МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
 РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«КУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет физики, математики, информатики

Кафедра программного обеспечения и администрирования информационных

систем

КУРСОВОЙ ПРОЕКТ

по дисциплине

«Основы проектирования информационных систем и баз данных»

на тему: ИНФОРМАЦИОННАЯ СИСТЕМА УПРАВЛЕНИЯ ПРОЕКТАМИ

Обучающегося 4 курса   
очной формы обучения направления подготовки

02.03.03 Математическое обеспечение и   
администрирование информационных   
систем

Направленность (профиль)   
Проектирование информационных   
систем и баз данных

Лебедева Данилы Викторовича

Руководитель: к.т.н., доцент кафедры ПОиАИС

Кривонос Алексей Владимирович

Допустить к защите:

/ Кривонос А.В./

«30» декабря 2024 г.

Курск, 2024

# содержание

[содержание 2](#_Toc187911228)

[введение 3](#_Toc187911229)

[1 Анализ требований к информационной системе 5](#_Toc187911230)

[1.1 Описание и анализ предметной области 5](#_Toc187911231)

[1.2 Обзор и анализ возможных альтернатив 7](#_Toc187911232)

[1.3 Анализ функциональных и эксплуатационных требований 11](#_Toc187911233)

[2 Проектирование информационной системы 44](#_Toc187911234)

[2.1 Разработка архитектуры технического обеспечения системы 44](#_Toc187911235)

[2.2 Разработка модели предметной области 44](#_Toc187911236)

[2.3 Проектирование классов предметной области 47](#_Toc187911237)

[2.4 Построение диаграммы кооперации 50](#_Toc187911238)

[2.5 Построение диаграммы классов 59](#_Toc187911239)

[2.6 Уточнение структуры классов предметной области и разработка алгоритмов методовт 61](#_Toc187911240)

[2.7 Проектирование интерфейса пользователя 63](#_Toc187911241)

[2.8 Разработка алгоритма функционирования системы 64](#_Toc187911242)

[3 Реализация системы 66](#_Toc187911243)

[3.1 Реализация программного обеспечения системы 66](#_Toc187911244)

[3.2 Реализация технического обеспечения 68](#_Toc187911245)

[3.3 Объекты интерфейса пользователя 69](#_Toc187911246)

[заключение 77](#_Toc187911247)

[список использованных источников 78](#_Toc187911248)

[приложение а 79](#_Toc187911249)

[приложение б 102](#_Toc187911250)

# введение

Современный мир тесно связан с информационными технологиями, и их роль в различных сферах бизнеса и жизни людей постоянно растет. IT-компании нуждаются в надежных и эффективных инструментах для организации бизнес-процессов и управления проектами. В связи с этим разработка специализированной информационной системы для сопровождения IT-проектов является актуальной задачей для организаций, работающих в данной области.

Одним из наиболее востребованных подходов к управлению проектами в сфере разработки программного обеспечения является методология Scrum в сочетании с Канбан-досками. Эти методы относятся к гибким (Agile) подходам, обеспечивая адаптивность и гибкость процесса разработки. Современные программные продукты должны соответствовать принципам Agile, что позволяет повысить их качество и адаптируемость.

Создание информационной системы для управления проектами позволит автоматизировать и упростить ключевые процессы, включая сбор, хранение и обработку данных, контроль выполнения задач, планирование и мониторинг хода работ. Кроме того, такая система способствует улучшению взаимодействия между участниками проекта и повышению общей эффективности работы.

При отсутствии специализированного программного обеспечения для управления IT-проектами компании могут сталкиваться с различными сложностями, такими как нерациональное распределение ресурсов, потеря данных, низкая прозрачность процессов и отсутствие точного прогнозирования сроков и бюджета. Внедрение информационной системы для сопровождения IT-проектов поможет избежать этих проблем и повысить продуктивность бизнес-процессов.

Цель работы - разработка современного веб-приложения для управления проектами, которое позволит эффективно организовывать, контролировать и анализировать задачи, проекты и команды. Приложение направлено на упрощение процессов планирования, распределения задач, отслеживания прогресса и генерации отчетов, что повысит продуктивность работы команд и улучшит управление проектами в организациях.

Структура курсового проекта включает следующие задачи:

1. Аналитический обзор предметной области, где рассматриваются существующие проблемы информационных систем по управлению проектами.
2. Проектирование информационной системы, включающее описание функциональных требований, архитектуры и основных компонентов системы.
3. Техническая реализация, в которой представлены технологии, инструменты и методы разработки.

# 1 Анализ требований к информационной системе

## 1.1 Описание и анализ предметной области

В условиях активного развития технологий и роста сложности бизнес-процессов управление проектами становится ключевой задачей для многих организаций. Компании, работающие в сфере IT, строительства, маркетинга и других отраслей, нуждаются в эффективных инструментах для координации задач, контроля сроков, распределения ресурсов и взаимодействия между участниками проекта. Программные системы управления проектами призваны автоматизировать эти процессы, минимизировать риски и повышать общую продуктивность команд.

Традиционные методы управления проектами, включающие использование электронных таблиц, бумажной документации и ручного контроля сроков, имеют ряд значительных недостатков. Такие подходы не позволяют эффективно управлять изменениями, усложняют коммуникацию между участниками и затрудняют мониторинг хода выполнения задач. В связи с этим возникает необходимость в специализированных программных продуктах, которые объединяют в себе инструменты для планирования, анализа и контроля всех этапов проекта.

На сегодняшний день существует множество решений в области управления проектами. Среди наиболее популярных программ можно выделить Jira, Trello, Microsoft Project. Эти системы предоставляют широкий функционал, включая создание задач, назначение ответственных, отслеживание сроков, управление ресурсами и интеграцию с другими сервисами. Однако у каждого решения есть свои ограничения: некоторые системы сложны в освоении, другие недостаточно гибкие для адаптации под специфические требования компаний.

Основные проблемы, с которыми сталкиваются пользователи существующих систем, включают:

* Ограниченная гибкость настроек – многие платформы имеют фиксированные рабочие процессы, которые не всегда подходят для уникальных требований бизнеса.
* Сложность освоения – некоторые системы требуют значительного времени на обучение пользователей, что замедляет процесс внедрения.
* Высокая стоимость – профессиональные решения могут быть недоступны для малого и среднего бизнеса.
* Фрагментация инструментов – необходимость использовать несколько разных сервисов для полноценного управления проектами (например, отдельные решения для задач, коммуникации, документации и отчетности).

Для решения этих проблем необходима разработка системы управления проектами, которая объединит основные требования бизнеса: интуитивно понятный интерфейс, гибкую настройку рабочих процессов, поддержку методологий Agile, Scrum и Kanban, а также интеграцию с популярными инструментами (например, Google Drive, GitHub и др.).

На основе анализа предметной области можно выделить ключевые задачи, которые должна решать современная система управления проектами:

1. Автоматизация процессов планирования и контроля – система должна помогать проектным командам эффективно распределять ресурсы и управлять задачами.
2. Гибкость методологий – поддержка различных подходов к управлению проектами (водопадная модель, Agile, Kanban, Scrum).
3. Интерактивные инструменты визуализации – доски задач, диаграммы Ганта, календари и отчеты, позволяющие наглядно представлять текущее состояние проекта.
4. Интеграция с внешними сервисами – возможность подключения систем коммуникации, облачного хранения данных и инструментов разработки.
5. Механизмы аналитики и отчетности – автоматическое формирование отчетов о ходе выполнения задач, загрузке сотрудников и возможных рисках.

Анализ показал, что современные системы управления проектами требуют постоянного развития и адаптации под новые вызовы бизнеса. Разработка удобной, гибкой и многофункциональной платформы позволит компаниям оптимизировать рабочие процессы, повысить прозрачность проектов и снизить риски, связанные с невыполнением задач в срок.

## 

## 1.2 Обзор и анализ возможных альтернатив

Существует широкий выбор программных решений для управления проектами. Одни из них обладают обширным функционалом, другие – более просты в использовании и ориентированы на небольшие команды.

Trello – это облачный сервис для ведения проектов, который позволяет создавать визуальные доски, карточки с задачами и перемещать их между списками для удобного контроля над ходом работы.

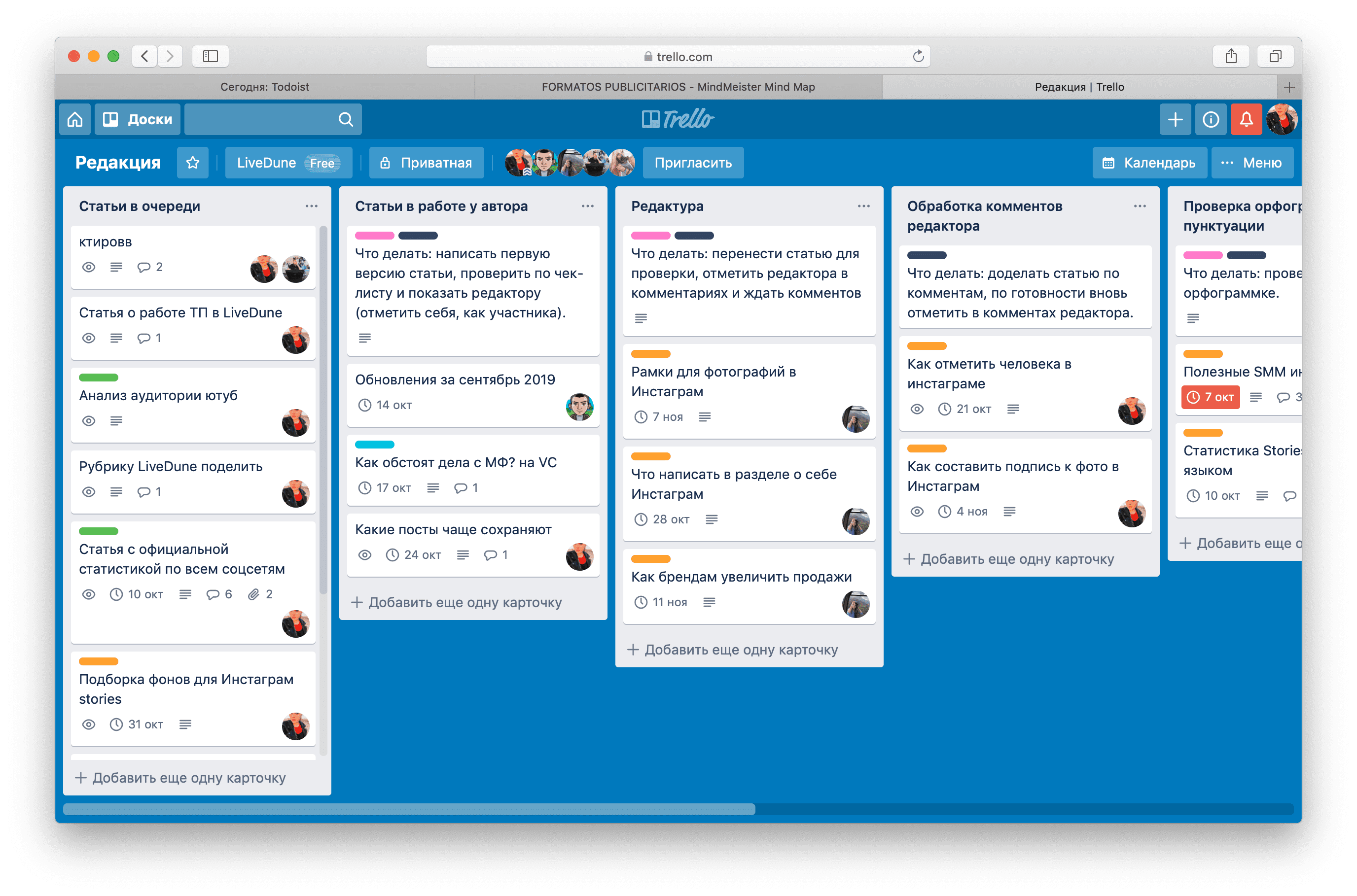


Рисунок 1 – Скриншот Trello

Основные функции:

* добавление вложений к задачам;
* установка сроков выполнения;
* возможность комментирования карточек;
* создание чек-листов;
* настройка уведомлений и другие.

Преимущества:

* удобный и интуитивно понятный интерфейс;
* бесплатная базовая версия;
* возможность совместной работы;
* поддержка мобильных устройств.

Недостатки:

* ограниченный набор инструментов;
* отсутствие диаграммы Ганта;
* ограниченный доступ для внешних пользователей.

Стоимость: бесплатная базовая версия, расширенные возможности – от $6 в месяц.

Microsoft Project – мощный инструмент для детального планирования проектов, включая создание диаграмм Ганта и управление ресурсами.

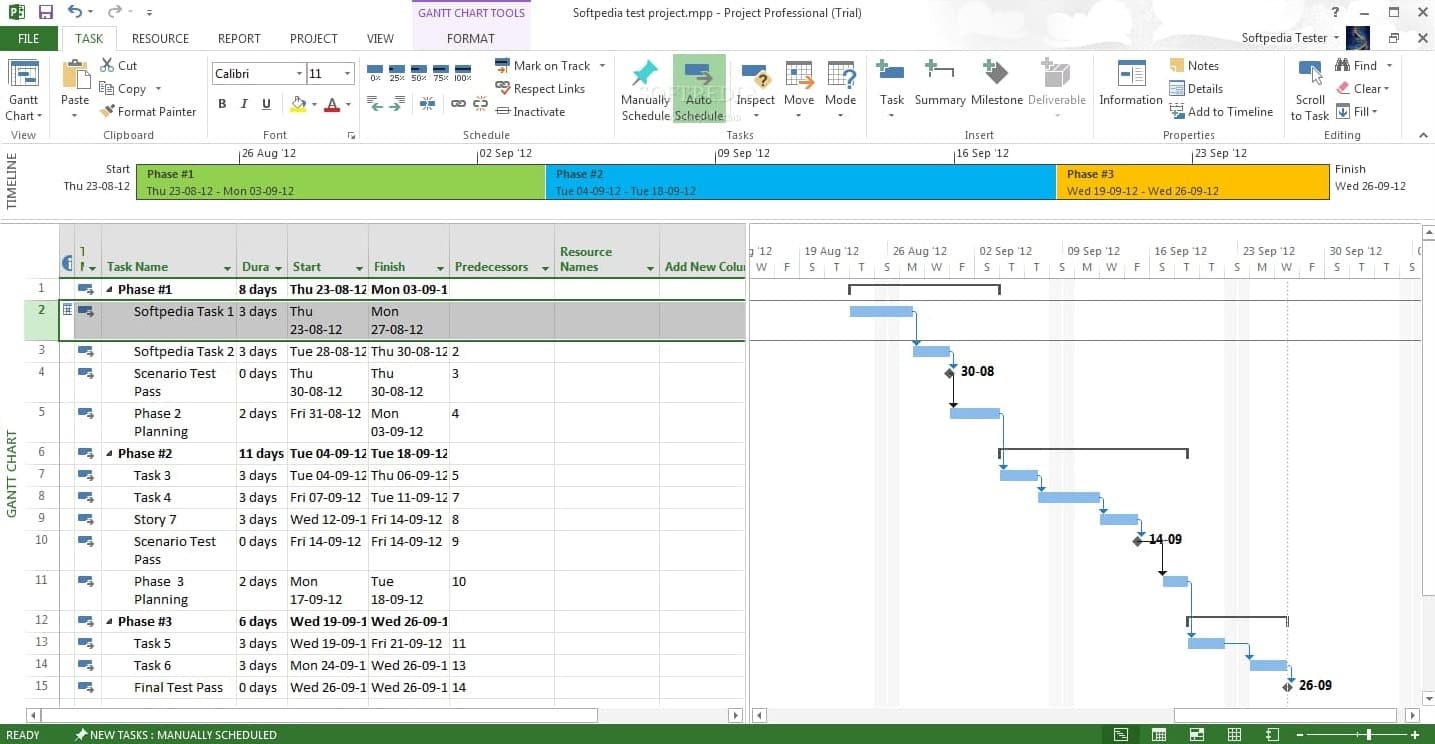


Рисунок 2 – Скриншот Microsoft Project

Основные функции:

* построение диаграмм Ганта;
* управление задачами и ресурсами;
* формирование отчетов;
* интеграция с другими сервисами.

Преимущества:

* широкий набор инструментов для управления проектами;
* поддержка командной работы;
* возможность интеграции с другими программными продуктами.

Недостатки:

* высокая стоимость;
* сложный процесс освоения;
* отсутствие облачного доступа в некоторых версиях.

Стоимость: от $10 в месяц за базовую версию, от $30 в месяц за профессиональную.

Jira – инструмент от компании Atlassian, активно используемый в сфере IT для управления процессом разработки, но подходящий и для других отраслей.

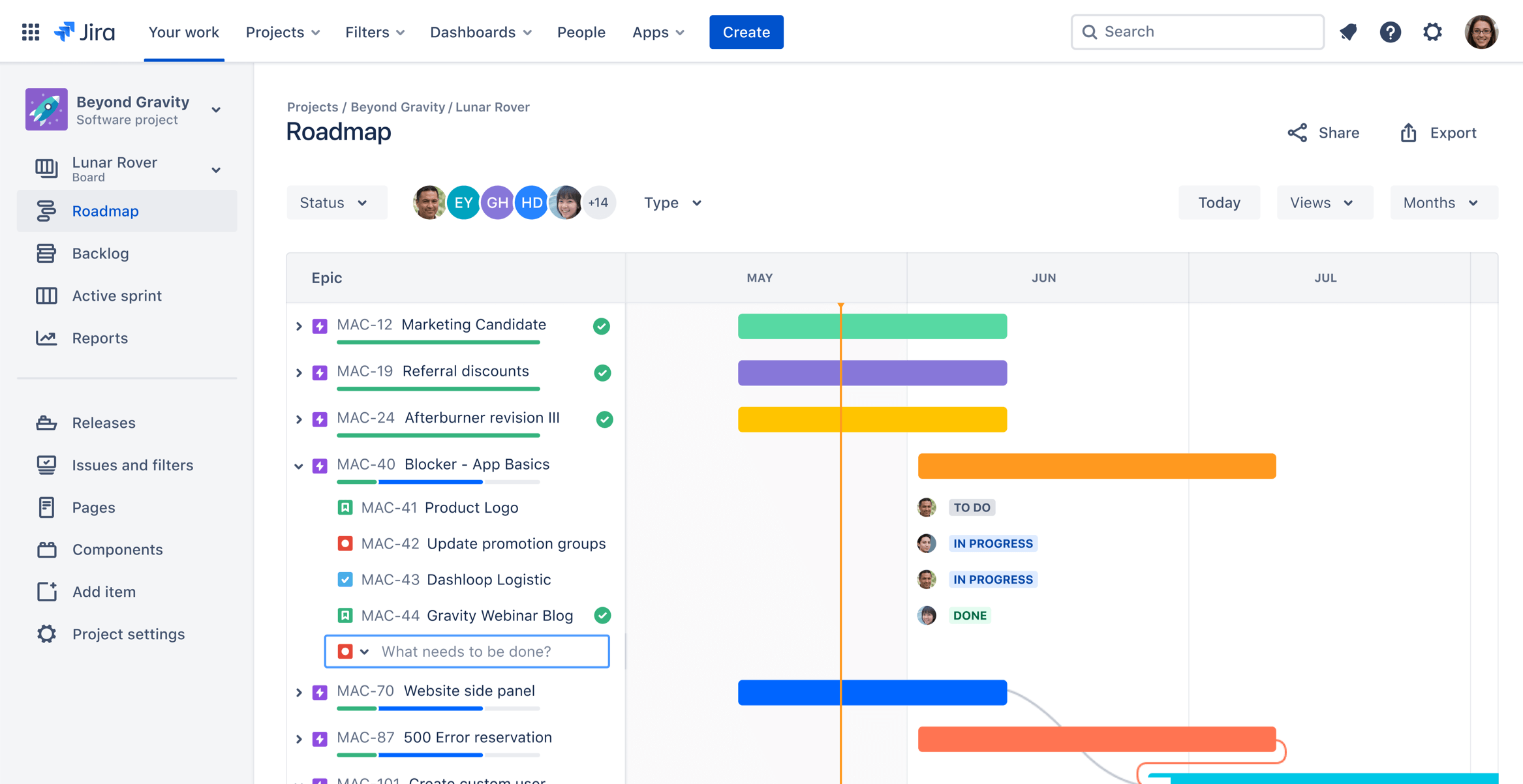


Рисунок 3 – Скриншот Jira

Основные функции:

* создание задач и подзадач;
* назначение ответственных лиц и сроков выполнения;
* визуализация процессов с помощью досок;
* отчетность и интеграция с другими сервисами.

Преимущества:

* мощный функционал для планирования, управления ресурсами и анализа данных;
* возможность интеграции с Confluence и Bitbucket.

Недостатки:

* сложность настройки и освоения;
* высокая стоимость продвинутых версий;
* менее гибкая структура по сравнению с Trello.

Стоимость: базовая версия от $10 в месяц, расширенные планы – до $14 300 в год для крупных компаний. Jira доступна как облачный сервис и как серверное ПО.

Все перечисленные системы предоставляют возможности для эффективного управления проектами, однако выбор зависит от специфики бизнеса, бюджета и необходимых функций. В некоторых случаях может потребоваться разработка собственного программного продукта, адаптированного под конкретные задачи компании.

При анализе представленных решений можно сделать вывод, что ни одна из них не является оптимальной. Большинство популярных программ являются зарубежными, что может осложнить процесс внедрения и дальнейшее сопровождение.

## 1.3 Анализ функциональных и эксплуатационных требований

### **1.3.1 Стандарты**

Программный продукт разрабатывается на основании следующих государственных стандартов:

1. Межгосударственный стандарт ГОСТ 7.32-2017. Система стандартов по информации, библиотечному и издательскому делу. Отчет о научно – исследовательской работе. Структура и правила оформления [1].
2. Международный стандарт ISO/IEC 12207. Информационные технологии. Процессы жизненного цикла программного обеспечения [2].
3. ГОСТ 34.601-90. Автоматизированные системы. Стадии создания [3].
4. ГОСТ 34.602-89. Информационная технология. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы [4].
5. ГОСТ 34.603-92. Информационная технология. Виды испытаний автоматизированных систем [5].

### **1.3.2 Функциональные требования пользователя**

Программный продукт, разрабатываемый в рамках курсового проекта, должен удовлетворять следующему перечню функциональных требований:

* управление пользователями и ролями;
* работа с задачами;
* визуализация и аналитика.

### **1.3.3 Входные данные**

Входными данными при работе с программным продуктом должны быть данные: ФИО пользователя, email, роль, вводимые пользователем вручную. Входной язык приложения – русский.

### **1.3.4 Выходные данные**

Выходными данными при работе программы являются данные, выводимые на экран пользователя: отчеты по проектам и задачам, статистика и аналитика, уведомления и оповещения. Язык выходных данных в приложении – русский.

### **1.3.5 Требования к интерфейсу**

Программный продукт должен содержать форму авторизации пользователей, которая будет открываться после открытия панели управления. Данная форма должна содержать поля для ввода пользователем своего пароля и выбора логина.

Главное меню программного продукта должно предоставить пользователю выбор функций, каждая функция должна открывать форму по запросу либо формировать отчет, согласно запрашиваемой информации.

Остальные окна программного продукта интуитивно понятны пользователю. Сообщения, выдаваемые программой при прохождении каких-либо действий, должны содержать краткое описание произведенных действий и содержать комментарии для облегчения дальнейшей работы пользователя.

### **1.3.6 Требования к надежности**

При работе с программным продуктом необходимо предусмотреть:

* контроль вводимой информации, т.е. возможность отслеживания ошибок, допускаемых пользователем, и последующей реакции программы на них;
* предусмотреть блокировку некорректных действий пользователя при работе с системой.

### **1.3.7 Требования к программной документации**

В состав сопровождающей документации программного продукта должны входить следующие компоненты:

1. Пояснительная записка на 80–100 листах, содержащая описание разработки.
2. Пояснительная записка, исходные тексты модулей на языке Python и HTML, задание на курсовой проект.
3. Откомпилированный EXE-файл на CD-диске

### **1.3.8 Требования к составу и параметрам технических средств**

Система должна работать на IBM совместимых персональных компьютерах. Минимальная конфигурация:

* тип процессора – процессор с архитектурами x86-64 и ARM64, совместимые с Windows, Linux и macOS;
* объем оперативного запоминающего устройства - 16 Мб;
* тип монитора - SVGA (15').

### **1.3.9 Модель вариантов использования**

#### **1.3.9.1 Диаграмма вариантов использования**

Действующие лица для диаграммы вариантов использования приведены в таблице 1.

Таблица 1 – Действующие лица

|  |  |
| --- | --- |
| **Термин** | **Значение** |
| Пользователь | Зарегистрированный пользователь системы, который может создавать задачи, управлять проектами, загружать отчеты и взаимодействовать с другими пользователями. Пользователь может быть как обычным участником команды, так и администратором, в зависимости от прав доступа. |
| База данных | Внешняя система, которая взаимодействует с приложением через запросы и ответы. |

На основании анализа требований пользователя были выделены следующие варианты использования, представленные в таблице 2.

Таблица 2 – Описание вариантов использования

|  |  |
| --- | --- |
| **Термин** | **Значение** |
| Зарегистрироваться | Пользователь может зарегистрироваться в системе, указав свои данные (имя, email, пароль) и выбрав команду. |
| Войти | Пользователь может войти в систему, используя свои учетные данные (email и пароль). |
| Создать команду | Пользователь может создать новую команду, указав её название. |
| Выбрать команду | Пользователь может выбрать команду, к которой он хочет присоединиться, из списка доступных команд. |
| Создать проект | Пользователь может создать новый проект, указав его название и описание. |
| Просмотреть проекты | Пользователь может просматривать список всех проектов, к которым он имеет доступ. |
| Удалить проект | Пользователь может удалить проект, что также приведет к удалению всех связанных с ним задач и отчетов. |
| Просмотреть детали проекта | Пользователь может просматривать детали проекта, включая список задач и отчетов, связанных с этим проектом. |
| Создать задачу | Пользователь может создать новую задачу, указав её название, описание, приоритет, статус, проект и ответственного. |
| Редактировать задачу | Пользователь может редактировать существующую задачу, изменяя её название, описание, приоритет, статус и ответственного. |
| Удалить задачу | Пользователь может удалить задачу. |
| Просмотреть детали задачи | Пользователь может просматривать детали задачи, включая её статус, приоритет, ответственного и проект. |
| Изменить статус задачи | Пользователь может изменять статус задачи (например, "To Do", "In Progress", "Done") через Kanban-доску. |
| Загрузить отчет | Пользователь может загрузить отчет, связанный с конкретным проектом, указав файл и выбрав проект. |
| Просмотреть отчеты | Пользователь может просматривать список отчетов, связанных с проектом, и скачивать их. |
| Просмотреть список пользователей | Пользователь может просматривать список всех пользователей системы. |
| Создать тестового пользователя | Пользователь может создать тестового пользователя для тестирования функциональности системы. |
| Просмотреть задачи на Kanban-доске | Пользователь может просматривать задачи, распределенные по колонкам (To Do, In Progress, Done) на Kanban-доске. |
| Переместить задачу между колонками | Пользователь может перетаскивать задачи между колонками, изменяя их статус. |
| Просмотреть диаграммы | Пользователь может просматривать различные диаграммы, такие как диаграмма Ганта или графики, связанные с проектами и задачами. |

На основании всех выше рассмотренных вариантов использования была составлена диаграмма вариантов использования, представленная на рисунке 1.

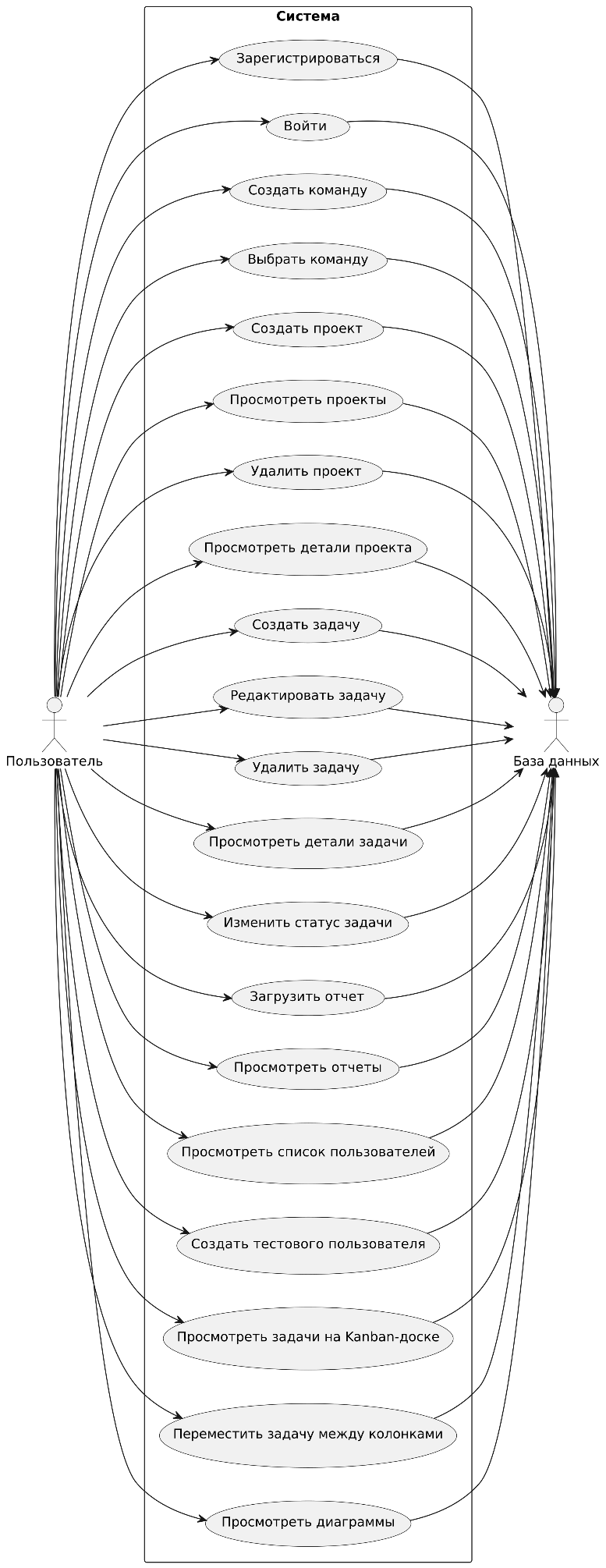


Рисунок 1 – Диаграмма вариантов использования

#### **1.3.9.2 Описание варианта использования** **«Зарегистрироваться»**

Действующие лица: Пользователь и База данных.

Заинтересованные лица и их требования:

* Пользователь хочет зарегистрироваться в системе для доступа к функционалу.

Предусловия:

* Пользователь не зарегистрирован в системе.
* Система доступна для регистрации новых пользователей.

Постусловия:

* Пользователь зарегистрирован в системе.
* Пользователь может войти в систему, используя свои учетные данные.

Таблица 3 – Основной сценарий

|  |  |
| --- | --- |
| **Действия актора** | **Действия(отклик) системы** |
| 1. Пользователь переходит на страницу регистрации. | 1. Система отображает форму регистрации с полями для ввода имени, email, пароля и выбора команды. |
| 1. Пользователь заполняет форму и нажимает кнопку "Зарегистрироваться". | 1. Система проверяет корректность введенных данных (уникальность email, заполнение обязательных полей). Если данные корректны, система создает нового пользователя и перенаправляет его на страницу входа. |
| Поток ошибок Е1. В случае ошибки при заполнении полей система должна уведомить пользователя о необходимости исправить данные и повторить попытку создания задачи. | |
| 1. Вариант использования завершается | |

#### **1.3.9.3 Описание варианта использования «Войти»**

Действующие лица: Пользователь и База данных.

Заинтересованные лица и их требования:

* Пользователь хочет зайти в систему для доступа к своему аккаунту.

Предусловия:

* Пользователь зарегистрирован в системе.
* Система доступна для входа.

Постусловия:

* Пользователь авторизован в системе.
* Пользователь перенаправлен на страницу личного кабинета (Dashboard).

Таблица 4 – Основной сценарий

|  |  |
| --- | --- |
| **Действия актора** | **Действия(отклик) системы** |
| 1. Пользователь переходит на страницу входа. | 1. Система отображает форму входа с полями для ввода email и пароля. |
| 1. Пользователь вводит email и пароль, нажимает кнопку "Войти". | 1. Система сохраняет изменения, отправляет уведомления ответственным. |
| 1. Менеджер управляет участниками проекта (добавляет, назначает роли, регулирует доступ). | 1. Система проверяет учетные данные. Если данные верны, система авторизует пользователя и перенаправляет его на страницу личного кабинета (Dashboard). |
| Поток ошибок Е1. В случае возникновения ошибок при управлении проектом, система должна уведомлять пользователя о причинах | |
| 1. Вариант использования завершается | |

#### **1.3.9.4 Описание варианта использования «Создать команду»**

Действующие лица: Пользователь и База данных.

Заинтересованные лица и их требования:

* Пользователь хочет создать новую команду для совместной работы.

Предусловия:

* Пользователь авторизован в системе.

Постусловия:

* Новая команда создана и доступна для выбора.
* Пользователь может выбрать созданную команду.

Таблица 5 – Основной сценарий

|  |  |
| --- | --- |
| **Действия актора** | **Действия(отклик) системы** |
| 1. Пользователь переходит на страницу создания команды. | 1. Система отображает форму с полем для ввода названия команды. |
| 1. Пользователь вводит название команды и нажимает кнопку "Создать команду". | 1. Система проверяет корректность данных (название не должно быть пустым). Если данные корректны, система создает новую команду и уведомляет пользователя об успешном создании. |
| Поток ошибок Е1. Если произошел сбой при загрузке, система выводит сообщение об ошибке и просит повторить попытку. | |
| 1. Вариант использования завершается | |

#### **1.3.9.5 Описание варианта использования «Выбрать команду»**

Действующие лица: Пользователь и База данных.

Заинтересованные лица и их требования:

* Пользователь хочет присоединиться к существующей команде

Предусловия:

* Пользователь авторизован в системе.
* В системе есть хотя бы одна команда.

Постусловия:

* Пользователь присоединен к выбранной команде.
* Пользователь может работать над проектами и задачами в рамках команды.

Таблица 6 – Основной сценарий

|  |  |
| --- | --- |
| **Действия актора** | **Действия(отклик) системы** |
| 1. Пользователь переходит на страницу выбора команды. | 1. Система отображает список доступных команд. |
| 1. Пользователь выбирает команду из списка и нажимает кнопку "Выбрать команду". | 1. Система связывает пользователя с выбранной командой и обновляет его профиль. |
| 1. Система уведомляет пользователя об успешном выборе команды. | 1. Система перенаправляет пользователя на страницу личного кабинета (Dashboard). |
| Поток ошибок Е1. Если произошел сбой при загрузке диаграммы, система выводит сообщение об ошибке и предлагает повторить попытку позже. | |
| 1. Вариант использования завершается | |

#### **1.3.9.6 Описание варианта использования «Создать проект»**

Действующие лица: Пользователь и База данных.

Заинтересованные лица и их требования:

* Пользователь очет создать новый проект для управления задачами.

Предусловия:

* Пользователь авторизован в системе.
* Пользователь состоит в команде.

Постусловия:

* Новый проект создан и доступен для управления.
* Пользователь может добавлять задачи и отчеты в проект.

Таблица 7 – Основной сценарий

|  |  |
| --- | --- |
| **Действия актора** | **Действия(отклик) системы** |
| 1. Пользователь переходит на страницу создания проекта. | 1. Система отображает форму с полями для ввода названия и описания проекта. |
| 1. Пользователь заполняет форму и нажимает кнопку "Создать проект". | 1. Система проверяет корректность данных (название не должно быть пустым). Если данные корректны, система создает новый проект и уведомляет пользователя об успешном создании.. |
| Поток ошибок Е1. Если произошел сбой при загрузке отчета, система выводит сообщение об ошибке и предлагает повторить попытку позже. | |
| 1. Вариант использования завершается | |

#### **1.3.9.7 Описание варианта использования «Просмотреть проекты»**

Действующие лица: Пользователь и База данных.

Заинтересованные лица и их требования:

* Пользователь хочет просмотреть список проектов, к которым имеет доступ.

Предусловия:

* Пользователь авторизован в системе.

Постусловия:

* Пользователь видит список проектов.
* Пользователь может перейти к деталям любого проекта.

Таблица 7 – Основной сценарий

|  |  |
| --- | --- |
| **Действия актора** | **Действия(отклик) системы** |
| 1. Пользователь переходит на страницу списка проектов. | 1. Система отображает список всех проектов, доступных пользователю, с возможностью перехода к деталям каждого проекта. |
| Поток ошибок Е1. Если произошел сбой при загрузке отчета, система выводит сообщение об ошибке и предлагает повторить попытку позже. | |
| 1. Вариант использования завершается | |

#### **1.3.9.8 Описание варианта использования «Удалить проект»**

Действующие лица: Пользователь и База данных.

Заинтересованные лица и их требования:

* Пользователь хочет удалить проект, который больше не нужен.

Предусловия:

* Пользователь авторизован в системе.
* Проект существует и принадлежит пользователю или его команде.

Постусловия:

* Проект удален вместе со всеми связанными задачами и отчетами.
* Пользователь уведомлен об успешном удалении.

Таблица 7 – Основной сценарий

|  |  |
| --- | --- |
| **Действия актора** | **Действия(отклик) системы** |
| 1. Пользователь выбирает проект для удаления. | 1. Система запрашивает подтверждение удаления. |
| 1. Пользователь подтверждает удаление. | 1. Система удаляет проект и все связанные с ним задачи и отчеты, уведомляя пользователя об успешном удалении. |
| Поток ошибок Е1. Если произошел сбой при загрузке отчета, система выводит сообщение об ошибке и предлагает повторить попытку позже. | |
| 1. Вариант использования завершается | |

#### **1.3.9.9 Описание варианта использования «Просмотреть детали проекта»**

Действующие лица: Пользователь и База данных.

Заинтересованные лица и их требования:

* Пользователь хочет просмотреть детали проекта, включая задачи и отчеты.

Предусловия:

* Пользователь авторизован в системе.
* Проект существует и принадлежит пользователю или его команде.

Постусловия:

* Пользователь видит детали проекта, включая список задач и отчетов.

Таблица 7 – Основной сценарий

|  |  |
| --- | --- |
| **Действия актора** | **Действия(отклик) системы** |
| 1. Пользователь выбирает проект для просмотра. | 1. Система отображает детали проекта, включая список задач и отчетов, связанных с этим проектом. |
| Поток ошибок Е1. Если произошел сбой при загрузке отчета, система выводит сообщение об ошибке и предлагает повторить попытку позже. | |
| 1. Вариант использования завершается | |

#### **1.3.9.10 Описание варианта использования «Создать задачу»**

Действующие лица: Пользователь и База данных.

Заинтересованные лица и их требования:

* Пользователь хочет создать новую задачу для проекта.

Предусловия:

* Пользователь авторизован в системе.
* Проект существует и принадлежит пользователю или его команде.

Постусловия:

* Новая задача создана и добавлена в проект.
* Пользователь уведомлен об успешном создании задачи.

Таблица 7 – Основной сценарий

|  |  |
| --- | --- |
| **Действия актора** | **Действия(отклик) системы** |
| 1. Пользователь переходит на страницу создания задачи. | 1. Система отображает форму с полями для ввода названия, описания, приоритета, статуса, проекта и ответственного. |
| 1. Пользователь заполняет форму и нажимает кнопку "Создать задачу". | 1. Система проверяет корректность данных (название и проект обязательны). Если данные корректны, система создает задачу и уведомляет пользователя об успешном создании. |
| Поток ошибок Е1. Если произошел сбой при загрузке отчета, система выводит сообщение об ошибке и предлагает повторить попытку позже. | |
| 1. Вариант использования завершается | |

#### **1.3.9.11 Описание варианта использования «Редактировать задачу»**

Действующие лица: Пользователь и База данных.

Заинтересованные лица и их требования:

* Пользователь хочет изменить данные существующей задачи

Предусловия:

* Пользователь авторизован в системе.
* Задача существует и принадлежит проекту пользователя.

Постусловия:

* Данные задачи обновлены.
* Пользователь уведомлен об успешном изменении задачи.

Таблица 7 – Основной сценарий

|  |  |
| --- | --- |
| **Действия актора** | **Действия(отклик) системы** |
| 1. Пользователь выбирает задачу для редактирования. | 1. Система отображает форму с текущими данными задачи. |
| 1. Пользователь вносит изменения и нажимает кнопку "Сохранить". | 1. Система проверяет корректность данных. Если данные корректны, система обновляет задачу и уведомляет пользователя об успешном изменении. |
| Поток ошибок Е1. Если произошел сбой при загрузке отчета, система выводит сообщение об ошибке и предлагает повторить попытку позже. | |
| 1. Вариант использования завершается | |

#### **1.3.9.12 Описание варианта использования «Удалить задачу»**

Действующие лица: Пользователь и База данных.

Заинтересованные лица и их требования:

* Пользователь хочет удалить задачу, которая больше не нужна.

Предусловия:

* Пользователь авторизован в системе.
* Задача существует и принадлежит проекту пользователя.

Постусловия:

* Задача удалена.
* Пользователь уведомлен об успешном удалении задачи.

Таблица 7 – Основной сценарий

|  |  |
| --- | --- |
| **Действия актора** | **Действия(отклик) системы** |
| 1. Пользователь выбирает задачу для удаления. | 1. Система запрашивает подтверждение удаления. |
| 1. Пользователь подтверждает удаление. | 1. Система удаляет задачу и уведомляет пользователя об успешном удалении. |
| Поток ошибок Е1. Если произошел сбой при загрузке отчета, система выводит сообщение об ошибке и предлагает повторить попытку позже. | |
| 1. Вариант использования завершается | |

#### **1.3.9.13 Описание варианта использования «Просмотреть детали задачи»**

Действующие лица: Пользователь и База данных.

Заинтересованные лица и их требования:

* Пользователь хочет просмотреть детали задачи.

Предусловия

* Пользователь авторизован в системе.
* Задача существует и принадлежит проекту пользователя.

Постусловия:

* Пользователь видит детали задачи, включая статус, приоритет и ответственного.

Таблица 7 – Основной сценарий

|  |  |
| --- | --- |
| **Действия актора** | **Действия(отклик) системы** |
| 1. Пользователь выбирает задачу для просмотра. | 1. Система отображает детали задачи, включая её статус, приоритет, ответственного и проект. |
| Поток ошибок Е1. Если произошел сбой при загрузке отчета, система выводит сообщение об ошибке и предлагает повторить попытку позже. | |
| 1. Вариант использования завершается | |

#### **1.3.9.14 Описание варианта использования «Изменить статус задачи»**

Действующие лица: Пользователь и База данных.

Заинтересованные лица и их требования:

* Пользователь хочет изменить статус задачи (например, "To Do", "In Progress", "Done").

Предусловия:

* Пользователь авторизован в системе.
* Задача существует и принадлежит проекту пользователя.

Постусловия:

* Статус задачи обновлен.
* Пользователь видит обновленный статус задачи на Kanban-доске.

Таблица 7 – Основной сценарий

|  |  |
| --- | --- |
| **Действия актора** | **Действия(отклик) системы** |
| 1. Пользователь перетаскивает задачу между колонками на Kanban-доске. | 1. Система автоматически обновляет статус задачи и сохраняет изменения. |
| Поток ошибок Е1. Если произошел сбой при загрузке отчета, система выводит сообщение об ошибке и предлагает повторить попытку позже. | |
| 1. Вариант использования завершается | |

#### **1.3.9.15 Описание варианта использования «Загрузить отчет»**

Действующие лица: Пользователь и База данных.

Заинтересованные лица и их требования:

* Пользователь хочет загрузить отчет, связанный с проектом.

Предусловия:

* Пользователь авторизован в системе.
* Проект существует и принадлежит пользователю или его команде.

Постусловия:

* Отчет загружен и связан с проектом.
* Пользователь уведомлен об успешной загрузке отчета.

Таблица 7 – Основной сценарий

|  |  |
| --- | --- |
| **Действия актора** | **Действия(отклик) системы** |
| 1. Пользователь переходит на страницу загрузки отчета. | 1. Система отображает форму с полями для выбора проекта и загрузки файла. |
| 1. Пользователь выбирает проект и загружает файл, нажимает кнопку "Загрузить отчет". | 1. Система проверяет корректность данных (файл должен быть в допустимом формате). Если данные корректны, система сохраняет отчет и уведомляет пользователя об успешной загрузке. |
| Поток ошибок Е1. Если произошел сбой при загрузке отчета, система выводит сообщение об ошибке и предлагает повторить попытку позже. | |
| 1. Вариант использования завершается | |

#### **1.3.9.16 Описание варианта использования «Просмотреть отчеты»**

Действующие лица: Пользователь и База данных.

Заинтересованные лица и их требования:

* Пользователь хочет просмотреть список отчетов, связанных с проектом.

Предусловия:

* Пользователь авторизован в системе.
* Проект существует и принадлежит пользователю или его команде.

Постусловия:

* Пользователь видит список отчетов, связанных с проектом.
* Пользователь может скачать любой отчет.

Таблица 7 – Основной сценарий

|  |  |
| --- | --- |
| **Действия актора** | **Действия(отклик) системы** |
| 1. Пользователь переходит на страницу списка отчетов. | 1. Система отображает список отчетов, связанных с выбранным проектом, и предоставляет возможность скачивания каждого отчета. |
| Поток ошибок Е1. Если произошел сбой при загрузке отчета, система выводит сообщение об ошибке и предлагает повторить попытку позже. | |
| 1. Вариант использования завершается | |

#### **1.3.9.17 Описание варианта использования «Просмотреть список пользователей»**

Действующие лица: Пользователь и База данных.

Заинтересованные лица и их требования:

* Пользователь хочет просмотреть список всех пользователей системы.

Предусловия:

* Пользователь авторизован в системе.

Постусловия:

* Пользователь видит список всех пользователей системы.

Таблица 7 – Основной сценарий

|  |  |
| --- | --- |
| **Действия актора** | **Действия(отклик) системы** |
| 1. Пользователь переходит на страницу списка пользователей. | 1. Система отображает список всех пользователей системы с их основными данными (имя, email). |
| Поток ошибок Е1. Если произошел сбой при загрузке отчета, система выводит сообщение об ошибке и предлагает повторить попытку позже. | |
| 1. Вариант использования завершается | |

#### **1.3.9.18 Описание варианта использования «Создать тестового пользователя»**

Действующие лица: Пользователь и База данных.

Заинтересованные лица и их требования:

* Пользователь хочет создать тестового пользователя для тестирования функциональности.

Предусловия:

* Пользователь авторизован в системе.

Постусловия:

* Тестовый пользователь создан.
* Пользователь уведомлен об успешном создании тестового пользователя.

Таблица 7 – Основной сценарий

|  |  |
| --- | --- |
| **Действия актора** | **Действия(отклик) системы** |
| 1. Пользователь нажимает кнопку "Создать тестового пользователя". | 1. Система генерирует тестового пользователя с уникальными данными и уведомляет пользователя об успешном создании. |
| Поток ошибок Е1. Если произошел сбой при загрузке отчета, система выводит сообщение об ошибке и предлагает повторить попытку позже. | |
| 1. Вариант использования завершается | |

#### **1.3.9.19 Описание варианта использования «Просмотреть задачи на Kanban-доске»**

Действующие лица: Пользователь и База данных.

Заинтересованные лица и их требования:

* Пользователь хочет просмотреть задачи, распределенные по колонкам на Kanban-доске.

Предусловия:

* Пользователь авторизован в системе.
* В системе есть хотя бы одна задача.

Постусловия:

* Пользователь видит задачи, распределенные по колонкам (To Do, In Progress, Done).

Таблица 7 – Основной сценарий

|  |  |
| --- | --- |
| **Действия актора** | **Действия(отклик) системы** |
| 1. Пользователь переходит на страницу Kanban-доски. | 1. Система отображает задачи, распределенные по колонкам (To Do, In Progress, Done). |
| Поток ошибок Е1. Если произошел сбой при загрузке отчета, система выводит сообщение об ошибке и предлагает повторить попытку позже. | |
| 1. Вариант использования завершается | |

#### **1.3.9.20 Описание варианта использования «Переместить задачу между колонками»**

Действующие лица: Пользователь и База данных.

Заинтересованные лица и их требования:

* Пользователь хочет изменить статус задачи, перемещая её между колонками на Kanban-доске.

Предусловия:

* Пользователь авторизован в системе.
* Задача существует и принадлежит проекту пользователя.

Постусловия:

* Статус задачи обновлен.
* Пользователь видит обновленный статус задачи на Kanban-доске.

Таблица 7 – Основной сценарий

|  |  |
| --- | --- |
| **Действия актора** | **Действия(отклик) системы** |
| 1. Пользователь перетаскивает задачу между колонками на Kanban-доске. | 1. Система автоматически обновляет статус задачи и сохраняет изменения. |
| Поток ошибок Е1. Если произошел сбой при загрузке отчета, система выводит сообщение об ошибке и предлагает повторить попытку позже. | |
| 1. Вариант использования завершается | |

#### **1.3.9.21 Описание варианта использования «Просмотреть диаграммы»**

Действующие лица: Пользователь и База данных.

Заинтересованные лица и их требования:

* Пользователь хочет просмотреть диаграммы (например, диаграмму Ганта или графики) для анализа прогресса.

Предусловия:

* Пользователь авторизован в системе.
* В системе есть данные для построения диаграмм.

Постусловия:

* Пользователь видит диаграммы, связанные с проектами и задачами.

Таблица 7 – Основной сценарий

|  |  |
| --- | --- |
| **Действия актора** | **Действия(отклик) системы** |
| 1. Пользователь переходит на страницу диаграмм. | 1. Система отображает диаграммы (например, диаграмму Ганта или графики), связанные с проектами и задачами. |
| Поток ошибок Е1. Если произошел сбой при загрузке отчета, система выводит сообщение об ошибке и предлагает повторить попытку позже. | |
| 1. Вариант использования завершается | |

### **1.3.10 Глоссарий проекта**

В таблице приведены термины предметной области и их значения.

Таблица 17 – Термины и их значения

|  |  |
| --- | --- |
| **Термин** | **Значение** |
| Пользователь | Постановщик или ответственный, зарегистрированный в системе |
| Проект | Временное начинание, направленное на создание уникального продукта, услуги или результата. |
| Задача | Отдельная единица работы, которая должна быть выполнена в рамках проекта. |
| Команда | Группа пользователей, объединённая в рамках проекта. |
| Отчет | Объект, содержащий информацию о продвижении проекта, такую как отчеты о выполнении задач и общий прогресс проекта. |

### **1.3.11 Проверка модели на полноту**

Проверка диаграммы вариантов использования на полноту производится по операциям, производимым над основными объектами, представлена в таблице 18.

Основными объектами, упоминаемыми в модели вариантов использования, являются: пользователь, проект, задача, команда, отчет.

Таблица 18 – Проверка на полноту по операциям

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Варианты использования  Объекты | Зарегистрироваться | Войти | Создать команду | Выбрать команду | Создать проект | Просмотреть проекты | Удалить проект | Просмотреть детали проекта | Создать задачу |
| Проект |  |  |  |  | 1 | 2 | 4 | 2 | 3 |
| Задача |  |  |  |  |  |  | 4 |  | 1 |
| Пользователь | 1 | 1 |  |  |  |  |  |  |  |
| Команда |  |  | 1 | 3 |  |  |  |  |  |
| Отчет |  |  |  |  |  |  | 4 |  |  |

\

Продолжение таблицы 18 – Проверка на полноту по операциям

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Варианты использования  Объекты | Редактировать задачу | Удалить задачу | Просмотреть детали задачи | Изменить статус задачи | Загрузить отчет | Просмотреть отчеты | Просмотреть список пользователей | Создать тестового пользователя |
| Проект |  |  |  |  | 3 |  |  |  |
| Задача | 3 | 4 | 2 | 3 |  |  |  |  |
| Пользователь |  |  |  |  |  |  | 2 | 1 |
| Команда |  |  |  |  |  |  |  |  |
| Отчет |  |  |  |  | 1 | 2,4 |  |  |

Продолжение таблицы 18 – Проверка на полноту по операциям

|  |  |  |  |
| --- | --- | --- | --- |
| Варианты использования  Объекты | Просмотреть задачи на Kanban-доске | Переместить задачу между колонками | Просмотреть диаграммы |
| Проект |  |  | 2 |
| Задача | 1,2 | 3 |  |
| Пользователь |  |  |  |
| Команда |  |  |  |
| Отчет |  |  |  |

В таблице обозначены виды операций:

1 – создание;

2 – просмотр;

3 – изменение;

4 – удаление.

Результаты анализа полноты выполнения функциональных требований пользователя в модели вариантов использования приведены в таблице 19. Все функциональные требования пользователя отражены в основных вариантах использования.

Таблица 19 – Анализ полноты выполнения требований пользователя

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Требования пользователя | Варианты использования | | | | | | | | |
| Зарегистрироваться | Войти | Создать команду | Выбрать команду | Создать проект | Просмотреть проекты | Удалить проект | Просмотреть детали проекта | Создать задачу |
| Управление пользователями и ролями | + | + | + | + |  |  |  |  |  |
| Работа с задачами |  |  |  |  | + | + | + | + | + |
| Визуализация и аналитика |  |  |  |  |  |  |  |  |  |

Продолжение таблицы 19 – Анализ полноты выполнения требований пользователя

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Требования пользователя | Варианты использования | | | | | | | |
| Редактировать задачу | Удалить задачу | Просмотреть детали задачи | Изменить статус задачи | Загрузить отчет | Просмотреть отчеты | Просмотреть список пользователей | Создать тестового пользователя |
| Управление пользователями и ролями |  |  |  |  |  |  | + | + |
| Работа с задачами | + | + | + | + |  |  |  |  |
| Визуализация и аналитика |  |  |  |  | + | + |  |  |

Продолжение таблицы 19 – Анализ полноты выполнения требований пользователя

|  |  |  |  |
| --- | --- | --- | --- |
| Требования пользователя | Варианты использования | | |
| Просмотреть задачи на Kanban-доске | Переместить задачу между колонками | Просмотреть диаграммы |
| Управление пользователями и ролями |  |  |  |
| Работа с задачами |  | + |  |
| Визуализация и аналитика | + |  | + |

# 2 Проектирование информационной системы

## 2.1 Разработка архитектуры технического обеспечения системы

Все данные, необходимые для работы разрабатываемого приложения, хранятся на устройстве пользователя.

На рисунке 2 приведена предварительная диаграмма развертывания разрабатываемого приложения – архитектура аппаратных средств системы.

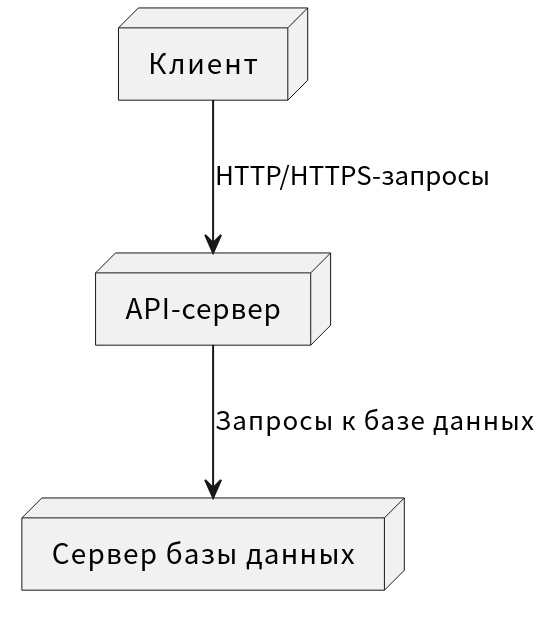


Рисунок 2 – Архитектура аппаратных средств системы

## 2.2 Разработка модели предметной области

Пользуясь списком категорий и методом анализа словесного описания вариантов использования, составлен список кандидатур на роль концептуальных классов для предметной области. Он соответствует требованиям и принятым упрощениям для всей предметной области.

Список концептуальных классов:

* Задача;
* Проект;
* Пользователь;
* Отчет;
* Команда.

На основании анализа словесного описания варианта использования, составлен список ассоциаций для предметной области, представленный в таблице 20.

Таблица 20 – Ассоциации для модели предметной области

|  |  |
| --- | --- |
| **Ассоциация** | **Описание ассоциации** |
| Включает | Проект включает Задачи и команды Задача включает Команды |
| Получает | Отчеты получают Задачи |
| Записывается в | Задача записывается в Отчеты |
| Выполняет | Пользователь выполняет Задачи и Команды Отчеты выполняются в Проекте Команды выполняют Задачи |

На основании анализа технического задания и описания вариантов использования выделены атрибуты классов для модели предметной области, представленные в таблице 21.

Таблица 21 – Атрибуты классов для модели предметной области

|  |  |
| --- | --- |
| **Название класса** | **Атрибуты класса** |
| Проект | ID  Название  Описание  Дата начала  Дата окончания  Статус  Ответственный  Список задач |
| Задача | ID  Название  Описание  Статус Дата начала  Дата окончания |
| Пользователь | Фамилия  Имя  Отчество  Логин  Пароль  Роль |
| Отчет | ID  Проект  Тип  Дата создания |
| Команда | ID  Название  Список пользователей  Проект |

В результате объединения концептуальных классов, ассоциаций и атрибутов классов концептуальная модель предметной области имеет вид, показанный на рисунке 3.

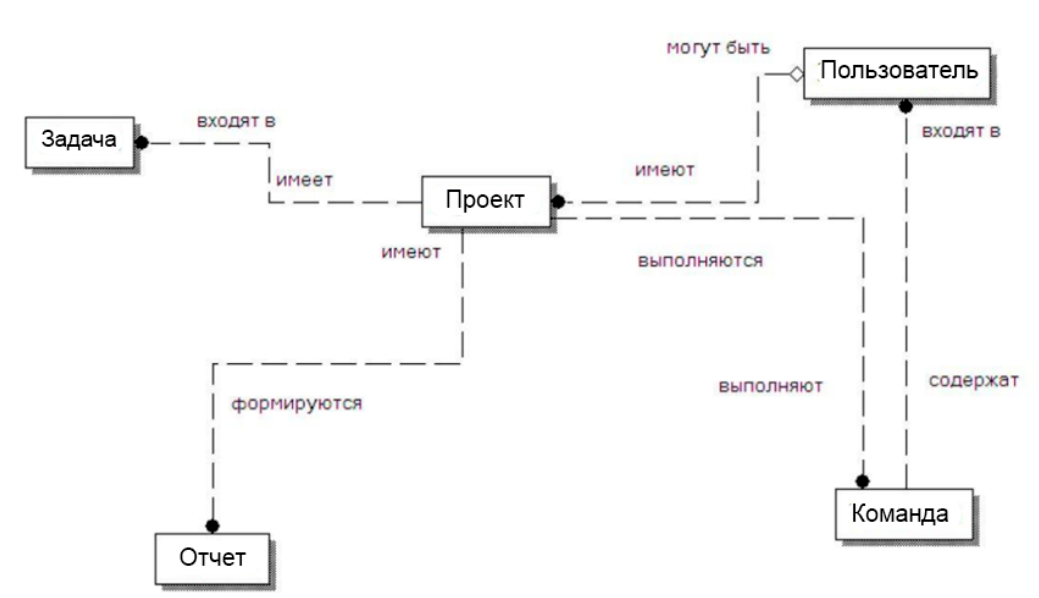


Рисунок 3 – Концептуальная модель предметной области

## 2.3 Проектирование классов предметной области

### **2.3.1 Построение диаграммы последовательности для варианта использования «Зарегистрироваться»**

Диаграмма последовательности, описывающая основной поток событий изображена на рисунке 4.

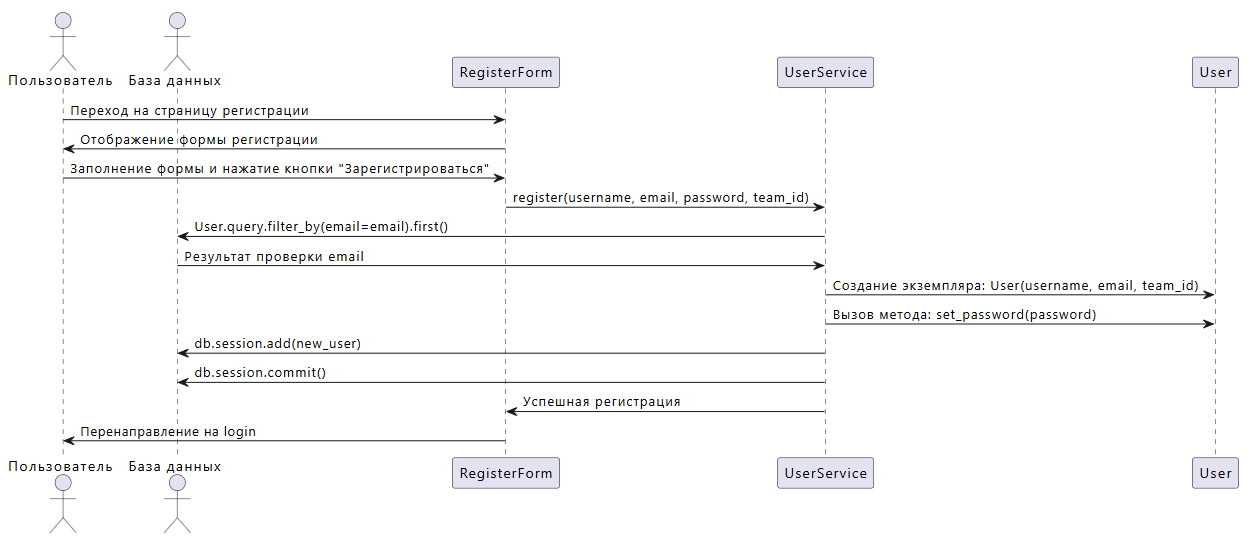


Рисунок 4 – Диаграмма последовательности для варианта использования «Зарегистрироваться»

### **2.3.2 Построение диаграммы последовательности для варианта использования «Войти»**

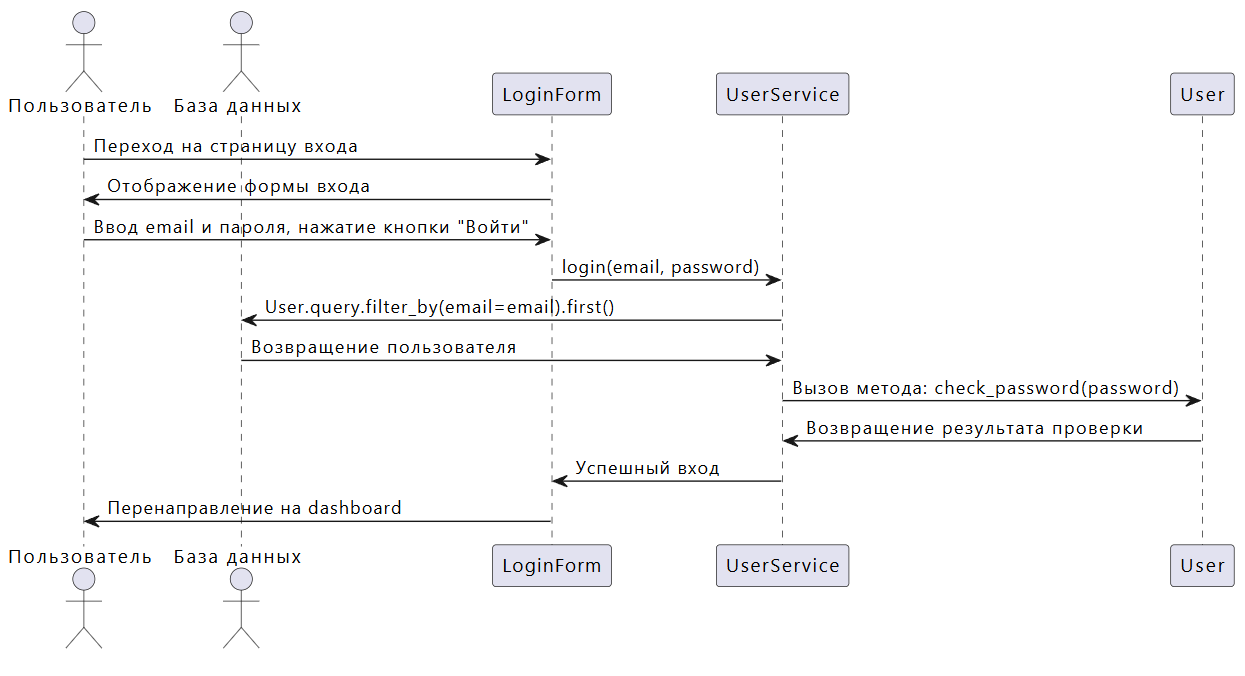
Диаграмма последовательности, описывающая основной и альтернативный потоки событий изображена на рисунке 5

Рисунок 5 – Диаграмма последовательности для варианта использования «Войти»

### **2.3.3 Построение диаграммы последовательности для варианта использования «Создать команду»**

Диаграмма последовательности, описывающая основной и альтернативный потоки событий изображена на рисунке 6.

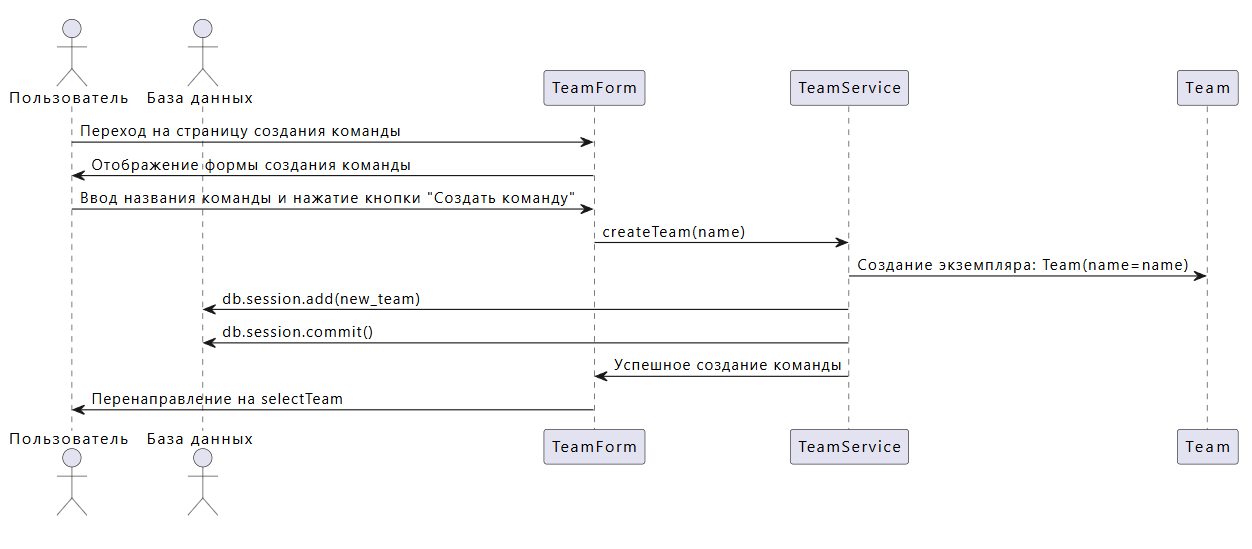


Рисунок 6 – Диаграмма последовательности для варианта использования «Создать команду»

### **2.3.4 Построение диаграммы последовательности для варианта использования «Выбрать команду»**

Диаграмма последовательности, описывающая основной поток событий изображена на рисунке 7.

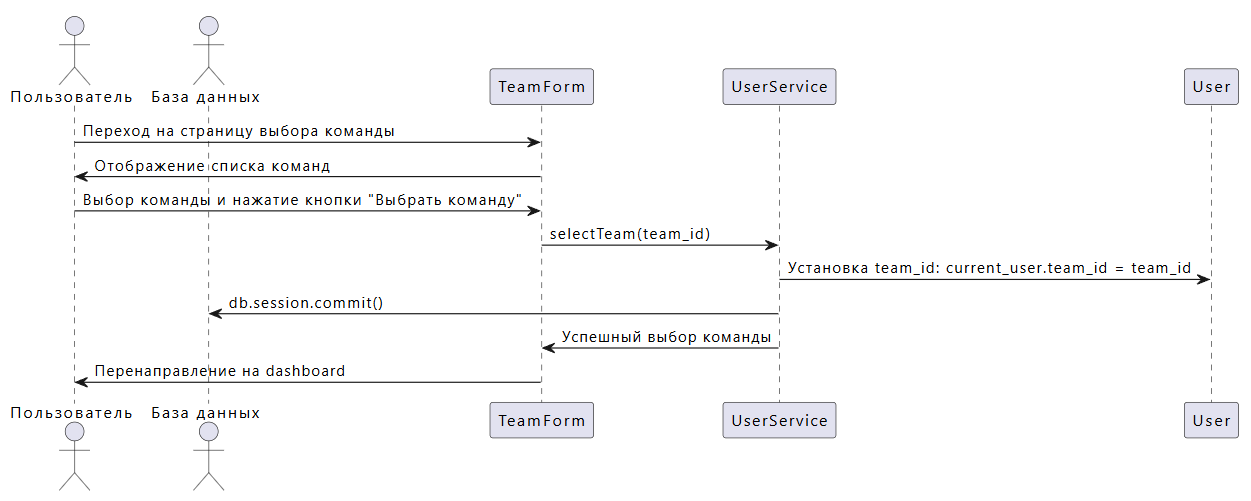


Рисунок 7 – Диаграмма последовательности для варианта использования «Выбрать команду»

### **2.3.5 Построение диаграммы последовательности для варианта использования «Создать проект»**

Диаграмма последовательности, описывающая основной поток событий изображена на рисунке 8.

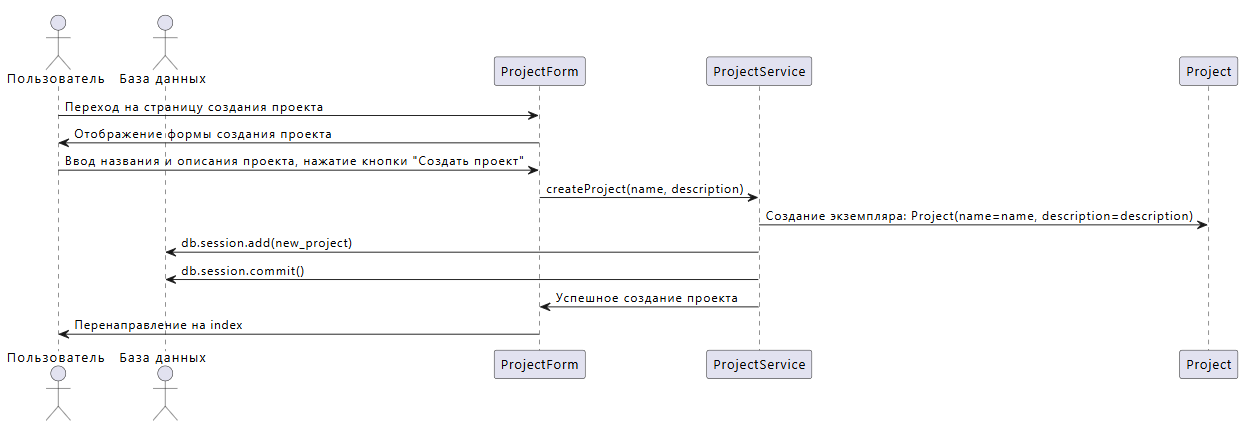


Рисунок 8 – Диаграмма последовательности для варианта использования «Создать проект

### **2.3.6 Построение диаграммы последовательности для варианта использования «Просмотреть проекты»**

Диаграмма последовательности, описывающая основной поток событий изображена на рисунке 9.

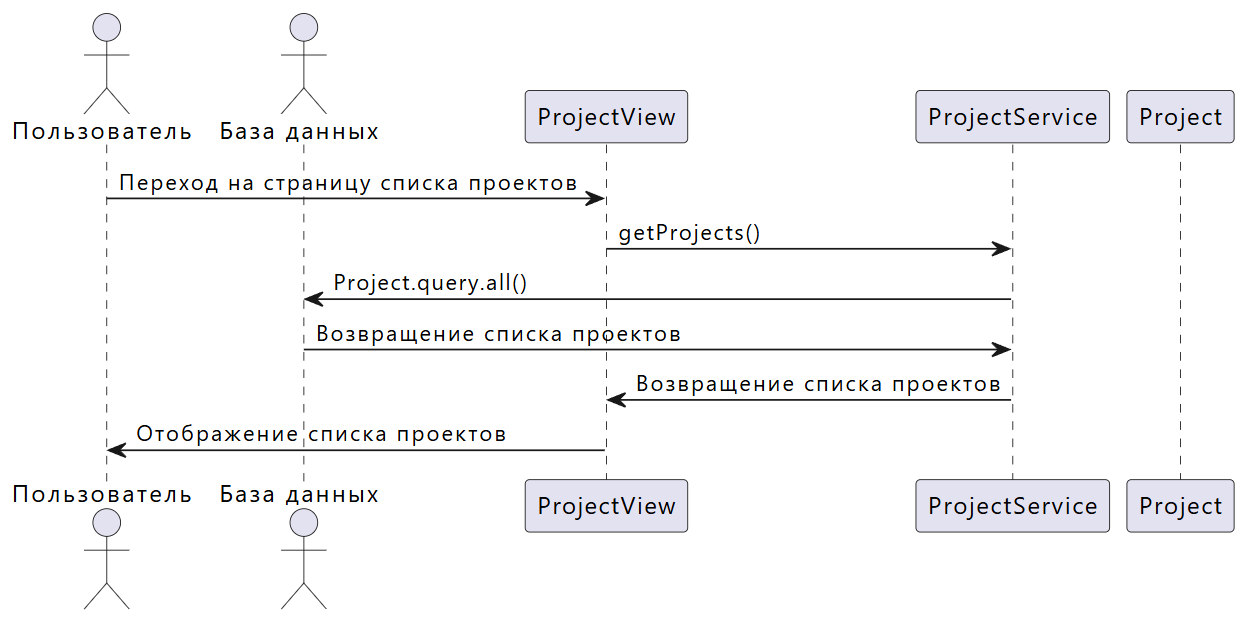


Рисунок 9 – Диаграмма последовательности для варианта использования «Просмотреть проекты»

### **2.3.7 Построение диаграммы последовательности для варианта использования «Удалить проект»**

Диаграмма последовательности, описывающая основной поток событий изображена на рисунке 10.

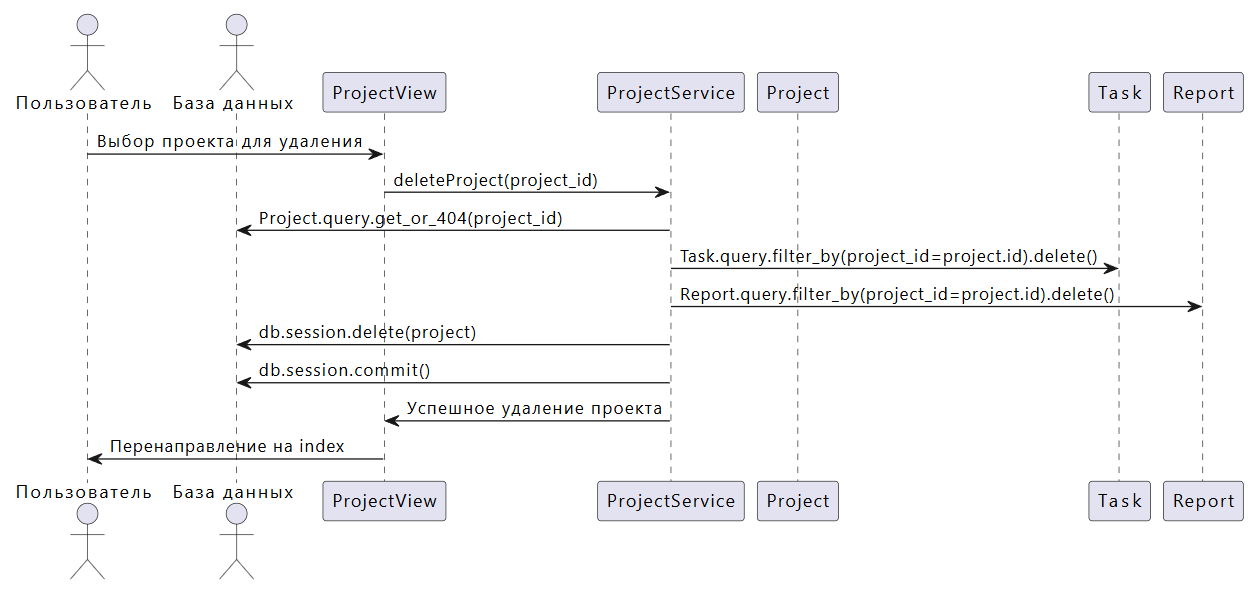


Рисунок 10 – Диаграмма последовательности для варианта использования «Удалить проект»

### **2.3.8 Построение диаграммы последовательности для варианта использования «Просмотреть детали проекта»**

Диаграмма последовательности, описывающая основной поток событий изображена на рисунке 11.

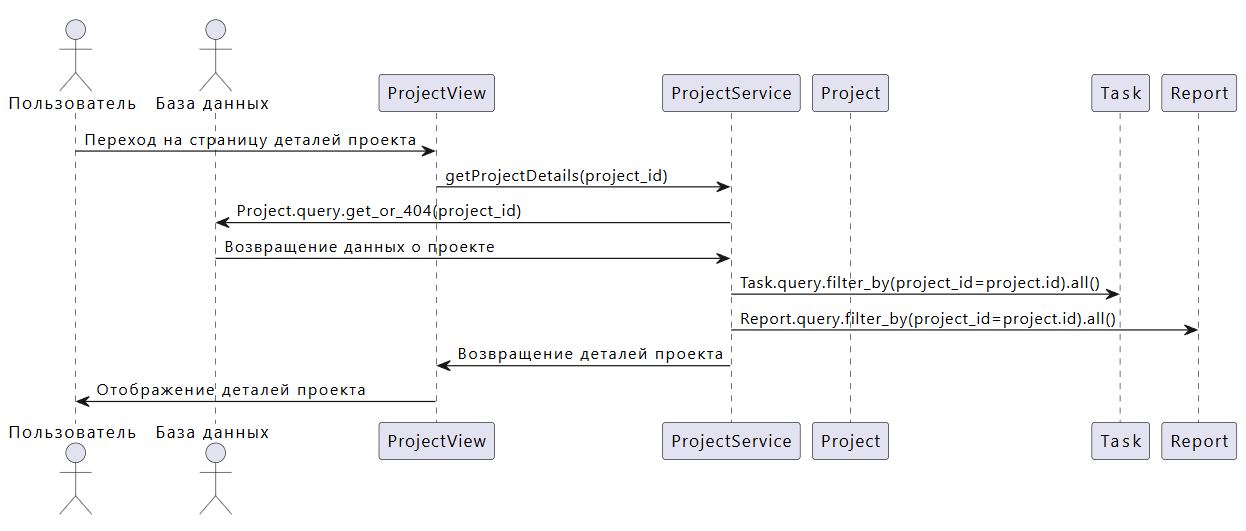


Рисунок 11 – Диаграмма последовательности для варианта использования «Просмотреть детали проекта»

### **2.3.9 Построение диаграммы последовательности для варианта использования «Создать задачу»**

Диаграмма последовательности, описывающая основной поток событий изображена на рисунке 12.

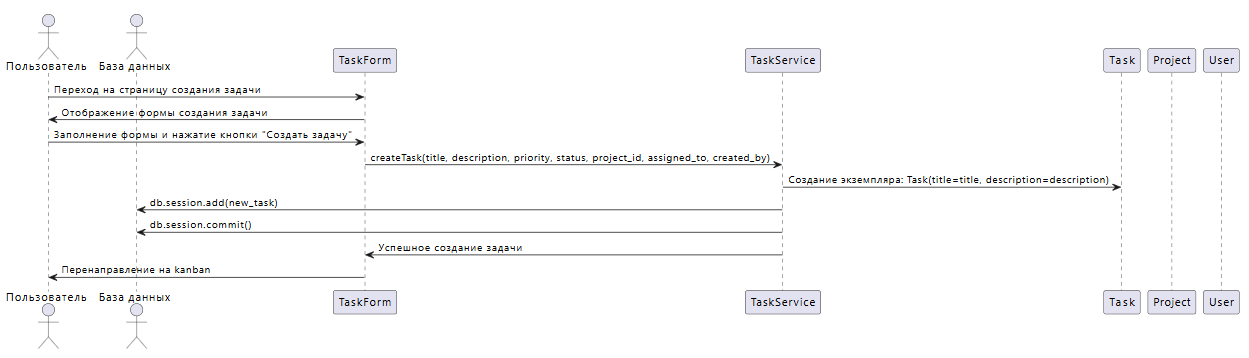


Рисунок 12 – Диаграмма последовательности для варианта использования «Создать задачу»

### **2.3.10 Построение диаграммы последовательности для варианта использования «Редактировать задачу»**

Диаграмма последовательности, описывающая основной поток событий изображена на рисунке 13.

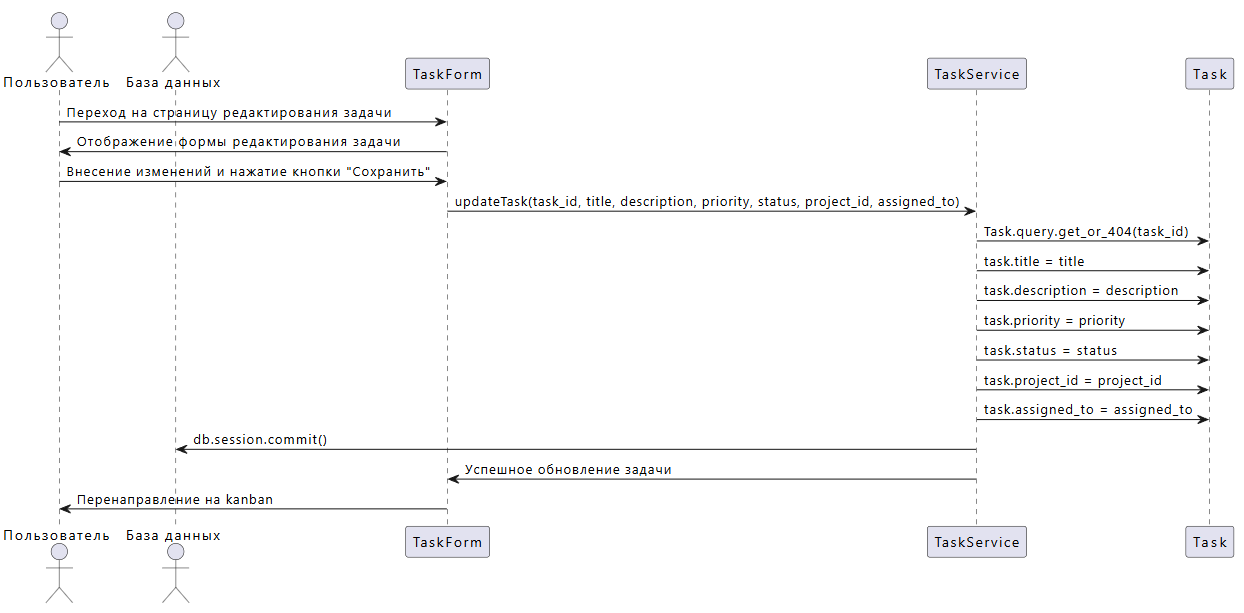


Рисунок 13 – Диаграмма последовательности для варианта использования «Редактировать задачу»

### **2.3.11 Построение диаграммы последовательности для варианта использования «Удалить задачу»**

Диаграмма последовательности, описывающая основной поток событий изображена на рисунке 14.

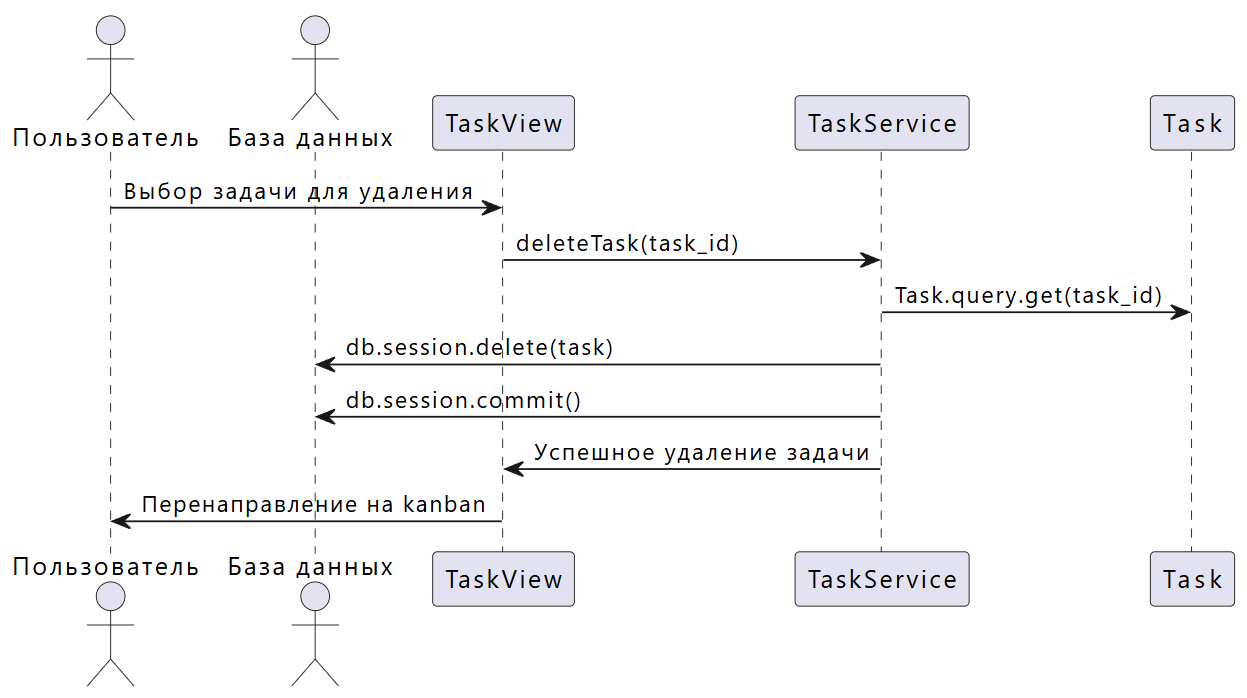


Рисунок 14 – Диаграмма последовательности для варианта использования «Удалить задачу»

### **2.3.12 Построение диаграммы последовательности для варианта использования «Просмотреть детали задачи»**

Диаграмма последовательности, описывающая основной поток событий изображена на рисунке 15.

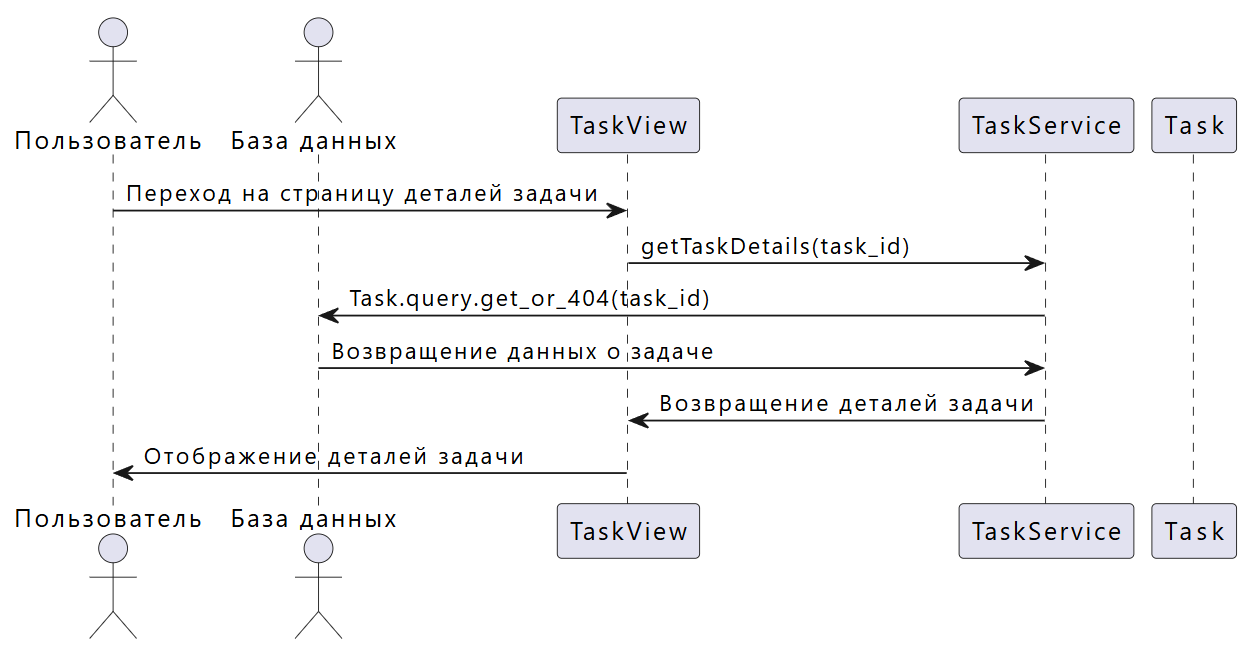


Рисунок 15 – Диаграмма последовательности для варианта использования «Просмотреть детали задачи»

### **2.3.13 Построение диаграммы последовательности для варианта использования «Изменить статус задачи»**

Диаграмма последовательности, описывающая основной поток событий изображена на рисунке 16.

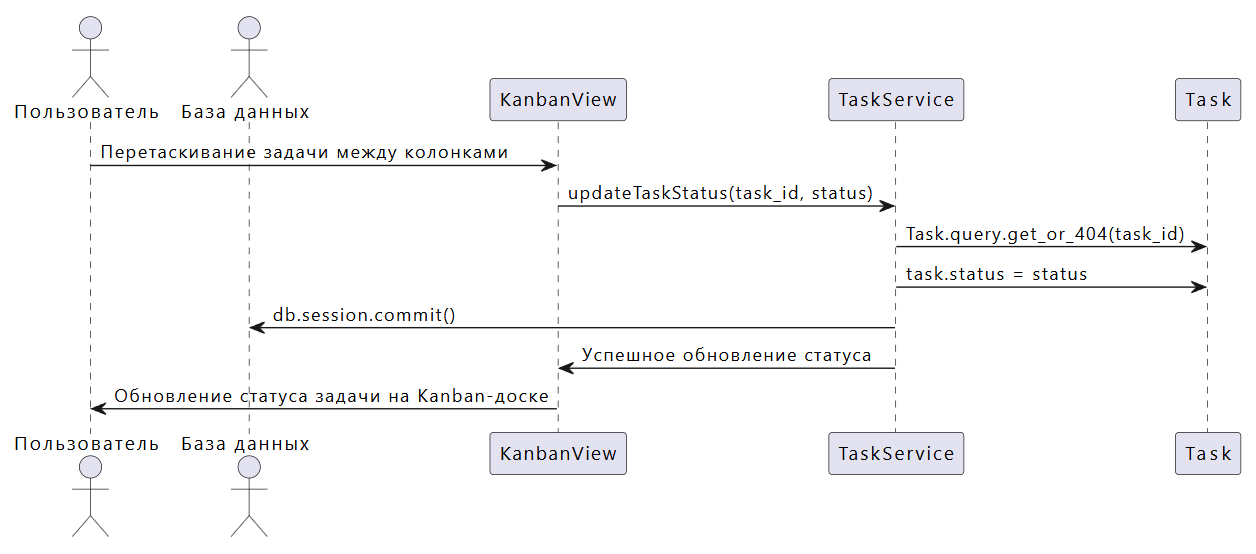


Рисунок 16 – Диаграмма последовательности для варианта использования «Изменить статус задачи»

### **2.3.14 Построение диаграммы последовательности для варианта использования «Загрузить отчет»**

Диаграмма последовательности, описывающая основной поток событий изображена на рисунке 17.

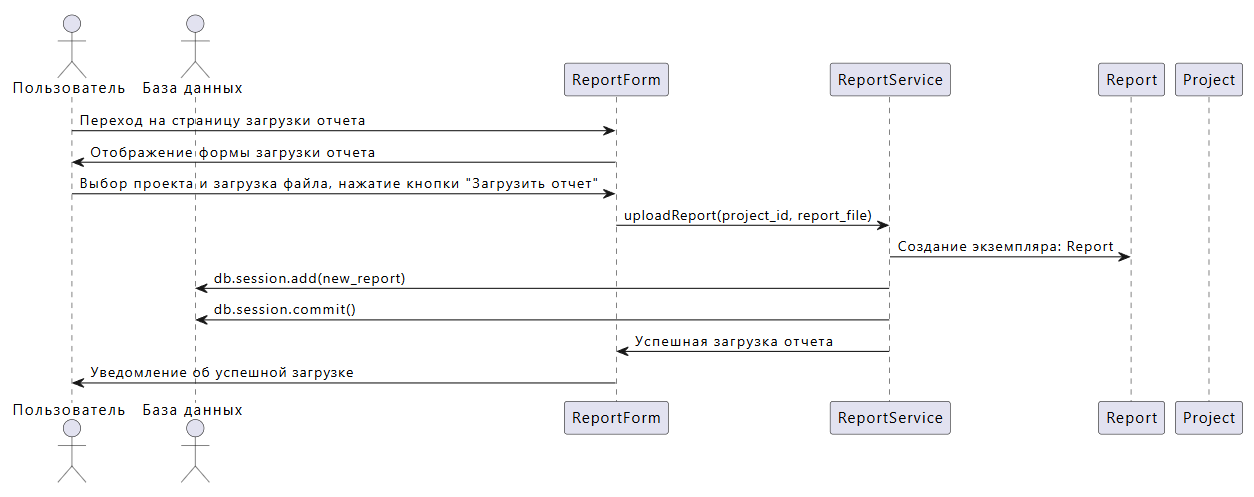


Рисунок 17 – Диаграмма последовательности для варианта использования «Загрузить отчет»

### **2.3.15 Построение диаграммы последовательности для варианта использования «Просмотреть отчеты»**

Диаграмма последовательности, описывающая основной поток событий изображена на рисунке 18.

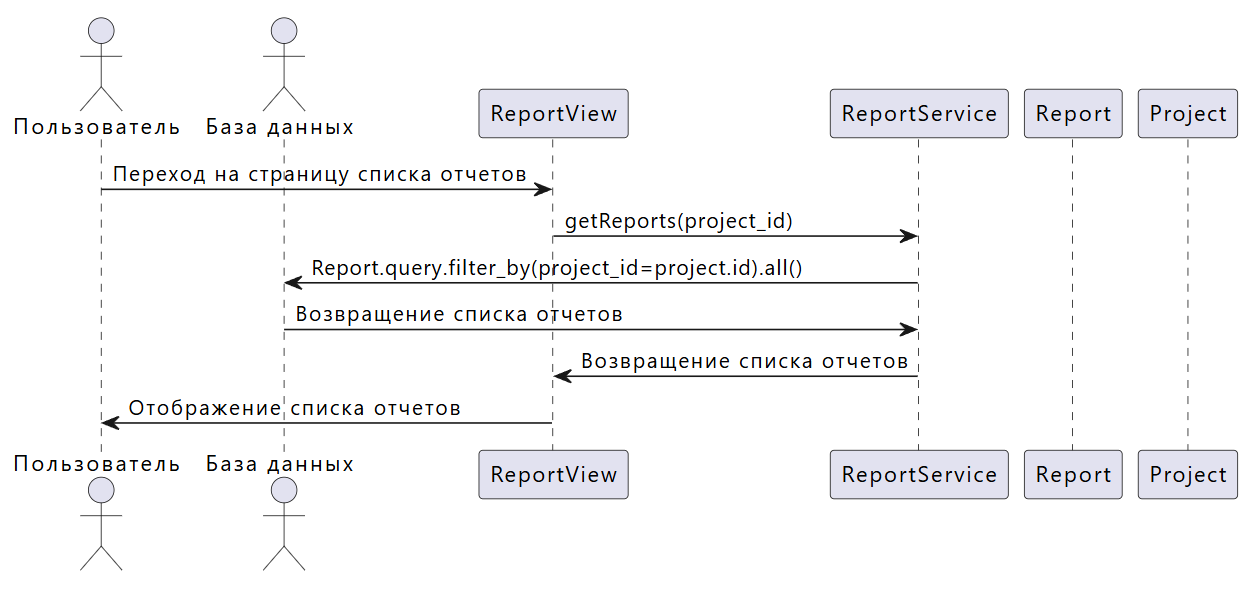


Рисунок 18 – Диаграмма последовательности для варианта использования «Просмотреть отчеты»

### **2.3.16 Построение диаграммы последовательности для варианта использования «Просмотреть список пользователей»**

Диаграмма последовательности, описывающая основной поток событий изображена на рисунке 19.

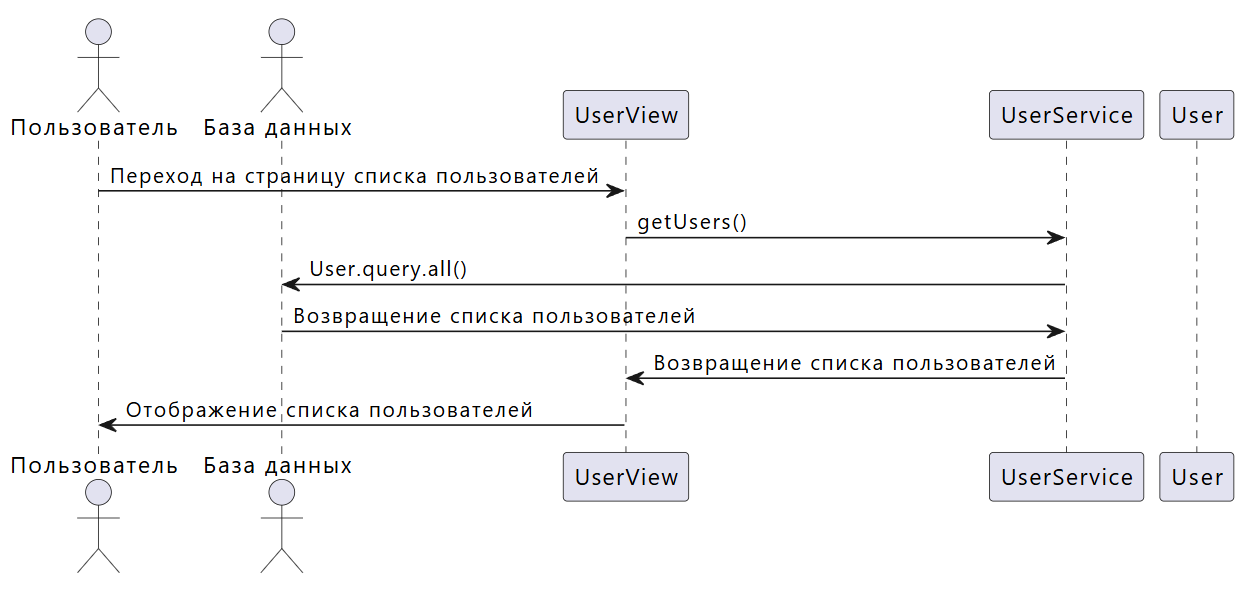


Рисунок 19 – Диаграмма последовательности для варианта использования «Просмотреть список пользователей»

### **2.3.17 Построение диаграммы последовательности для варианта использования «Создать тестового пользователя»**

Диаграмма последовательности, описывающая основной поток событий изображена на рисунке 20.

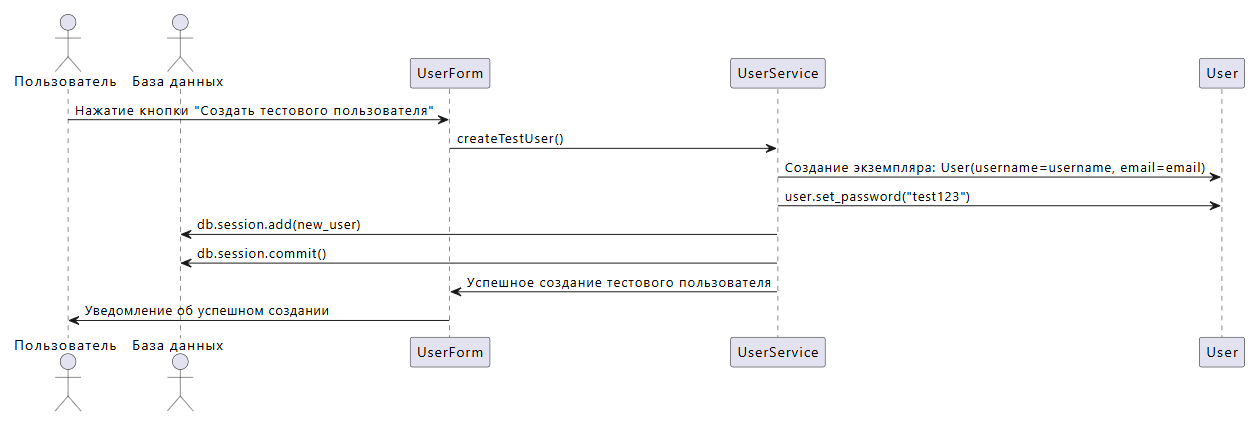


Рисунок 20 – Диаграмма последовательности для варианта использования «Создать тестового пользователя»

### **2.3.18 Построение диаграммы последовательности для варианта использования «Просмотреть задачи на Kanban-доске»**

Диаграмма последовательности, описывающая основной поток событий изображена на рисунке 21.

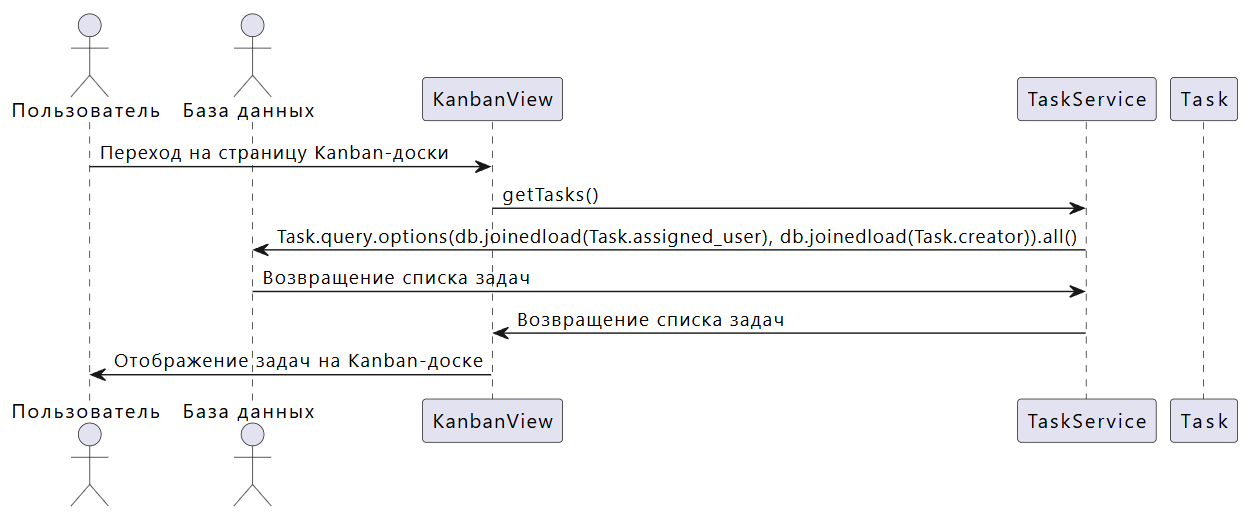


Рисунок 21 – Диаграмма последовательности для варианта использования «Просмотреть задачи на Kanban-доске»

### **2.3.19 Построение диаграммы последовательности для варианта использования «Переместить задачу между колонками»**

Диаграмма последовательности, описывающая основной поток событий изображена на рисунке 22.

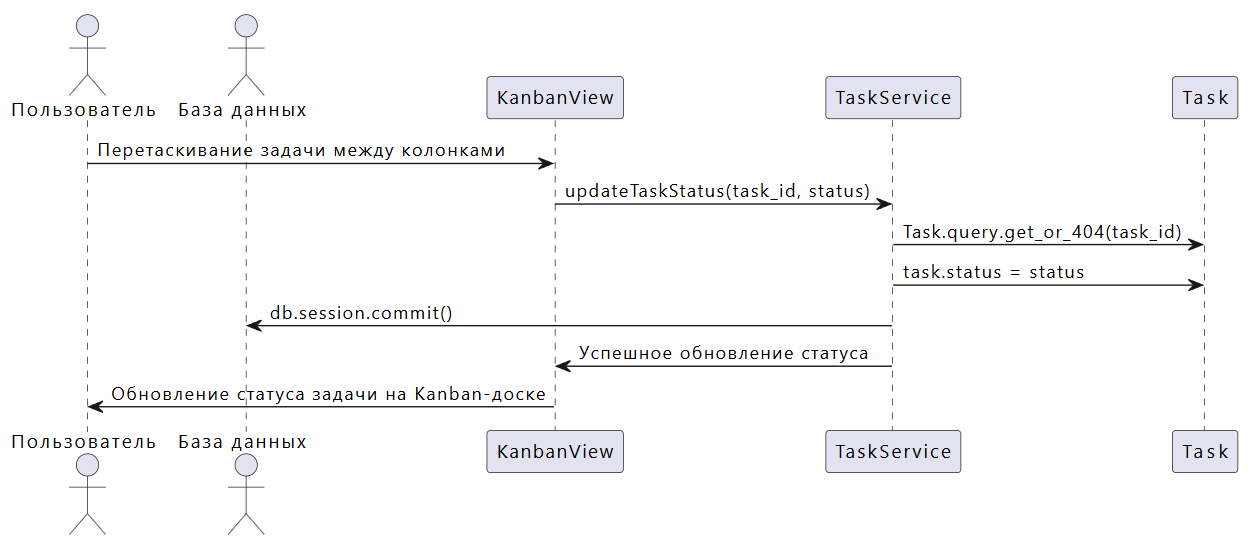


Рисунок 22 – Диаграмма последовательности для варианта использования «Переместить задачу между колонками»

### **2.3.20 Построение диаграммы последовательности для варианта использования «Просмотреть диаграммы»**

Диаграмма последовательности, описывающая основной поток событий изображена на рисунке 23.

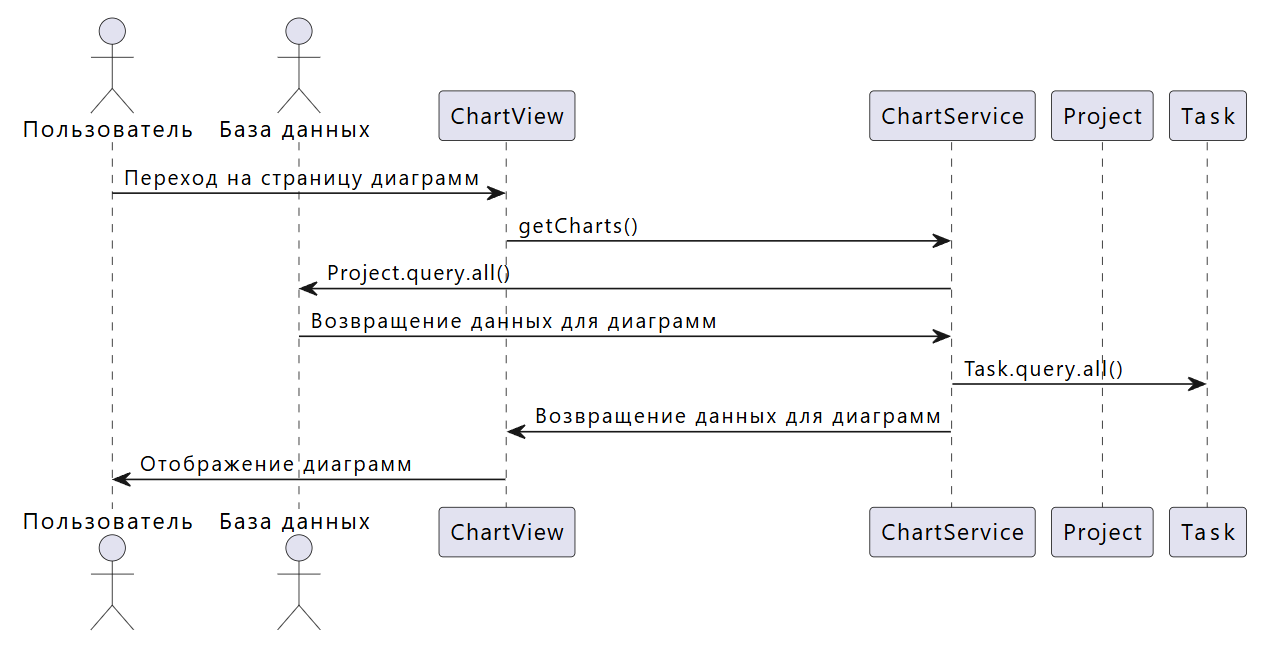


Рисунок 23 – Диаграмма последовательности для варианта использования «Просмотреть диаграммы»

## 2.4 Построение диаграммы кооперации

Структурные особенности передачи и приема сообщений между объектами представлены на диаграмме кооперации на рисунке 24.

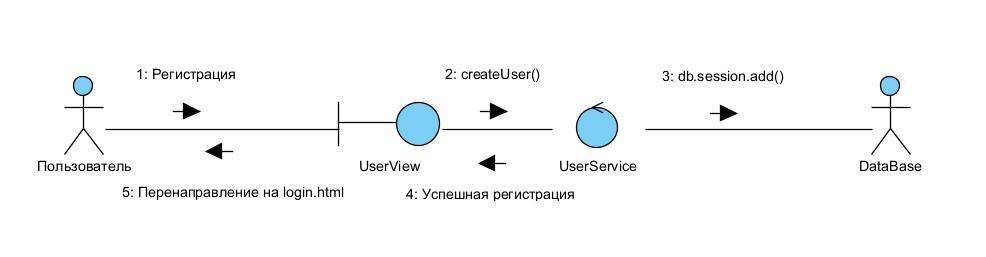


Рисунок 24 – Диаграмма кооперации

## 2.5 Построение диаграммы классов

Диаграмма классов представлена на рисунке 25.

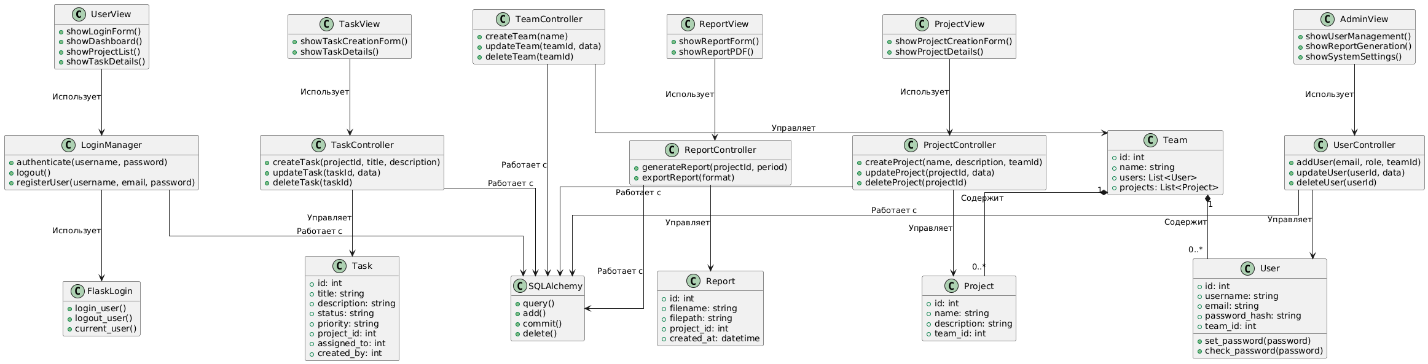


Рисунок 25 – Диаграмма классов

Класс «UserView» предоставляет интерфейс для взаимодействия пользователя с системой, включая отображение форм входа, списка проектов и деталей задач.

Класс «AdminView» отвечает за отображение интерфейса администратора, включая управление пользователями, генерацию отчетов и настройки системы.

Класс «ProjectView» предоставляет интерфейс для работы с проектами, включая создание, редактирование и просмотр деталей проекта.

Класс «TaskView» отвечает за отображение интерфейса для работы с задачами, включая создание, редактирование и просмотр деталей задачи.

Класс «ReportView» предоставляет функциональность для формирования и отображения отчетов, включая выбор параметров и просмотр PDF-документов.

Класс «LoginManager» предоставляет функциональность для управления авторизацией пользователей, включая вход, выход и загрузку данных пользователя.

Класс «ProjectController» отвечает за управление проектами, включая создание, обновление и удаление проектов, а также взаимодействие с базой данных.

Класс «TaskController» предоставляет функциональность для управления задачами, включая создание, обновление и удаление задач, а также их связь с проектами.

Класс «ReportController» отвечает за генерацию и экспорт отчетов, включая обработку данных и формирование отчетов в различных форматах.

Класс «UserController» предоставляет функциональность для управления пользователями, включая добавление, обновление и удаление пользователей, а также их связь с командами.

Класс «TeamController» отвечает за управление командами, включая создание, обновление и удаление команд, а также их связь с пользователями и проектами.

Класс «User» представляет сущность пользователя, включая его идентификатор, имя, email, хэш пароля и связь с командой.

Класс «Team» представляет сущность команды, включая её идентификатор, название, список пользователей и проектов.

Класс «Project» представляет сущность проекта, включая его идентификатор, название, описание и связь с командой.

Класс «Task» представляет сущность задачи, включая её идентификатор, название, описание, статус, приоритет и связь с проектом и пользователями.

Класс «Report» представляет сущность отчета, включая его идентификатор, имя файла, путь к файлу, связь с проектом и дату создания.

Класс «SQLAlchemy» предоставляет функциональность для работы с базой данных, включая выполнение запросов, добавление, обновление и удаление данных.

Класс «FlaskLogin» предоставляет функциональность для управления аутентификацией пользователей, включая вход, выход и управление текущей сессией.

## 2.5 Уточнение структуры классов предметной области и разработка алгоритмов методов

Таблица 22 – Список атрибутов информационных элементов

|  |  |  |  |
| --- | --- | --- | --- |
| **Класс** | **Атрибут** | **Тип** | **Описание** |
| User | ID | int | Уникальный идентификатор записи в таблице. |
| name | string | Имя пользователя. |
| email | string | Электронная почта пользователя. |
| password | string | Пароль для аутентификации пользователя. |
| role | string | Роль пользователя. |
| Team | ID | int | ИД группы. |
| name | string | Название или номер группы. |
| users | array | Список пользователей группы. |
| project | int | Принадлежность команды к проекту |
| Task | ID | int | ИД задачи |
| name | string | Название задачи. |
| description | string | Описание задачи. |
| deadline | date | Крайний срок выполнения. |
| status | string | Статус задачи. |
| Report | ID | int | ИД отчета. |
| project | int | Принадлежность отчета к проекту. |
| date | date | Дата создания отчета. |
| Project | ID | int | ИД проекта. |
| name | string | Название проекта. |
| description | string | Описание проекта. |
| deadline | date | Крайний срок выполнения. |
| status | array | Статус проекта. |
| assigned | string | Ответственный за выполнение проекта |
| tasks | array | Список задач в проекте |

Таблица 23 – Список операций информационных элементов

|  |  |  |
| --- | --- | --- |
| **Класс** | **Операция** | **Описание** |
| SQLAlchemy | create\_all() | Создать таблицы в БД() |
| session.add() | Добавить объект() |
| session.commit() | Сохранить изменения() |
| session.delete() | Удалить объект из БД() |
| Flask | route() | Определить маршрут() |
| run() | Запустить() |
| SortableJS | onEnd() | Обновить статус() |
| FileUpload | allowed\_file() | Проверить файл() |
| secure\_filename() | Очистить имя файла от небезопасных символов() |
| Chart | generate() | Генерировать диаграмму() |
| UserView | showLoginForm() | Отображает форму входа в систему. |
| showDashboard() | Отображает главную страницу пользователя. |
| showProjectList() | Отображает список проектов, доступных пользователю. |
| showTaskDetails() | Отображает детали задачи. |
| AdminView | showUserManagement() | Отображает интерфейс управления пользователями. |
| showReportGeneration() | Отображает интерфейс для генерации отчетов. |
| showSystemSettings() | Отображает настройки системы. |
| ProjectView | showProjectCreationForm() | Отображает форму создания проекта. |
| showProjectDetails() | Отображает детали проекта. |
| TaskView | showTaskCreationForm() | Отображает форму создания задачи. |
| showTaskDetails() | Отображает детали задачи. |
| ReportView | showReportForm() | Отображает форму для выбора параметров отчета. |
| showReportPDF() | Отображает сгенерированный отчет в формате PDF. |
| LoginManager | authenticate() | Проверяет учетные данные пользователя и выполняет вход в систему. |
| logout() | Завершает сессию пользователя. |
| registerUser() | Регистрирует нового пользователя в системе. |
| ProjectController | createProject() | Создает новый проект. |
| updateProject() | Обновляет данные проекта. |
| deleteProject() | Удаляет проект. |
| TaskController | createTask() | Создает новую задачу. |
| updateTask() | Обновляет данные задачи. |
| deleteTask() | Удаляет задачу. |
| ReportController | generateReport() | Генерирует отчет на основе данных проекта. |
| exportReport() | Экспортирует отчет в выбранном формате. |
| UserController | addUser() | Добавляет нового пользователя в систему. |
| updateUser() | Обновляет данные пользователя. |
| deleteUser() | Удаляет пользователя из системы. |
| TeamController | createTeam() | Создает новую команду. |
| updateTeam() | Обновляет данные команды. |
| deleteTeam() | Удаляет команду. |

## 2.7 Проектирование интерфейса пользователя

На основании диаграмм последовательностей разработана диаграмма состояний, представленная на рисунке 26.

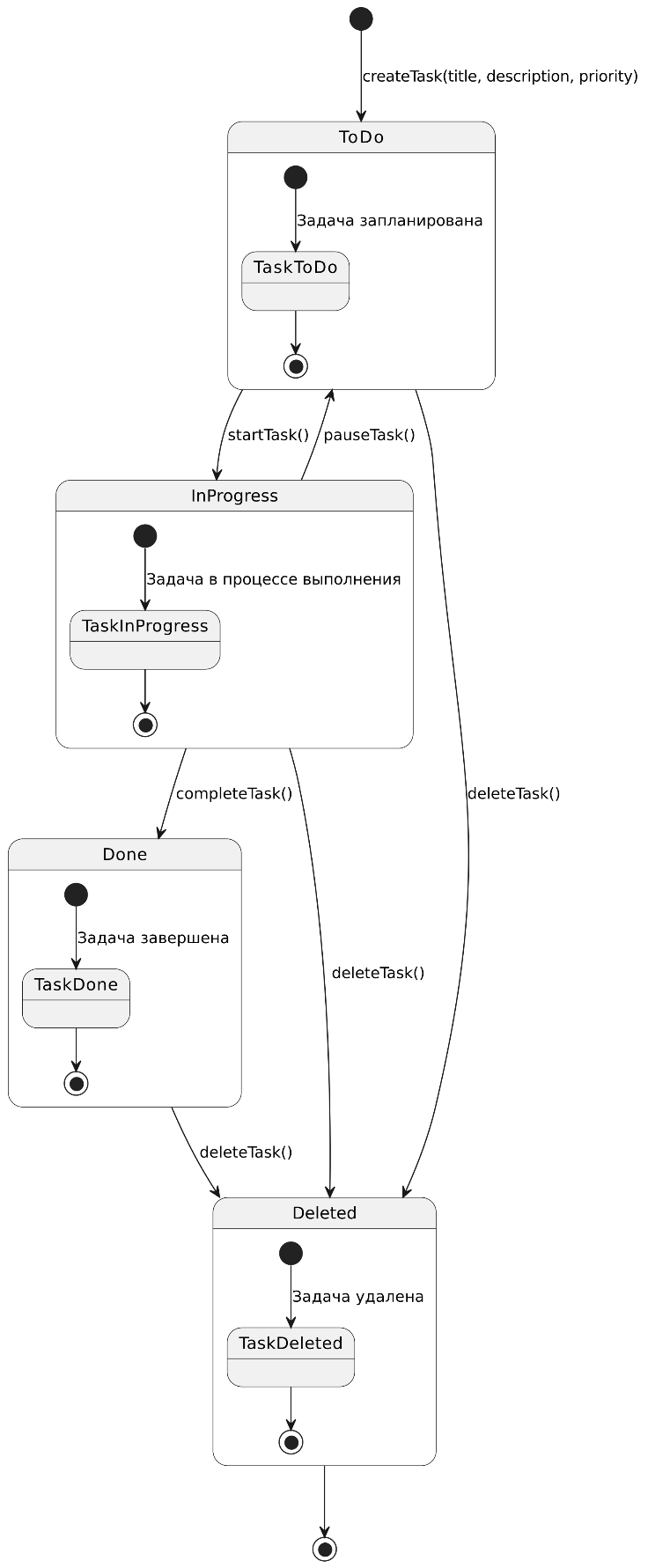


Рисунок 26 – Диаграмма состояний для объекта «Задача»

## 2.8 Разработка алгоритма функционирования системы

Алгоритм работы требования «Управление рабочим процессом» в виде диаграммы деятельностей приведен на рисунке 27.

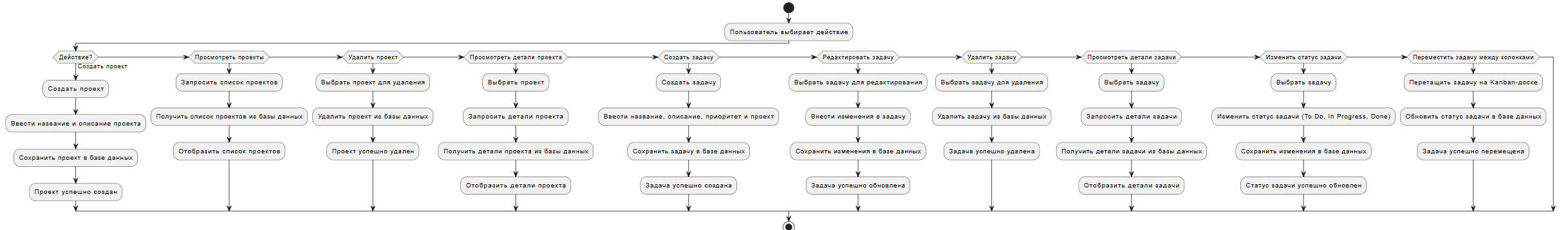


Рисунок 27 – Диаграмма деятельностей «Работа с задачами»

# 

# 3 Реализация системы

## 3.1 Реализация программного обеспечения системы

При разработке приложения был использован язык программирования – Python - объектно-ориентированный язык программирования общего назначения – для создания общей структуры и классовой иерархии; для создания и описания графических компонентов веб сайта был использован расширяемый язык разметки (HyperText Markup Language) HTML. Также были использованы такие инструменты как CSS и JS. HTML и CSS используются для создания структуры и стилей веб-страниц, а JavaScript — для интерактивности.

### **3.1.1 Разработка диаграммы компонентов**

Реализация программного обеспечения системы представлена на изображении в виде диаграммы компонентов. Она определяет архитектуру разрабатываемой системы на физическом уровне и представляет зависимости между программными компонентами.

Система включает в себя следующие компоненты (см. изображение 23):

**Модуль управления пользователями**:

login.html: Шаблон для страницы входа в систему.

register.html: Шаблон для страницы регистрации пользователя.

users.html: Шаблон для отображения списка пользователей.

**Модуль управления командами**:

create\_team.html: Шаблон для создания новой команды.

select\_team.html: Шаблон для выбора команды.

**Модуль управления проектами**:

create\_project.html: Шаблон для создания нового проекта.

project\_detail.html: Шаблон для отображения деталей проекта.

projects.html: Шаблон для отображения списка проектов.

**Модуль управления задачами**:

task\_creator.html: Шаблон для создания новой задачи.

edit\_task.html: Шаблон для редактирования задачи.

task\_detail.html: Шаблон для отображения деталей задачи.

kanban.html: Шаблон для отображения канбан-доски с задачами.

**Модуль аналитики и отчетности:**

reports.html: Шаблон для отображения списка отчетов.

charts.html: Шаблон для отображения графиков и диаграмм.

**Модуль администрирования**:

dashboard.html: Шаблон для панели управления.

**Базовые шаблоны**:

base.html: Базовый шаблон для всех страниц.

index.html: Шаблон для главной страницы.

**Основные файлы сервера**:

app.py: Основной файл сервера, содержащий маршруты и логику приложения.

database.py: Файл для работы с базой данных, содержащий модели и настройки.

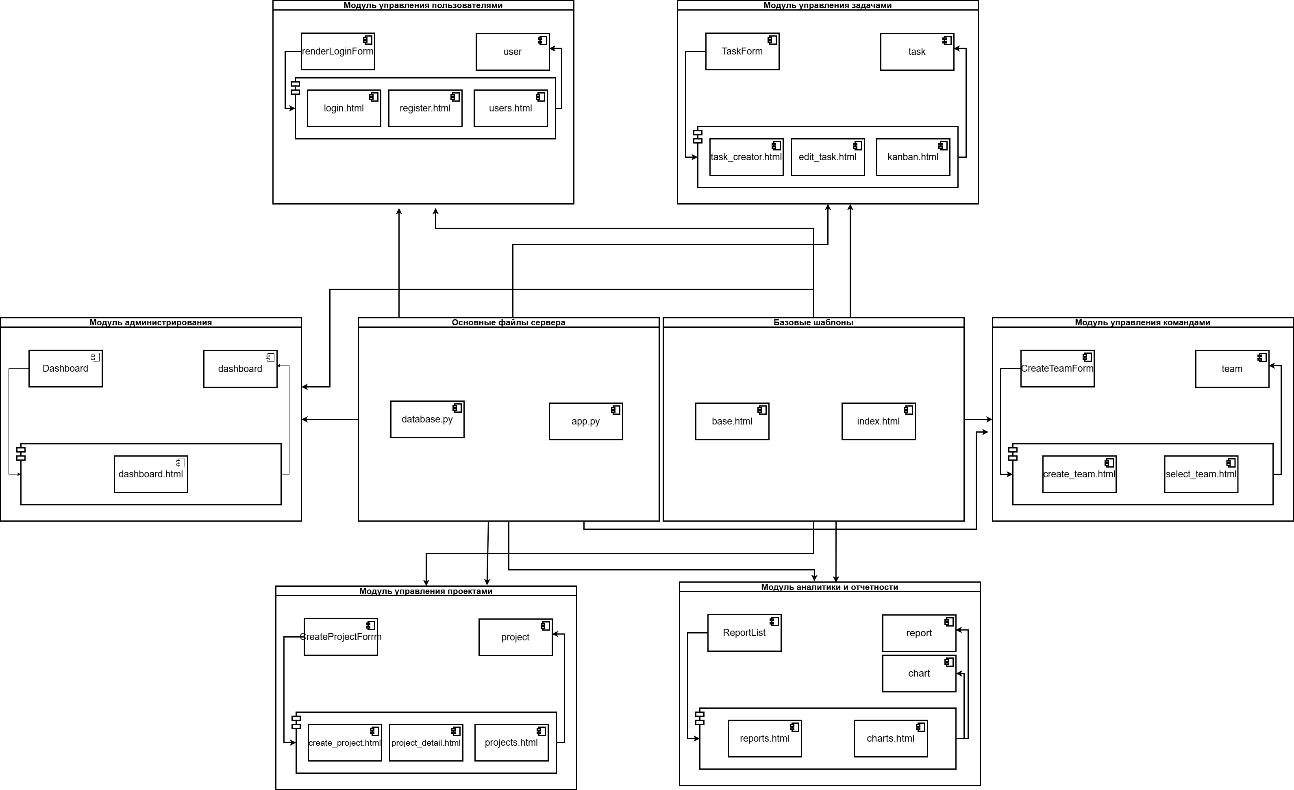


Рисунок 28 – Диаграмма компонентов приложения

## 3.2 Реализация технического обеспечения

Полная диаграмма развертывания приложения приведена на рисунке 24.

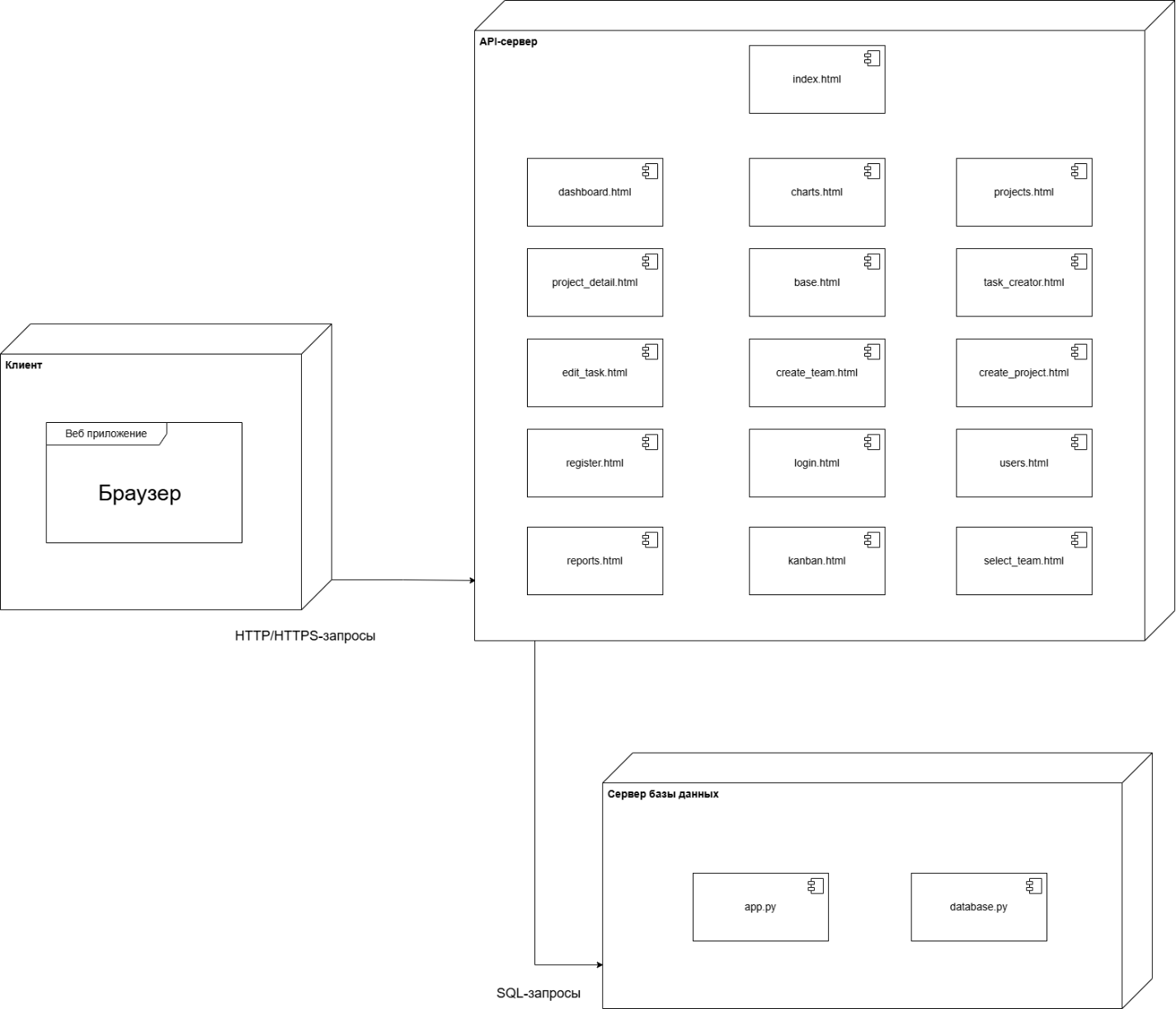


Рисунок 29 – Диаграмма развертывания приложения

## 3.3 Объекты интерфейса пользователя

Внешний вид формы главного меню раздела «Проекты» представлен на рисунке 30.

6

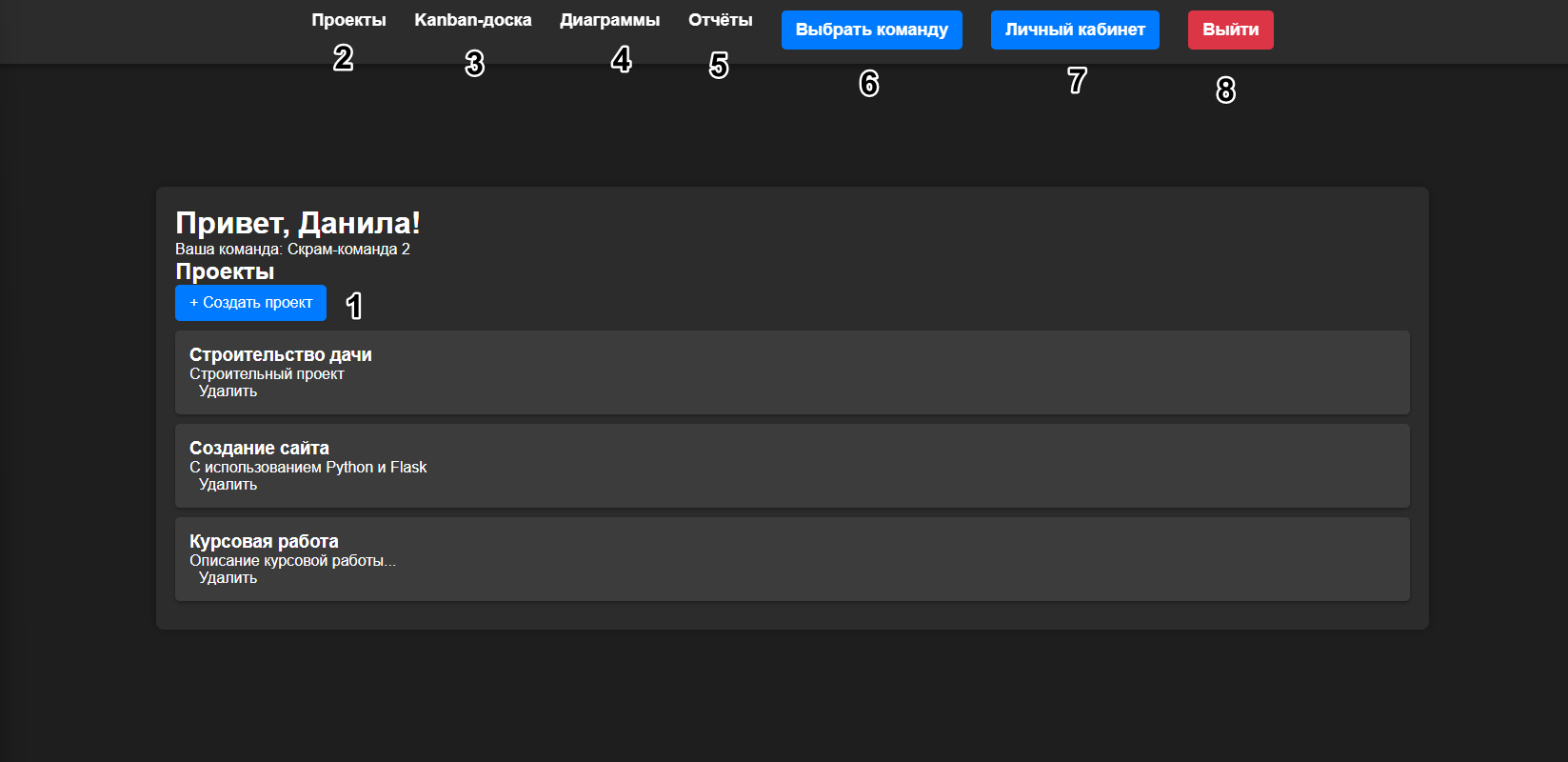


Рисунок 30 – Форма главного меню раздела «Проекты»

В таблице 24 представлены расположенные на форме компоненты.

Таблица 24 – Компоненты формы для авторизованного пользователя

|  |  |  |  |
| --- | --- | --- | --- |
| **№ на рисунке** | **Наименование компонента** | **Тип компонента** | **Назначение** |
| 1 | button | Button | Кнопка для создания проекта |
| 2 | button | Button | Кнопка для в раздел «Проекты» |
| 3 | button | Button | Кнопка для перехода и просмотра раздела «Kanban-доска» |
| 4 | button | Button | Кнопка для перехода и просмотра раздела «Диаграммы» |
| 5 | button | Button | Кнопка для перехода и просмотра раздела «Отчеты» |
| 6 | button | Button | Кнопка для перехода и просмотра раздела «Команды» |
| 7 | button | Button | Кнопка для перехода и просмотра раздела «Личный кабинет» |
| 8 | button | Button | Кнопка для выхода из аккаунта |

Внешний вид формы главного меню раздела «Kanban-доска» для авторизованного пользователя представлен на рисунке 31.

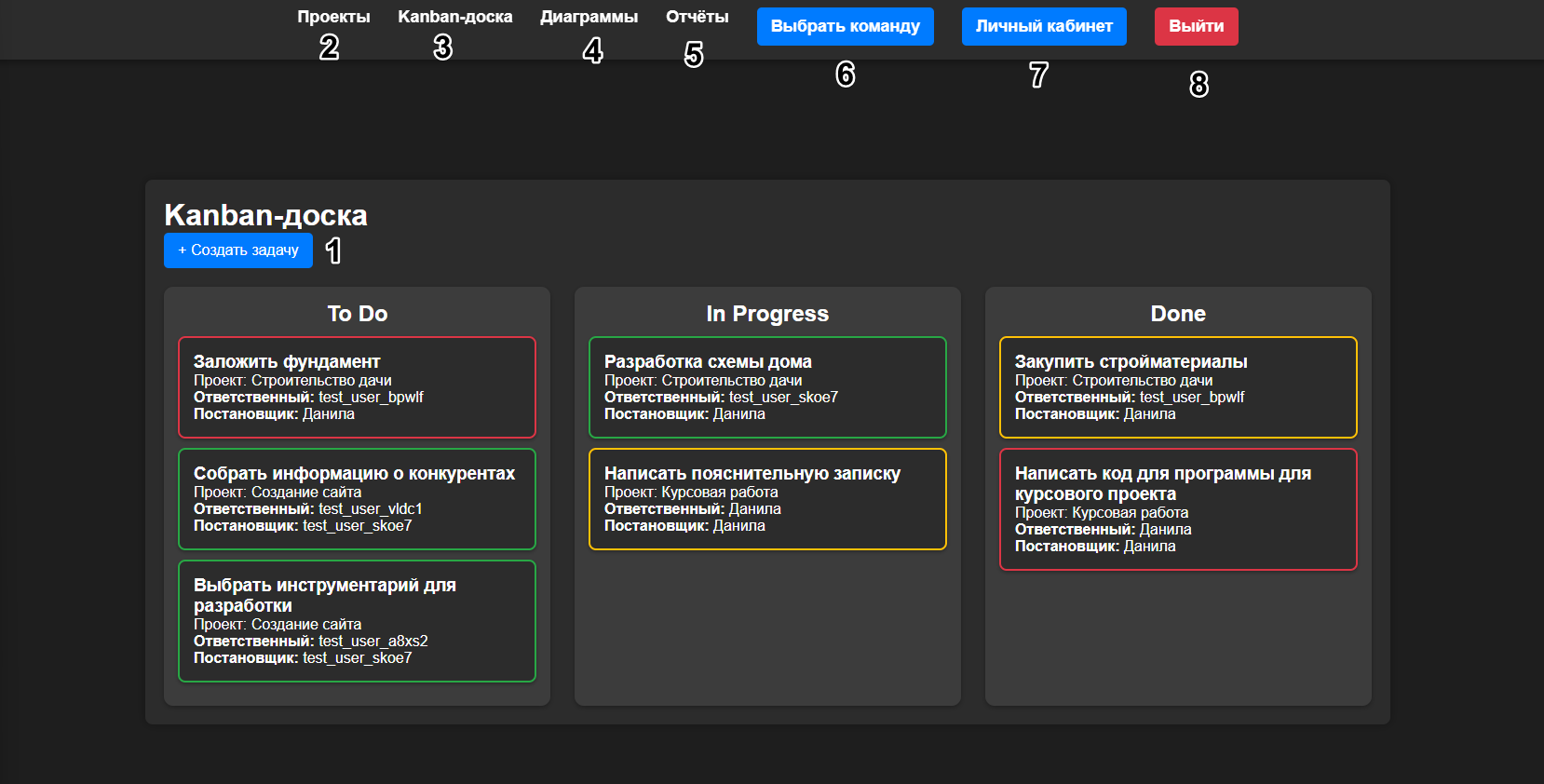


Рисунок 31 – Форма главного меню раздела «Kanban-доска»

В таблице 25 представлены расположенные на форме компоненты.

Таблица 25 – Компоненты формы для авторизованного пользователя

|  |  |  |  |
| --- | --- | --- | --- |
| **№ на рисунке** | **Наименование компонента** | **Тип компонента** | **Назначение** |
| 1 | button | Button | Кнопка для создания новой задачи |
| 2 | button | Button | Кнопка для перехода и просмотра раздела «Проекты» |
| 3 | button | Button | Кнопка для перехода и просмотра раздела «Kanban-доска» |
| 4 | button | Button | Кнопка для перехода и просмотра раздела «Диаграммы» |
| 5 | button | Button | Кнопка для перехода и просмотра раздела «Отчеты» |
| 6 | button | Button | Кнопка для перехода и просмотра раздела «Команды» |
| 7 | button | Button | Кнопка для перехода и просмотра раздела «Личный кабинет» |
| 8 | button | Button | Кнопка для выхода из учетной записи |

Внешний вид формы главного меню раздела «Диаграммы» представлен на рисунке 32.

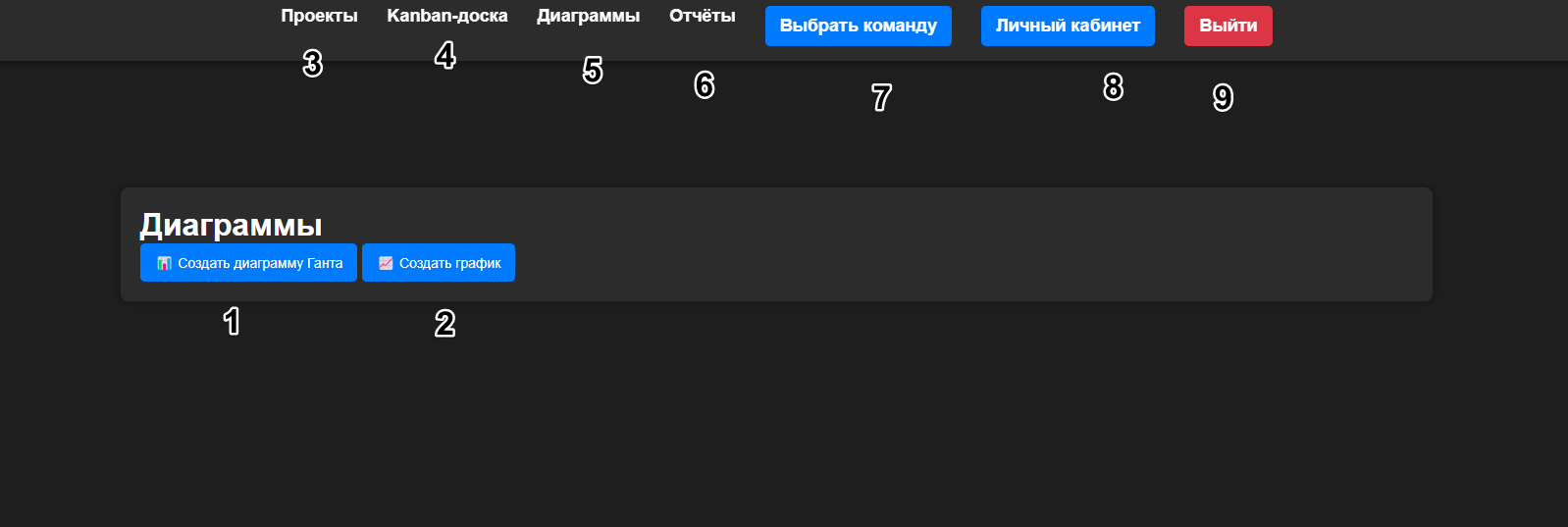


Рисунок 32 – Форма главного меню раздела «Диаграммы»

В таблице 26 представлены расположенные на форме компоненты.

Таблица 26 – Компоненты формы

|  |  |  |  |
| --- | --- | --- | --- |
| **№ на рисунке** | **Наименование компонента** | **Тип компонента** | **Назначение** |
| 1 | button | Button | Кнопка для создания диаграммы Ганта |
| 2 | button | Button | Кнопка для создания графика |
| 3 | button | Button | Кнопка для перехода и просмотра раздела «Проекты» |
| 4 | button | Button | Кнопка для перехода и просмотра раздела «Kanban-доска» |
| 5 | button | Button | Кнопка для перехода и просмотра раздела «Диаграммы» |
| 6 | button | Button | Кнопка для перехода и просмотра раздела «Отчеты» |
| 7 | button | Button | Кнопка для перехода и просмотра раздела «Команды» |

Внешний вид формы главного меню раздела «Отчеты» представлен на рисунке 33.

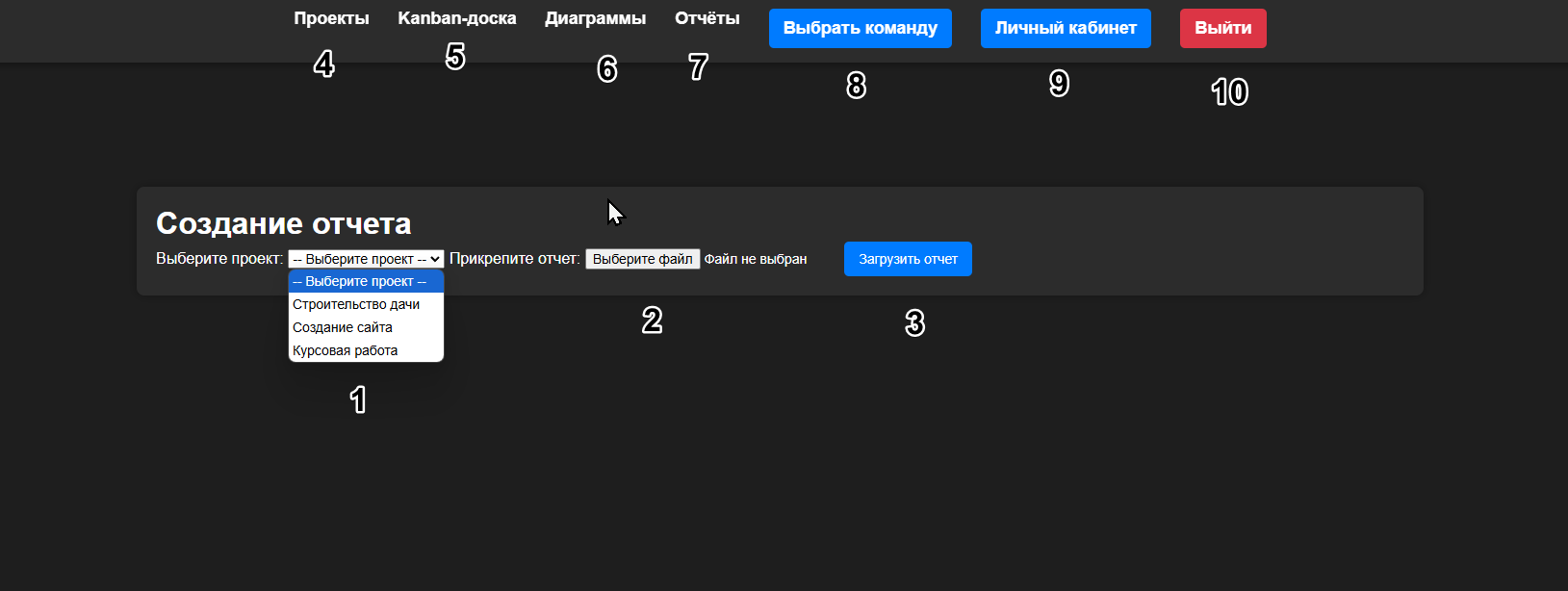


Рисунок 33 – Форма главного меню раздела «Отчеты»

В таблице 27 представлены расположенные на форме компоненты.

Таблица 27 – Компоненты формы

|  |  |  |  |
| --- | --- | --- | --- |
| **№ на рисунке** | **Наименование компонента** | **Тип компонента** | **Назначение** |
| 1 | list | Drop down list | Выпадающий список с выбором проекта |
| 2 | button | Button | Кнопка для прикрепления файла |
| 3 | button | Button | Кнопка для загрузки файла на сервер |
| 4 | button | Button | Кнопка для перехода и просмотра раздела «Проекты» |
| 5 | button | Button | Кнопка для перехода и просмотра раздела «Kanban-доска» |
| 6 | button | Button | Кнопка для перехода и просмотра раздела «Диаграммы» |
| 7 | button | Button | Кнопка для перехода и просмотра раздела «Отчеты» |
| 8 | button | Button | Кнопка для перехода и просмотра раздела «Команды» |
| 9 | button | Button | Кнопка для перехода и просмотра раздела «Личный кабинет» |

Внешний вид формы главного меню раздела «Создать задачу» для авторизованного пользователя представлен на рисунке 34.

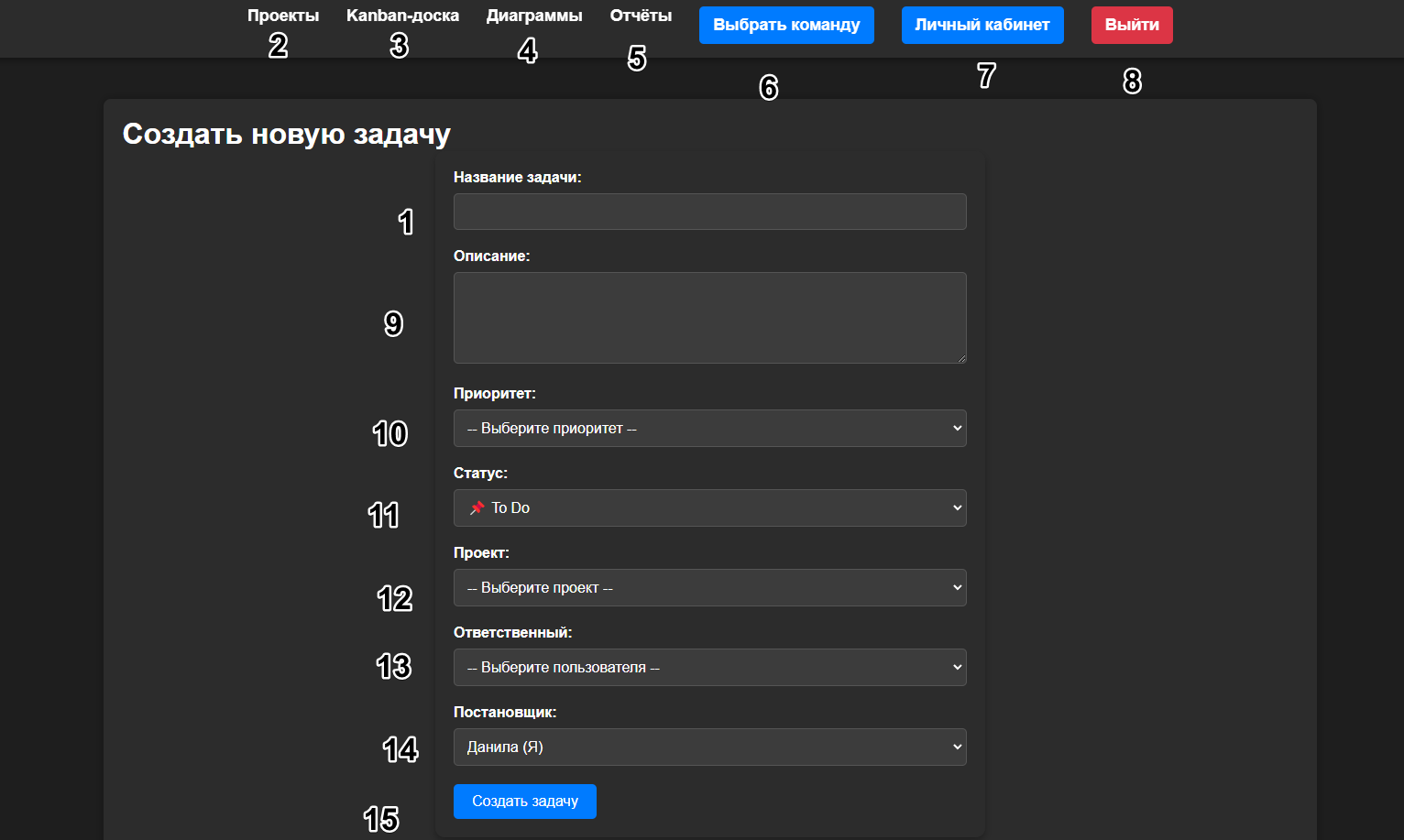


Рисунок 34 – Форма главного меню раздела «Создать задачу»

В таблице 28 представлены расположенные на форме компоненты.

Таблица 28 – Компоненты формы

|  |  |  |  |
| --- | --- | --- | --- |
| **№ на рисунке** | **Наименование компонента** | **Тип компонента** | **Назначение** |
| 1 | textbox | Textbox | Поле для ввода названия задачи |
| 2 | button | Button | Кнопка для перехода и просмотра раздела «Проекты» |
| 3 | button | Button | Кнопка для перехода и просмотра раздела «Kanban-доска» |
| 4 | button | Button | Кнопка для перехода и просмотра раздела «Диаграммы» |
| 5 | button | Button | Кнопка для перехода и просмотра раздела «Отчеты» |
| 6 | button | Button | Кнопка для перехода и просмотра раздела «Команды» |
| 7 | button | Button | Кнопка для перехода и просмотра раздела «Личный кабинет» |
| 8 | button | Button | Кнопка для выхода из учетной записи |
| 9 | textbox | Textbox | Поле для ввода описания |
| 10 | list | Drop down list | Выпадающий список с выбором приоритета |
| 11 | list | Drop down list | Выпадающий список с выбором статуса |
| 12 | list | Drop down list | Выпадающий список с выбором проекта |
| 13 | list | Drop down list | Выпадающий список с выбором ответственного |
| 14 | list | Drop down list | Выпадающий список с выбором постановщика |
| 15 | button | Button | Кнопка для создания задачи |

Внешний вид формы главного меню раздела «Создать проект» представлен на рисунке 35.

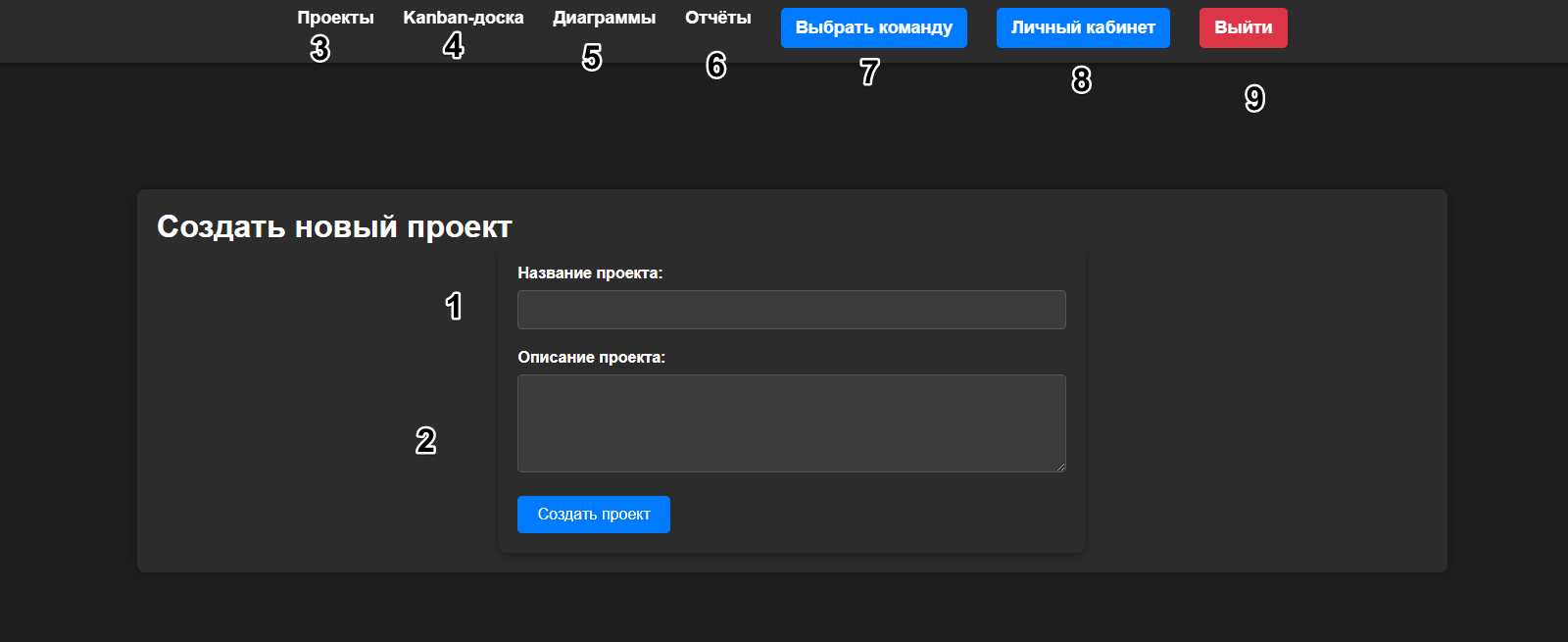


Рисунок 35 – Форма главного меню раздела «Создать проект»

В таблице 29 представлены расположенные на форме компоненты.

Таблица 29 – Компоненты формы

|  |  |  |  |
| --- | --- | --- | --- |
| **№ на рисунке** | **Наименование компонента** | **Тип компонента** | **Назначение** |
| 1 | textbox | Textbox | Поле для ввода названия |
| 2 | textbox | Textbox | Поле для ввода описания |
| 3 | button | Button | Кнопка для перехода и просмотра раздела «Проекты» |
| 4 | button | Button | Кнопка для перехода и просмотра раздела «Kanban-доска» |
| 5 | button | Button | Кнопка для перехода и просмотра раздела «Диаграммы» |
| 6 | button | Button | Кнопка для перехода и просмотра раздела «Отчеты» |
| 7 | button | Button | Кнопка для перехода и просмотра раздела «Команды» |
| 8 | button | Button | Кнопка для перехода и просмотра раздела «Личный кабинет» |
| 9 | button | Button | Кнопка для выхода из учетной записи |

Внешний вид формы главного меню раздела «Команды» представлен на рисунке 36.

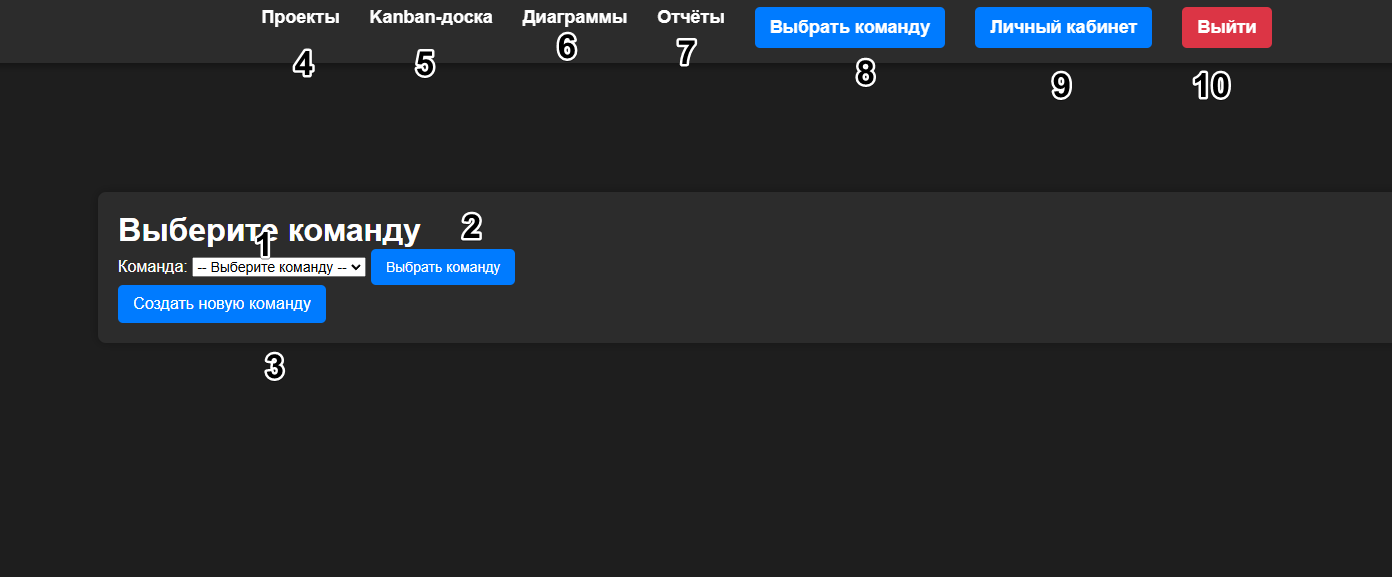


Рисунок 31 – Форма главного меню раздела «Команды»

В таблице 30 представлены расположенные на форме компоненты.

Таблица 30 – Компоненты формы для Студента

|  |  |  |  |
| --- | --- | --- | --- |
| **№ на рисунке** | **Наименование компонента** | **Тип компонента** | **Назначение** |
| 1 | list | Drop down list | Выпадающий список с выбором команды |
| 2 | button | Button | Кнопка для выбора команды |
| 3 | button | Button | Кнопка для создания команды |
| 4 | button | Button | Кнопка для перехода и просмотра раздела «Проекты» |
| 5 | button | Button | Кнопка для перехода и просмотра раздела «Kanban-доска» |
| 6 | button | Button | Кнопка для перехода и просмотра раздела «Диаграммы» |
| 7 | button | Button | Кнопка для перехода и просмотра раздела «Отчеты» |
| 8 | button | Button | Кнопка для перехода и просмотра раздела «Команды» |
| 9 | button | Button | Кнопка для перехода и просмотра раздела «Личный кабинет» |
| 10 | button | Button | Кнопка для выхода из учетной записи |

Программа для отображения компонентов форм включает несколько экранов (форм), реализованных на языке Python и HTML с использованием CSS-стилей и JavaScript. Эти экраны обеспечивают интерфейс для взаимодействия пользователя с приложением и управления рабочим процессом и управления проектами.

# заключение

В ходе выполнения курсового проекта была достигнута основная цель — разработана информационная система для управления проектами, направленная на повышение эффективности организации и контроля рабочих процессов. Приложение обеспечивает удобное управление задачами, проектами, командами и отчётами, что способствует улучшению координации и прозрачности в работе над проектами.

В первой главе проекта проведён аналитический обзор предметной области. Рассмотрены существующие подходы к управлению проектами, выявлены проблемы, такие как недостаточная гибкость, отсутствие интеграции между задачами и проектами, а также сложность в отслеживании прогресса. Эти данные сформировали требования к разрабатываемому приложению.

Во второй главе разработаны функциональные и архитектурные требования, включая модули управления пользователями, командами, проектами, задачами и отчётами. Описана структура системы, обеспечивающая удобство использования, масштабируемость и интеграцию с внешними инструментами.

В третьей главе реализована разработка приложения с использованием современных технологий, таких как Flask, SQLAlchemy и Bootstrap. Интеграция компонентов, включая интерфейс и серверную часть, обеспечила его работоспособность. Тестирование подтвердило стабильность и соответствие поставленным задачам.

Таким образом, проект достиг целей, и созданное приложение станет эффективным инструментом для управления проектами. Оно имеет потенциал для масштабирования и применения в различных организациях, включая образовательные учреждения, IT-компании и другие сферы, где требуется чёткое управление задачами и проектами.

.

# список использованных источников

1. Межгосударственный стандарт ГОСТ 7.32-2017. Система стандартов по информации, библиотечному и издательскому делу. Отчет о научно – исследовательской работе. Структура и правила оформления: [сайт]. – URL: https://www.rea.ru/ru/org/managements/orgnirupr/Documents/gost\_7.32-2017.pdf (дата обращения 22.12.2024).
2. Международный стандарт ISO/IEC 12207. Информационные технологии. Процессы жизненного цикла программного обеспечения: [сайт]. – URL: https://kaf401.rloc.ru/TRPO/GOST\_ISOIEC\_12207.pdf (дата обращения 01.11.2024).
3. ГОСТ 34.601-90. Автоматизированные системы. Стадии создания: [сайт]. – URL: https://www.astoni.ru/upload/iblock/2d4/GOST-34.601\_90.pdf (дата обращения 15.10.2024).
4. ГОСТ 34.602-89. Информационная технология. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы: [сайт]. – URL: https://irkobl.ru/sites/saio/rgistp/td/ГОСТ\_34-602-89\_ТЗ\_на\_создание\_АС.pdf (дата обращения 02.11.2024).
5. ГОСТ 34.603-92. Информационная технология. Виды испытаний автоматизированных систем: [сайт]. – URL: http://vsegost.com/Catalog/19/19032.shtml (дата обращения 13.10.2024).
6. Васильев, А. Самоучитель Java с примерами и программами. 3-е издание. / А. Н. Васильев – СПб.: Наука и Техника, 2016. – 368 с.: ил.
7. Шилдт, Г. Java 8: руководство для начинающих, 6-е изд.: Пер. с англ. /Г. Шилдт. – М. ООО "И.Д. Вильяме", 2015. – 720 с.: ил. - Парал. тит. англ. – ISBN 978-5-8459-1955-7.
8. Васильев, А. Java для всех. / А. Васильев – СПб.: Питер, 2020. – 512 с.: ил. – (Библиотека программиста). – ISBN 978-5-4461-1382-8

# приложение а

**Текст программы**

App.py  
from flask import Flask, render\_template, request, redirect, url\_for, flash, send\_from\_directory, jsonify

from flask\_sqlalchemy import SQLAlchemy

from flask\_login import LoginManager, login\_user, login\_required, logout\_user, current\_user

from database import db, User, Project, Task, Report, Team

import os, random, string

from werkzeug.utils import secure\_filename

from datetime import datetime

# Инициализация приложения

app = Flask(\_\_name\_\_, template\_folder="../templates", static\_folder="../static")

app.config['SQLALCHEMY\_DATABASE\_URI'] = 'sqlite:///../project\_management.db'

app.config['SQLALCHEMY\_TRACK\_MODIFICATIONS'] = False

app.config['SECRET\_KEY'] = 'supersecretkey'

# Настройка папки для загрузки файлов

UPLOAD\_FOLDER = 'uploads'

ALLOWED\_EXTENSIONS = {'pdf', 'doc', 'docx', 'xls', 'xlsx', 'txt'}

app.config['UPLOAD\_FOLDER'] = UPLOAD\_FOLDER

# Создание папки для загрузки файлов, если она не существует

if not os.path.exists(UPLOAD\_FOLDER):

    os.makedirs(UPLOAD\_FOLDER)

# Проверка допустимых расширений файлов

def allowed\_file(filename):

    return '.' in filename and filename.rsplit('.', 1)[1].lower() in ALLOWED\_EXTENSIONS

db.init\_app(app)

login\_manager = LoginManager()

login\_manager.init\_app(app)

login\_manager.login\_view = 'login'

def allowed\_file(filename):

    return '.' in filename and filename.rsplit('.', 1)[1].lower() in ALLOWED\_EXTENSIONS

@login\_manager.user\_loader

def load\_user(user\_id):

    return User.query.get(int(user\_id))

@app.route('/create\_team', methods=['GET', 'POST'])

@login\_required

def create\_team():

    if request.method == 'POST':

        name = request.form['name']

        new\_team = Team(name=name)

        db.session.add(new\_team)

        db.session.commit()

        flash('Команда успешно создана!', 'success')

        return redirect(url\_for('select\_team'))

    return render\_template('create\_team.html')

# Маршрут для выбора команды

@app.route('/select\_team', methods=['GET', 'POST'])

@login\_required

def select\_team():

    if request.method == 'POST':

        team\_id = request.form['team\_id']

        current\_user.team\_id = team\_id

        db.session.commit()

        flash('Команда успешно выбрана!', 'success')

        return redirect(url\_for('home'))

    teams = Team.query.all()

    return render\_template('select\_team.html', teams=teams)

@app.route('/update\_task\_status/<int:task\_id>', methods=['POST'])

@login\_required

def update\_task\_status(task\_id):

    task = Task.query.get\_or\_404(task\_id)

    new\_status = request.json.get('status')

    task.status = new\_status

    db.session.commit()

    return jsonify({'message': 'Статус задачи обновлен'})

@app.route('/register', methods=['GET', 'POST'])

def register():

    if request.method == 'POST':

        username = request.form.get('username')

        email = request.form.get('email')

        password = request.form.get('password')

        team\_id = request.form.get('team\_id')

        if not username or not email or not password:

            flash("Все поля обязательны!", "danger")

            return redirect(url\_for('register'))

        existing\_user = User.query.filter\_by(email=email).first()

        if existing\_user:

            flash("Пользователь с таким email уже зарегистрирован!", "warning")

            return redirect(url\_for('register'))

        new\_user = User(username=username, email=email, team\_id=team\_id)

        new\_user.set\_password(password)

        db.session.add(new\_user)

        db.session.commit()

        flash("Регистрация успешна! Войдите в систему.", "success")

        return redirect(url\_for('login'))

    teams = Team.query.all()

    return render\_template('register.html', teams=teams)

@app.route('/login', methods=['GET', 'POST'])

def login():

    if request.method == 'POST':

        email = request.form['email']

        password = request.form['password']

        user = User.query.filter\_by(email=email).first()

        if user and user.check\_password(password):

            login\_user(user)

            return redirect(url\_for('dashboard'))

        else:

            flash("Ошибка входа! Проверьте данные.", "danger")

    return render\_template('login.html')

@app.route('/logout')

@login\_required

def logout():

    logout\_user()

    flash("Вы вышли из системы.", "info")

    return redirect(url\_for('login'))

@app.route('/dashboard')

@login\_required

def dashboard():

    return render\_template('dashboard.html', user=current\_user, team=current\_user.team)

@app.route('/users', methods=['GET', 'POST'])

@login\_required

def users():

    if request.method == 'POST':

        # Генерация уникального имени и email

        username = f"test\_user\_{''.join(random.choices(string.ascii\_lowercase + string.digits, k=5))}"

        email = f"{username}@example.com"

        # Проверка, что сгенерированные имя и email уникальны

        while User.query.filter\_by(username=username).first() or User.query.filter\_by(email=email).first():

            username = f"test\_user\_{''.join(random.choices(string.ascii\_lowercase + string.digits, k=5))}"

            email = f"{username}@example.com"

        # Создание нового пользователя

        new\_user = User(username=username, email=email)

        new\_user.set\_password("test123")  # Установка пароля

        db.session.add(new\_user)

        db.session.commit()

        flash(f"Тестовый пользователь {username} создан!", "success")

    users = User.query.all()

    return render\_template('users.html', users=users)

@app.route('/task/<int:task\_id>')

@login\_required

def task\_detail(task\_id):

    task = Task.query.get\_or\_404(task\_id)

    return render\_template('task\_detail.html', task=task)

@app.route('/reports', methods=['GET', 'POST'])

@login\_required

def reports():

    if request.method == 'POST':

        project\_id = request.form['project\_id']

        file = request.files['report\_file']

        if file and allowed\_file(file.filename):

            filename = secure\_filename(file.filename)

            filepath = os.path.join(app.config['UPLOAD\_FOLDER'], filename)

            file.save(filepath)

            new\_report = Report(filename=filename, filepath=filepath, project\_id=project\_id)

            db.session.add(new\_report)

            db.session.commit()

            flash('Отчет успешно загружен!', 'success')

        else:

            flash('Недопустимый формат файла!', 'danger')

    projects = Project.query.all()

    return render\_template('reports.html', projects=projects)

@app.route('/report/<int:report\_id>')

@login\_required

def view\_report(report\_id):

    report = Report.query.get\_or\_404(report\_id)

    return send\_from\_directory(app.config['UPLOAD\_FOLDER'], report.filename)

@app.route('/charts')

@login\_required

def charts():

    return render\_template('charts.html')

@app.route('/')

def home():

    if current\_user.is\_authenticated:

        projects = Project.query.all()

        return render\_template('index.html', projects=projects)

    else:

        return render\_template('index.html')

@app.route('/kanban')

def kanban():

    tasks = Task.query.options(

        db.joinedload(Task.assigned\_user),

        db.joinedload(Task.creator)

    ).all()

    return render\_template('kanban.html', tasks=tasks)

@app.route('/task\_creator', methods=['GET', 'POST'])

@login\_required

def task\_creator():

    if request.method == 'POST':

        title = request.form.get('title', '').strip()

        description = request.form.get('description', '').strip()

        priority = request.form.get('priority')

        status = request.form.get('status', 'To Do')

        project\_id = request.form.get('project\_id')

        assigned\_to = request.form.get('assigned\_to')

        created\_by = request.form.get('created\_by', current\_user.id)  # По умолчанию текущий пользователь

        if not title:

            flash("Название задачи не может быть пустым!", "danger")

            return redirect(url\_for('task\_creator'))

        if not priority:

            flash("Необходимо выбрать приоритет!", "danger")

            return redirect(url\_for('task\_creator'))

        if not project\_id:

            flash("Необходимо выбрать проект!", "danger")

            return redirect(url\_for('task\_creator'))

        new\_task = Task(

            title=title,

            description=description if description else None,

            priority=priority,

            status=status,

            project\_id=int(project\_id),

            assigned\_to=int(assigned\_to) if assigned\_to else None,

            created\_by=int(created\_by)  # Указываем постановщика

        )

        db.session.add(new\_task)

        db.session.commit()

        flash("Задача успешно создана!", "success")

        return redirect(url\_for('kanban'))

    return render\_template('task\_creator.html', projects=Project.query.all(), users=User.query.all())

@app.route('/edit\_task/<int:task\_id>', methods=['GET', 'POST'])

@login\_required

def edit\_task(task\_id):

    task = Task.query.get\_or\_404(task\_id)

    if request.method == 'POST':

        task.title = request.form.get('title', '').strip()

        task.description = request.form.get('description', None)

        task.priority = request.form.get('priority', 'Средний')

        task.status = request.form.get('status', 'To Do')

        task.project\_id = request.form.get('project\_id')

        task.assigned\_to = request.form.get('assigned\_to', None)

        if not task.title:

            flash("Название задачи не может быть пустым!", "danger")

            return redirect(url\_for('edit\_task', task\_id=task\_id))

        db.session.commit()

        flash("Задача успешно обновлена!", "success")

        return redirect(url\_for('kanban'))

    return render\_template('task\_creator.html', task=task, projects=Project.query.all(), users=User.query.all())

@app.route('/delete\_task/<int:task\_id>')

def delete\_task(task\_id):

    task = Task.query.get(task\_id)

    db.session.delete(task)

    db.session.commit()

    return redirect(url\_for('kanban'))

@app.route('/projects')

@login\_required

def projects():

    projects = Project.query.all()

    return render\_template('index.html', projects=projects)

@app.route('/project/<int:project\_id>')

@login\_required

def project\_detail(project\_id):

    project = Project.query.get\_or\_404(project\_id)

    return render\_template('project\_detail.html', project=project)

@app.route('/delete\_project/<int:project\_id>')

@login\_required

def delete\_project(project\_id):

    project = Project.query.get\_or\_404(project\_id)

    # Удаляем все задачи, связанные с проектом

    Task.query.filter\_by(project\_id=project.id).delete()

    # Удаляем все отчеты, связанные с проектом

    Report.query.filter\_by(project\_id=project.id).delete()

    # Удаляем сам проект

    db.session.delete(project)

    db.session.commit()

    flash("Проект и все связанные задачи и отчеты успешно удалены!", "success")

    return redirect(url\_for('home'))

@app.route('/create\_project', methods=['GET', 'POST'])

@login\_required

def create\_project():

    if request.method == 'POST':

        name = request.form['name']

        description = request.form.get('description', '').strip()  # Получаем описание

        if not name:

            flash("Название проекта не может быть пустым!", "danger")

            return redirect(url\_for('create\_project'))

        new\_project = Project(name=name, description=description if description else None)

        db.session.add(new\_project)

        db.session.commit()

        flash("Проект успешно создан!", "success")

        return redirect(url\_for('home'))

    return render\_template('create\_project.html')

if \_\_name\_\_ == '\_\_main\_\_':

    with app.app\_context():

        db.create\_all()

    app.run(debug=True)

Database.py

from flask\_sqlalchemy import SQLAlchemy

from flask\_login import UserMixin

from datetime import datetime

from werkzeug.security import generate\_password\_hash, check\_password\_hash

db = SQLAlchemy()

class Team(db.Model):

    id = db.Column(db.Integer, primary\_key=True)

    name = db.Column(db.String(100), unique=True, nullable=False)

    users = db.relationship('User', backref='team', lazy=True)

    projects = db.relationship('Project', back\_populates='team', lazy=True)  # Use back\_populates

class User(db.Model, UserMixin):

    id = db.Column(db.Integer, primary\_key=True)

    username = db.Column(db.String(100), unique=True, nullable=False)

    email = db.Column(db.String(150), unique=True, nullable=False)

    password\_hash = db.Column(db.String(200), nullable=False)

    team\_id = db.Column(db.Integer, db.ForeignKey('team.id'), nullable=True)

    def set\_password(self, password):

        self.password\_hash = generate\_password\_hash(password)

    def check\_password(self, password):

        return check\_password\_hash(self.password\_hash, password)

class Project(db.Model):

    id = db.Column(db.Integer, primary\_key=True)

    name = db.Column(db.String(100), nullable=False)

    description = db.Column(db.Text, nullable=True)

    tasks = db.relationship('Task', back\_populates='project', lazy=True)  # Use back\_populates

    reports = db.relationship('Report', back\_populates='project', lazy=True)  # Use back\_populates

    team\_id = db.Column(db.Integer, db.ForeignKey('team.id'), nullable=True)  # Corrected indentation

    # Отношение к команде

    team = db.relationship('Team', back\_populates='projects')  # Matches back\_populates in Team

class Task(db.Model):

    id = db.Column(db.Integer, primary\_key=True)

    title = db.Column(db.String(100), nullable=False)

    description = db.Column(db.Text, nullable=True)

    priority = db.Column(db.String(20), nullable=False, default='Средний')

    status = db.Column(db.String(50), nullable=False, default='To Do')

    created\_at = db.Column(db.DateTime, default=datetime.utcnow)

    project\_id = db.Column(db.Integer, db.ForeignKey('project.id'), nullable=False)

    assigned\_to = db.Column(db.Integer, db.ForeignKey('user.id'), nullable=True)

    created\_by = db.Column(db.Integer, db.ForeignKey('user.id'), nullable=False)

    project = db.relationship('Project', back\_populates='tasks')  # Matches back\_populates in Project

    assigned\_user = db.relationship('User', foreign\_keys=[assigned\_to], backref='assigned\_tasks')

    creator = db.relationship('User', foreign\_keys=[created\_by], backref='created\_tasks')

class Report(db.Model):

    id = db.Column(db.Integer, primary\_key=True)

    filename = db.Column(db.String(200), nullable=False)  # Имя файла

    filepath = db.Column(db.String(200), nullable=False)  # Путь к файлу

    project\_id = db.Column(db.Integer, db.ForeignKey('project.id'), nullable=False)

    created\_at = db.Column(db.DateTime, default=datetime.utcnow)

    project = db.relationship('Project', back\_populates='reports')  # Matches back\_populates in Project

Create\_team.html  
{% extends 'base.html' %}

{% block title %}Создать команду{% endblock %}

{% block content %}

<div class="container">

    <h1>Создать новую команду</h1>

    <form method="POST" action="{{ url\_for('create\_team') }}">

        <label for="name">Название команды:</label>

        <input type="text" id="name" name="name" required>

        <button type="submit" class="btn">Создать команду</button>

    </form>

</div>

{% endblock %}

Select\_team.html  
{% extends 'base.html' %}

{% block title %}Выбрать команду{% endblock %}

{% block content %}

<div class="container">

    <h1>Выберите команду</h1>

    <form method="POST" action="{{ url\_for('select\_team') }}">

        <label for="team\_id">Команда:</label>

        <select id="team\_id" name="team\_id" required>

            <option value="">-- Выберите команду --</option>

            {% for team in teams %}

            <option value="{{ team.id }}">{{ team.name }}</option>

            {% endfor %}

        </select>

        <button type="submit" class="btn">Выбрать команду</button>

    </form>

    <a href="{{ url\_for('create\_team') }}" class="btn">Создать новую команду</a>

</div>

{% endblock %}

Users.html

{% extends 'base.html' %}

{% block content %}

<div class="container">

    <h1>Список пользователей</h1>

    <table class="user-table">

        <tr>

            <th>Имя пользователя</th>

            <th>Email</th>

        </tr>

        {% for user in users %}

        <tr>

            <td>{{ user.username }}</td>

            <td>{{ user.email }}</td>

        </tr>

        {% endfor %}

    </table>

    <h2>Добавить тестового пользователя</h2>

    <form method="POST">

        <button type="submit" class="btn">Создать тестового пользователя</button>

    </form>

</div>

{% endblock %}

Login.html  
{% extends 'base.html' %}

{% block content %}

<div class="auth-container">

    <div class="auth-box">

        <h2>Вход</h2>

        <form method="POST">

            <input type="email" name="email" placeholder="Email" required>

            <input type="password" name="password" placeholder="Пароль" required>

            <button type="submit">Войти</button>

        </form>

        <p>Нет аккаунта? <a href="{{ url\_for('register') }}">Регистрация</a></p>

    </div>

</div>

{% endblock %}

Register.html  
{% extends 'base.html' %}

{% block content %}

<div class="auth-container">

    <div class="auth-box">

        <h2>Вход</h2>

        <form method="POST">

            <input type="email" name="email" placeholder="Email" required>

            <input type="password" name="password" placeholder="Пароль" required>

            <button type="submit">Войти</button>

        </form>

        <p>Нет аккаунта? <a href="{{ url\_for('register') }}">Регистрация</a></p>

    </div>

</div>

{% endblock %}

Dashboard.html  
{% extends 'base.html' %}

{% block title %}Личный кабинет{% endblock %}

{% block content %}

<div class="container">

    <h1>Привет, {{ user.username }}!</h1>

    <p>Ваша команда: {{ team.name if team else "Не состоит в команде" }}</p>

    <h2>Навигация</h2>

    <ul>

        <li><a href="{{ url\_for('projects') }}">📂 Проекты</a></li>

        <li><a href="{{ url\_for('kanban') }}">📌 Kanban-доска</a></li>

        <li><a href="{{ url\_for('reports') }}">📄 Отчёты</a></li>

        <li><a href="{{ url\_for('users') }}">👥 Список пользователей</a></li>

        <li><a href="{{ url\_for('logout') }}">🚪 Выйти</a></li>

    </ul>

</div>

{% endblock %}

Base.html

<!DOCTYPE html>

<html lang="ru">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>{% block title %}Проектное управление{% endblock %}</title>

    <link rel="stylesheet" href="{{ url\_for('static', filename='styles.css') }}">

    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>

</head>

<body>

    <nav class="navbar">

        <a href="{{ url\_for('projects') }}">Проекты</a>

        <a href="{{ url\_for('kanban') }}">Kanban-доска</a>

        <a href="{{ url\_for('charts') }}">Диаграммы</a>

        <a href="{{ url\_for('reports') }}">Отчёты</a>

        {% if current\_user.is\_authenticated %}

            <a href="{{ url\_for('select\_team') }}" class="btn">Выбрать команду</a>

            <a href="{{ url\_for('dashboard') }}" class="btn">Личный кабинет</a>

            <a href="{{ url\_for('logout') }}" class="btn btn-danger">Выйти</a>

        {% else %}

            <a href="{{ url\_for('login') }}" class="btn">Войти</a>

            <a href="{{ url\_for('register') }}" class="btn">Зарегистрироваться</a>

        {% endif %}

    </nav>

    <div class="content">

        {% block content %}{% endblock %}

    </div>

</body>

</html>

Create\_project.html  
{% extends 'base.html' %}

{% block title %}Создать проект{% endblock %}

{% block content %}

<div class="container">

    <h1>Создать новый проект</h1>

    <form method="POST" class="project-form">

        <div class="form-group">

            <label for="name">Название проекта:</label>

            <input type="text" id="name" name="name" required>

        </div>

        <div class="form-group">

            <label for="description">Описание проекта:</label>

            <textarea id="description" name="description"></textarea>

        </div>

        <button type="submit" class="btn btn-primary">Создать проект</button>

    </form>

</div>

{% endblock %}

Index.html  
{% extends 'base.html' %}

{% block title %}Главная{% endblock %}

{% block content %}

<div class="container">

    {% if not current\_user.is\_authenticated %}

        <!-- Приветствие для неавторизованных пользователей -->

        <div class="auth-container">

            <div class="auth-box">

                <h2>Добро пожаловать в систему управления проектами!</h2>

                <p>Пожалуйста, <a href="{{ url\_for('login') }}">войдите</a> или <a href="{{ url\_for('register') }}">зарегистрируйтесь</a>, чтобы начать работу.</p>

            </div>

        </div>

    {% else %}

        <!-- Контент для авторизованных пользователей -->

        <h1>Привет, {{ current\_user.username }}!</h1>

        <p>Ваша команда: {{ current\_user.team.name if current\_user.team else "Не состоит в команде" }}</p>

        <h2>Проекты</h2>

        <a href="{{ url\_for('create\_project') }}" class="btn">+ Создать проект</a>

        <!-- Список проектов -->

        <ul class="project-list">

            {% for project in projects %}

            <li>

                <h3><a href="{{ url\_for('project\_detail', project\_id=project.id) }}">{{ project.name }}</a></h3>

                <p>{{ project.description | default("Описание отсутствует", true) }}</p>

                <a href="{{ url\_for('delete\_project', project\_id=project.id) }}" class="delete-btn">Удалить</a>

            </li>

            {% endfor %}

        </ul>

    {% endif %}

</div>

{% endblock %}

Charts.html  
{% extends 'base.html' %}

{% block title %}Диаграммы{% endblock %}

{% block content %}

<div class="container">

    <h1>Диаграммы</h1>

    <button class="btn">📊 Создать диаграмму Ганта</button>

    <button class="btn">📈 Создать график</button>

</div>

{% endblock %}

Kanban.html  
{% extends 'base.html' %}

{% block title %}Kanban-доска{% endblock %}

{% block content %}

<div class="container">

    <h1>Kanban-доска</h1>

    <a href="{{ url\_for('task\_creator') }}" class="btn">+ Создать задачу</a>  <!-- Кнопка "Создать задачу" -->

    <div class="kanban-board">

        <!-- Колонка "To Do" -->

        <div class="kanban-column" id="todo-column">

            <h2>To Do</h2>

            {% for task in tasks if task.status == 'To Do' %}

            <div class="task-card {% if task.priority == 'Высокий' %}high-priority{% elif task.priority == 'Средний' %}medium-priority{% elif task.priority == 'Низкий' %}low-priority{% endif %}" data-task-id="{{ task.id }}">

                <h3><a href="{{ url\_for('task\_detail', task\_id=task.id) }}">{{ task.title }}</a></h3>

                <p>Проект: <a href="{{ url\_for('project\_detail', project\_id=task.project\_id) }}">{{ task.project.name }}</a></p>

                <p><strong>Ответственный:</strong> {{ task.assigned\_user.username if task.assigned\_user else "Не назначен" }}</p>

                <p><strong>Постановщик:</strong> {{ task.creator.username }}</p>

            </div>

            {% endfor %}

        </div>

        <!-- Колонка "In Progress" -->

        <div class="kanban-column" id="in-progress-column">

            <h2>In Progress</h2>

            {% for task in tasks if task.status == 'In Progress' %}

            <div class="task-card {% if task.priority == 'Высокий' %}high-priority{% elif task.priority == 'Средний' %}medium-priority{% elif task.priority == 'Низкий' %}low-priority{% endif %}" data-task-id="{{ task.id }}">

                <h3><a href="{{ url\_for('task\_detail', task\_id=task.id) }}">{{ task.title }}</a></h3>

                <p>Проект: <a href="{{ url\_for('project\_detail', project\_id=task.project\_id) }}">{{ task.project.name }}</a></p>

                <p><strong>Ответственный:</strong> {{ task.assigned\_user.username if task.assigned\_user else "Не назначен" }}</p>

                <p><strong>Постановщик:</strong> {{ task.creator.username }}</p>

            </div>

            {% endfor %}

        </div>

        <!-- Колонка "Done" -->

        <div class="kanban-column" id="done-column">

            <h2>Done</h2>

            {% for task in tasks if task.status == 'Done' %}

            <div class="task-card {% if task.priority == 'Высокий' %}high-priority{% elif task.priority == 'Средний' %}medium-priority{% elif task.priority == 'Низкий' %}low-priority{% endif %}" data-task-id="{{ task.id }}">

                <h3><a href="{{ url\_for('task\_detail', task\_id=task.id) }}">{{ task.title }}</a></h3>

                <p>Проект: <a href="{{ url\_for('project\_detail', project\_id=task.project\_id) }}">{{ task.project.name }}</a></p>

                <p><strong>Ответственный:</strong> {{ task.assigned\_user.username if task.assigned\_user else "Не назначен" }}</p>

                <p><strong>Постановщик:</strong> {{ task.creator.username }}</p>

            </div>

            {% endfor %}

        </div>

    </div>

</div>

{% endblock %}

Task\_detail.html  
{% extends 'base.html' %}

{% block title %}{{ task.title }}{% endblock %}

{% block content %}

<div class="container">

    <h1>{{ task.title }}</h1>

    <p><strong>Описание:</strong> {{ task.description or "Не указано" }}</p>

    <p><strong>Приоритет:</strong> <span class="{{ task.priority }}">{{ task.priority }}</span></p>

    <p><strong>Ответственный:</strong> {{ task.assigned\_to.username if task.assigned\_to else "Не назначен" }}</p>

    <p><strong>Постановщик:</strong> {{ task.created\_by.username }}</p>

    <p><strong>Проект:</strong> <a href="{{ url\_for('project\_detail', project\_id=task.project\_id) }}">{{ task.project.name }}</a></p>

    <a href="{{ url\_for('edit\_task', task\_id=task.id) }}" class="btn">✏ Редактировать</a>

    <a href="{{ url\_for('delete\_task', task\_id=task.id) }}" class="btn btn-danger">❌ Удалить</a>

</div>

{% endblock %}

Project\_detail.html  
{% extends 'base.html' %}

{% block title %}{{ project.name }}{% endblock %}

{% block content %}

<div class="container">

    <h1>{{ project.name }}</h1>

    <p>{{ project.description | default("Описание отсутствует", true) }}</p>

    <h2>Задачи проекта:</h2>

    <ul>

        {% for task in project.tasks %}

        <li>

            <a href="{{ url\_for('task\_detail', task\_id=task.id) }}">{{ task.title }}</a> ({{ task.status }})

        </li>

        {% else %}

        <li>Задачи пока отсутствуют.</li>

        {% endfor %}

    </ul>

    <h2>Отчеты проекта:</h2>

    <ul>

        {% for report in project.reports %}

        <li>

            <a href="{{ url\_for('view\_report', report\_id=report.id) }}">{{ report.filename }}</a>

        </li>

        {% else %}

        <li>Отчеты пока отсутствуют.</li>

        {% endfor %}

    </ul>

    <a href="{{ url\_for('reports') }}" class="btn">Добавить отчет</a>

</div>

{% endblock %}

Edit\_task.html  
{% extends 'base.html' %}

{% block title %}Редактировать задачу{% endblock %}

{% block content %}

<div class="container">

    <h1>Редактировать задачу</h1>

    <form action="{{ url\_for('edit\_task', task\_id=task.id) }}" method="POST">

        <label for="title">Название</label>

        <input type="text" name="title" value="{{ task.title }}" required>

        <label for="status">Статус</label>

        <select name="status">

            <option value="To Do" {% if task.status == 'To Do' %}selected{% endif %}>To Do</option>

            <option value="In Progress" {% if task.status == 'In Progress' %}selected{% endif %}>In Progress</option>

            <option value="Done" {% if task.status == 'Done' %}selected{% endif %}>Done</option>

        </select>

        <button type="submit">Сохранить изменения</button>

    </form>

</div>

{% endblock %}

Projects.html

{% extends 'base.html' %}

{% block title %}Проекты{% endblock %}

{% block content %}

<div class="container">

    <h1>Список проектов</h1>

    <form action="{{ url\_for('projects') }}" method="POST" class="form">

        <input type="text" name="name" placeholder="Название проекта" required>

        <textarea name="description" placeholder="Описание проекта"></textarea>

        <button type="submit">Добавить проект</button>

    </form>

    <ul class="project-list">

        {% for project in projects %}

        <li>

            <h3><a href="{{ url\_for('project\_detail', project\_id=project.id) }}">{{ project.name }}</h3>

            <p>{{ project.description }}</p>

            <a href="{{ url\_for('delete\_project', project\_id=project.id) }}" class="delete-btn">Удалить</a>

        </li>

        {% endfor %}

    </ul>

</div>

{% endblock %}

Reports.html  
{% extends 'base.html' %}

{% block title %}Создание отчета{% endblock %}

{% block content %}

<div class="container">

    <h1>Создание отчета</h1>

    <form method="POST" enctype="multipart/form-data">

        <label for="project\_id">Выберите проект:</label>

        <select id="project\_id" name="project\_id" required>

            <option value="">-- Выберите проект --</option>

            {% for project in projects %}

            <option value="{{ project.id }}">{{ project.name }}</option>

            {% endfor %}

        </select>

        <label for="report\_file">Прикрепите отчет:</label>

        <input type="file" id="report\_file" name="report\_file" required>

        <button type="submit" class="btn">Загрузить отчет</button>

    </form>

</div>

{% endblock %}

Task\_creator.html  
{% extends 'base.html' %}

{% block title %}Создать задачу{% endblock %}

{% block content %}

<div class="container">

    <h1>Создать новую задачу</h1>

    <form method="POST" class="task-form">

        <div class="form-group">

            <label for="title">Название задачи:</label>

            <input type="text" id="title" name="title" required>

        </div>

        <div class="form-group">

            <label for="description">Описание:</label>

            <textarea id="description" name="description"></textarea>

        </div>

        <div class="form-group">

            <label for="priority">Приоритет:</label>

            <select id="priority" name="priority" required>

                <option value="">-- Выберите приоритет --</option>

                <option value="Высокий">🔴 Высокий</option>

                <option value="Средний">🟡 Средний</option>

                <option value="Низкий">🟢 Низкий</option>

            </select>

        </div>

        <div class="form-group">

            <label for="status">Статус:</label>

            <select id="status" name="status" required>

                <option value="To Do">📌 To Do</option>

                <option value="In Progress">⚙ In Progress</option>

                <option value="Done">✅ Done</option>

            </select>

        </div>

        <div class="form-group">

            <label for="project\_id">Проект:</label>

            <select id="project\_id" name="project\_id" required>

                <option value="">-- Выберите проект --</option>

                {% for project in projects %}

                <option value="{{ project.id }}">{{ project.name }}</option>

                {% endfor %}

            </select>

        </div>

        <div class="form-group">

            <label for="assigned\_to">Ответственный:</label>

            <select id="assigned\_to" name="assigned\_to">

                <option value="">-- Выберите пользователя --</option>

                {% for user in users %}

                <option value="{{ user.id }}">{{ user.username }}</option>

                {% endfor %}

            </select>

        </div>

        <div class="form-group">

            <label for="created\_by">Постановщик:</label>

            <select id="created\_by" name="created\_by" required>

                <option value="{{ current\_user.id }}">{{ current\_user.username }} (Я)</option>

                {% for user in users %}

                    {% if user.id != current\_user.id %}

                        <option value="{{ user.id }}">{{ user.username }}</option>

                    {% endif %}

                {% endfor %}

            </select>

        </div>

        <button type="submit" class="btn btn-primary">Создать задачу</button>

    </form>

</div>

<script>

document.getElementById('task-form').addEventListener('submit', function(event) {

    let title = document.getElementById('title').value.trim();

    let priority = document.getElementById('priority').value;

    let project = document.getElementById('project\_id').value;

    if (title === '') {

        alert('Название задачи не может быть пустым!');

        event.preventDefault();

    }

    if (priority === '') {

        alert('Выберите приоритет!');

        event.preventDefault();

    }

    if (project === '') {

        alert('Выберите проект!');

        event.preventDefault();

    }

});

</script>

{% endblock %}

# приложение б

**Внешний вид графического материала**

Внешний вид графического материала, выполненного на отдельных листах представлен на рисунках Б.1 – Б.3.

Перечень графического материала:

лист 1: Диаграмма вариантов использования (рисунок Б.1),

лист 2: Диаграмма классов (рисунок Б.2),

лист 3: Диаграмма компонентов (рисунок Б.3).

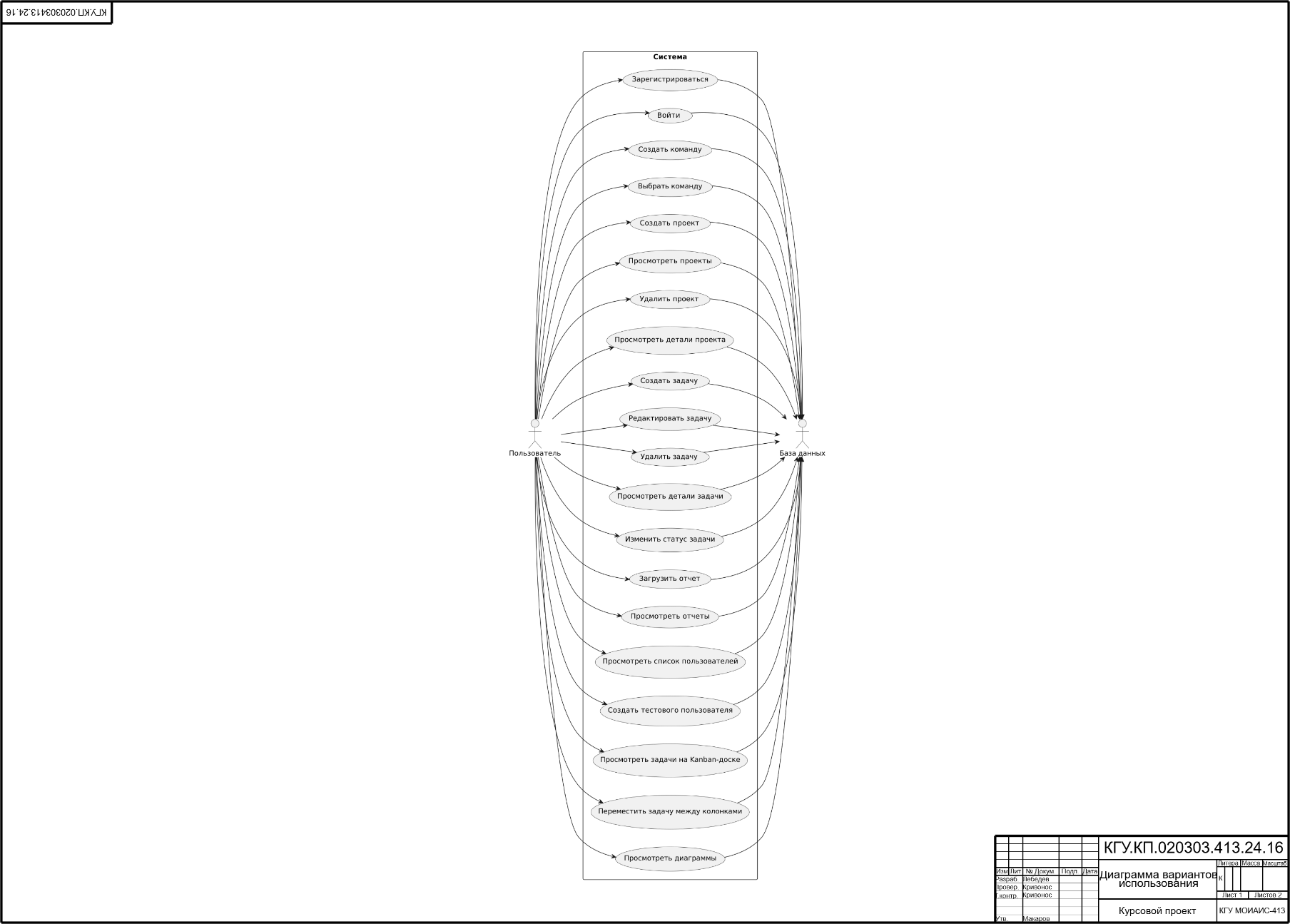


Рисунок Б.1 – Диаграмма вариантов использования

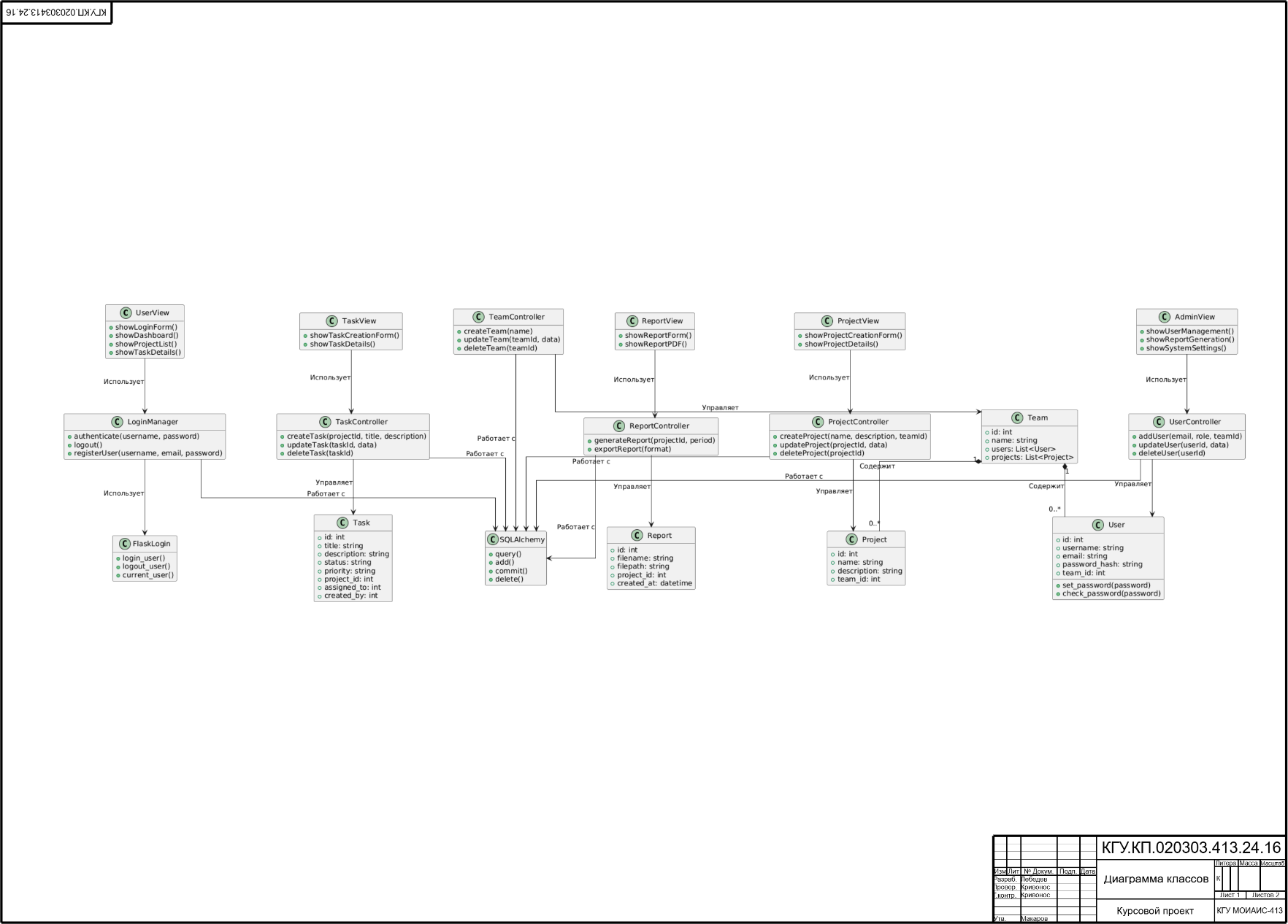


Рисунок Б.2 – Диаграмма классов

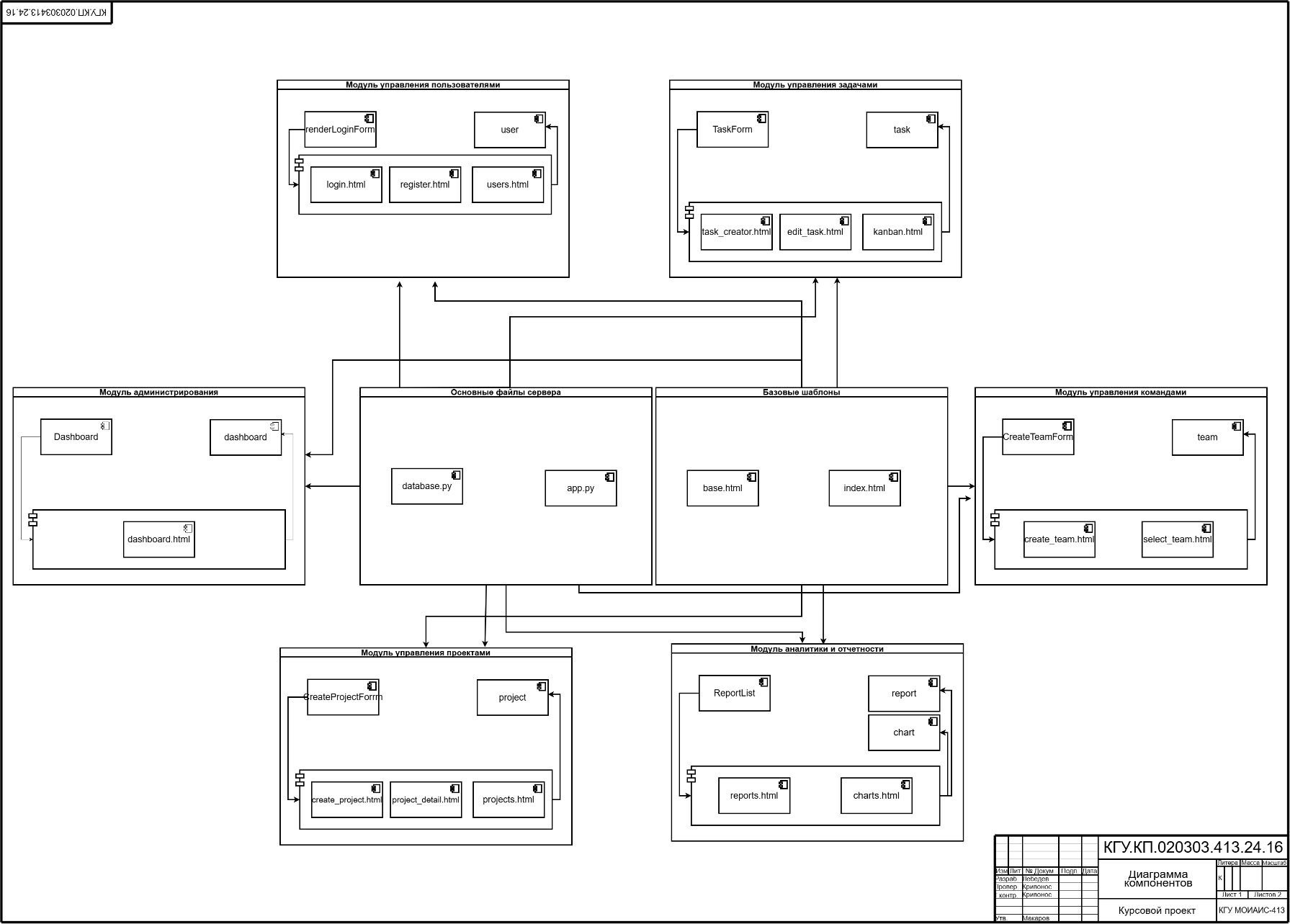


Рисунок Б.3 – Диаграмма компонентов