровне в таких областях, как разработка игр, офшорное программирование и наукоемкие программные решения.

В последние годы IT-проекты в России развиваются под сильным влиянием глобальных тенденций, таких как распространение облачных технологий [10], больших данных, искусственного интеллекта, мобильных приложений и интернета вещей. Однако на первый план все активнее выходит задача обеспечения цифрового суверенитета и импортозамещения. В ответ на внешние вызовы государство и бизнес наращивают усилия по созданию отечественных программных и аппаратных платформ, операционных систем, систем управления базами данных и бизнес-приложений. Яркими примерами таких проектов являются национальная платежная система "Мир развитие отечественных операционных систем на базе Linux, а также государственные инициативы по поддержке разработки российского ПО и микроэлектроники.

Особое внимание уделяется проектам в области кибербезопасности, что обусловлено как ростом глобальных киберугроз, так и стремлением защитить критическую информационную инфраструктуру страны. Активно развиваются проекты, связанные с цифровизацией промышленности, сельского хозяйства, здравоохранения и образования. Внедрение технологий искусственного интеллекта становится приоритетным направлением во многих отраслях, от финансового сектора до государственного управления.

Современные IT-проекты в России характеризуются растущей сложностью, необходимостью интеграции разнообразных технологий и высоким уровнем конкуренции [2]. Несмотря на существующие вызовы, такие как кад-

ровый голод в отдельных сегментах и необходимость адаптации к меняющимся экономическим условиям, российская IT-отрасль демонстрирует значительный потенциал для дальнейшего роста и инноваций, играя все более важную роль в развитии страны. Фокус на собственных разработках и стремление к технологической независимости определяют ключевые векторы развития IT-проектов в России на ближайшие годы.

1.5 Динамика и перспективы развития IT-сферы

Российская IT-сфера, по состоянию на май 2025 года, переживает период активной трансформации, где эффективное управление проектами становится залогом успеха в условиях курса на цифровой суверенитет и адаптации к новым экономическим реалиям [1]. Динамика отрасли характеризуется как масштабными государственными инициативами, так и гибкостью коммерческого сектора.

В управлении IT-проектами наблюдается сосуществование различных подходов. Крупные государственные проекты, особенно в сфере импортозамещения и развития критической инфраструктуры, по-прежнему требуют структурированного планирования и контроля, часто опираясь на каскадные или гибридные модели [2]. Одновременно частный бизнес и IT-компании активно применяют гибкие методологии, такие как Agile, для быстрой разработки и вывода продуктов на рынок. Растет значение гибридных подходов, сочетающих дисциплину традиционного управления с адаптивностью гибких практик, что требует от менеджеров проектов высокой квалификации и умения подбирать оптимальные инструменты. Основными вызовами остаются обеспечение качества в сжатые сроки, управление ресурсами и адаптация к высокой степени неопределенности. Государственная поддержка IT стимулирует проектную деятельность, но и повышает требования к прозрачности и эффективности управления.

Перспективы российской IT-отрасли тесно связаны с развитием отечественных программных продуктов, аппаратных решений и цифровых платформ. Это формирует устойчивый спрос на IT-проекты в таких областях, как

разработка системного и прикладного ПО [41], кибербезопасность и облачные сервисы [10]. Роль управления проектами в этих условиях только усиливается: востребованы специалисты, способные вести сложные комплексные проекты, управлять портфелями и обеспечивать эффективное взаимодействие команд. Ожидается дальнейшая цифровизация самого процесса управления проектами за счет внедрения специализированных ИС и аналитических инструментов [42]. Ключевыми компетенциями становятся управление рисками и глубокие знания в предметных областях реализуемых проектов [6].

Несмотря на существующие вызовы, российская IT-сфера демонстрирует потенциал к развитию, где профессиональное управление проектами играет критически важную роль. По мере дальнейшего технологического прогресса, включая интеграцию решений на базе искусственного интеллекта и нейронных сетей, будут появляться новые типы IT-проектов, ставящие перед менеджерами еще более сложные и интересные задачи.

2 Техническое задание

2.1 Основание для разработки

Полное наименование системы: «Программная реализация системы управления проектами». Основанием для разработки программы является приказ ректора ЮЗГУ от «04» апреля 2025 г. № 1696-сприказа «Об утверждении тем выпускных квалификационных работ и руководителей выпускных квалификационных работ».

2.2 Цель и назначение разработки

Система управления проектами предназначена для эффективной организации, отслеживания и координации рабочих процессов команд, использующих Kanban-подход, с целью повышения прозрачности, предсказуемости и продуктивности выполнения проектов. Пользователи должны иметь возможность визуализировать этапы работы с помощью настраиваемых Kanban-досок, управлять потоком задач от создания до завершения, отслеживать прогресс в реальном времени и анализировать эффективность процессов для непрерывного улучшения. Система должна предоставлять интуитивно понятный интерфейс для совместной работы и обеспечивать легкий доступ ко всей необходимой проектной информации. Задачами данной разработки являются:

1. Создание основной архитектуры и серверной логики для управления проектами, Kanban-досками, задачами, пользователями и их ролями.
2. Разработка веб-интерфейса пользователя с интерактивной Kanban-доской, поддерживающей создание и настройку колонок и задач, их перемещение и визуализацию статусов.
3. Реализация функционала управления задачами, включая их создание, редактирование, назначение ответственных, установку приоритетов, сроков выполнения и добавление описаний.

4. Обеспечение базовых функций администрирования системы, включая управление пользователями, настройку прав доступа и управление проектными пространствами.

2.3 Актуальность темы разработки

В условиях современной цифровой экономики, характеризующейся высокой скоростью изменений, глобализацией рынков и постоянно растущей конкуренцией, способность компаний быстро и эффективно реализовывать проекты становится критически важным фактором успеха. Особенно остро это проявляется в сфере информационных технологий и разработки программного обеспечения, где жизненные циклы продуктов сокращаются, а требования заказчиков и пользователей эволюционируют с беспрецедентной скоростью. По состоянию на 2025 год, организации всех размеров активно ищут инструменты и подходы, позволяющие им не только управлять проектами, но и гибко адаптироваться к непредвиденным обстоятельствам, сохраняя при этом фокус на создании ценности.

Актуальность разработки такой системы обусловлена, прежде всего, повышенным спросом на гибкость и адаптивность, поскольку современные проекты требуют от команд способности быстро перестраиваться, изменять приоритеты и инкорпорировать новые требования без значительных потерь времени и ресурсов, чему хорошо отвечает подход Kanban. Эта потребность в адаптивных инструментах становится особенно острой в условиях роста популярности удаленных и гибридных моделей работы, что порождает необходимость в эффективных цифровых решениях, способных обеспечить прозрачность, синхронизацию и эффективную коллаборацию всех участников проекта независимо от их местоположения; здесь программно реализованная Kanban-доска выступает как единый источник правды для команды. Кроме того, интегрированная система напрямую способствует стремлению к повышению производительности и сокращению потерь, позволяя не только визуализировать задачи, но и отслеживать метрики производительности, такие как время цикла и пропускная способность, а также применять WIP-лимиты для

предотвращения перегрузок и фокусировки на завершении начатой работы, что ведет к повышению общей эффективности.

Несмотря на наличие на рынке множества систем управления проектами, сохраняется выраженный запрос на инструменты, которые можно легко адаптировать под специфические процессы конкретной команды или организации, и которые обладают интуитивно понятным интерфейсом и низким порогом вхождения для пользователей. В конечном итоге, предлагаемая программная система поддерживает культуру непрерывного улучшения, поощряемую Kanban, предоставляя ценные данные для проведения ретроспектив и анализа эффективности потока работ.

2.4 Этапы разработки

Для реализации программной системы предполагается выполнение следующих этапов:

1. Анализ предметной области и определение ключевых требований к функционалу Kanban-доски.
2. Проектирование архитектуры приложения.
3. Разработка структуры базы данных для хранения информации об основных сущностях системы, их состояниях и прочей технической информации.
4. Реализация серверной логики и API для управления основными сущностями системы.
5. Создание пользовательского интерфейса для удобного пользования приложением.
6. Проведение функционального тестирования для выявления критичных несоответствий.

2.5 Требования к программной системе

2.5.1 Требования к данным

Входными данными для системы являются:

- учетные данные пользователя;
- параметры задач, проектов, досок;
- комментарии к задачам.

Выходными данными для системы являются:

- полная и детализированная информация о пользователях, задачах, проектах, досках;
- системные уведомления и информационные сообщения.

2.5.2 Функциональные требования

В разрабатываемой системе управления проектами должно быть предусмотрено наличие трех основных ролей пользователей с различными наборами прав и функциональных возможностей в рамках каждого проекта: «Участник проекта», «Владелец проекта» и «Администратор».

Пользователю с ролью «Участник проекта» должны быть доступны следующие функции:

- регистрация и аутентификация;
- просмотр и изменение персональной информации;
- удаление профиля;
- просмотр проектов, в которых он состоит, и существующих в них досок;
- просмотр, создание, редактирование, перемещение всех задач;
- удаление собственных задач.

Пользователю с ролью «Владелец проекта» должны быть доступны следующие функции:

- регистрация и аутентификация;
- просмотр и изменение персональной информации;
- удаление профиля;
- просмотр, создание, редактирование, удаление собственных проектов и досок;
- просмотр, создание, редактирование, перемещение, удаление любых задач в собственных проектах независимо от автора;

- управление участниками собственных проектов.

Пользователю с ролью «Администратор» должны быть доступны следующие функции:

- регистрация и аутентификация;
- просмотр и изменение персональной информации;
- удаление профиля;
- просмотр, создание, редактирование, удаление всех существующих проектов, досок и задач (а также их перемещение);
- управление участниками всех проектов.

На рисунке 2.1 представлена диаграмма прецедентов.

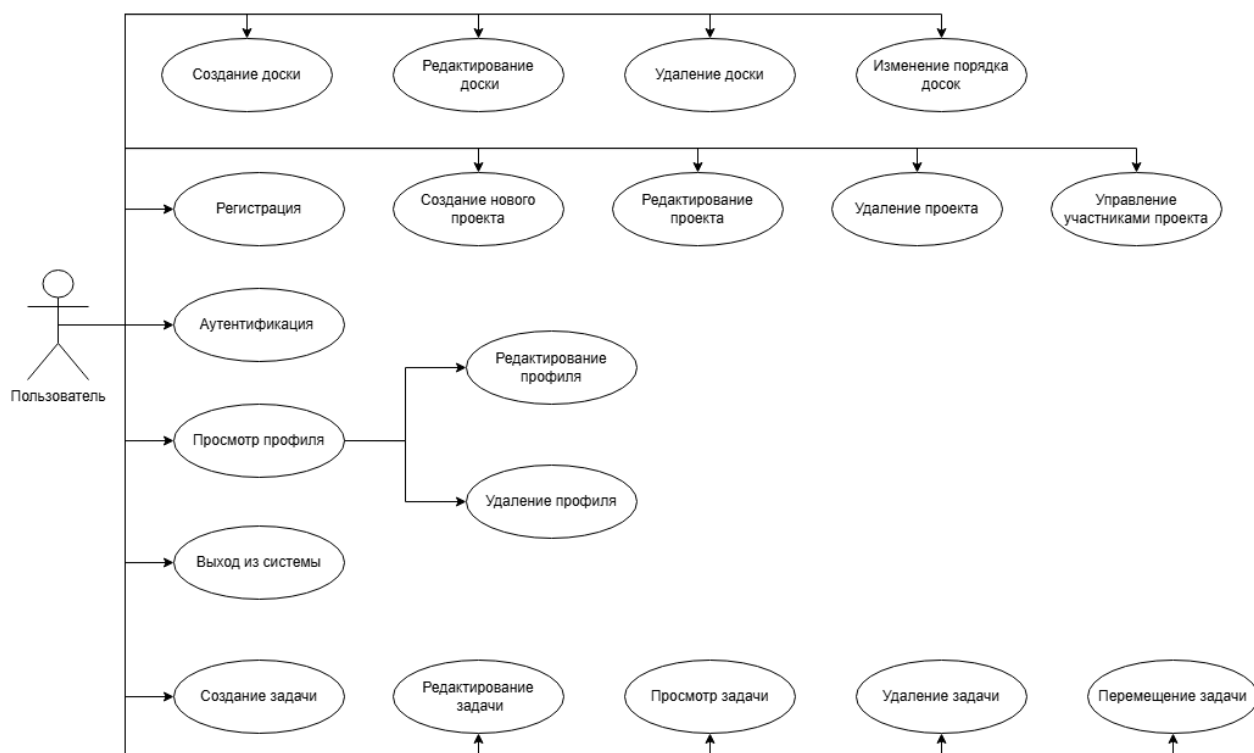


Рисунок 2.1 – Диаграмма прецедентов

2.5.3 Сценарий прецедента «Регистрация»

Основной успешный сценарий:

1. Пользователь переходит на страницу аутентификации web-приложения.
2. Пользователь нажимает на ссылку/кнопку «Создать».

3. Пользователь вводит свое имя в поле «Имя».
4. Пользователь вводит свою фамилию в поле «Фамилия».
5. Пользователь вводит свой адрес электронной почты в поле «Адрес электронной почты».
6. Пользователь вводит свой пароль в поле «Пароль», соблюдая указанные критерии.
7. Пользователь подтверждает свой пароль в поле «Подтвердите пароль».
8. Пользователь нажимает кнопку «Создать аккаунт».
9. Система проверяет корректность введенных данных и уникальность email.
10. Система регистрирует нового пользователя.
11. Система автоматически аутентифицирует пользователя, сохраняет токен и перенаправляет его на основной интерфейс приложения (доску Kanban).

2.5.4 Сценарий прецедента «Аутентификация»

Основной успешный сценарий:

1. Пользователь переходит на страницу аутентификации web-приложения.
2. Пользователь вводит свой адрес электронной почты в поле «Адрес электронной почты».
3. Пользователь вводит свой пароль в поле «Пароль».
4. Пользователь нажимает кнопку «Войти».
5. Система проверяет корректность введенных учетных данных.
6. Система аутентифицирует пользователя, сохраняет токен и перенаправляет его на основной интерфейс приложения (доску Kanban).

2.5.5 Сценарий прецедента «Выход из системы»

Основной успешный сценарий:

1. Аутентифицированный пользователь нажимает на иконку «Выйти» на панели настроек.
2. Система отображает модальное окно с запросом подтверждения выхода.
3. Пользователь нажимает кнопку «Выйти» в модальном окне.
4. Система завершает сеанс пользователя (удаляет токен).
5. Система перенаправляет пользователя на страницу аутентификации.

2.5.6 Сценарий прецедента «Просмотр профиля»

Основной успешный сценарий:

1. Аутентифицированный пользователь нажимает на иконку «Профиль пользователя» на панели настроек.
2. Система отображает страницу профиля пользователя с его данными (ФИО, email, дата регистрации, аватар).

2.5.7 Сценарий прецедента «Редактирование профиля»

Основной успешный сценарий:

1. Пользователь находится на странице своего профиля.
2. Пользователь нажимает кнопку «Редактировать профиль».
3. Система отображает модальное окно редактирования профиля с текущими данными.
4. Пользователь изменяет необходимые поля (имя, фамилия, отчество, URL аватара).
5. Пользователь нажимает кнопку «Сохранить изменения».
6. Система проверяет корректность введенных данных.
7. Система обновляет данные профиля пользователя в базе данных.
8. Система обновляет данные профиля в интерфейсе и закрывает модальное окно.

2.5.8 Сценарий прецедента «Удаление профиля»

Основной успешный сценарий:

1. Пользователь находится на странице своего профиля.
2. Пользователь нажимает кнопку «Удалить профиль».
3. Система отображает модальное окно с запросом подтверждения удаления профиля.
4. Пользователь нажимает кнопку «Да, удалить мой профиль».
5. Система удаляет профиль пользователя из базы данных.
6. Система завершает сеанс пользователя и перенаправляет на страницу аутентификации.

Управление Проектами

2.5.9 Сценарий прецедента «Создание нового проекта»

Основной успешный сценарий:

1. Аутентифицированный пользователь нажимает на иконку «Добавить новый проект» на панели настроек.
2. Система отображает модальное окно для добавления проекта.
3. Пользователь вводит название проекта в поле «Название проекта».
4. Пользователь опционально вводит описание проекта в поле «Описание».
5. Пользователь нажимает кнопку «Создать проект».
6. Система создает новый проект в базе данных, назначая текущего пользователя владельцем.
7. Система добавляет владельца в участники проекта.
8. Система обновляет список проектов в интерфейсе, новый проект становится текущим (если это первый проект или по логике приложения).
9. Модальное окно закрывается.

2.5.10 Сценарий прецедента «Редактирование проекта»

Условия: Пользователь является владельцем проекта или администратором. Основной успешный сценарий:

1. Пользователь (владелец/админ) нажимает на иконку «Редактировать текущий проект» на панели настроек (кнопка активна, если проект выбран).
2. Система отображает модальное окно редактирования текущего проекта с его данными.
3. Пользователь изменяет название и/или описание проекта.
4. Пользователь нажимает кнопку «Сохранить изменения».
5. Система обновляет данные проекта в базе данных.
6. Система обновляет название/описание проекта в интерфейсе. Модальное окно может закрыться или показать сообщение об успехе.

2.5.11 Сценарий прецедента «Удаление проекта»

Условия: Пользователь является владельцем проекта или администратором. Основной успешный сценарий:

1. Пользователь (владелец/админ) открывает модальное окно редактирования проекта.
2. Пользователь нажимает кнопку «Удалить проект».
3. Система отображает модальное окно подтверждения удаления проекта.
4. Пользователь нажимает кнопку «Да, удалить проект».
5. Система удаляет проект, все связанные с ним доски и задачи из базы данных.
6. Система обновляет список проектов в интерфейсе. Если удален текущий проект, выбирается другой или отображается состояние "нет проектов".
7. Модальные окна закрываются.

2.5.12 Сценарий прецедента «Управление участниками проекта»

Условия: Пользователь является владельцем проекта или администратором. Основной успешный сценарий (Просмотр):

1. Пользователь (владелец/админ) нажимает на иконку «Управление участниками проекта» на панели настроек (кнопка активна, если проект выбран).

2. Система отображает модальное окно со списком текущих участников проекта.

Основной успешный сценарий (Добавление участника):

1. В модальном окне управления участниками пользователь вводит email существующего пользователя системы в поле «Введите email пользователя для добавления».

2. Пользователь нажимает кнопку «Добавить».

3. Система проверяет, существует ли пользователь с таким email и не является ли он уже участником.

4. Система добавляет пользователя в участники проекта.

5. Список участников в модальном окне обновляется.

Основной успешный сценарий (Удаление участника):

1. В модальном окне управления участниками пользователь нажимает иконку удаления напротив участника (не владельца).

2. Система удаляет пользователя из участников проекта.

3. Список участников в модальном окне обновляется.

2.5.13 Сценарий прецедента «Создание доски»

Условия: Пользователь является владельцем текущего проекта или администратором. Проект выбран. Основной успешный сценарий:

1. Пользователь (владелец/админ) нажимает на иконку «Добавить новую доску» на панели настроек.

2. Система отображает модальное окно для добавления доски.

3. Пользователь вводит название доски в поле «Заголовок доски».

4. Пользователь опционально вводит описание доски.

5. Пользователь нажимает кнопку «Добавить доску».

6. Система создает новую доску в текущем проекте.

7. Система обновляет интерфейс, отображая новую доску в конце списка досок текущего проекта.

8. Модальное окно закрывается.

2.5.14 Сценарий прецедента «Редактирование доски»

Условия: Пользователь является владельцем проекта, к которому принадлежит доска, или администратором. Основной успешный сценарий:

1. Пользователь нажимает на иконку редактирования на заголовке доски.
2. Система отображает модальное окно редактирования доски с ее текущими данными.
3. Пользователь изменяет название и/или описание доски.
4. Пользователь нажимает кнопку «Сохранить».
5. Система обновляет данные доски в базе данных.
6. Система обновляет отображение доски в интерфейсе. Модальное окно закрывается.

2.5.15 Сценарий прецедента «Удаление доски»

Условия: Пользователь является владельцем проекта, к которому принадлежит доска, или администратором. Основной успешный сценарий:

1. Пользователь открывает модальное окно редактирования доски.
2. Пользователь нажимает кнопку «Удалить доску».
3. Система (опционально) запрашивает подтверждение.
4. Система удаляет доску и все связанные с ней задачи.
5. Система обновляет интерфейс, убирая доску. Модальное окно закрывается.

2.5.16 Сценарий прецедента «Изменение порядка досок»

Основной успешный сценарий:

1. Аутентифицированный пользователь наводит курсор на заголовок доски.

2. Пользователь зажимает левую кнопку мыши на заголовке доски и перетаскивает доску на новую позицию среди других досок текущего проекта.
3. Пользователь отпускает кнопку мыши.
4. Система визуально обновляет порядок досок на экране.
5. Система сохраняет новый порядок отображения досок для текущего проекта в базе данных.

2.5.17 Сценарий прецедента «Создание задачи»

Условия: Пользователь аутентифицирован, проект выбран, и пользователь является участником проекта (или владельцем/администратором). Основной успешный сценарий:

1. Пользователь нажимает на иконку «Добавить новую задачу» на панели настроек.
2. Система отображает модальное окно для добавления задачи.
3. Пользователь выбирает доску из выпадающего списка досок текущего проекта.
4. Пользователь вводит заголовок задачи.
5. Пользователь опционально вводит описание задачи.
6. Пользователь выбирает срок выполнения, приоритет и тип задачи.
7. Пользователь нажимает кнопку «Добавить задачу».
8. Система создает новую задачу и связывает ее с выбранной доской и текущим проектом. Текущий пользователь назначается автором.
9. Система обновляет интерфейс, отображая новую задачу на соответствующей доске.
10. Модальное окно закрывается.

2.5.18 Сценарий прецедента «Просмотр задачи»

Основной успешный сценарий:

1. Аутентифицированный пользователь нажимает на карточку задачи на одной из досок.

2. Система переключает вид основной области контента на страницу просмотра деталей задачи.

3. На странице отображается полная информация о задаче: заголовок, описание, проект, доска, срок выполнения, приоритет, тип, автор, разработчик (если назначен), QA (если назначен).

2.5.19 Сценарий прецедента «Редактирование задачи»

Условия: Пользователь является автором задачи или администратором.

Основной успешный сценарий:

1. Пользователь находится на странице просмотра деталей задачи.
2. Пользователь нажимает кнопку «Редактировать задачу».
3. Система отображает модальное окно редактирования задачи с ее текущими данными (заголовок, описание, доска, приоритет; срок выполнения и тип НЕ редактируются в этой модали).
4. Пользователь изменяет необходимые поля.
5. Пользователь нажимает кнопку «Сохранить изменения».
6. Система обновляет данные задачи в базе данных.
7. Система обновляет отображение деталей задачи на странице и на карточке (если она видна). Модальное окно закрывается.

2.5.20 Сценарий прецедента «Удаление задачи»

Условия: Пользователь является автором задачи или администратором.

Основной успешный сценарий:

1. Пользователь находится на странице просмотра деталей задачи.
2. Пользователь нажимает кнопку «Удалить задачу».
3. Система отображает модальное окно подтверждения удаления.
4. Пользователь нажимает кнопку «Да, удалить».
5. Система удаляет задачу из базы данных.
6. Система перенаправляет пользователя на доску Kanban и обновляет интерфейс.

2.5.21 Сценарий прецедента «Перемещение задачи»

Основной успешный сценарий:

1. Аутентифицированный пользователь наводит курсор на карточку задачи.
2. Пользователь зажимает левую кнопку мыши на карточке задачи и перетаскивает ее на другую доску в рамках текущего проекта.
3. Пользователь отпускает кнопку мыши над целевой доской.
4. Система визуально перемещает карточку задачи на новую доску и обновляет ее позицию в списке задач новой доски.
5. Система обновляет board_id и display_order задачи в базе данных.

2.6 Требования пользователя к интерфейсу web-интерфейса

Интерфейс должен быть интуитивно понятным, чтобы пользователи могли легко взаимодействовать с системой. На рисунке 2.2 представлен макет страницы.

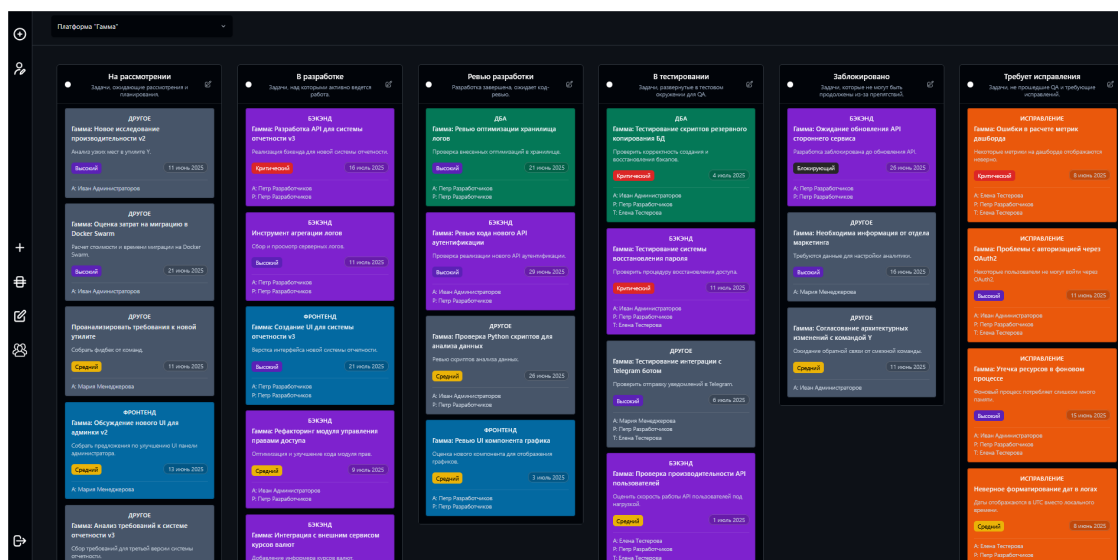


Рисунок 2.2 – Макет страницы

2.7 Нефункциональные требования

2.7.1 Требования к программному обеспечению

Для разработки и полноценной работы системы управления проектами необходимы следующие программные компоненты:

1. Среда выполнения JavaScript - Node.js.
2. Инструмент для сборки и разработки на основе Node.js - Vite.
3. Библиотека React для построения пользовательских интерфейсов.
4. Библиотека Zustand для управления состоянием приложения.
5. Библиотека react-beautiful-dnd для реализации функциональности перетаскивания.
6. CSS-фреймворк Tailwind CSS для стилизации интерфейса.
7. Библиотека prop-types для проверки типов свойств React-компонентов.

2.8 Ограничения

Необходимо учитывать следующие ограничения:

1. Для доступа ко всем функциям системы, включая совместную работу и синхронизацию данных в реальном времени, необходимо постоянное и стабильное подключение к сети Интернет.
2. Система ориентирована на управление проектами с использованием Kanban-методологии и может не покрывать все потребности проектов, требующих других подходов или расширенного инструментария.
3. Точность и эффективность специфических аналитических функций напрямую зависят от полноты, актуальности и качества данных, вводимых пользователями в систему.
4. Производительность системы при работе с очень большим количеством одновременных пользователей или значительными объемами данных будет зависеть от характеристик и масштабируемости серверной инфраструктуры.

2.8.1 Требования к аппаратному обеспечению

Для работы приложения требуется дисковое пространство не менее 5 Гб, минимум 16 Гб оперативной памяти и подключение к Интернету. Рекомендуется использовать процессор с 6 или более ядрами и частотой 2 ГГц или выше.

2.9 Требования к оформлению документации

Разработка программной документации и программного изделия должна производиться согласно ГОСТ 19.102-77 и ГОСТ 34.601-90. Единая система программной документации.

3 Технический проект

3.1 Общая характеристика организации решения задачи

Задача заключается в разработке и программной реализации веб-приложения для управления проектами, основанного на использовании Kanban-доски и принципов Agile-методологий. Приложение предназначено для командной работы и поможет руководителям проектов, а также членам команд эффективно планировать спринты и текущую деятельность, визуализировать рабочие процессы, отслеживать выполнение задач и улучшать взаимодействие, что способствует повышению общей продуктивности и успешности проектов.

Приложение будет представлять собой многопользовательское веб-приложение, доступное через сеть Интернет. Основными элементами интерфейса будут являться интерактивная Kanban-доска для визуального управления задачами, страницы для настройки проектов и досок, а также инструменты для работы с задачами.

3.2 Обоснование выбора технологии проектирования

Для создания приложения выбраны технологии, которые обеспечивают высокую производительность и удобство для пользователей.

3.2.1 Язык программирования JavaScript

JavaScript — это высокоуровневый, мультипарадигменный язык программирования, являющийся ключевой технологией для создания интерактивных веб-сайтов и приложений. Он выполняется непосредственно в браузере пользователя, обеспечивая динамическое обновление контента и взаимодействие без перезагрузки страницы, а также может использоваться на серверной стороне благодаря среде Node.js [21].

3.2.2 Среда выполнения Node.js

Node.js — это кроссплатформенная среда выполнения JavaScript, построенная на движке V8 от Google Chrome. Она позволяет исполнять JavaScript-код на стороне сервера, что открывает возможности для создания быстрых и масштабируемых сетевых приложений [31].

3.2.3 Библиотека React

React — это популярная JavaScript-библиотека для создания пользовательских интерфейсов, разработанная и поддерживаемая Facebook (ныне Meta). В основе React лежит концепция компонентного подхода, позволяющая разбивать сложный интерфейс на независимые, переиспользуемые части — компоненты, каждый из которых управляет собственным состоянием [26].

3.2.4 PostgreSQL

PostgreSQL — это объектно-реляционная система управления базами данных, известная своей надежностью, гибкостью и расширяемостью. Она поддерживает широкий спектр функциональности, включая сложные запросы, транзакции, уровни изоляции и возможность создания пользовательских типов данных [39].

3.3 Диаграмма компонентов

На рисунке 3.1 изображена диаграмма компонентов системы.

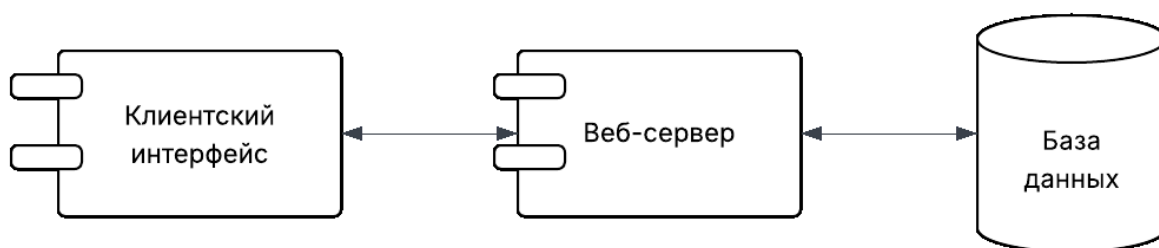


Рисунок 3.1 – Диаграмма компонентов

На представленной диаграмме показана архитектура веб-приложения для управления проектами, основанного на Kanban-доске и Agile-методиках. Схема включает три основных компонента: «Клиентский интерфейс», «Веб-сервер» и «База данных». Эти компоненты соединены стрелками, которые указывают направление потоков данных и взаимодействия между ними. Каждая часть системы выполняет свои специфические функции, обеспечивая совместную и эффективную работу всего приложения.

«Клиентский интерфейс» — это веб-приложение, которое запускается в браузере пользователя и служит основной точкой взаимодействия с системой. Через него пользователи могут регистрироваться и аутентифицироваться, создавать новые проекты, настраивать Kanban-доски, добавлять, редактировать и перемещать задачи между колонками. На страницах интерфейса в реальном времени отображается актуальное состояние проектов, Kanban-доски, списки задач, бэклоги и другая информация, необходимая для организации работы по Agile-принципам.

«Веб-сервер» функционирует как центральный узел архитектуры, обрабатывающий всю бизнес-логику системы. Он получает HTTP-запросы от «Клиентского интерфейса», выполняет соответствующие операции, взаимодействует с «Базой данных» для сохранения или извлечения необходимой информации, и отправляет ответы «Клиентскому интерфейсу» для обновления отображаемых данных или подтверждения выполненных действий.

«База данных» отвечает за надежное и долговременное хранение всей информации, используемой в системе управления проектами. В ней содержатся структурированные данные о зарегистрированных пользователях, созданных проектах, деталях задач, конфигурациях Kanban-досок и другие сведения, необходимые для функционирования Agile-процессов. «Веб-сервер» постоянно обращается к «Базе данных» для выполнения операций чтения и записи данных при каждом значимом взаимодействии пользователя с системой.

3.4 Структура программы

Система управления проектами представляет собой веб-приложение, чья структура организована в виде набора модулей, каждый из которых выполняет определённые функции.

3.4.1 Клиентская часть

Клиентская часть приложения разработана с использованием библиотеки React и отвечает за пользовательский интерфейс и взаимодействие с пользователем. Она включает в себя набор компонентов, каждый из которых выполняет определенную функцию, начиная от отображения основных элементов управления и заканчивая сложными модальными окнами для управления данными:

1. Модуль `App.jsx`. Главный компонент React-приложения. Управляет состоянием аутентификации пользователя, маршрутизацией между основными представлениями, отображением глобальных модальных окон. Инициализирует загрузку основных данных при входе пользователя и при их обновлении.

2. Модуль `layoutObject.jsx`. Компонент, определяющий основной макет интерфейса для аутентифицированных пользователей. Включает боковую панель настроек, верхнюю панель со списком проектов и поиском, а также основную область контента, где динамически отображаются либо доски Kanban с задачами, либо страница детального просмотра задачи. Реализует логику перетаскивания для досок и задач.

3. Модуль `authPage.jsx`. Компонент страницы аутентификации. Предоставляет пользователю интерфейс для регистрации новой учетной записи и входа в существующую. Включает валидацию вводимых данных и отображение требований к паролю.

4. Модуль `userProfilePage.jsx`. Компонент страницы профиля пользователя. Отображает информацию о текущем аутентифицированном пользова-

теле и предоставляет функционал для редактирования и удаления профиля через соответствующие модальные окна.

5. Модуль `taskPage.jsx`. Компонент страницы детального просмотра задачи. Отображает всю информацию о выбранной задаче, включая ее заголовок, описание, проект, доску, срок выполнения, приоритет, тип, автора, а также назначенных разработчика и QA-специалиста. Предоставляет пользователю возможность назначить себя на роль разработчика или QA, если эти роли свободны. Также содержит кнопки для редактирования и удаления задачи.

6. Модуль `boardObject.jsx`. Компонент, представляющий одну доску Kanban в рамках проекта. Отображает заголовок и описание доски. Содержит список задач, принадлежащих этой доске, и обеспечивает возможность перетаскивания задач. Предоставляет кнопку для редактирования/удаления доски.

7. Модуль `taskObject.jsx`. Компонент, представляющий карточку отдельной задачи на доске. Отображает заголовок задачи, ее тип, приоритет, срок выполнения, а также краткую информацию об авторе, разработчике и QA-специалисте. Позволяет пользователю кликнуть по карточке для перехода на страницу детального просмотра задачи.

8. Модуль `settingsPanel.jsx`. Компонент боковой панели навигации и действий. Содержит кнопки для добавления новой задачи, перехода к профилю пользователя, добавления нового проекта, добавления новой доски, редактирования текущего проекта, управления участниками текущего проекта и выхода из системы. Видимость некоторых кнопок зависит от роли пользователя и выбранного проекта.

9. Модуль `projectsPanel.jsx`. Компонент, отображаемый в верхней части интерфейса. Содержит выпадающий список для выбора текущего проекта из числа тех, в которых пользователь состоит.

10. Модуль `tooltipObject.jsx`. Вспомогательный компонент для отображения всплывающих текстовых подсказок при наведении курсора на элементы интерфейса.

11. Модуль `modalObject.jsx`. Компонент модального окна для создания новой задачи. Позволяет выбрать доску, ввести заголовок, описание, срок выполнения, а также выбрать приоритет и тип задачи.

12. Модуль `addBoardModal.jsx`. Компонент модального окна для добавления новой доски в выбранный проект. Позволяет ввести заголовок и описание доски.

13. Модуль `addProjectModal.jsx`. Компонент модального окна для создания нового проекта. Позволяет ввести название и описание проекта.

14. Модуль `alertModal.jsx`. Общий компонент для отображения простых информационных сообщений или оповещений об ошибках с одной кнопкой «ОК».

15. Модуль `confirmModal.jsx`. Общий компонент для запроса подтверждения у пользователя перед выполнением деструктивных или важных действий. Предоставляет кнопки «Подтвердить» и «Отмена».

16. Модуль `editBoardModal.jsx`. Компонент модального окна для редактирования существующей доски. Позволяет изменить заголовок и описание доски, а также удалить доску.

17. Модуль `editProfileModal.jsx`. Компонент модального окна для редактирования персональной информации пользователя.

18. Модуль `editProjectModal.jsx`. Компонент модального окна для редактирования существующего проекта. Позволяет изменить название и описание проекта, а также удалить проект.

19. Модуль `editTaskModal.jsx`. Компонент модального окна для редактирования существующей задачи. Позволяет изменить заголовок, описание, доску и приоритет задачи.

20. Модуль `manageMembersModal.jsx`. Компонент модального окна для управления участниками выбранного проекта. Позволяет просматривать список участников, добавлять новых участников по email и удалять существующих.

3.4.2 Серверная часть

Серверная часть приложения, построенная на Node.js с использованием фреймворка Express.js, обеспечивает обработку всех API-запросов, управление бизнес-логикой, взаимодействие с базой данных и реализацию механизмов безопасности, таких как аутентификация и авторизация. Список связанных модулей:

1. Модуль `server.cjs`. Основной файл серверной части приложения, реализованный на Node.js с использованием фреймворка Express.js. Отвечает за обработку всех API-запросов от клиентской части. Реализует эндпоинты для:

- регистрации и аутентификации пользователей;
- управления профилями пользователей;
- получения списка всех пользователей системы;
- управления проектами;
- управления участниками проектов;
- управления досками;
- управления задачами.

Взаимодействует с базой данных PostgreSQL для хранения и извлечения данных. Реализует логику авторизации, проверяя права пользователя (например, владелец проекта, администратор, участник проекта) перед выполнением защищенных операций.

3.4.3 Клиентские сервисы

Клиентские сервисы представляют собой вспомогательные JavaScript-модули, которые инкапсулируют специфическую логику, используемую различными компонентами клиентской части. Они помогают упростить взаимодействие с API и управление состоянием аутентификации. Список связанных модулей:

1. Модуль `apiService.js`. Модуль на стороне клиента, предоставляющий функцию `authenticatedFetch` для упрощения выполнения аутентифицирован-

ных HTTP-запросов к API серверной части. Автоматически добавляет JWT токен аутентификации в заголовки запросов. Содержит логику для обработки специфических ошибок от сервера и вызова глобального обработчика для принудительного выхода пользователя из системы при проблемах с сессией или токеном.

2. Модуль `tokenService.js`. Модуль на стороне клиента, предоставляющий утилиты для работы с JWT токеном. Включает функцию `isTokenActive` для проверки наличия токена в `localStorage` и валидации его срока действия путем декодирования с использованием `jwt-decode`.

3.4.4 Управление состоянием и стилизация

Для эффективного управления состоянием приложения на стороне клиента используется библиотека `Zustand`, обеспечивающая централизованное и реактивное хранилище данных. Визуальное оформление реализуется с помощью `Tailwind CSS` и кастомных CSS-правил для создания консистентного и адаптивного пользовательского интерфейса. Список связанных модулей:

1. Модуль `store.js`. Файл, определяющий глобальное хранилище состояния приложения с использованием библиотеки `Zustand`. Содержит состояние для проектов, досок, задач, информации о текущем пользователе, списке всех пользователей, текущем выбранном проекте и задаче, виде отображения приложения, а также различные флаги для модальных окон и оповещений. Реализует действия для изменения этого состояния. Использует `middleware persist` для сохранения части состояния в `localStorage` браузера.

2. `index.css`. Основной файл стилей приложения. Включает базовые стили и утилиты `Tailwind CSS`, а также кастомные CSS-правила для оформления полос прокрутки и других элементов интерфейса для обеспечения единого визуального стиля.

3.4.5 Точка входа и конфигурация БД

Этот раздел описывает основные файлы, отвечающие за запуск клиентского приложения, а также скрипт для инициализации и настройки структу-

ры базы данных PostgreSQL, включая создание таблиц и заполнение начальными данными. Список связанных элементов:

1. Модуль `main.jsx`. Точка входа клиентского React-приложения. Инициализирует корневой компонент `App` и рендерит его в DOM-элемент с `id root`. Подключает основные стили из `index.css`.

2. `db.sql`. SQL-скрипт, предназначенный для инициализации структуры базы данных PostgreSQL. Содержит команды создания для всех необходимых сущностей, определяет первичные и внешние ключи, ограничения, значения по умолчанию и триггеры для автоматического обновления временных меток. Также включает секцию для заполнения базы данных начальными тестовыми данными для демонстрации и тестирования функционала.

Структура и отношения между модулями представлены на диаграмме модулей.

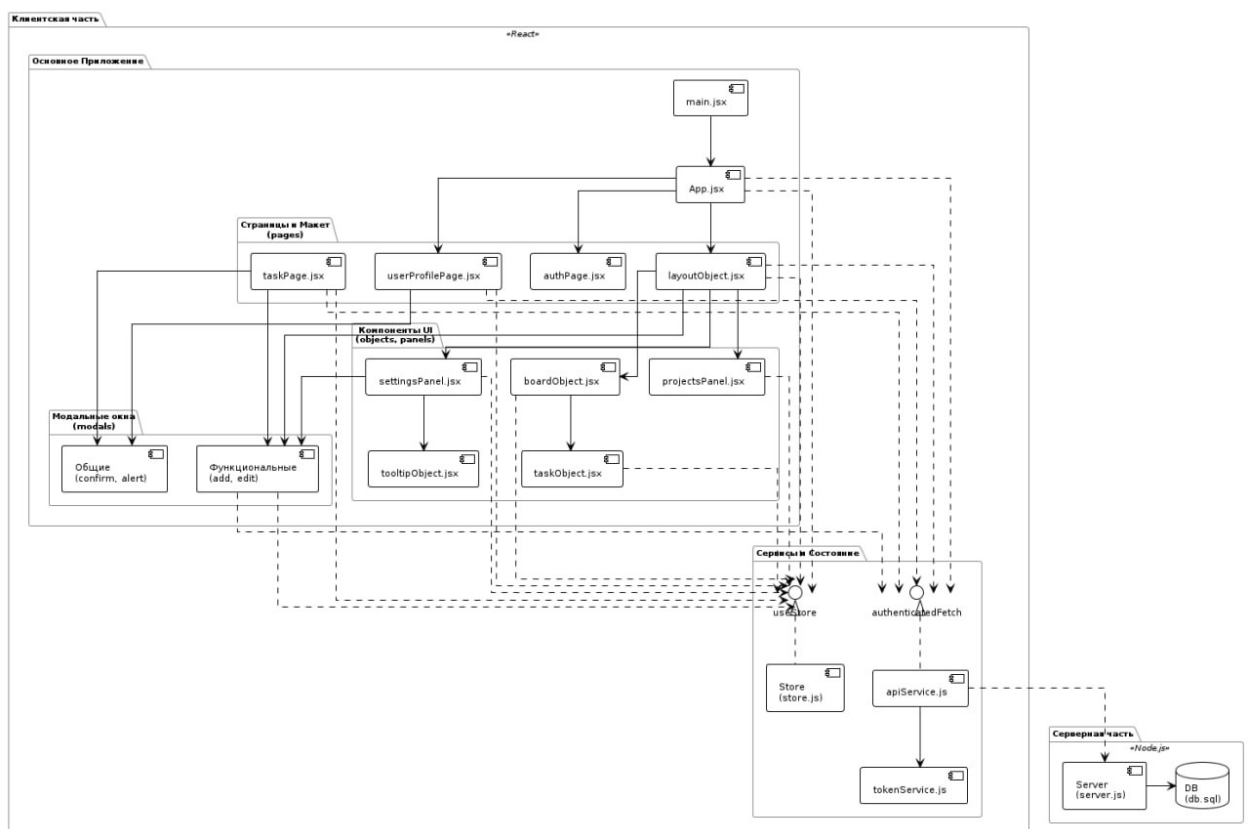


Рисунок 3.2 – Диаграмма модулей

3.5 Структура базы данных

Сущности и отношения между ними отображены на ER-диаграмме.

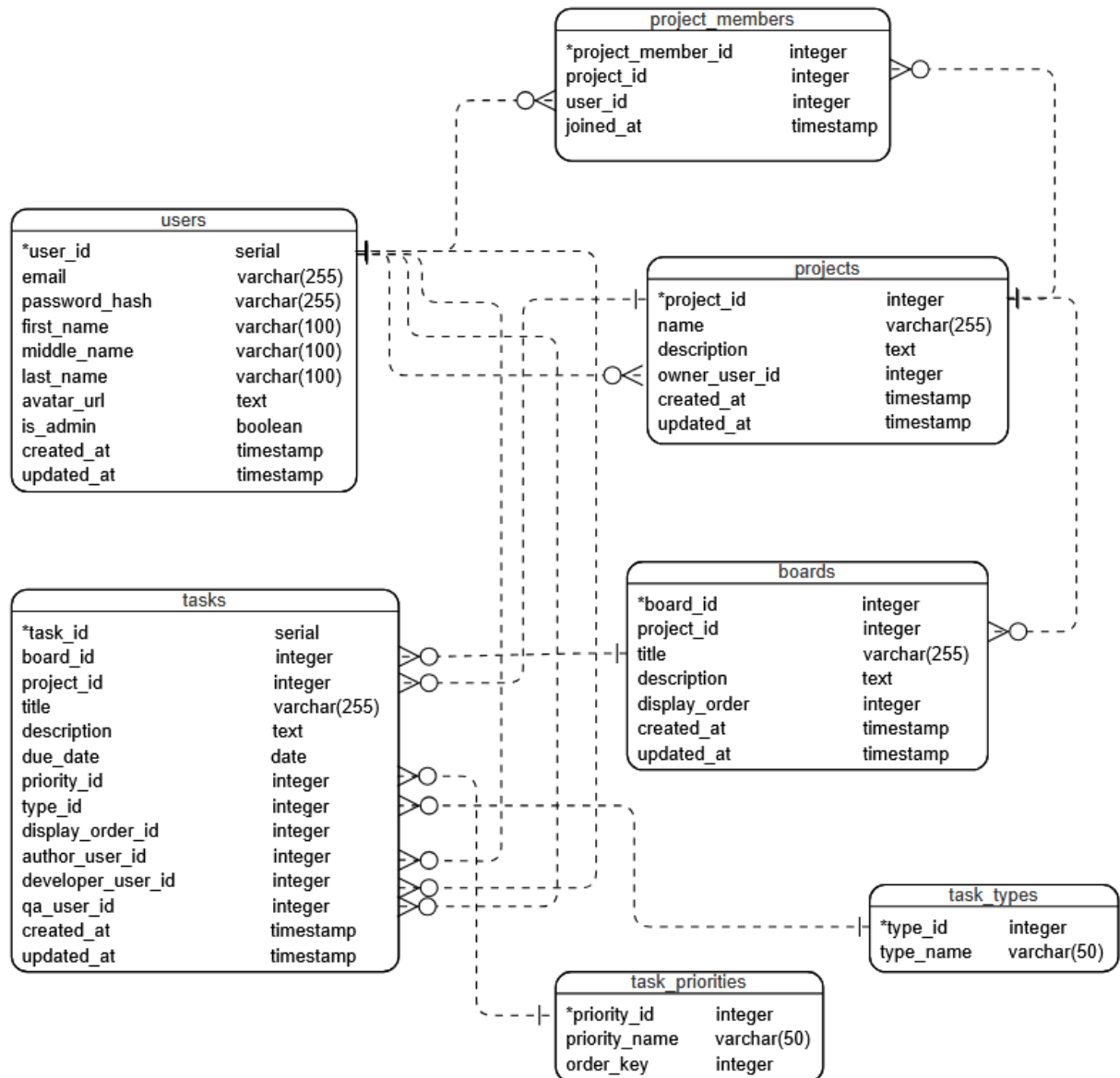


Рисунок 3.3 – ER-диаграмма

Таблица 3.1 – Атрибуты сущности «Users»

Поле	Тип	Обязательное	Описание
1	2	3	4
user_id	serial	да	Первичный ключ, уникальный идентификатор пользователя.
email	varchar(255)	да	Адрес электронной почты пользователя, уникальный.
password_hash	varchar(255)	да	Хеш пароля пользователя.
first_name	varchar(100)	нет	Имя пользователя.
middle_name	varchar(100)	нет	Отчество пользователя.
last_name	varchar(100)	нет	Фамилия пользователя.
avatar_url	text	нет	URL-адрес аватара пользователя.
is_admin	boolean	да	Флаг, указывающий, является ли пользователь администратором.
created_at	timestamp with time zone	да	Дата и время создания записи.
updated_at	timestamp with time zone	да	Дата и время последнего обновления записи.

Таблица 3.2 – Атрибуты сущности «Projects»

Поле	Тип	Обязательное	Описание
1	2	3	4
project_id	serial	да	Первичный ключ, уникальный идентификатор проекта.
name	varchar(255)	да	Название проекта.
description	text	нет	Описание проекта.
owner_user_id	integer	да	Внешний ключ, ссылается на users(user_id). Идентификатор пользователя-владельца проекта.
created_at	timestamp with time zone	да	Дата и время создания записи.
updated_at	timestamp with time zone	да	Дата и время последнего обновления записи.

Таблица 3.3 – Атрибуты сущности «Project Members»

Поле	Тип	Обязательное	Описание
1	2	3	4
project_member_id	serial	да	Первичный ключ, уникальный идентификатор записи об участии.
project_id	integer	да	Внешний ключ, ссылается на projects(project_id). Идентификатор проекта.

Продолжение таблицы 3.3

1	2	3	4
user_id	integer	да	Внешний ключ, ссылается на users(user_id). Идентификатор пользователя-участника.
joined_at	timestamp with time zone	да	Дата и время присоединения пользователя к проекту.

Таблица 3.4 – Атрибуты сущности «Boards»

Поле	Тип	Обязательное	Описание
1	2	3	4
board_id	serial	да	Первичный ключ, уникальный идентификатор доски.
project_id	integer	да	Внешний ключ, ссылается на projects(project_id). Идентификатор проекта, к которому принадлежит доска.
title	varchar(255)	да	Название доски.
description	text	нет	Описание доски.
display_order	integer	да	Порядок отображения доски в проекте.
created_at	timestamp with time zone	да	Дата и время создания записи.
updated_at	timestamp with time zone	да	Дата и время последнего обновления записи.

Таблица 3.5 – Атрибуты сущности «Task Priorities»

Поле	Тип	Обязательное	Описание
1	2	3	4
priority_id	serial	да	Первичный ключ, уникальный идентификатор приоритета.
priority_name	varchar(50)	да	Название приоритета (например, 'Low', 'Medium').
order_key	integer	да	Ключ для сортировки приоритетов.

Таблица 3.6 – Атрибуты сущности «Task Types»

Поле	Тип	Обязательное	Описание
1	2	3	4
type_id	serial	да	Первичный ключ, уникальный идентификатор типа задачи.
type_name	varchar(50)	да	Название типа задачи (например, 'FRONTEND', 'BUGFIX').

Таблица 3.7 – Атрибуты сущности «Tasks»

Поле	Тип	Обязательное	Описание
1	2	3	4
task_id	serial	да	Первичный ключ, уникальный идентификатор задачи.

Продолжение таблицы 3.7

1	2	3	4
board_id	integer	да	Внешний ключ, ссылается на boards(board_id). Идентификатор доски, на которой находится задача.
project_id	integer	да	Внешний ключ, ссылается на projects(project_id). Идентификатор проекта, к которому принадлежит задача.
title	varchar(255)	да	Заголовок задачи.
description	text	нет	Описание задачи.
due_date	date	нет	Срок выполнения задачи.
priority_id	integer	нет	Внешний ключ, ссылается на task_priorities(priority_id). Идентификатор приоритета задачи.
type_id	integer	нет	Внешний ключ, ссылается на task_types(type_id). Идентификатор типа задачи.
display_order	integer	да	Порядок отображения задачи на доске.
author_user_id	integer	да	Внешний ключ, ссылается на users(user_id). Идентификатор пользователя-автора задачи.
developer_user_id	integer	нет	Внешний ключ, ссылается на users(user_id). Идентификатор пользователя-разработчика.

Продолжение таблицы 3.7

1	2	3	4
qa_user_id	integer	нет	Внешний ключ, ссылается на users(user_id). Идентификатор пользователя-тестировщика.
created_at	timestamp with time zone	да	Дата и время создания записи.
updated_at	timestamp with time zone	да	Дата и время последнего обновления записи.

4 Рабочий проект

4.1 Спецификация системы

4.1.1 Модуль App.jsx

Модуль App.jsx реализует главный компонент приложения, который управляет аутентификацией, начальной загрузкой данных и навигацией между различными представлениями приложения, такими как доска Kanban, профиль пользователя и страница аутентификации. Он координирует общее состояние приложения и обрабатывает глобальные задачи, такие как истечение сессии и обновление данных. В таблице 4.1 приведено описание состояний модуля.

Таблица 4.1 – Описание состояний, используемых в App.jsx

Название состояния	Описание состояния
isAuthenticated	Логический флаг для отслеживания, аутентифицирован ли пользователь в данный момент.
isLoadingData	Логический флаг, указывающий, когда приложение находится в процессе загрузки начальных данных.
isAddProjectModalOpenFromEmptyState	Логический флаг для управления видимостью модального окна Добавить проект, когда у пользователя нет проектов.
tokenExpiredAlert	Состояние для модального окна оповещения об истечении срока действия токена, полученное из useStore.
registrationAlert	Состояние для модального окна оповещения об успешной регистрации, полученное из useStore.

Продолжение таблицы 4.1

projects	Массив, содержащий все проекты, доступные пользователю, полученный из useStore.
currentProjectId	ID текущего выбранного проекта, полученный из useStore.
dataRefreshTrigger	Счетчик, который при изменении запускает повторную загрузку данных, полученный из useStore.
appView	Строка, определяющая текущее активное представление в приложении (например, kanban, profile), полученная из useStore.
logoutConfirmModal	Состояние для модального окна подтверждения выхода, полученное из useStore.

4.1.2 Модуль index.css

Модуль index.css реализует глобальные и утилитные стили для приложения с использованием Tailwind CSS. Он определяет базовые стили, пользовательские стили для полос прокрутки и корректировки макета для всего приложения.

Таблица 4.2 – Описание параметров функций, используемых в index.css

Название параметра	Описание параметра
Нет	

4.1.3 Модуль main.jsx

Модуль main.jsx реализует точку входа в приложение React. Он импортирует главный компонент App и глобальный CSS-файл, а затем отображает компонент App в элементе DOM с идентификатором root.

4.1.4 Модуль store.js

Модуль store.js реализует глобальное управление состоянием приложения с использованием Zustand. Он определяет центральное хранилище, включая переменные состояния и действия для управления проектами, досками, задачами, профилями пользователей, представлением приложения и различными состояниями пользовательского интерфейса, такими как модальные окна и оповещения. Он также включает в себя промежуточное ПО для сохранения частей состояния в локальное хранилище. В таблице 4.3 приведено описание состояний модуля.

Таблица 4.3 – Описание состояний, используемых в store.js

Название состояния	Описание состояния
appView	Определяет текущее активное представление в приложении (например, kanban, profile, taskView).
selectedTaskId	Хранит ID задачи, выбранной для детального просмотра.
currentUserProfile	Хранит объект с данными профиля текущего вошедшего в систему пользователя.
allUsers	Хранит список всех пользователей в системе, обычно для выпадающих списков назначения.
mainLogoutHandler	Заполнитель для функции выхода из системы, который будет установлен компонентом App.
dataRefreshTrigger	Счетчик, который можно увеличить для запуска повторной загрузки данных в компонентах.
projects	Массив, содержащий все проекты, доступные пользователю.
currentProjectId	ID текущего выбранного проекта.

Продолжение таблицы 4.3

boards	Массив, содержащий все доски для всех проектов пользователя.
tasks	Массив, содержащий все задачи для всех проектов пользователя.
registrationAlert	Объект для управления состоянием модального окна оповещения о регистрации (показать, заголовок, сообщение).
logoutConfirmModal	Объект для управления состоянием модального окна подтверждения выхода.
tokenExpiredAlert	Объект для управления состоянием модального окна оповещения об истечении срока действия токена.

4.1.5 Модуль projectsPanel.jsx

Модуль projectsPanel.jsx реализует компонент React, который отображает выпадающий список доступных проектов. Он позволяет пользователю просматривать и переключаться между различными проектами, обновляя текущий выбранный идентификатор проекта в приложении. В таблице 4.4 приведено описание состояний модуля.

Таблица 4.4 – Описание состояний, используемых в projectsPanel.jsx

Название состояния	Описание состояния
selectedProjectIdForDropdown	Хранит идентификатор проекта, выбранного в данный момент в выпадающем меню, для управления контролируемым вводом компонента.

4.1.6 Модуль settingsPanel.jsx

Модуль settingsPanel.jsx реализует компонент боковой панели, который предоставляет действия пользователя и навигацию. Он включает кнопки для добавления задач, просмотра профиля пользователя, управления проектами и досками, а также выхода из системы. Доступ к определенным действиям контролируется на основе прав пользователя (владелец или администратор). В таблице 4.5 приведено описание состояний модуля.

Таблица 4.5 – Описание состояний, используемых в settingsPanel.jsx

Название состояния	Описание состояния
isAddBoardModalOpen	Логический флаг для управления видимостью модального окна Добавить доску.
isAddProjectModalOpen	Логический флаг для управления видимостью модального окна Добавить проект.
isEditProjectModalOpen	Логический флаг для управления видимостью модального окна Редактировать проект.
isManageMembersModalOpen	Логический флаг для управления видимостью модального окна Управление участниками.

4.1.7 Модуль authPage.jsx

Модуль authPage.jsx реализует компонент для аутентификации пользователя, обрабатывающий как вход, так и регистрацию. Он включает форму, которая динамически адаптируется для любого из этих представлений, с клиентской валидацией пользовательских вводов, таких как электронная почта и пароль. В таблице 4.6 приведено описание состояний модуля.

Таблица 4.6 – Описание состояний, используемых в authPage.jsx

Название состояния	Описание состояния
isRegisterView	Логический флаг для переключения между формами входа и регистрации.
firstName	Хранит значение поля ввода имени для регистрации.
lastName	Хранит значение поля ввода фамилии для регистрации.
middleName	Хранит значение поля ввода отчества для регистрации.
email	Хранит значение поля ввода электронной почты.
password	Хранит значение поля ввода пароля.
confirmPassword	Хранит значение поля подтверждения пароля для регистрации.
showPassword	Логический флаг для переключения видимости пароля.
isLoading	Логический флаг, указывающий, когда выполняется запрос на аутентификацию.
firstNameError	Хранит сообщение об ошибке для поля имени.
lastNameError	Хранит сообщение об ошибке для поля фамилии.
emailError	Хранит сообщение об ошибке для поля электронной почты.
passwordError	Хранит сообщение об ошибке для поля пароля.
confirmPasswordError	Хранит сообщение об ошибке для поля подтверждения пароля.
generalSubmitError	Хранит общее сообщение об ошибке при неудачной отправке формы.

Продолжение таблицы 4.6

passwordCriteria	Массив объектов, представляющих критерии валидации пароля и их текущий статус (выполнено или нет).
showPasswordCriteria	Логический флаг для управления видимостью списка требований к паролю.

4.1.8 Модуль taskPage.jsx

Модуль taskPage.jsx реализует компонент, который отображает детальное представление одной выбранной задачи. Он показывает все атрибуты задачи, такие как описание, приоритет, срок выполнения и назначенных пользователей. Он также предоставляет действия для редактирования, удаления или назначения ролей для задачи. В таблице 4.7 приведено описание состояний модуля.

Таблица 4.7 – Описание состояний, используемых в taskPage.jsx

Название состояния	Описание состояния
alertInfo	Объект для управления состоянием модального окна оповещения (показать, сообщение, заголовок).
isEditModalOpen	Логический флаг для управления видимостью модального окна Редактировать задачу.
isDeleteConfirmOpen	Логический флаг для управления видимостью модального окна подтверждения удаления.

4.1.9 Модуль userProfilePage.jsx

Модуль userProfilePage.jsx реализует компонент, который отображает информацию о профиле текущего вошедшего в систему пользователя. Он

извлекает данные пользователя и предоставляет опции для редактирования профиля или удаления учетной записи пользователя. В таблице 4.8 приведено описание состояний модуля.

Таблица 4.8 – Описание состояний, используемых в userProfilePage.jsx

Название состояния	Описание состояния
isLoading	Логический флаг, указывающий, когда извлекаются данные профиля пользователя.
error	Хранит любое сообщение об ошибке, возникающее при извлечении данных профиля.
isEditModalOpen	Логический флаг для управления видимостью модального окна Редактировать профиль.
isDeleteConfirmOpen	Логический флаг для управления видимостью модального окна подтверждения удаления учетной записи.
generalAlert	Объект для управления состоянием общего модального окна оповещения (показать, сообщение, заголовок).

4.1.10 Модуль layoutObject.jsx

Модуль layoutObject.jsx реализует основной компонент макета, который организует главный интерфейс приложения после входа в систему. Он объединяет панель настроек, панель проектов и основную область контента, которая может отображать либо представление доски Kanban, либо детальное представление задачи. Он также управляет логикой перетаскивания для задач и досок. В таблице 4.9 приведено описание состояний модуля.

Таблица 4.9 – Описание состояний, используемых в layoutObject.jsx

Название состояния	Описание состояния
isAddTaskModalOpen	Логический флаг для управления видимостью модального окна Добавить задачу.

4.1.11 Модуль taskObject.jsx

Модуль taskObject.jsx реализует одну карточку задачи на доске Kanban. Она перетаскиваемая и отображает ключевую информацию о задаче, такую как заголовок, тип, приоритет и срок выполнения. Нажатие на карточку переводит пользователя в детальное представление задачи. В таблицах 4.10, 4.11 и 4.12 приведено описание параметров функций модуля.

Таблица 4.10 – Описание параметров функции TaskObject в taskObject.jsx

Название параметра	Описание параметра
index	Индекс задачи в списке, используется для ключей и логики перетаскивания.
state: taskProp	Объект, содержащий все данные для отображаемой задачи.

Таблица 4.11 – Описание параметров функции getCardBgColorByType в taskObject.jsx

Название параметра	Описание параметра
typeValue	Строковое значение типа задачи (например, FRONTEND, BACKEND).

Таблица 4.12 – Описание параметров функции `getUserShortDisplay` в `taskObject.jsx`

Название параметра	Описание параметра
<code>userId</code>	Числовой ID пользователя, информацию о котором нужно отобразить.
<code>rolePrefix</code>	Строковый префикс для отображения перед именем пользователя (например, А:, Р:).

4.1.12 Модуль `tooltipObject.jsx`

Модуль `tooltipObject.jsx` реализует многоразовый компонент, который предоставляет простую текстовую подсказку при наведении на любой дочерний элемент, который он оборачивает. В таблице 4.13 приведено описание состояний модуля.

Таблица 4.13 – Описание состояний, используемых в `tooltipObject.jsx`

Название состояния	Описание состояния
<code>isVisible</code>	Логический флаг для управления видимостью подсказки.

4.1.13 Модуль `boardObject.jsx`

Модуль `boardObject.jsx` реализует одну колонку (доску) в представлении Kanban. Это перетаскиваемый контейнер, который содержит и отображает список задач. Он также предоставляет функциональность для редактирования деталей доски. В таблице 4.14 приведено описание состояний модуля.

Таблица 4.14 – Описание состояний, используемых в boardObject.jsx

Название состояния	Описание состояния
isModalEditOpen	Логический флаг для управления видимостью модального окна Редактировать доску для этой конкретной доски.

4.1.14 Модуль editBoardModal.jsx

Модуль editBoardModal.jsx реализует модальный компонент для редактирования деталей (заголовка и описания) существующей доски. Он также содержит функциональность для удаления доски. В таблице 4.15 приведено описание состояний модуля.

Таблица 4.15 – Описание состояний, используемых в editBoardModal.jsx

Название состояния	Описание состояния
title	Хранит значение поля ввода заголовка доски.
description	Хранит значение поля ввода описания доски.
isAlertOpen	Логический флаг для управления видимостью модального окна оповещения для уведомлений.
alertMessage	Хранит сообщение, которое будет отображаться в модальном окне оповещения.
populatedForBoardId	Хранит идентификатор доски, данные которой в настоящее время заполнены в модальном окне, чтобы предотвратить повторное отображение со старыми данными.

4.1.15 Модуль editProfileModal.jsx

Модуль editProfileModal.jsx реализует модальный компонент, который позволяет текущему пользователю редактировать информацию своего профиля, такую как имя и URL-адрес аватара. В таблице 4.16 приведено описание состояний модуля.

Таблица 4.16 – Описание состояний, используемых в editProfileModal.jsx

Название состояния	Описание состояния
firstName	Хранит значение поля ввода имени.
lastName	Хранит значение поля ввода фамилии.
middleName	Хранит значение поля ввода отчества.
avatarUrl	Хранит значение поля ввода URL-адреса аватара.
isLoading	Логический флаг, указывающий, когда выполняется запрос на обновление профиля.
errorAlert	Объект для управления состоянием модального окна оповещения об ошибке (показать, сообщение).

4.1.16 Модуль editProjectModal.jsx

Модуль editProjectModal.jsx реализует модальный компонент для редактирования деталей (названия и описания) текущего выбранного проекта. Он также включает функциональность для удаления проекта. В таблице 4.17 приведено описание состояний модуля.

Таблица 4.17 – Описание состояний, используемых в editProjectModal.jsx

Название состояния	Описание состояния
projectName	Хранит значение поля ввода названия проекта.

Продолжение таблицы 4.17

projectDescription	Хранит значение поля ввода описания проекта.
isSubmitting	Логический флаг, указывающий, когда выполняется запрос на обновление или удаление.
alertInfo	Объект для управления состоянием модального окна оповещения (показать, сообщение, заголовок).
confirmDeleteModalOpen	Логический флаг для управления видимостью модального окна подтверждения удаления проекта.
populatedForProjectId	Хранит идентификатор проекта, данные которого в настоящее время заполнены в модальном окне.

4.1.17 Модуль editTaskModal.jsx

Модуль editTaskModal.jsx реализует модальный компонент для редактирования деталей существующей задачи, включая ее заголовок, описание, срок выполнения, статус (доску) и приоритет. В таблице 4.18 приведено описание состояний модуля.

Таблица 4.18 – Описание состояний, используемых в editTaskModal.jsx

Название состояния	Описание состояния
title	Хранит значение поля ввода заголовка задачи.
description	Хранит значение поля ввода описания задачи.
dueDate	Хранит значение поля ввода срока выполнения.
selectedBoardId	Хранит идентификатор доски, выбранной для задачи.

Продолжение таблицы 4.18

selectedPriority	Хранит уровень приоритета, выбранный для задачи.
canEditDueDate	Логический флаг для определения, имеет ли текущий пользователь право редактировать срок выполнения.
isAlertOpen	Логический флаг для управления видимостью модального окна оповещения.
alertMessage	Хранит сообщение для модального окна оповещения.
populatedForTaskId	Хранит идентификатор задачи, данные которой в настоящее время заполнены в модальном окне.
minDate	Хранит минимально допустимую дату для выбора срока выполнения.

4.1.18 Модуль `manageMembersModal.jsx`

Модуль `manageMembersModal.jsx` реализует модальный компонент для управления участниками проекта. Он позволяет владельцам проектов или администраторам просматривать, добавлять и удалять пользователей из текущего проекта. В таблице 4.19 приведено описание состояний модуля.

Таблица 4.19 – Описание состояний, используемых в `manageMembersModal.jsx`

Название состояния	Описание состояния
members	Массив для хранения списка текущих участников проекта.
isLoadingMembers	Логический флаг, указывающий, когда извлекается список участников.
emailToAdd	Хранит электронную почту пользователя для добавления в проект.

Продолжение таблицы 4.19

isSubmitting	Логический флаг, указывающий, когда выполняется запрос на добавление или удаление участника.
alertInfo	Объект для управления состоянием модального окна оповещения (показать, сообщение, заголовок).

4.1.19 Модуль modalObject.jsx

Модуль modalObject.jsx реализует общий модальный компонент, используемый специально для создания новой задачи. Он предоставляет форму для ввода всех необходимых деталей задачи, таких как заголовок, описание, доска, приоритет и срок выполнения. В таблице 4.20 приведено описание состояний модуля.

Таблица 4.20 – Описание состояний, используемых в modalObject.jsx

Название состояния	Описание состояния
selectedBoardId	Хранит идентификатор доски, выбранной для новой задачи.
selectedPriority	Хранит уровень приоритета, выбранный для новой задачи.
selectedType	Хранит тип, выбранный для новой задачи.
title	Хранит значение поля ввода заголовка новой задачи.
description	Хранит значение поля ввода описания новой задачи.
dueDate	Хранит значение поля ввода срока выполнения для новой задачи.
minDate	Хранит минимально допустимую дату для выбора срока выполнения.

Продолжение таблицы 4.20

isAlertOpen	Логический флаг для управления видимостью модального окна оповещения.
alertMessage	Хранит сообщение для модального окна оповещения.

4.1.20 Модуль addBoardModal.jsx

Модуль addBoardModal.jsx реализует модальный компонент, который предоставляет форму для создания новой доски (колонки) в текущем выбранном проекте. В таблице 4.21 приведено описание состояний модуля.

Таблица 4.21 – Описание состояний, используемых в addBoardModal.jsx

Название состояния	Описание состояния
title	Хранит значение поля ввода заголовка новой доски.
description	Хранит значение поля ввода описания новой доски.
isAlertOpen	Логический флаг для управления видимостью модального окна оповещения.
alertMessage	Хранит сообщение для модального окна оповещения.

4.1.21 Модуль addProjectModal.jsx

Модуль addProjectModal.jsx реализует модальный компонент, который предоставляет форму для создания нового проекта. В таблице 4.22 приведено описание состояний модуля.

Таблица 4.22 – Описание состояний, используемых в addProjectModal.jsx

Название состояния	Описание состояния
projectName	Хранит значение поля ввода названия нового проекта.
projectDescription	Хранит значение поля ввода описания нового проекта.
isSubmitting	Логический флаг, указывающий, когда выполняется запрос на создание проекта.
alertInfo	Объект для управления состоянием модального окна оповещения (показать, сообщение, заголовок).

4.1.22 Модуль alertModal.jsx

Модуль alertModal.jsx реализует многоразовый модальный компонент, предназначенный для отображения простого оповещения или уведомления пользователю с заголовком и кнопкой ОК для его закрытия. В таблице 4.23 приведено описание параметров функции модуля.

Таблица 4.23 – Описание параметров функции AlertModal в alertModal.jsx

Название параметра	Описание параметра
isOpen	Логический флаг, управляющий видимостью модального окна.
onClose	Функция обратного вызова для закрытия модального окна.
title	Строка для заголовка модального окна.
message	Строка с сообщением для отображения в модальном окне.

4.1.23 Модуль confirmModal.jsx

Модуль confirmModal.jsx реализует многоразовый модальный компонент, используемый для запроса подтверждения у пользователя перед выполнением критического действия, такого как удаление. Он предоставляет опции Подтвердить и Отмена. В таблице 4.24 приведено описание параметров функции модуля.

Таблица 4.24 – Описание параметров функции ConfirmModal в confirmModal.jsx

Название параметра	Описание параметра
isOpen	Логический флаг, управляющий видимостью модального окна.
onClose	Функция обратного вызова для закрытия модального окна.
onConfirm	Функция обратного вызова, выполняемая при подтверждении действия.
title	Строка для заголовка модального окна.
message	Строка с сообщением для отображения в модальном окне.
confirmButtonText	Необязательный текст для кнопки подтверждения.
cancelButtonText	Необязательный текст для кнопки отмены.

4.1.24 Модуль server.js

Модуль server.js реализует бэкенд-сервер для приложения, созданный с помощью Node.js и Express. Он обрабатывает API-запросы для аутентификации (регистрация, вход), управления данными (проекты, доски, задачи, пользователи) и авторизации. Он подключается к базе данных PostgreSQL для со-

хранения данных. В таблицах 4.25, 4.26 и 4.27 приведено описание параметров функций модуля.

Таблица 4.25 – Описание параметров функции isUserAdministrator в server.js

Название параметра	Описание параметра
userId	ID пользователя для проверки статуса администратора.

Таблица 4.26 – Описание параметров функции generateToken в server.js

Название параметра	Описание параметра
user	Объект пользователя, для которого генерируется токен.

Таблица 4.27 – Описание параметров функции authenticateToken в server.js

Название параметра	Описание параметра
req	Объект запроса Express.
res	Объект ответа Express.
next	Функция обратного вызова для перехода к следующему middleware.

4.1.25 Модуль tokenService.js

Модуль tokenService.js реализует клиентский утилитарный модуль для обработки операций с JWT (JSON Web Token). Его основная функция - проверять наличие токена в локальном хранилище и не истек ли его срок действия.

4.1.26 Модуль apiService.js

Модуль apiService.js реализует клиентский сервисный модуль, который централизует обмен данными с API. Он предоставляет функцию authenticatedFetch, которая автоматически прикрепляет JWT пользователя к исходящим запросам и обрабатывает глобальные сбои аутентификации, та-

кие как истекший токен, вызывая принудительный выход из системы. В таблицах 4.28 и 4.29 приведено описание параметров функций модуля.

Таблица 4.28 – Описание параметров функции `setupGlobalAuthFailureHandler` в `apiService.js`

Название параметра	Описание параметра
<code>callback</code>	Функция обратного вызова для выполнения принудительного выхода из системы.

Таблица 4.29 – Описание параметров функции `authenticatedFetch` в `apiService.js`

Название параметра	Описание параметра
<code>path</code>	Путь к API эндпоинту (например, <code>/api/users</code>).
<code>options</code>	Необязательный объект с опциями для <code>fetch</code> -запроса.

4.2 Функциональное тестирование веб-приложения

На рисунке 4.1 показана страница с элементами для аутентификации.

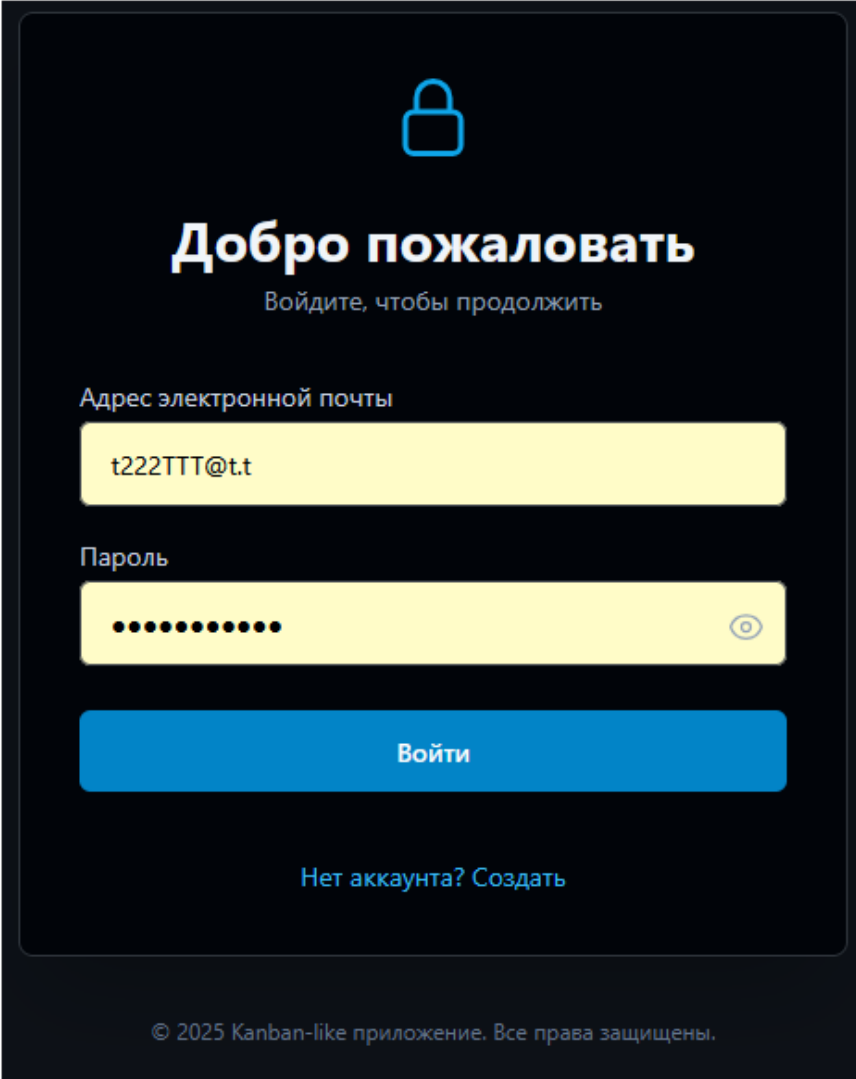
The illustration shows a login interface on a dark background. At the top center is a blue padlock icon. Below it, the text 'Добро пожаловать' is written in a large, bold, white font, followed by 'Войдите, чтобы продолжить' in a smaller white font. There are two input fields: the first is labeled 'Адрес электронной почты' and contains the text 't222TTT@t.t'; the second is labeled 'Пароль' and contains ten dots, with a blue eye icon on the right side to toggle visibility. Below the password field is a large blue button with the white text 'Войти'. Underneath the button is a link that says 'Нет аккаунта? Создать'. At the very bottom, in small white text, is the copyright notice '© 2025 Kanban-like приложение. Все права защищены.'

Рисунок 4.1 – Окно аутентификации

На рисунке 4.2 показано сообщение в случае попытки входа с неверными данными.

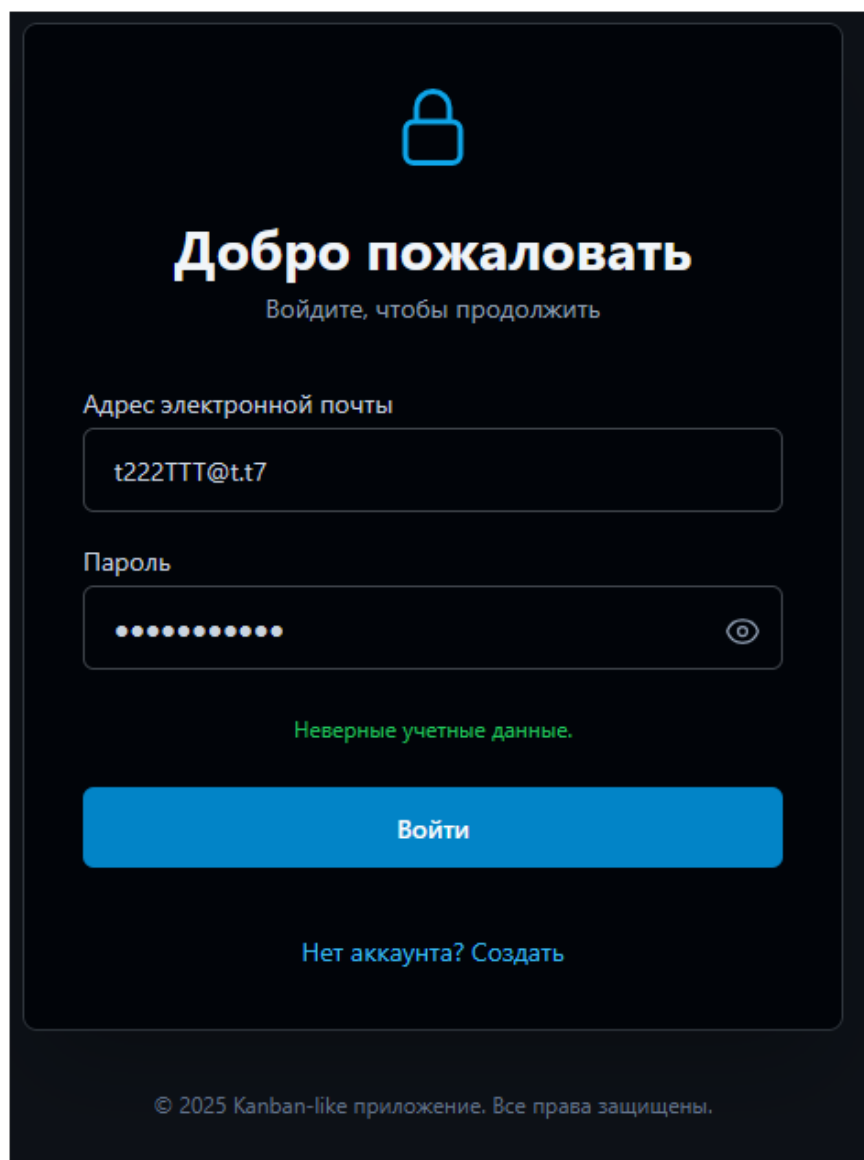
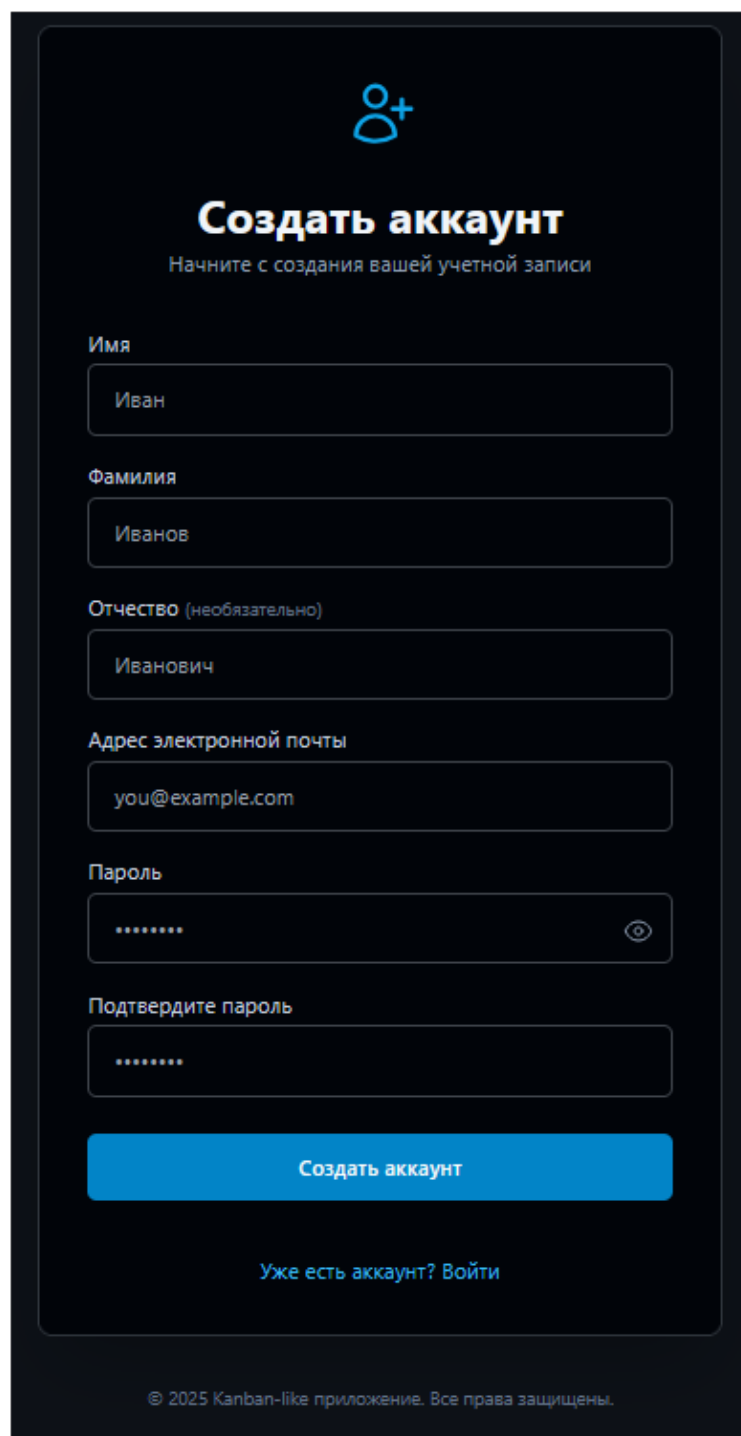



Рисунок 4.2 – Сообщение о неверных данных

На рисунке 4.3 показана страница с элементами для регистрации.





Создать аккаунт

Начните с создания вашей учетной записи

Имя

Иван

Фамилия

Иванов

Отчество (необязательно)

Иванович

Адрес электронной почты

you@example.com

Пароль

.....

Подтвердите пароль

.....


Создать аккаунт

[Уже есть аккаунт? Войти](#)

© 2025 Kanban-like приложение. Все права защищены.

Рисунок 4.3 – Окно регистрации

На рисунке 4.4 показано сообщение в случае отсутствия заполненных полей для регистрации.



Создать аккаунт

Начните с создания вашей учетной записи

Имя

Имя обязательно для заполнения.

Фамилия

Фамилия обязательна для заполнения.

Отчество (необязательно)

Адрес электронной почты

Email обязателен для заполнения.

Пароль

Пароль обязателен для заполнения.

Подтвердите пароль

Подтверждение пароля обязательно.

Создать аккаунт

[Уже есть аккаунт? Войти](#)

© 2025 Kanban-like приложение. Все права защищены.

Рисунок 4.4 – Проваленная валидация на заполненность полей

На рисунке 4.5 показано сообщение успешной валидации пароля.

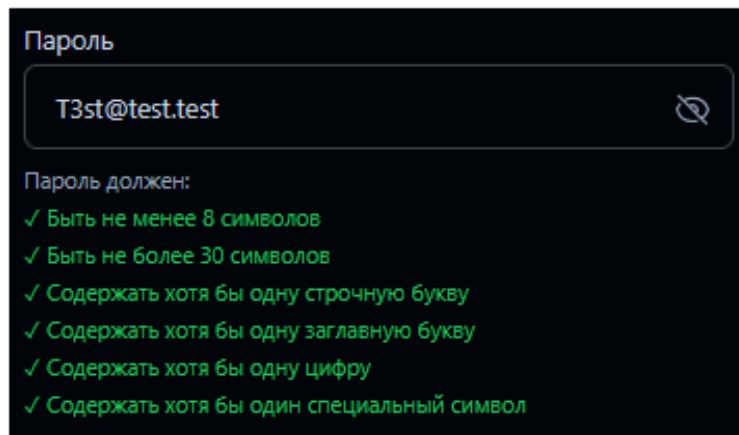


Рисунок 4.5 – Успешная валидация на значение пароля

На рисунке 4.6 показано сообщение проваленной валидации пароля.

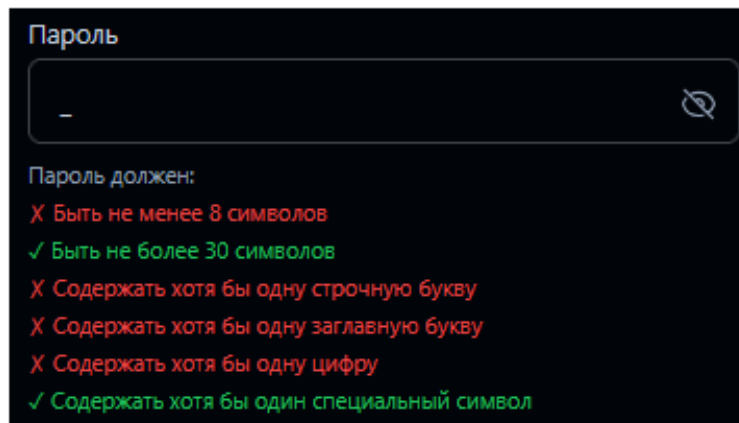


Рисунок 4.6 – Проваленная валидация на значение пароля

На рисунке 4.7 показано сообщение в случае успешной регистрации.

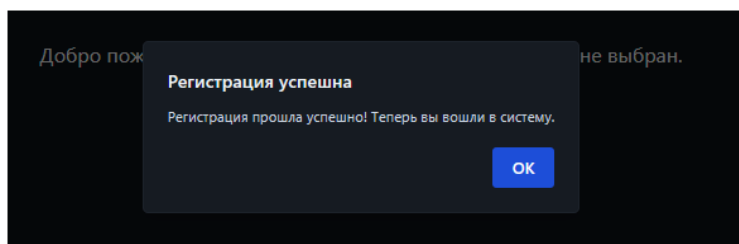


Рисунок 4.7 – Успешная регистрацию

На рисунке 4.8 показано окно подтверждения выхода из профиля.

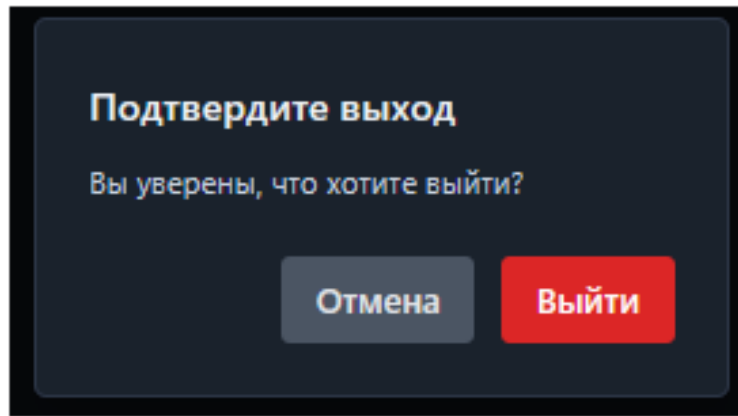


Рисунок 4.8 – Выход из профиля

На рисунке 4.9 показана главная страница веб-приложение.

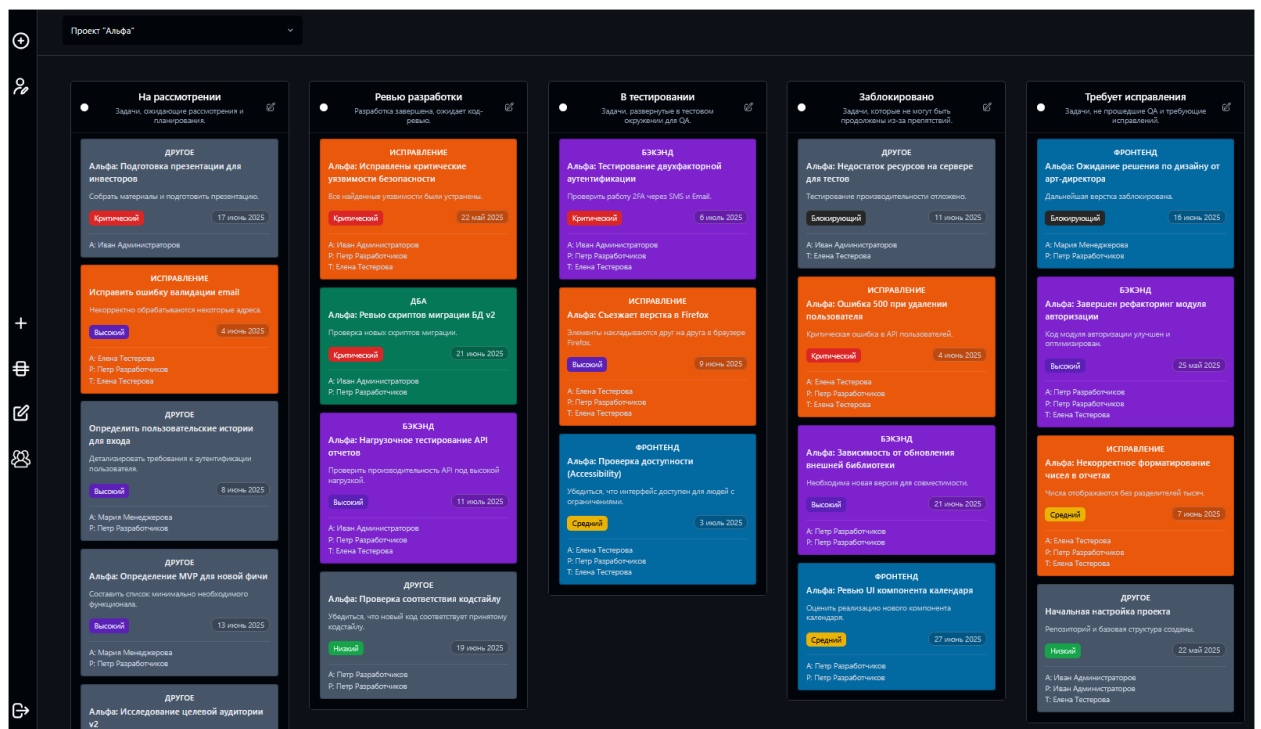


Рисунок 4.9 – Главная страница веб-приложения

На рисунке 4.10 показано окно создания проекта.

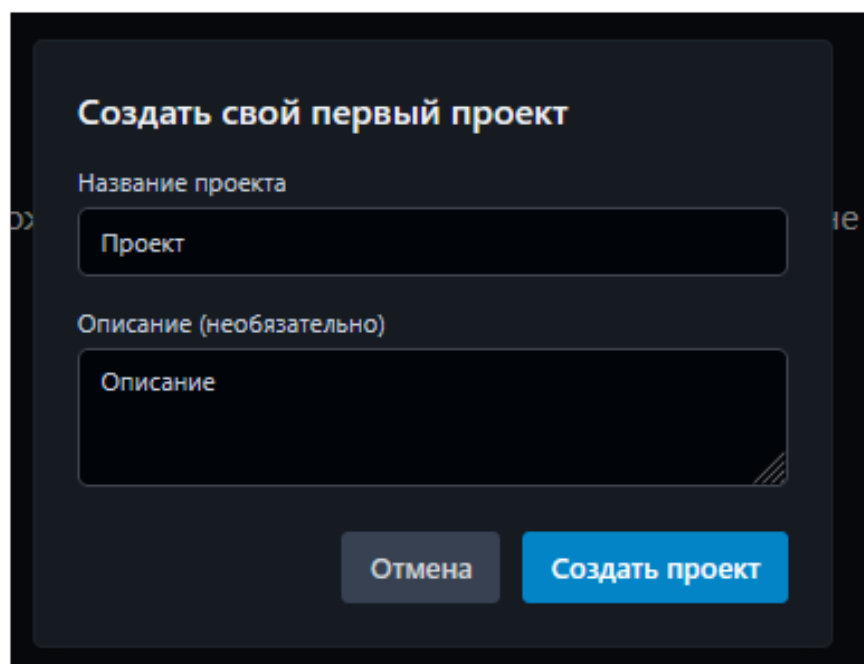
A dark-themed user interface for creating a project. At the top, the title 'Создать свой первый проект' is displayed in white. Below it, the label 'Название проекта' is followed by a text input field containing the word 'Проект'. Underneath, the label 'Описание (необязательно)' is followed by a larger text area containing the word 'Описание'. At the bottom right, there are two buttons: a grey 'Отмена' button and a blue 'Создать проект' button.

Рисунок 4.10 – Создание проекта

На рисунке 4.11 показано раскрываемое поле, отображающее все проекты, доступные пользователю.

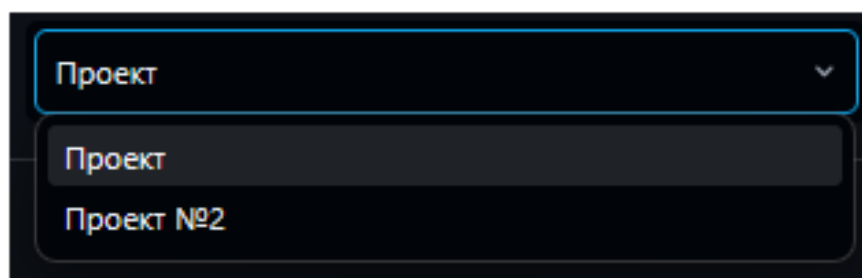
A dark-themed dropdown menu. The top bar shows 'Проект' with a downward arrow. The menu is open, showing a list of items: 'Проект' and 'Проект №2'.

Рисунок 4.11 – Список проектов

На рисунке 4.12 показана поясняющая надпись у элементов панели инструментов.

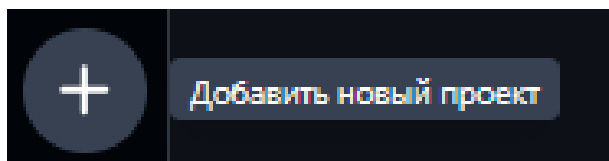


Рисунок 4.12 – Поясняющая надпись

На рисунке 4.13 показана проваленная валидация в окне создания создания проекта.

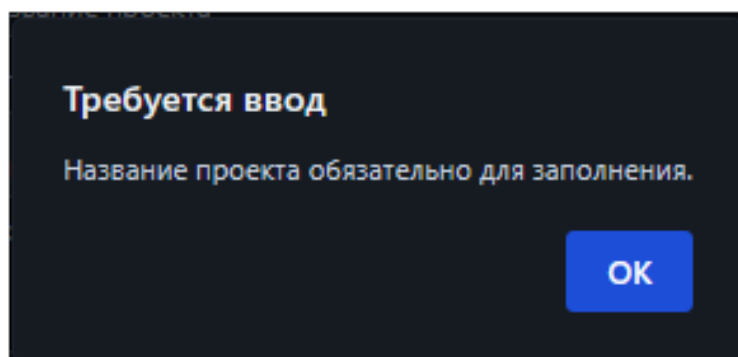


Рисунок 4.13 – Проваленная валидация на название проекта

На рисунке 4.14 показано окно создания доски в выбранном проекте.

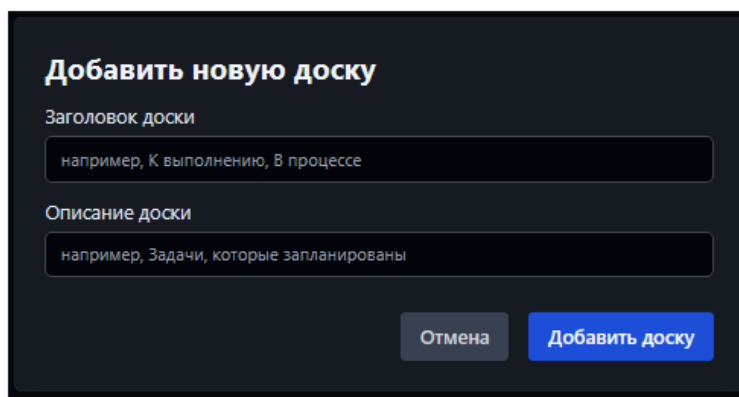


Рисунок 4.14 – Создание доски

На рисунке 4.15 показана проваленная валидация в окне создания доски.

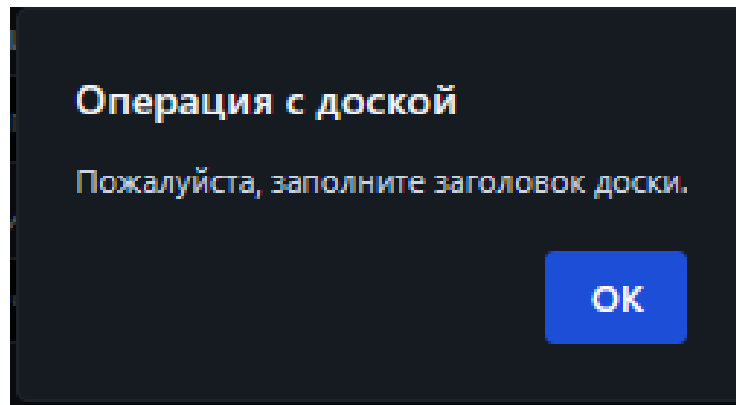


Рисунок 4.15 – Проваленная валидация на заголовок доски

На рисунке 4.16 показана успешно созданная в выбранном проекте доска.

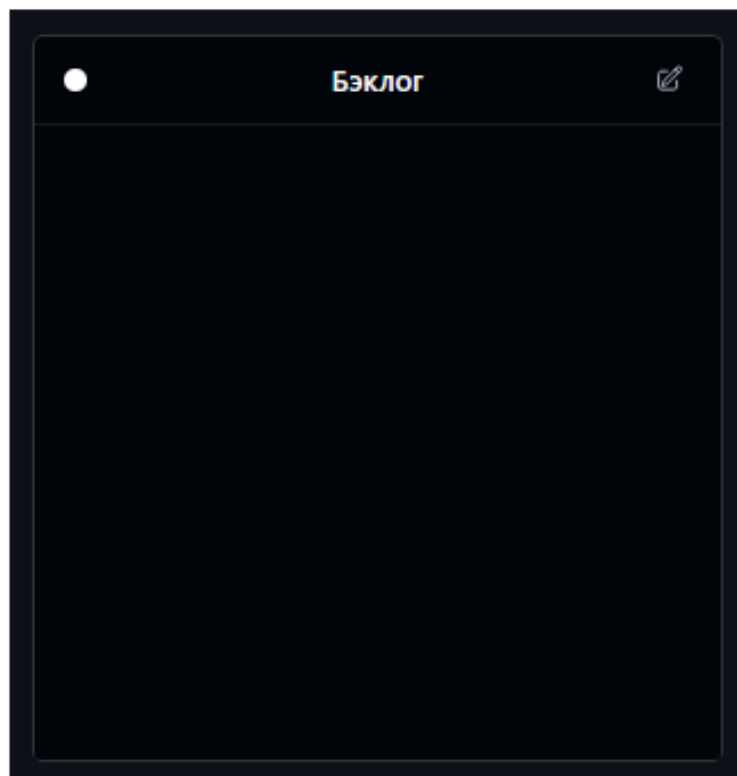


Рисунок 4.16 – Успешно созданная доска

На рисунке 4.17 показано окно редактирования доски проекта.

Редактировать "В разработке"

Заголовок доски

В разработке

Описание доски

Задачи, над которыми активно ведется работа.

Удалить доску Отмена Сохранить

Рисунок 4.17 – Редактирование доски

На рисунке 4.18 показано окно создания задачи в выбранном проекте.

Добавить новую задачу

Доска

Бэклог

Заголовок

Заголовок задачи (например, Реализовать страницу входа)

Описание

Описание задачи

Срок выполнения dd . мм . гggg

Приоритет Средний

Тип ФРОНТЕНД

Отмена Добавить задачу

Рисунок 4.18 – Создание задачи

На рисунке 4.19 показана проваленная валидация на заголовок в окне создания задачи.

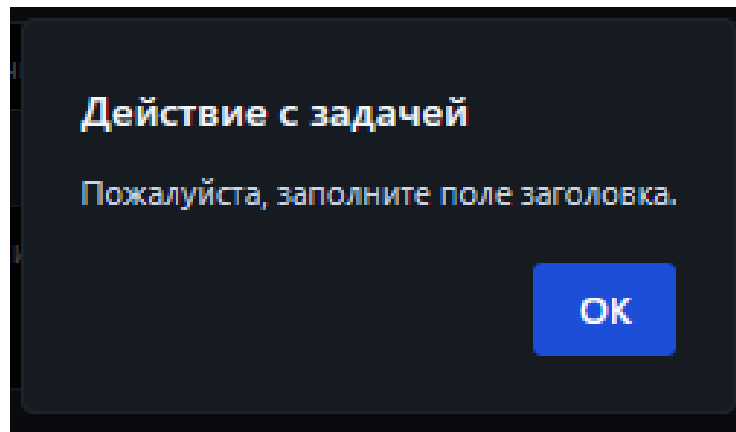


Рисунок 4.19 – Проваленная валидация при создании задачи

На рисунке 4.20 показана проваленная валидация на дату сдачи в окне создания задачи.

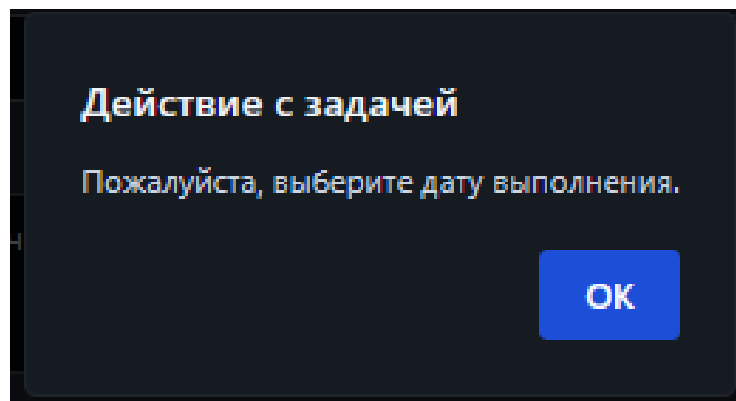


Рисунок 4.20 – Проваленная валидация при создании задачи

На рисунке 4.21 показана проваленная валидация на доску в окне создания задачи.

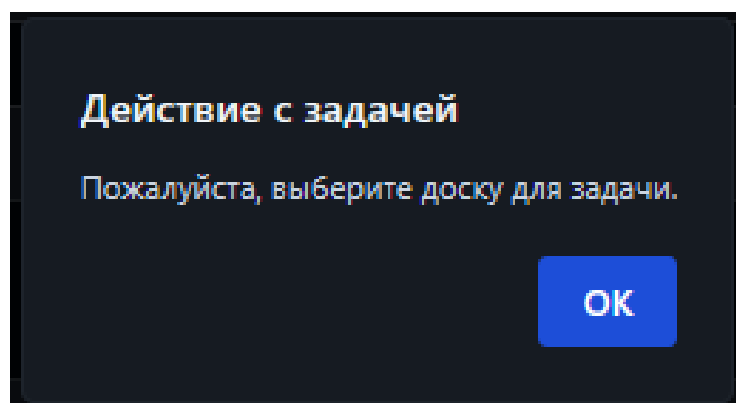


Рисунок 4.21 – Проваленная валидация при создании задачи

На рисунке 4.22 показано отображение успешно созданной задачи.

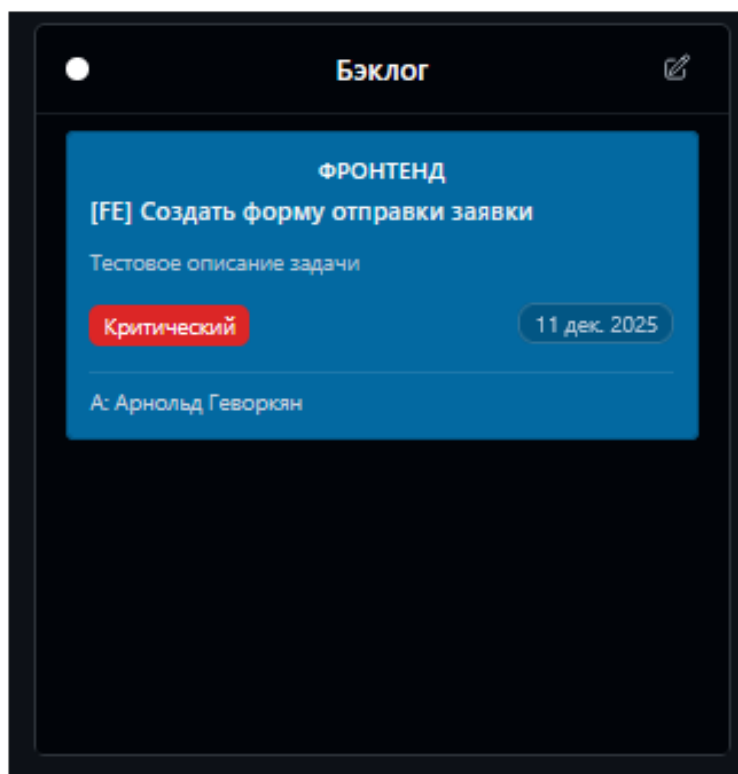


Рисунок 4.22 – Созданная задача

На рисунке 4.23 показано окно подтверждения удаления проекта.

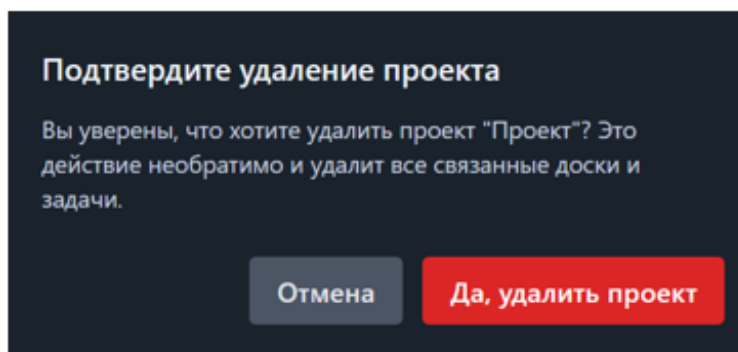


Рисунок 4.23 – Удаление проекта

На рисунках 4.24 и 4.25 показано успешное перемещение задачи между досками.

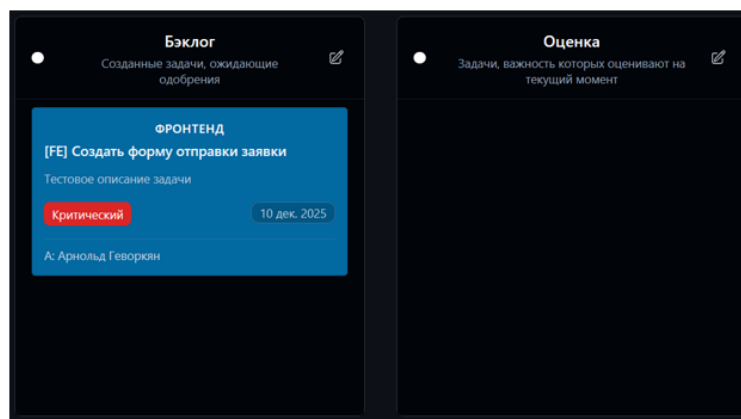


Рисунок 4.24 – Начальная позиция задачи

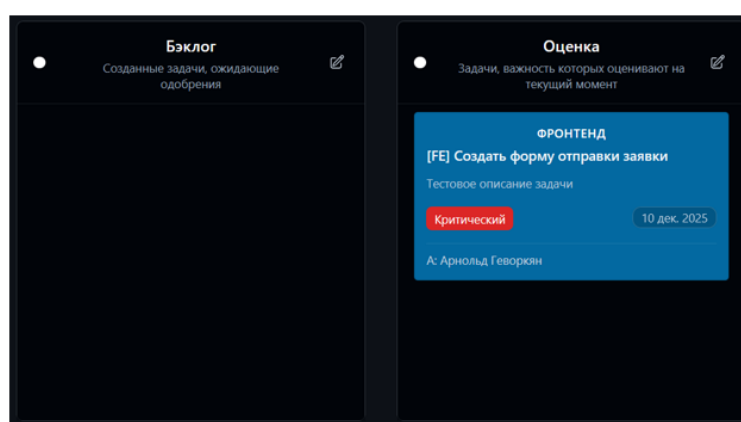


Рисунок 4.25 – Конечная позиция задачи

На рисунках 4.26 и 4.27 показано успешное перемещение досок.



Рисунок 4.26 – Начальные позиции досок

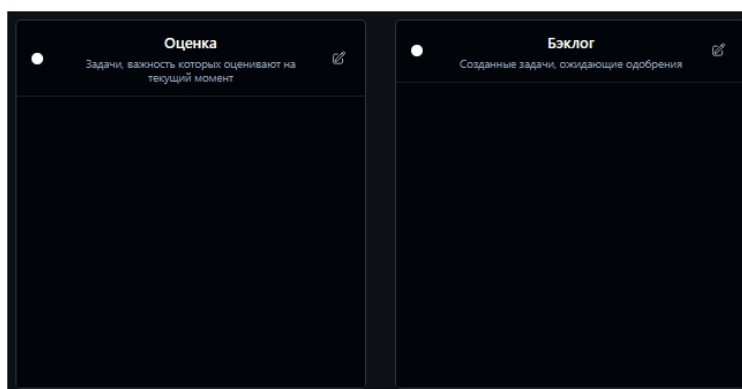


Рисунок 4.27 – Конечные позиции досок

На рисунке 4.28 показано окно редактирования проекта.

Рисунок 4.28 – Редактирование проекта

На рисунке 4.29 показано окно просмотра задачи.

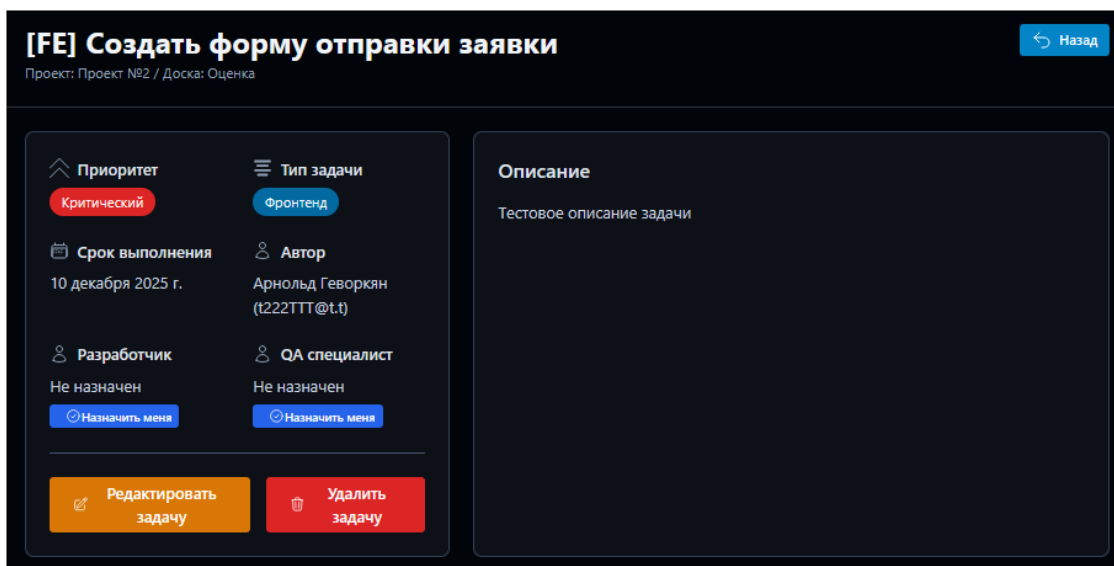


Рисунок 4.29 – Просмотр задачи

На рисунке 4.30 показан результат назначения пользователя на роль исполнителя задачи и соответствующее сообщение.

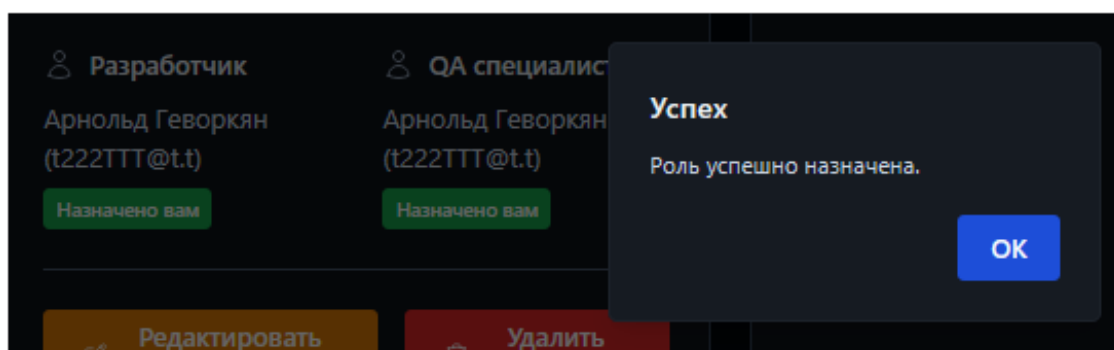


Рисунок 4.30 – Назначение на роль

На рисунке 4.31 показано окно редактирования задачи со всеми доступными полями.

Редактировать задачу: [FE] Создать форму отправки заявки

Заголовок

[FE] Создать форму отправки заявки

Описание

Тестовое описание задачи

Доска

Оценка

Срок выполнения

10.12.2025

Приоритет

Низкий

Отмена Сохранить изменения

Рисунок 4.31 – Редактирование задачи

На рисунке 4.32 показано окно подтверждения удаления задачи.

Подтвердите удаление задачи

Вы уверены, что хотите удалить задачу "[FE] Создать форму отправки заявки"? Это действие необратимо.

Отмена Да, удалить

Рисунок 4.32 – Удаление задачи

На рисунке 4.33 показано окно управления участниками проекта.

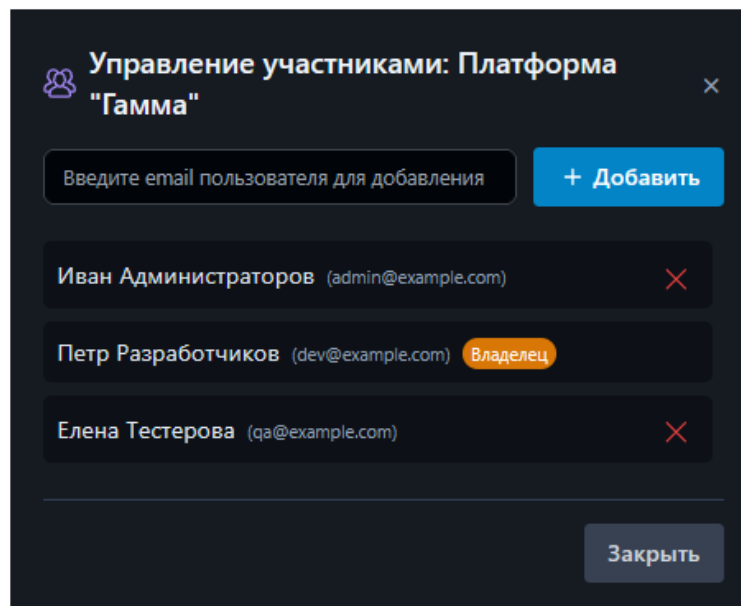


Рисунок 4.33 – Участники проекта

На рисунке 4.34 показано успешное удаление участника проекта.

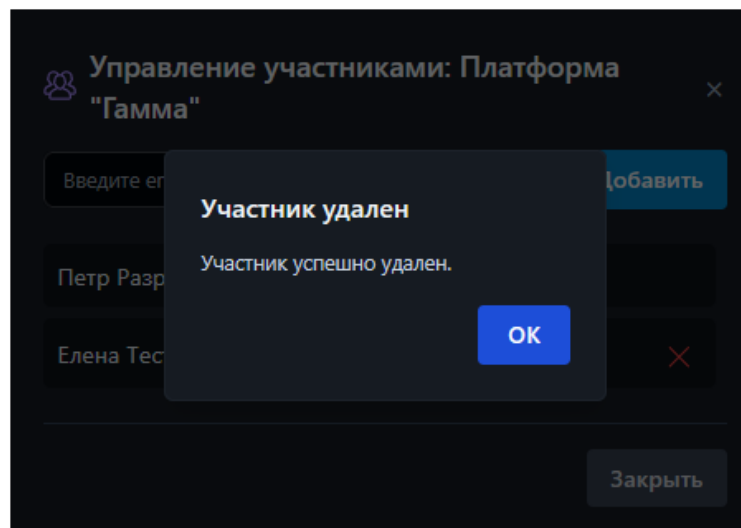


Рисунок 4.34 – Удаление участника

На рисунке 4.35 показана попытка добавления несуществующего участника.

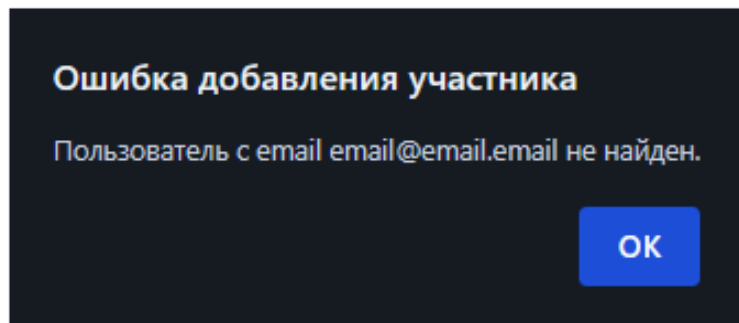


Рисунок 4.35 – Ошибка добавления участника

На рисунке 4.36 показано успешное добавление участника проекта.

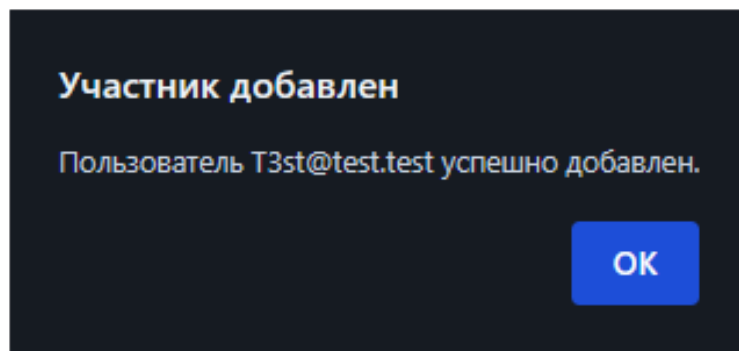


Рисунок 4.36 – Успешное добавление участника

На рисунке 4.37 показано окно профиля пользователя со всей информацией о нём.

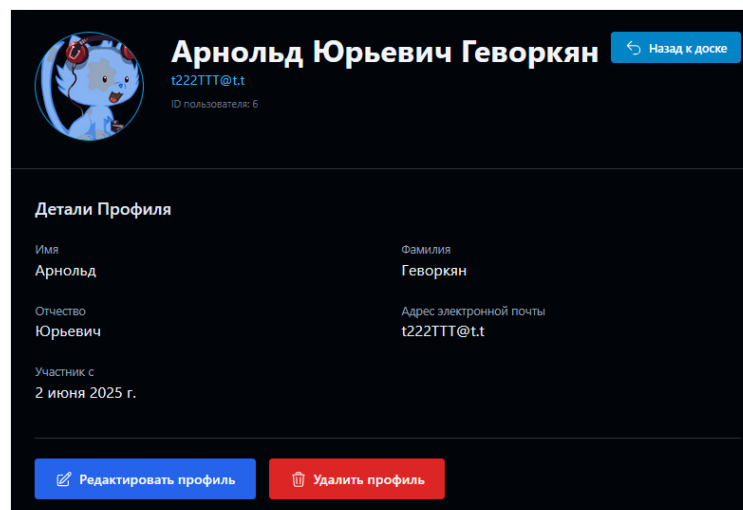
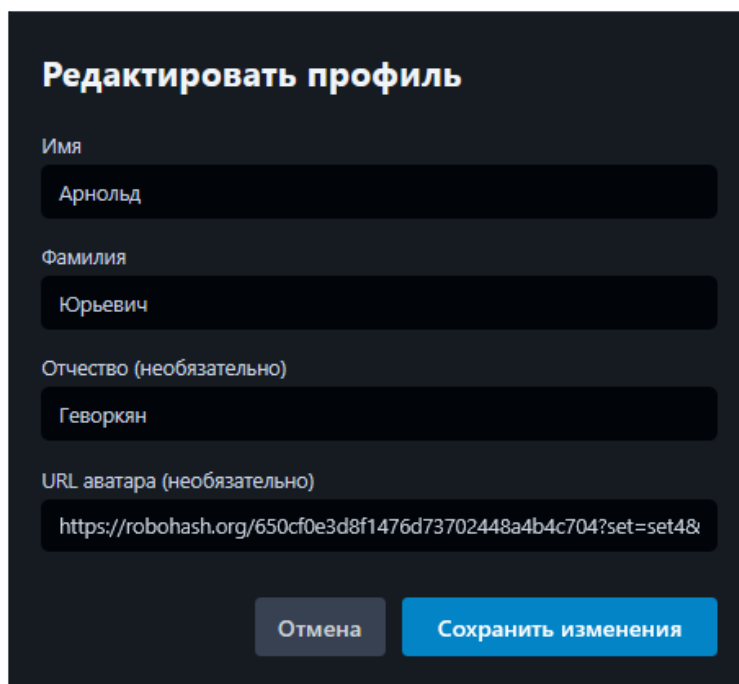


Рисунок 4.37 – Профиль пользователя

На рисунке 4.38 показано окно редактирования профиля пользователя со всеми доступными полями.



Редактировать профиль

Имя
Арнольд

Фамилия
Юрьевич

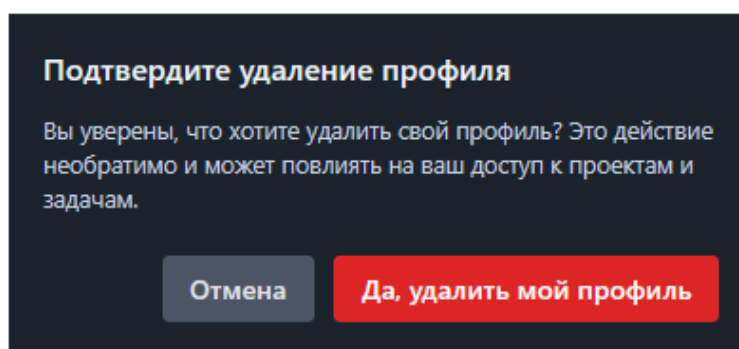
Отчество (необязательно)
Геворкян

URL аватара (необязательно)
<https://robohash.org/650cf0e3d8f1476d73702448a4b4c704?set=set48&>

Отмена Сохранить изменения

Рисунок 4.38 – Редактирование пользователя

На рисунке 4.39 показано окно подтверждения удаления профиля пользователя.



Подтвердите удаление профиля

Вы уверены, что хотите удалить свой профиль? Это действие необратимо и может повлиять на ваш доступ к проектам и задачам.

Отмена Да, удалить мой профиль

Рисунок 4.39 – Удаление профиля

На рисунке 4.40 показано оповещение об успешном удалении пользователя.

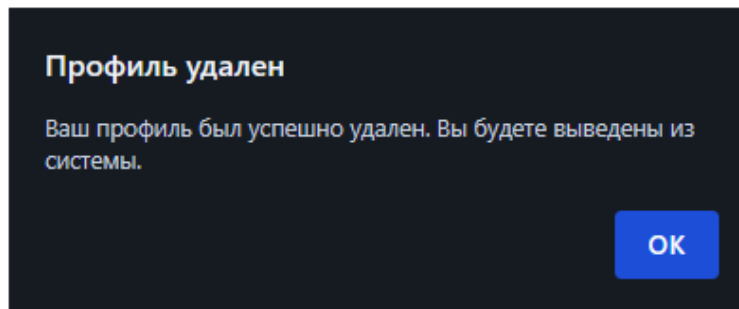


Рисунок 4.40 – Сообщение об успешном удалении

На рисунке 4.41 показано отображение успешно созданной задачи.

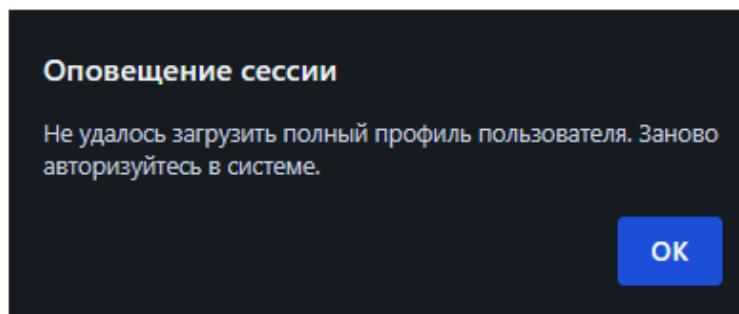


Рисунок 4.41 – Ошибка валидации сессии

На рисунке 4.42 показано оповещение об истекшей сессии авторизации.

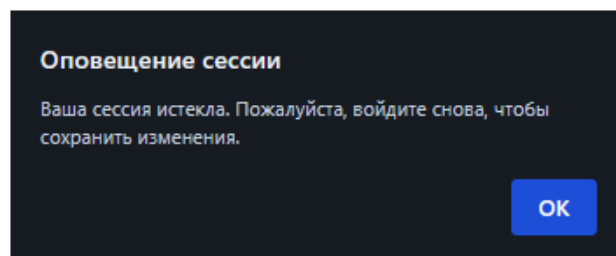


Рисунок 4.42 – Конец сессии

На рисунке 4.43 показана сортировка задач в рамках одной доски по приоритету.

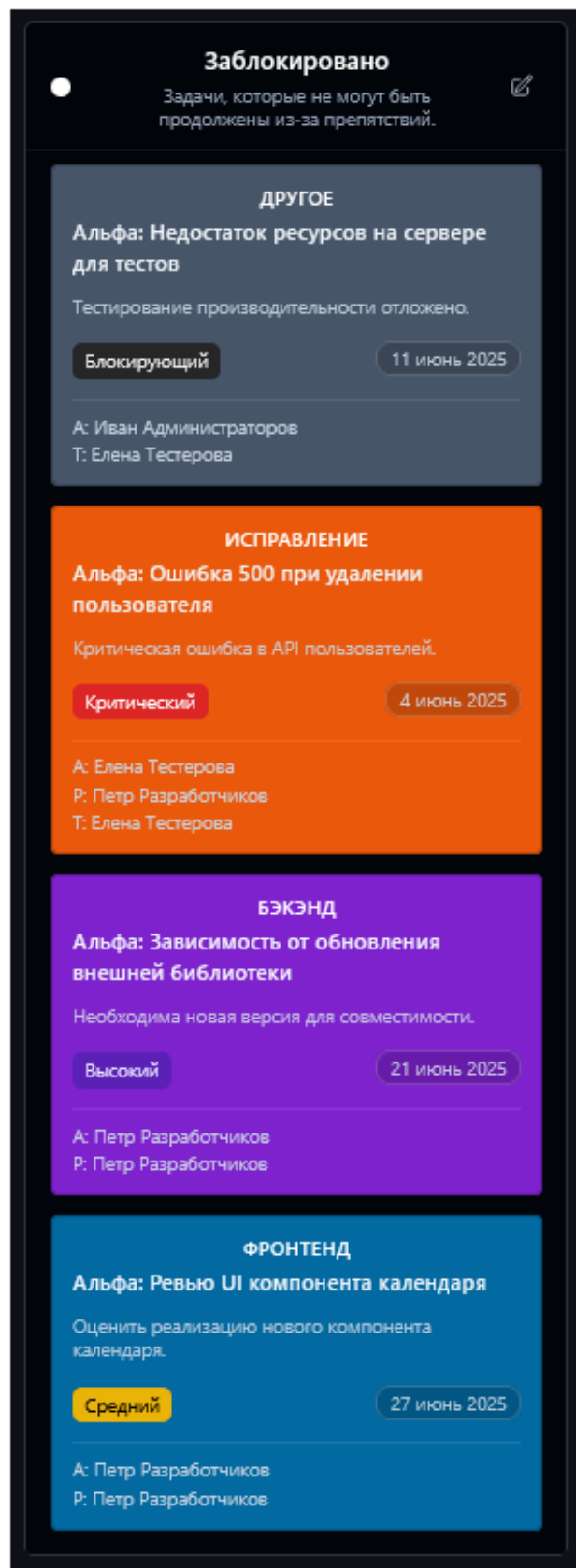


Рисунок 4.43 – Сортировка по приоритету

ЗАКЛЮЧЕНИЕ

В ходе выполнения данной работы была разработана комплексная веб-система для управления проектами по методологии Kanban. Система реализована с использованием современных технологий веб-разработки, включая React для клиентской части, Node.js для серверной части и PostgreSQL в качестве СУБД.

Основные результаты работы:

1. Проведен анализ предметной области. Рассмотрены основные преимущества Kanban-методологии, проанализированы удобства ее применения.

2. Спроектирована и реализована архитектура системы. Разработана клиент-серверная архитектура, обеспечивающая четкое разделение логики представления и бизнес-логики.

3. Разработано веб-приложение с интуитивно понятным пользовательским интерфейсом. С использованием React созданы компоненты для отображения всех необходимых элементов для работы с профилем, проектами, досками и задачами, обеспечивая удобное взаимодействие пользователя с системой.

4. Проведено тестирование и отладка системы. Осуществлена проверка работоспособности основного функционала, включая API-эндпоинты, взаимодействие с базой данных, корректность отображения данных и пользовательского взаимодействия на клиентской стороне.

Все основные требования, характерные для систем управления проектами Kanban-типа, были успешно реализованы. Разработанное приложение представляет собой масштабируемый и гибкий инструмент, предназначенный для эффективной организации командной работы, отслеживания прогресса выполнения задач и визуализации рабочих процессов. Система готова к дальнейшему развитию, включая возможное добавление новых функций и улучшений.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Орлов, С. А. Основы управления ИТ-проектами / С. А. Орлов – Москва : ДМК Пресс, 2023. - 320 с. – ISBN 978-5-9706-0881-4. – Текст : непосредственный.
2. Семенов, М. И. Управление программными проектами: стандарты, инструменты, практика / М. И. Семенов, И. В. Волков – Санкт-Петербург : БХВ-Петербург, 2024. - 450 с. – ISBN 978-5-9775-4321-7. – Текст : непосредственный.
3. Ковалев, А. П. Риск-менеджмент в проектах разработки программного обеспечения / А. П. Ковалев – Москва : Инфра-М, 2022. - 280 с. – ISBN 978-5-16-017854-9. – Текст : непосредственный.
4. Липаев, В. В. Качество программного обеспечения: принципы и методы оценки / В. В. Липаев – Москва : Янус-К, 2023. - 512 с. – ISBN 978-5-8037-0912-5. – Текст : непосредственный.
5. Петров, К. Д. Эффективные коммуникации в ИТ-командах / К. Д. Петров – Москва : Альпина Паблишер, 2024. - 198 с. – ISBN 978-5-9614-8234-1. – Текст : непосредственный.
6. Вигерс, К. Разработка требований к программному обеспечению. 3-е издание / К. Вигерс, Дж. Битти – Москва : БИНОМ. Лаборатория знаний, 2023. - 736 с. – ISBN 978-5-9909700-3-8. – Текст : непосредственный.
7. Группа Акселос. ITIL® 4: Основы / Группа Акселос – Москва : VAN HAREN PUBLISHING, 2024. - 320 с. – ISBN 978-9-40180-888-0. – Текст : непосредственный.
8. Лебедев, А. Н. Лидерство в технологических командах: создание культуры инноваций и сотрудничества / А. Н. Лебедев – Санкт-Петербург : Питер, 2023. - 288 с. – ISBN 978-5-4461-2345-6. – Текст : непосредственный.
9. Ладушкин, В. С. Управление данными и Data Governance: стратегии и внедрение / В. С. Ладушкин – Москва : ДМК Пресс, 2024. - 410 с. – ISBN 978-5-9706-1234-0. – Текст : непосредственный.

10. Ершов, М. П. Основы облачных вычислений: технологии, модели и сервисы / М. П. Ершов – Москва : Техносфера, 2023. - 350 с. – ISBN 978-5-94836-987-0. – Текст : непосредственный.
11. Кон, М. Agile: оценка и планирование проектов / М. Кон – Москва : Вильямс, 2022. - 450 с. – ISBN 978-5-6045556-8-7. – Текст : непосредственный.
12. Вольф, К. С. Пользовательские истории: искусство гибкой разработки ПО / К. С. Вольф, Л. Гофман – Москва : ДМК Пресс, 2023. - 304 с. – ISBN 978-5-9706-0901-9. – Текст : непосредственный.
13. Дерби, Э. Agile-ретроспектива: как превратить хорошую команду в великую / Э. Дерби, Д. Ларсен – Москва : Эксмо, 2024. - 224 с. – ISBN 978-5-04-178834-3. – Текст : непосредственный.
14. Пуппол, М. Принципы Agile. Гибкое управление проектами для начинающих / М. Пуппол – Москва : АСТ, 2023. - 128 с. – ISBN 978-5-17-154321-0. – Текст : непосредственный.
15. Сидоренко, В. П. Коучинг Agile-команд: практическое руководство / В. П. Сидоренко – Москва : Доброе слово, 2024. - 280 с. – ISBN 978-5-9909876-5-4. – Текст : непосредственный.
16. Андерсон, Д. Канбан: альтернативный путь в Agile / Д. Андерсон – Москва : Символ-Плюс, 2023. - 352 с. – ISBN 978-5-93286-234-5. – Текст : непосредственный.
17. Бергманн, Б. Канбан изнутри: два года успешного применения / Б. Бергманн, П. Ахмар – Санкт-Петербург : Питер, 2024. - 240 с. – ISBN 978-5-496-03456-7. – Текст : непосредственный.
18. Леопольд, К. Практический Канбан: от идеи до поставки / К. Леопольд, З. Маурис – Москва : ДМК Пресс, 2022. - 288 с. – ISBN 978-5-9706-0777-0. – Текст : непосредственный.
19. Зайцев, М. В. Канбан для команд разработки ПО: внедрение и оптимизация / М. В. Зайцев – Москва : Техносфера, 2023. - 192 с. – ISBN 978-5-94836-789-0. – Текст : непосредственный.

20. Бенсон, Дж. Персональный Kanban: карта вашей работы / Дж. Бенсон, Т. де Мариа Бенсон – Москва : Эксмо, 2024. - 208 с. – ISBN 978-5-04-165432-1. – Текст : непосредственный.
21. Флэнаган, Д. JavaScript: подробное руководство. 7-е издание / Д. Флэнаган – Санкт-Петербург : Символ-Плюс, 2022. - 720 с. – ISBN 978-5-93286-228-4. – Текст : непосредственный.
22. Хавербеке, М. Выразительный JavaScript. 3-е издание / М. Хавербеке – Санкт-Петербург : Питер, 2023. - 480 с. – ISBN 978-5-4461-1700-0. – Текст : непосредственный.
23. Стефанов, С. JavaScript. Шаблоны / С. Стефанов – Москва : Вильямс, 2024. - 272 с. – ISBN 978-5-8459-2299-9. – Текст : непосредственный.
24. Сойер, К. Современный JavaScript для нетерпеливых / К. Сойер – Санкт-Петербург : БХВ-Петербург, 2024. - 512 с. – ISBN 978-5-9775-1234-5. – Текст : непосредственный.
25. Макфарланд, Д. Новая большая книга CSS / Д. Макфарланд – Москва : Эксмо, 2023. - 640 с. – ISBN 978-5-04-112345-6. – Текст : непосредственный.
26. Бэнкс, А. React и Redux: функциональная веб-разработка / А. Бэнкс, Е. Порселло – Санкт-Петербург : Питер, 2023. - 336 с. – ISBN 978-5-4461-1987-5. – Текст : непосредственный.
27. Фримен, А. React для профессионалов / А. Фримен – Москва : ДМК Пресс, 2024. - 800 с. – ISBN 978-5-9706-0999-6. – Текст : непосредственный.
28. Васильев, Д. А. React. Быстрый старт с хуками и TypeScript / Д. А. Васильев – Москва : Солон-Пресс, 2023. - 352 с. – ISBN 978-5-91359-555-1. – Текст : непосредственный.
29. Шварцмюллер, М. React – полное руководство (включая Hooks, React Router, Redux) / М. Шварцмюллер – Москва : Эксмо, 2025. - 688 с. – ISBN 978-5-04-188765-4. – Текст : непосредственный.
30. Громов, И. П. Сборка современных веб-приложений с Vite: React, Vue, Svelte / И. П. Громов – Санкт-Петербург : Наука и Техника, 2024. - 280 с. – ISBN 978-5-94387-888-9. – Текст : непосредственный.

31. Кантелон, М. Node.js. Разработка серверных приложений / М. Кантелон, Т. Хартер, Н. Раджмохан – Санкт-Петербург : Питер, 2023. - 576 с. – ISBN 978-5-4461-1654-0. – Текст : непосредственный.
32. Чен, Ф. Разработка веб-приложений с помощью Node.js и Express / Ф. Чен – Москва : Вильямс, 2024. - 432 с. – ISBN 978-5-907114-77-7. – Текст : непосредственный.
33. Тейшейра, П. Профессиональный Node.js: создание масштабируемых приложений / П. Тейшейра – Москва : ДМК Пресс, 2022. - 624 с. – ISBN 978-5-9706-0811-1. – Текст : непосредственный.
34. Белов, А. В. Node.js: шаблоны проектирования и лучшие практики / А. В. Белов – Москва : Бином. Лаборатория знаний, 2023. - 368 с. – ISBN 978-5-9963-6789-0. – Текст : непосредственный.
35. Голдберг, А. REST API на Node.js и MongoDB / А. Голдберг – Санкт-Петербург : БХВ-Петербург, 2024. - 312 с. – ISBN 978-5-9775-3210-5. – Текст : непосредственный.
36. Карвин, Б. SQL. Антипаттерны: как избежать ошибок при проектировании баз данных / Б. Карвин – Санкт-Петербург : Питер, 2023. - 320 с. – ISBN 978-5-4461-1543-3. – Текст : непосредственный.
37. Петров, А. И. SQL для анализа данных: практическое руководство / А. И. Петров – Москва : ДМК Пресс, 2024. - 384 с. – ISBN 978-5-9706-1001-5. – Текст : непосредственный.
38. Ульман, Дж. Д. Введение в системы баз данных / Дж. Д. Ульман, Дж. Уидом – Москва : Вильямс, 2021. - 1072 с. – ISBN 978-5-8459-2153-3. – Текст : непосредственный.
39. Дуглас, К. PostgreSQL для начинающих. От основ к профессиональной разработке / К. Дуглас – Москва : Лори, 2024. - 480 с. – ISBN 978-5-85582-456-7. – Текст : непосредственный.
40. Васкес, Л. PostgreSQL. Администрирование баз данных для профессионалов / Л. Васкес – Москва : Символ-Плюс, 2022. - 704 с. – ISBN 978-5-93286-301-4. – Текст : непосредственный.

41. Мартин, Р. Чистая архитектура: Искусство разработки программного обеспечения / Р. Мартин – Санкт-Петербург : Питер, 2023. - 352 с. – ISBN 978-5-4461-0772-8. – Текст : непосредственный.

42. Ким, Дж. Проект «Феникс». Роман о том, как DevOps меняет бизнес к лучшему / Дж. Ким, К. Бер, Дж. Спэффорд – Москва : Манн, Иванов и Фербер, 2022. - 480 с. – ISBN 978-5-00169-567-4. – Текст : непосредственный.

43. Манифест гибкой разработки программного обеспечения : сайт / AgileManifesto.org. – [Б. м.] : AgileManifesto.org, 2001 – . – URL: <https://agilemanifesto.org/iso/ru/manifesto.html> (дата обращения: 17.04.2025). – Текст: электронный.

44. Kanban University : сайт / Kanban University. – [Б. м.] : Kanban University, 2010 – . – URL: <https://kanban.university/> (дата обращения: 19.04.2025). – Текст: электронный.

45. React – официальная документация : сайт / React team. – [Б. м.] : Meta Platforms, Inc., 2013 – . – URL: <https://react.dev/> (дата обращения: 19.04.2025). – Текст: электронный.

46. Vite – официальная документация : сайт / Vite team. – [Б. м.] : Vite team, 2020 – . – URL: <https://vitejs.dev/guide/> (дата обращения: 19.04.2025). – Текст: электронный.

47. Node.js – официальная документация : сайт / OpenJS Foundation. – [Б. м.] : OpenJS Foundation, 2009 – . – URL: <https://nodejs.org/ru/docs/> (дата обращения: 19.04.2025). – Текст: электронный.

48. PostgreSQL – официальная документация : сайт / The PostgreSQL Global Development Group. – [Б. м.] : The PostgreSQL Global Development Group, 1996 – . – URL: <https://www.postgresql.org/docs/> (дата обращения: 23.04.2025). – Текст: электронный.

49. JavaScript | MDN : сайт / Mozilla Developer Network. – [Б. м.] : Mozilla, 2005 – . – URL: <https://developer.mozilla.org/ru/docs/Web/JavaScript> (дата обращения: 18.04.2025). – Текст: электронный.

50. Канбан-метод в разработке: от теории к практике : сайт / Хабр. – Москва : Хабр, 2023 – . – URL: <https://habr.com/ru/companies/otus/articles/780000/> (дата обращения: 17.04.2025). – Текст: электронный.

51. Эффективное управление состоянием в React приложениях : сайт / Smashing Magazine. – Freiburg : Smashing Media AG, 2024 – . – URL: <https://www.smashingmagazine.com/tag/react/> (дата обращения: 19.04.2025). – Текст: электронный.

ПРИЛОЖЕНИЕ А

Представление графического материала

Графический материал, выполненный на отдельных листах, изображен на рисунках А.1–А.6.

Сведения о ВКРБ

Минобрнауки России

Юго-Западный государственный университет

Кафедра программной инженерии

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА ПО ПРОГРАММЕ БАКАЛАВРИАТА

«Программная реализация системы управления проектами»

Руководитель ВКРБ
д.т.н, доцент
Томакова Римма Александровна

Автор ВКРБ
студент группы ПО-116
Геворкян Арнольд Юрьевич

ВКРБ 21060040.09.03.04.25.008			
Аспирант	Васильев В. С.	Васильев В. С.	Васильев В. С.
Руководитель	Томакова Р. А.	Томакова Р. А.	Томакова Р. А.
Рецензент	Томакова Р. А.	Томакова Р. А.	Томакова Р. А.
Сведения о ВКРБ			
Выпускная квалификационная работа бакалавра			
ЮЗГУ ПО-116			

Рисунок А.1 – Сведения о ВКРБ

Цель и задачи разработки

Цель данной работы – разработка веб-приложения для управления проектами по методологии Kanban, предоставляющего пользователям инструменты для эффективной организации, отслеживания и совместной работы над проектными задачами.

В соответствии с целью работы были сформулированы следующие задачи:

- провести анализ предметной области;
- разработать концептуальную модель системы управления проектами на основе Kanban-методологии;
- спроектировать модель системы и веб-приложения;
- реализовать и протестировать программную систему.

				ВКРБ 21060040.09.03.04.25.008			
Автор работы	Колесов В. С.	Научный	Докл.	Цель и задачи разработки			
Рецензент	Колесов А. Ю.						
Рецензент	Колесов В. А.						
Рецензент	Колесов В. А.			Выпускная квалификационная работа бакалавра			
				ЮЗГУ ПО-116			

Рисунок А.2 – Цель и задачи разработки



Рисунок А.3 – Диаграмма прецедентов

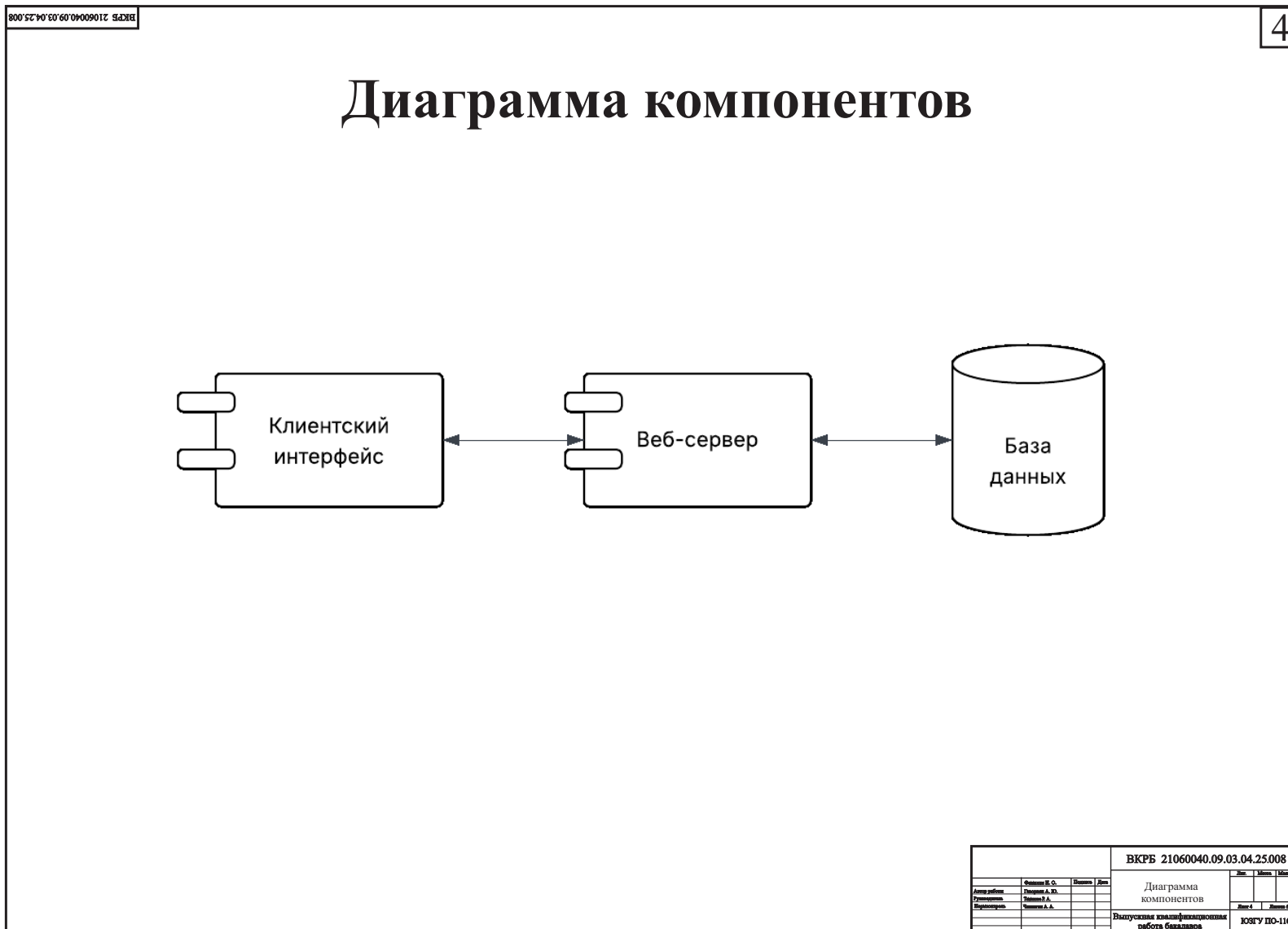
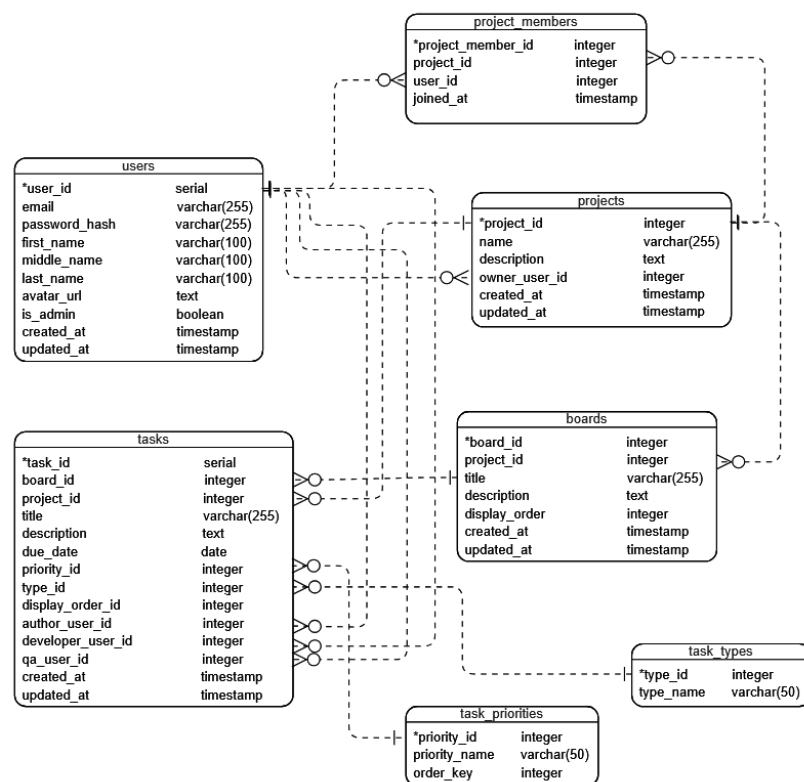


Рисунок А.4 – Диаграмма компонентов

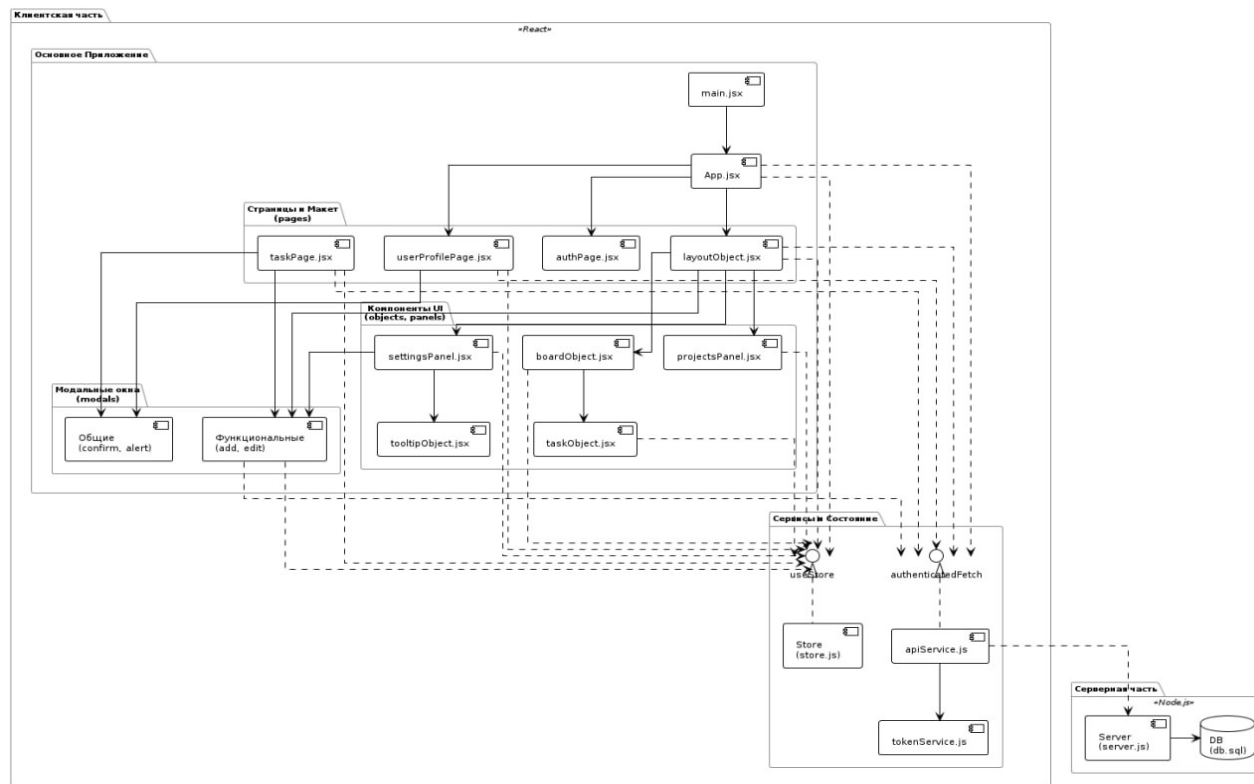
ER – диаграмма



БКРБ 21060040.09.03.04.25.008			
Акт работы	Выполнено	Дата	Место
Утверждено	Выполнено	Дата	Место
Внесено	Выполнено	Дата	Место
Выпускная квалификационная работа бакалавра			ЮЗГУ ПО-116

Рисунок А.5 – ER – диаграмма

Структура модулей



ВКРБ 21060040.09.03.04.25.008			
Автор работы	Петров В. С.	Проверил	Иванов И. И.
Рецензент	Петров В. С.	Проверил	Иванов И. И.
Рецензент	Петров В. С.	Проверил	Иванов И. И.
Выпускная квалификационная работа бакалавра			
ЮЗГУ ПО-116			

Рисунок А.6 – Структура модулей

ПРИЛОЖЕНИЕ Б

Фрагменты исходного кода программы

App.jsx

```
1 // App.jsx
2 // Главный компонент приложения
3 import React, { useState, useEffect, useCallback } from 'react';
4 import { jwtDecode } from 'jwt-decode';
5 import LayoutObject from './components/pages/layoutObject.jsx';
6 import AuthPage from './components/pages/authPage';
7 import UserProfilePage from './components/pages/userProfilePage.jsx';
8 import AlertModal from './components/modals/alertModal.jsx';
9 import AddProjectModal from './components/modals/addProjectModal.jsx';
10 import { useStore } from './store';
11 import { setupGlobalAuthFailureHandler, authenticatedFetch } from './backend/
    services/apiService.js';
12 import ConfirmModal from './components/modals/confirmModal.jsx';
13
14 function App() {
15     const [isAuthenticated, setIsAuthenticated] = useState(!!localStorage.
        getItem('authToken'));
16     const [isLoadingData, setIsLoadingData] = useState(true);
17     const [isAddProjectModalOpenFromEmptyState,
        setIsAddProjectModalOpenFromEmptyState] = useState(false);
18
19     const {
20         tokenExpiredAlert, showTokenExpiredAlert, clearTokenExpiredAlert,
21         registrationAlert, clearRegistrationAlert,
22         projects, setProjects, currentProjectId, setCurrentProjectId,
23         setBoards, setTasks, dataRefreshTrigger,
24         appView, setAppView,
25         setCurrentUserProfile, clearCurrentUserProfile,
26         setAllUsers,
27         logoutConfirmModal, hideLogoutConfirmModal,
28         setMainLogoutHandler,
29     } = useStore(state => ({
30         tokenExpiredAlert: state.tokenExpiredAlert,
31         showTokenExpiredAlert: state.showTokenExpiredAlert,
32         clearTokenExpiredAlert: state.clearTokenExpiredAlert,
33         registrationAlert: state.registrationAlert,
34         clearRegistrationAlert: state.clearRegistrationAlert,
35         projects: state.projects,
36         setProjects: state.setProjects,
37         currentProjectId: state.currentProjectId,
38         setCurrentProjectId: state.setCurrentProjectId,
39         setBoards: state.setBoards,
40         setTasks: state.setTasks,
41         dataRefreshTrigger: state.dataRefreshTrigger,
42         appView: state.appView,
43         setAppView: state.setAppView,
44         setCurrentUserProfile: state.setCurrentUserProfile,
45         clearCurrentUserProfile: state.clearCurrentUserProfile,
46         setAllUsers: state.setAllUsers,
```

```

47     logoutConfirmModal: state.logoutConfirmModal,
48     hideLogoutConfirmModal: state.hideLogoutConfirmModal,
49     setMainLogoutHandler: state.setMainLogoutHandler,
50   }));
51
52   const handleLogout = useCallback(() => {
53     localStorage.removeItem('authToken');
54     setIsAuthenticated(false);
55     setProjects([]);
56     setBoards([]);
57     setTasks([]);
58     setCurrentProjectId(null);
59     clearCurrentUserProfile();
60     setAllUsers([]);
61     setAppView('kanban');
62     hideLogoutConfirmModal();
63     console.log("App.jsx: Пользователь вышел, все данные очищены, вид сброшен");
64   }, [setProjects, setBoards, setTasks, setCurrentProjectId,
        clearCurrentUserProfile, setAppView, hideLogoutConfirmModal, setAllUsers
        ]);
65
66   useEffect(() => {
67     setMainLogoutHandler(handleLogout);
68   }, [handleLogout, setMainLogoutHandler]);
69
70   const forceLogoutAndShowAlert = useCallback((message) => {
71     handleLogout();
72     showTokenExpiredAlert(message);
73   }, [handleLogout, showTokenExpiredAlert]);
74
75   useEffect(() => {
76     setupGlobalAuthFailureHandler(forceLogoutAndShowAlert);
77   }, [forceLogoutAndShowAlert]);
78
79   useEffect(() => {
80     const token = localStorage.getItem('authToken');
81     let isActiveAttempt = true;
82
83     if (!token) {
84       setIsAuthenticated(false);
85       setIsLoadingData(false);
86       clearCurrentUserProfile();
87       setAllUsers([]);
88       setProjects([]);
89       setBoards([]);
90       setTasks([]);
91       setCurrentProjectId(null);
92       return;
93     }
94
95     setIsLoadingData(true);
96
97     const verifyTokenAndFetchInitialData = async (currentToken) => {

```

```

98   try {
99     const decodedToken = jwtDecode(currentToken);
100    const currentTime = Date.now() / 1000;
101
102    if (decodedToken.exp < currentTime) {
103      if (isActiveAttempt) forceLogoutAndShowAlert("Ваша сессия истекла.
104        Пожалуйста, войдите снова.");
105      return;
106    }
107
108    if (isActiveAttempt) {
109      if (!isAuthenticated) setIsAuthenticated(true);
110      console.log("App.jsx: Токен пользователя действителен. Загрузка
111        начальных данных...");
112    }
113
114    const [profileResponse, projectsResponse, allUsersResponse] = await
115      Promise.all([
116        authenticatedFetch('/api/users/me'),
117        authenticatedFetch('/api/projects'),
118        authenticatedFetch('/api/users')
119      ]);
120
121    if (isActiveAttempt) {
122      try {
123        if (!profileResponse.ok) {
124          const errorBody = await profileResponse.text().catch(() => "Не
125            удалось прочитать тело ответа с ошибкой для загрузки профиля
126            .");
127          console.error(`App.jsx: Не удалось загрузить профиль
128            пользователя. Статус: ${profileResponse.status}, Тело: ${
129              errorBody}`);
130          const profileError = new Error(`Ошибка загрузки профиля: ${
131            profileResponse.status} - ${errorBody}`);
132          profileError.isProfileError = true;
133          profileError.status = profileResponse.status;
134          throw profileError;
135        }
136        const userProfile = await profileResponse.json();
137        setCurrentUserProfile(userProfile);
138        console.log("App.jsx: Профиль пользователя загружен с API и
139          установлен.", userProfile);
140      } catch (profileError) {
141        console.error("App.jsx: Ошибка во время загрузки профиля
142          пользователя:", profileError);
143        const isAuthRelatedError = profileError.status === 401 ||
144          profileError.status === 403 || (profileError.message && (
145            profileError.message.toLowerCase().includes('unauthorized') ||
146            profileError.message.toLowerCase().includes('session') ||
147            profileError.message.toLowerCase().includes('не авторизован'))
148          );
149        if (isAuthRelatedError) {
150          throw profileError;
151        } else {

```



```

137         console.warn("App.jsx: Используются резервные данные профиля из
            токена из-за ошибки API при загрузке профиля.
            Принудительный выход.");
138         const fallbackProfile = {
139             id: decodedToken.userId, email: decodedToken.email,
140             firstName: decodedToken.firstName || "Пользователь", lastName
                : decodedToken.lastName || "",
141             middleName: decodedToken.middleName || "", avatarUrl: null,
142             createdAt: new Date().toISOString(),
143         };
144         if (isActiveAttempt) {
145             setCurrentUserProfile(fallbackProfile);
146             if (!tokenExpiredAlert.show) {
147                 forceLogoutAndShowAlert("Не удалось загрузить полный
                    профиль пользователя. Заново авторизуйтесь в системе.");
148             }
149         }
150         return;
151     }
152 }
153 }
154
155 if (isActiveAttempt) {
156     if (!projectsResponse.ok) {
157         const errorBody = await projectsResponse.text().catch(() => "Не
            удалось прочитать тело ответа с ошибкой для загрузки проектов.
            ");
158         console.error(`App.jsx: Не удалось загрузить проекты. Статус: ${
            projectsResponse.status}, Тело: ${errorBody}`);
159         setProjects([]);
160     } else {
161         const fetchedProjects = await projectsResponse.json();
162         setProjects(fetchedProjects);
163         console.log(`App.jsx: Проекты (${fetchedProjects.length})
            загружены.`);
164     }
165 }
166
167 if (isActiveAttempt) {
168     if (!allUsersResponse.ok) {
169         const errorBody = await allUsersResponse.text().catch(() => "Не
            удалось прочитать тело ответа с ошибкой для загрузки списка
            пользователей.");
170         console.error(`App.jsx: Не удалось загрузить список пользователей
            . Статус: ${allUsersResponse.status}, Тело: ${errorBody}`);
171         setAllUsers([]);
172     } else {
173         const fetchedAllUsers = await allUsersResponse.json();
174         setAllUsers(fetchedAllUsers);
175         console.log(`App.jsx: Список всех пользователей (${
            fetchedAllUsers.length}) загружен.`);
176     }
177 }
178

```

```

179
180   } catch (error) {
181     console.error("App.jsx: КРИТИЧЕСКАЯ ОШИБКА во время процесса
182       начальной загрузки данных или валидации токена:", error);
183     if (isActiveAttempt) {
184       const errorMessageText = error.message ? error.message.toLowerCase
185         () : "";
186       const isAuthError = errorMessageText.includes('unauthorized') ||
187         errorMessageText.includes('session') || errorMessageText.
188         includes('не авторизован') || error.status === 401 || error.
189         status === 403;
190
191       if (!tokenExpiredAlert.show && !isAuthError) {
192         showTokenExpiredAlert("Не удалось загрузить начальные данные
193           приложения. Пожалуйста, попробуйте позже.");
194       } else if (isAuthError && !tokenExpiredAlert.show) {
195         forceLogoutAndShowAlert(errorMessageText.includes("expired") ||
196           errorMessageText.includes("истек") ? "Ваша сессия истекла.
197             Пожалуйста, войдите снова." : "Ошибка аутентификации.
198               Пожалуйста, войдите снова.");
199       }
200     } finally {
201       if (isActiveAttempt) setIsLoadingData(false);
202     }
203   };
204
205   verifyTokenAndFetchInitialData(token);
206
207   return () => { isActiveAttempt = false; };
208 }, [isAuthenticated, dataRefreshTrigger, clearCurrentUserProfile,
209   setProjects, setBoards, setTasks, setCurrentProjectId,
210   showTokenExpiredAlert, forceLogoutAndShowAlert, setCurrentUserProfile,
211   setAllUsers, tokenExpiredAlert.show, handleLogout]);
212
213 useEffect(() => {
214   let isActiveAttempt = true;
215   const fetchBoardAndTaskData = async () => {
216     if (isAuthenticated && currentProjectId !== null) {
217       console.log(`App.jsx: currentProjectId равен ${currentProjectId}.
218         Загрузка досок и задач.`);
219       setIsLoadingData(true);
220       try {
221         const [boardsResponse, tasksResponse] = await Promise.all([
222           authenticatedFetch('/api/boards'),
223           authenticatedFetch('/api/tasks')
224         ]);
225
226         if (!boardsResponse.ok) {
227           const errorText = await boardsResponse.text();
228           throw new Error(`Не удалось загрузить доски: ${boardsResponse.
229             status} - ${errorText}`);
230         }
231
232         if (!tasksResponse.ok) {

```

```

219     const errorText = await tasksResponse.text();
220     throw new Error(`Не удалось загрузить задачи: ${tasksResponse.
        status} - ${errorText}`);
221 }
222
223 const fetchedBoards = await boardsResponse.json();
224 const fetchedTasks = await tasksResponse.json();
225
226 if (isActiveAttempt) {
227     const currentProjectBoards = fetchedBoards.filter(b => b.
        projectId === currentProjectId);
228     const currentProjectBoardIds = currentProjectBoards.map(b => b.id
        );
229     const currentProjectTasks = fetchedTasks.filter(t =>
        currentProjectBoardIds.includes(t.boardId));
230
231     setBoards(currentProjectBoards);
232     setTasks(currentProjectTasks);
233     console.log(`App.jsx: Для проекта ${currentProjectId}, Доски (${
        currentProjectBoards.length}) и Задачи (${currentProjectTasks.
        length}) установлены.`);
234 }
235 } catch (error) {
236     console.error(`App.jsx: Ошибка загрузки данных для проекта ${
        currentProjectId}:`, error);
237     const errorMessageText = error.message ? error.message.toLowerCase
        () : "";
238     const isAuthError = errorMessageText.includes('unauthorized') ||
        errorMessageText.includes('session') || errorMessageText.
        includes('не авторизован');
239     if (isActiveAttempt && !isAuthError && !tokenExpiredAlert.show) {
240         showTokenExpiredAlert("Не удалось загрузить детали проекта.
            Пожалуйста, попробуйте еще раз.");
241     }
242     if (isActiveAttempt && !isAuthError) {
243         setBoards([]);
244         setTasks([]);
245     }
246 } finally {
247     if (isActiveAttempt) setIsLoadingData(false);
248 }
249 } else if (!isAuthenticated && currentProjectId === null) {
250     setBoards([]);
251     setTasks([]);
252 }
253 };
254
255 if (isAuthenticated && currentProjectId !== null) {
256     fetchBoardAndTaskData();
257 } else if (!isAuthenticated) {
258     setBoards([]);
259     setTasks([]);
260 }
261 return () => { isActiveAttempt = false; setIsLoadingData(false); };

```

```

262 }, [isAuthenticated, currentProjectId, dataRefreshTrigger,
    tokenExpiredAlert.show, setBoards, setTasks, showTokenExpiredAlert]);
263
264
265 const handleLoginSuccess = (token) => {
266   if (token) {
267     localStorage.setItem('authToken', token);
268     setIsAuthenticated(true);
269     setAppView('kanban');
270     console.log("App.jsx: Вход успешен, токен сохранен. Главный useEffect
        перезапустит загрузку данных.");
271   } else {
272     console.error("App.jsx: handleLoginSuccess вызван без токена.");
273   }
274 };
275
276 if (isLoadingData && localStorage.getItem('authToken')) {
277   return <div className="app-container bg-secBlack h-screen w-screen flex
        items-center justify-center text-white text-xl">Загрузка данных...</
        div>;
278 }
279
280 let content;
281 if (isAuthenticated) {
282   if (appView === 'profile') {
283     content = <UserProfilePage />;
284   } else if (appView === 'kanban' || appView === 'taskView') {
285     if (currentProjectId !== null && projects.length > 0) {
286       content = <LayoutObject onLogout={handleLogout} />;
287     } else if (appView === 'kanban' && (currentProjectId === null ||
        projects.length === 0)) {
288       content = (
289         <div className="app-container bg-secBlack h-screen w-screen flex
            flex-col items-center justify-center text-white">
290           <p className="text-xl mb-4">Добро пожаловать! У вас пока нет
                проектов или ни один не выбран.</p>
291           <p className="mb-6">Пожалуйста, создайте проект, чтобы начать.</p>
                >
292           <button
                onClick={() => setIsAddProjectModalOpenFromEmptyState(true)}
                className="px-4 py-2 bg-sky-600 hover:bg-sky-700 rounded text-
                white font-semibold"
293             >
294             Создать свой первый проект
295           </button>
296           <button onClick={handleLogout} className="mt-4 text-sky-400 hover
                :text-sky-300">Выйти</button>
297         </div>
298       );
299     } else {
300       content = <LayoutObject onLogout={handleLogout} />;
301     }
302   } else {
303
304

```

```

305     console.warn(`App.jsx: Неизвестный appView "${appView}", переключение
        на 'kanban'.`);
306     setAppView('kanban');
307     content = <LayoutObject onLogout={handleLogout} />;
308   }
309   } else {
310     content = <AuthPage onLoginSuccess={handleLoginSuccess} />;
311   }
312
313   return (
314     <div className="app-container bg-secBlack h-screen w-screen overflow -
        hidden">
315       {content}
316       <AddProjectModal
317         isOpen={isAddProjectModalOpenFromEmptyState}
318         onClose={() => setIsAddProjectModalOpenFromEmptyState(false)}
319       >
320         <h2 className="text-xl font-semibold text-slate-100 mb-4">Создать
            свой первый проект</h2>
321       </AddProjectModal>
322       <AlertModal isOpen={registrationAlert.show} onClose={
            clearRegistrationAlert} title={registrationAlert.title} message={
            registrationAlert.message} />
323       <AlertModal isOpen={tokenExpiredAlert.show} onClose={
            clearTokenExpiredAlert} title="Оповещение сессии" message={
            tokenExpiredAlert.message} />
324       <ConfirmModal
325         isOpen={logoutConfirmModal.isOpen}
326         onClose={hideLogoutConfirmModal}
327         onConfirm={handleLogout}
328         title="Подтвердите выход"
329         message="Вы уверены, что хотите выйти?"
330         confirmButtonText="Выйти"
331         cancelButtonText="Отмена"
332       />
333     </div>
334   );
335 }
336
337 export default App;

```

Автор ВКР

(подпись, дата)

А. Ю. Геворкян

Руководитель ВКР

(подпись, дата)

Р. А. Томакова

Нормоконтроль

(подпись, дата)

А. А. Чаплыгин

Место для диска