МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

ДОПУСТИТЬ К ЗАЩИТЕ

Заведующий кафедрой, д.ф.-м.н.,
профессор

_____ Л.Б. Соколинский

«___»_____ 2024 г.

Разработка веб-приложения для управления проектами на основе методологии Scrumban

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 09.03.04.2024.308-578.ВКР

Научный руководитель,
ст. преподаватель кафедры СП
_____ Л.Н. Петрова

Автор работы,
студент группы КЭ-404
_____ А.В. Паклин

Ученый секретарь
(нормоконтролер)
_____ И.Д. Володченко
«___»_____ 2024 г.

Челябинск, 2024 г.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

УТВЕРЖДАЮ
Зав. кафедрой СП
_____ Л.Б. Соколинский
29.01.2024 г.

ЗАДАНИЕ
на выполнение выпускной квалификационной работы бакалавра
студенту группы КЭ-404
Паклину Александру Вячеславовичу,
обучающемуся по направлению
09.03.04 «Программная инженерия»

1. **Тема работы** (утверждена приказом ректора от 22.04.2024 г. № 764-13/12)
Разработка веб-приложения для управления проектами на основе методологии Scrumban.
2. **Срок сдачи студентом законченной работы:** 03.06.2024 г.
3. **Исходные данные к работе**

3.1. Django documentation. [Электронный ресурс] URL: https://docs.djangoproject.com/en/5.0/ (дата обращения: 01.02.2024 г.).
3.2. Чан У., Биссекс П., Форсье Д. Django. Разработка веб-приложений на Python. // СПб.: Символ, 2020. – 456 с.
3.3. Изучение веб-разработки. [Электронный ресурс] URL: https://developer.mozilla.org/ru/docs/Learn (дата обращения: 01.02.2024 г.).
3.4. Веб-приложение. [Электронный ресурс] URL: https://aws.amazon.com/ru/what–is/web–application/ (дата обращения: 01.02.2024 г.).

4. Перечень подлежащих разработке вопросов

4.1. Провести анализ предметной области.

4.2. Определить требования и спроектировать веб-приложение.

4.3. Реализовать веб-приложение.

4.4. Протестировать веб-приложение.

5. Дата выдачи задания: 29.01.2024 г.

Научный руководитель,
ст. преподаватель кафедры СП Л.Н. Петрова

Задание принял к исполнению А.В. Паклин

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность

В условиях современной динамично растущей деловой среды управление проектами приобретает все большую актуальность. В наши дни сложно представить крупную компанию, которая бы не имела сервиса для сотрудников по работе с задачами. Способность эффективно организовывать работу команды становится ключевым фактором успеха для многих организаций. В достижении максимально эффективного и качественного рабочего процесса способны помочь различные методологии. Одной из наиболее популярных и действенных методологий в этой области является Scrumban. Данная методология представляет собой гибридный подход, сочетающий принципы Scrum и Kanban.

Помимо растущего спроса на приложения для управления проектами, в наши дни, с развитием интернета и проникновением его в повседневную жизнь, веб-приложения становятся неотъемлемой ее частью. Они обеспечивают доступ к информации независимо от времени и местоположения. Веб-приложения обладают высокой степенью доступности и совместимости с различными платформами и устройствами, что делает их удобными и предпочтительными для пользователей. Помимо пользователей, веб-приложение облегчает задачу и разработчикам, где приложение – это единая версия, которая автоматически становится доступной для всех пользователей, что значительно упрощает процесс поддержки приложений.

Исходя из всего вышеперечисленного, можно сделать вывод, что веб-приложение для управления проектами на основе методологии Scrumban – это отличный инструмент, который может помочь организациям повысить эффективность, сократить время выполнения задач. Помимо этого, использование гибкой методологии Scrumban позволит всегда иметь возможность изменить направление развития проекта, а также адаптироваться к изменениям внешней среды, быстро реагировать на запросы клиентов и требования рынка. Данная методология, в отличии от Scrum или других методологий,

проста во внедрении, так как является простой в освоении и не требует больших временных ресурсов для обучения пользователей.

Постановка задачи

Целью выпускной квалификационной работы является разработка веб-приложения для управления проектами на основе методологии Scrumban. Для достижения поставленной цели необходимо решить следующие задачи:

1) провести анализ предметной области;
2) определить требования и спроектировать веб-приложение;
3) реализовать веб-приложение;
4) протестировать веб-приложение.

Структура и содержание работы

Работа состоит из введения, четырех глав, заключения и списка литературы. Объем работы составляет 43 страницы, объем списка литературы – 22 источника.

В первой главе описывается анализ предметной области, а также обзор аналогичных веб-приложений, анализ существующие решения для реализации проекта, а также делаются выводы о том, каким должно быть веб-приложение и с помощью каких инструментов будет вестись разработка.

Вторая глава посвящена проектированию веб-приложения, описанию его функциональных, нефункциональных требований и архитектуры. Помимо этого, в данной главе приведены диаграммы вариантов использования, а также разработана ER-диаграмма базы данных.

В третьей главе описывается создание проекта, реализация базы данных и веб-приложения.

Четвертая глава посвящена проведению тестирования, разработанного веб-приложения.

В приложении содержится описание спецификаций вариантов использования системы управления проектами.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

В разделе 1.1 приводится обзор методологии Scrumban для управления проектами, а также описываются возможности, которыми должно обладать веб-приложение. В разделе 1.2 приводится обзор на существующие аналоги. В разделе 1.3 приводится анализ существующих решений для реализации веб-приложения.

1.1. Предметная область проекта

Разрабатываемое в данной выпускной квалификационной работе веб-приложение [1] предназначено для управления проектами [2] на основе методологии Scrumban.

Методология Scrumban [3] объединяет лучшие практики из таких подходов, как Scrum и Kanban. Scrum разбивает проект на более мелкие и управляемые задачи, а также вводит идею спринтов (итераций), которые позволяют выделить время и сосредоточиться на текущих задачах. В то время, как Kanban обеспечивает прозрачность и контроль над текущим состоянием проекта с помощью доски, разделенной на столбцы. Scrum эффективен при работе над конкретными проблемами и имеет фиксированную длительность спринта, в то время как Kanban позволяет ограничить незавершенную работу, предотвращая перегрузку команды.

В методологии Scrumban существует четыре ключевых аспекта, описанных ниже.

1. Триггер планирования. Команда планирует предстоящий спринт в Scrumban, основываясь на прошлых результатах и оценках. Триггер планирования помогает определить, когда начинать планирование следующей итерации, показывая, сколько задач должно остаться в невыполненной работе после совещания по планированию.

2. Kanban-доска. Команды используют доску Kanban для отслеживания выполнения задач. На доске отображается вся запланированная, выпол-

няемая и завершенная работа. Kanban-доски могут различаться в зависимости от команды, но обычно включают невыполненную работу, процесс и столбец «Выполнено».

3. Лимит незавершенного производства. Команды устанавливают ограничение на количество задач, над которыми они могут работать одновременно. Этот предел, называемый лимитом незавершенного производства, позволяет точнее оценивать сроки выполнения задач и ускоряет их выполнение. Обычно этот лимит зависит от размера команды. Например, если в команде шесть человек, то максимальное количество незавершенных задач равно шести, чтобы каждый член мог сосредоточиться на одной задаче за раз.

4. Сегменты планирования. Команды используют метод долгосрочного планирования, называемый сегментами планирования. Карта команды имеет блок с задачами, которые планируются. Когда команда решает приступить к выполнению планов, задачи добавляются в бэклог и реализуются в следующей итерации.

Все описанные выше особенности методологии Scrumban должны быть учтены в разрабатываемом веб-приложении. Основная задача веб-приложения сделать взаимодействие проектного менеджера с исполнителями эффективным и удобным. Для этого веб-приложение должно предоставлять администраторам и проектным менеджерам возможность:

– создавать проекты с указанием: названия, дат, информации о клиенте, а также дополнительной информации о проекте;

– просматривать проекты;

– изменять информацию о проектах, а также их статусы;

– создавать задачи с указанием: названия, дат, информации о задаче, а также закреплять их за определенным исполнителем;

– просматривать задачи;

– изменять информацию о задачах и их статусы;

– оставлять комментарии к задачам;
– добавлять и удалять тэги;
– присваивать пользователям теги в соответствии с их должностью;
– просматривать список пользователей;
– добавлять и удалять пользователей;
– просматривать статистику по проектам и задачам.

Веб-приложение должно иметь возможность осуществлять обмен данными между базой данных [4] и приложением, чтобы данные всегда были актуальны.

1.2. Анализ аналогичных проектов

В ходе проведенного обзора было выявлено небольшое количество аналогов. Многие аналоги являются иностранными и не имеют поддержку русского языка. Для нас это является преимуществом ввиду малой конкуренции на российском рынке. Ниже представлен подробный анализ каждого из конкурентов.

Первым из ближайших конкурентов является Trello [5]. Trello – облачная программа для управления проектами небольших групп, разработанная Fog Creek Software, которая использует парадигму для управления проектами, известную как Kanban. Trello является самым распространенным и популярным инструментом для ведения личных и командных проектов. Достоинства и недостатки данного сервиса приведены в таблице 1.

Таблица 1 – Достоинства и недостатки Trello

Достоинства	Недостатки
1. Бесплатной версии достаточно для большинства проектов. 2. Быстрый и мощный поиск. 3. Метки для создания категорий. 4. Для продвинутого пользования есть улучшения.	1. Акцент на текущей работе, а не на датах, приоритетах и дедлайнах. 2. Прекращение деятельности на территории РФ.

Скриншот рабочего стола данного сервиса представлен на рисунке 1.

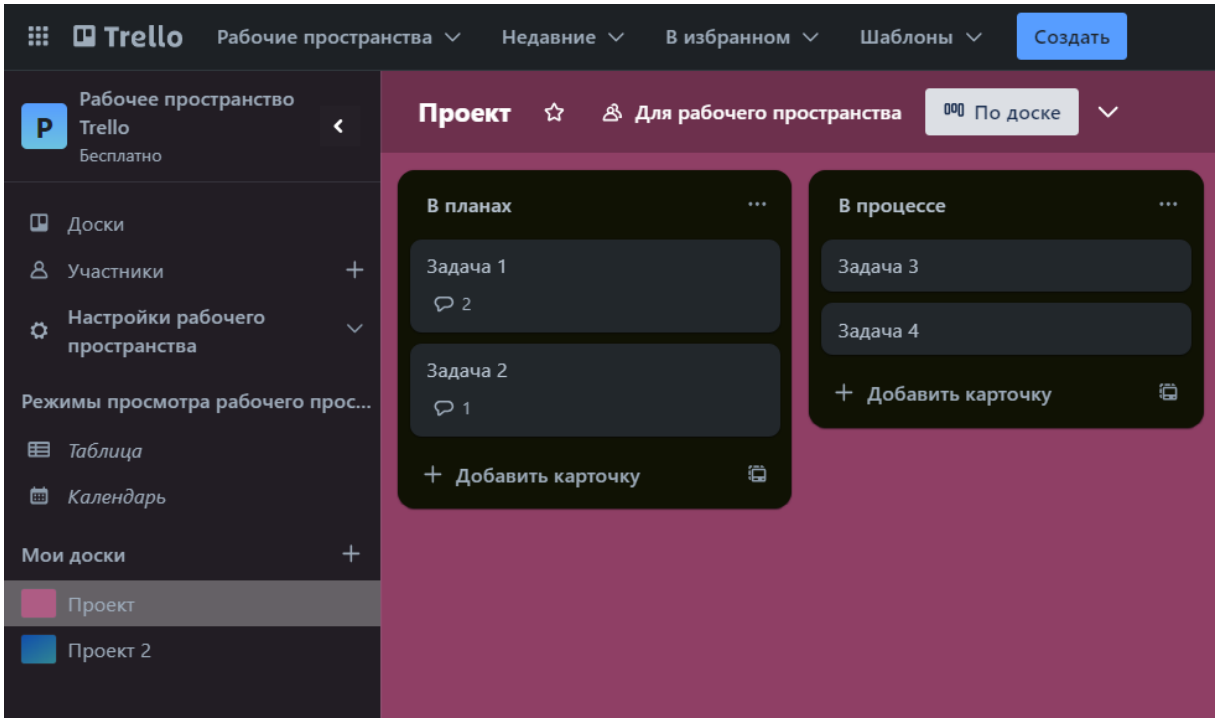

Рисунок 1 – Скриншот из сервиса Trello

Вторым из конкурентов является Bitrix24 [6]. Bitrix24 – это CRM-система на основе облачного сервиса. Разработан специально для того, чтобы максимально оптимизировать работу в компании. И не просто оптимизировать, а связать воедино различные процессы в разных отделах. В данной системе возможность управлять проектами и задачами представлена как одна из других возможностей. Достоинства и недостатки данного сервиса приведены в таблице 2.

Таблица 2 – Достоинства и недостатки Bitrix24

Достоинства	**Недостатки**
1. Есть приложение для iOS и Android. 2. Интерфейс и служба поддержки на русском языке. 3. Роботы для постановки задач по заданному заранее сценарию. 4. Возможность оценивать эффективность работы сотрудников.	1. Не интуитивный интерфейс. 2. Требует время на изучение, обучение и внедрение. 3. Огромное количество лишних возможностей.

Скриншот рабочего стола данного сервиса представлен на рисунке 2.

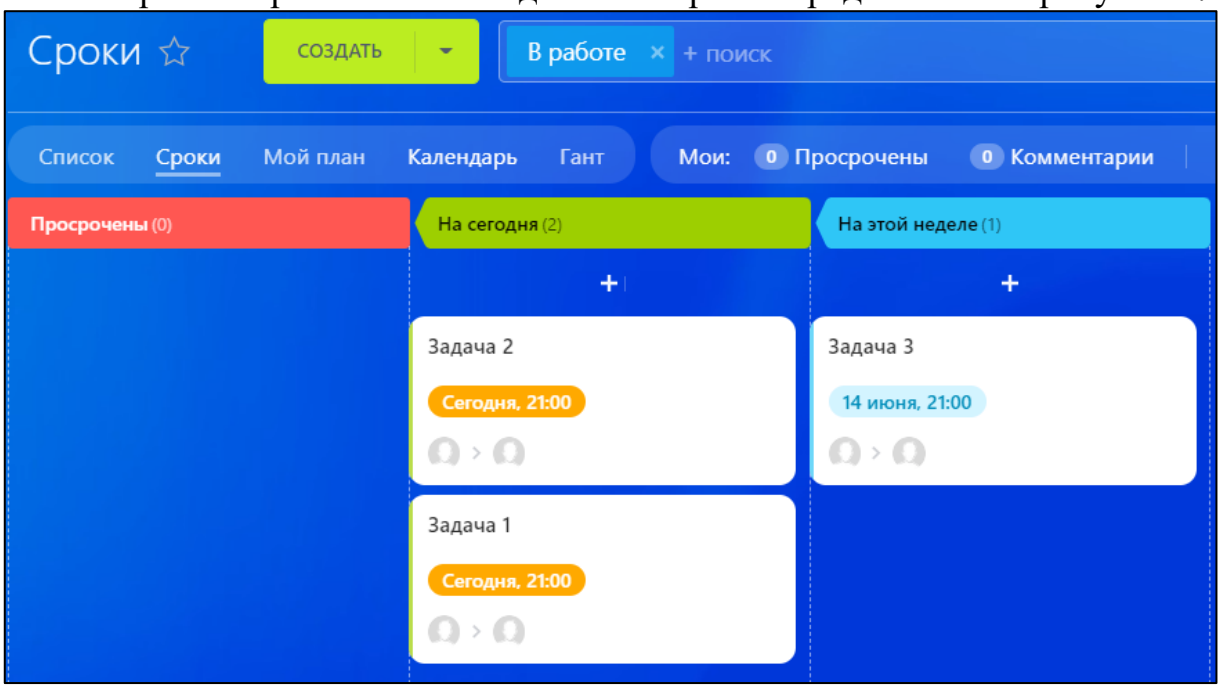

Рисунок 2 – Скриншот из сервиса Bitrix24

Скриншот рабочего стола с проектами в данном сервисе представлен на рисунке 3.

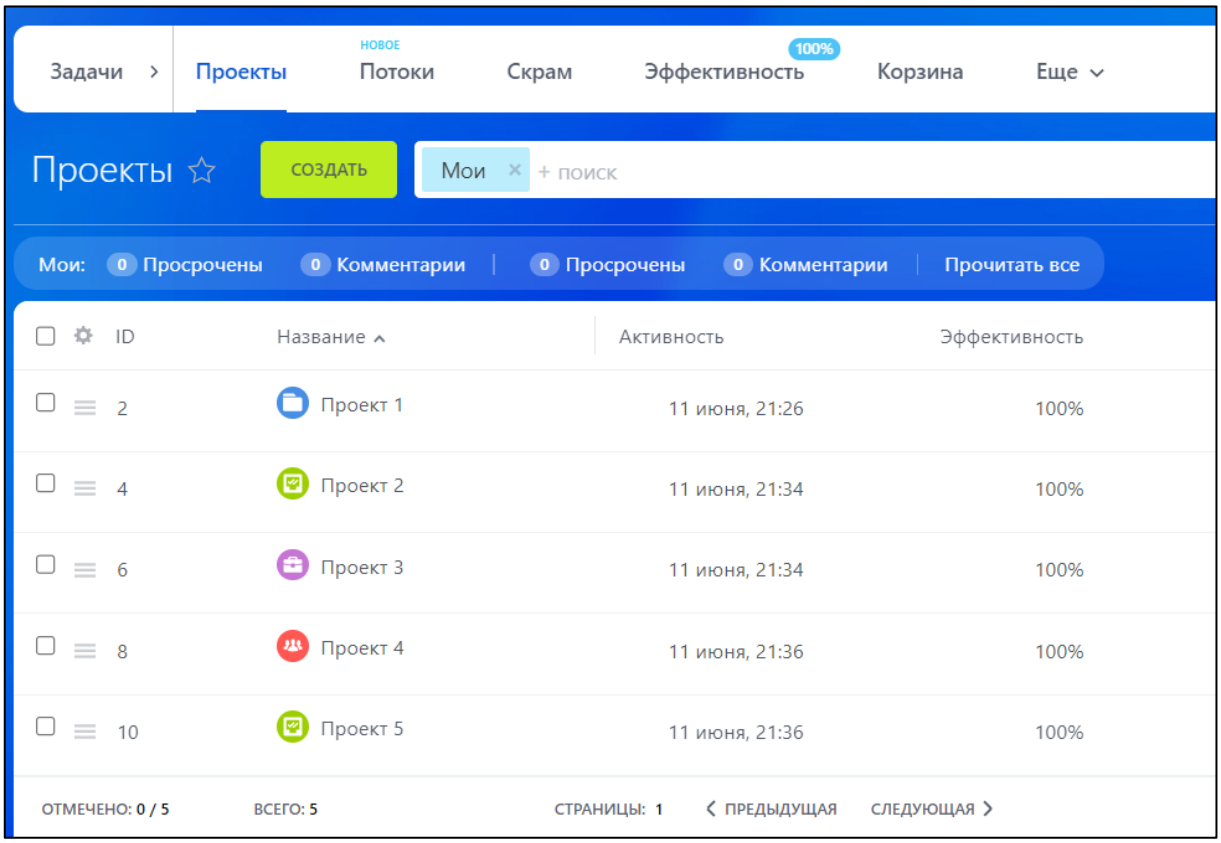

Рисунок 3 – Скриншот страницы с проектами в сервисе Bitrix24

Третьим из найденных конкурентов является GanttPRO [7]. GanttPRO – это онлайн-программа для управления проектами, которая облегчает планирование и реализацию проектов с помощью диаграмм Ганта. Данное приложение для планирования проектов позволяет создавать диаграммы Ганта для простых и сложных проектов, отслеживать прогресс, систематизировать задачи и подзадачи нужным образом. Достоинства и недостатки данного сервиса приведены в таблице 3.

Таблица 3 – Достоинства и недостатки GanttPRO

Достоинства	Недостатки
1. Красивый дизайн. 2. Интерфейс и служба поддержки на русском языке. 3. Интеграция с JIRA Cloud, Google Drive, Slack.	1. Не интуитивный интерфейс. 2. Нет быстрого решения для работы с повторяющимися задачами. 3. Акцент на диаграмме Ганта.

Скриншот рабочего стола данного сервиса представлен на рисунке 4.

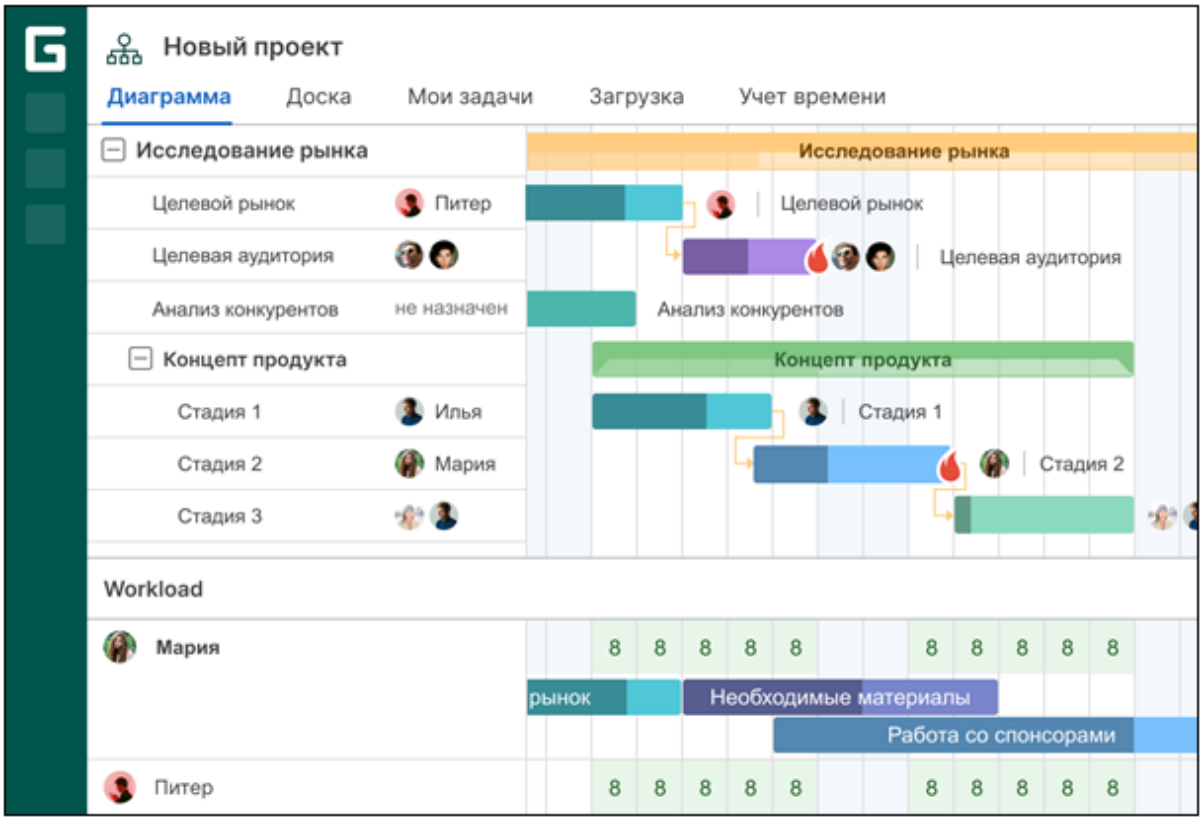

Рисунок 4 – Скриншот из сервиса GanttPRO

После рассмотрения достоинств и недостатков конкурентов можно сделать вывод, что в разрабатываемом веб-приложении хотелось бы видеть удобный и дружелюбный интерфейс, который будет понятен и прост в освоении. Также веб-приложение должно иметь средство для напоминаний по предстоящим задачам, систему комментариев, возможность выбора визуального исполнения в виде таблицы или kanban доски.

1.3. Анализ существующих решений для реализации проекта

Для создания веб-приложения возможно использование множества языков и специальных фреймворков. Таких как: Java (с использованием Java Servlet API), PHP + Laravel, Python + Django, Node.js, языки платформы .NET (C#) + ASP.NET, Ruby + Ruby on Rails и Go. Различные языки и фреймворки имеют различное быстродействие и удобство разработки, что позволяет выбирать, исходя из целей и задач создаваемого веб-приложения.

Главными критериями для данного веб-приложения являются:

1) наличие библиотек для работы с формами;
2) удобство работы с базами данных;
3) наличие большого количества понятной и актуальной документации.

Вывод по первой главе

В результате изучения существующих решений и литературы [8] был выбран язык программирования Python с фреймворком Django [9]. Данный язык и фреймворк отлично подойдут для решения поставленной задачи. Они соответствует всем необходимым требованиям и являются удобным решением для разработки веб-приложений.

2. ПРОЕКТИРОВАНИЕ

В разделе 2.1 приводятся функциональные и нефункциональные требования к проектируемой системе. В разделе 2.2 представлены диаграммы вариантов использования с описанием актеров, которые взаимодействуют с системой.

2.1. Требования к проектируемой системе

Функциональные требования

Функциональные требования [10] представляют собой заявление о том, как должна вести себя система. Ниже описаны функциональные требования к проектируемой системе.

1. Система управления проектами должна выводить пользователям актуальную информацию о проектах и задачах.
2. Система управления проектами должна позволять проектному менеджеру создавать новые, а также редактировать уже имеющиеся проекты и задачи.
3. Система управления проектами должна позволять проектному менеджеру закреплять исполнителей за задачами.
4. Система управления проектами должна позволять пользователям изменять статус задач.
5. Система управления проектами должна напоминать пользователям о предстоящих задачах.
6. Система управления проектами должна показывать проектному менеджеру задачи всех пользователей.
7. Система управления проектами должна позволять проектному менеджеру и исполнителю оставлять комментарии к задачам.
8. Система управления проектами должна показывать статистику по закрепленным за пользователем задачам.
9. Система управления проектами должна позволять проектному менеджеру создавать и удалять пользователей.

10. Система управления проектами должна позволять пользователям изменять информацию о себе.

11. Система управления проектами должна позволять проектным менеджерам создавать, удалять и присваивать пользователям тэги.

Нефункциональные требования

Нефункциональные требования [11] представляют собой определяющие свойства, которые система должна демонстрировать, или ограничения, которые она должна соблюдать, не относящиеся к поведению системы. Ниже описаны нефункциональные требования к проектируемой системе.

1. Система управления проектами должна быть написана на Python с использованием фреймворка Django.

2. Система управления проектами должна быть защищена от несанкционированного доступа путем шифрования данных и аутентификации пользователей.

3. Система управления проектами должна быть защищена от несанкционированного доступа путем контроля допуска к информации для пользователей разных уровней.

4. Система управления проектами должна быть масштабируема.

5. Система управления проектами должна иметь удобный интуитивно понятный интерфейс для пользователей.

6. Система управления проектами должна быть совместима с различными браузерами и операционными системами.

2.2. Диаграммы вариантов использования

Диаграмма вариантов использования [12] представляет собой рисунок, который отображает взаимодействие между вариантами использования, представляющими функции системы, и действующими лицами, представляющими людей или системы, получающие или передающие информацию в данную систему. Из диаграмм вариантов использования можно получить довольно много информации о системе. Этот тип диаграмм описывает

общую функциональность системы. Пользователи, менеджеры проектов, аналитики, разработчики, специалисты по контролю качества и все, кого интересует система в целом, могут, изучая диаграммы вариантов использования, понять, что система должна делать. На рисунке 5 представлена диаграмма вариантов использования для актеров «Посетитель», «Исполнитель» и «Время».

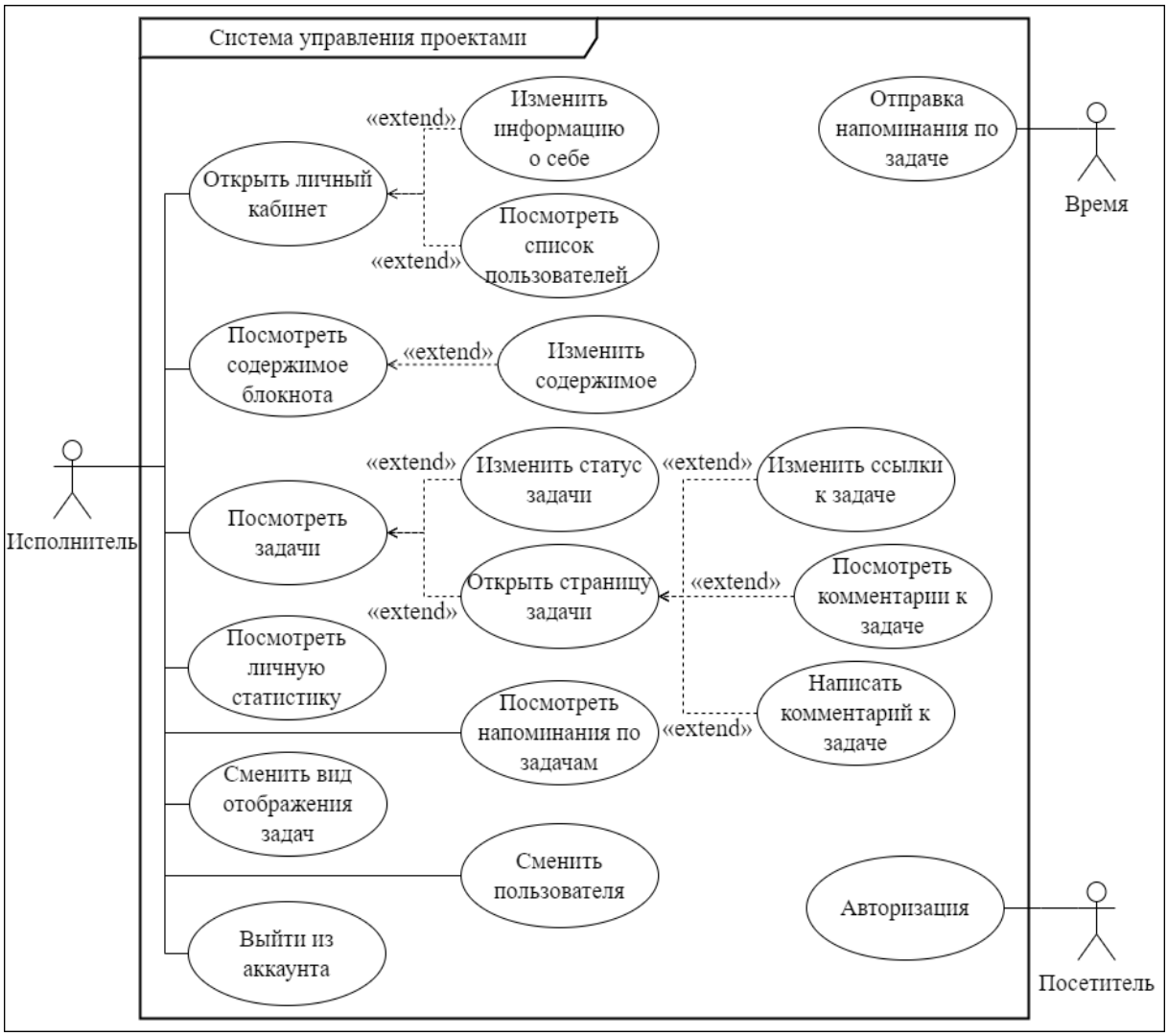

Рисунок 5 – Диаграмма вариантов использования для актеров «Посетитель», «Исполнитель» и «Время»

Исходя из получившейся диаграммы вариантов использования были выявлены актеры, взаимодействующие с системой и их краткое описание.

«Посетитель» – данный актер представляет собой исполнителя, который еще не вошел в систему.

«Исполнитель» – данный актер представляет собой пользователя, за которым будут закрепляться задачи.

«Время» – данный актер представляет собой время и имеет возможность отправлять уведомления пользователям.

На рисунке 6 показана диаграмма вариантов использования для актера «Администратор».

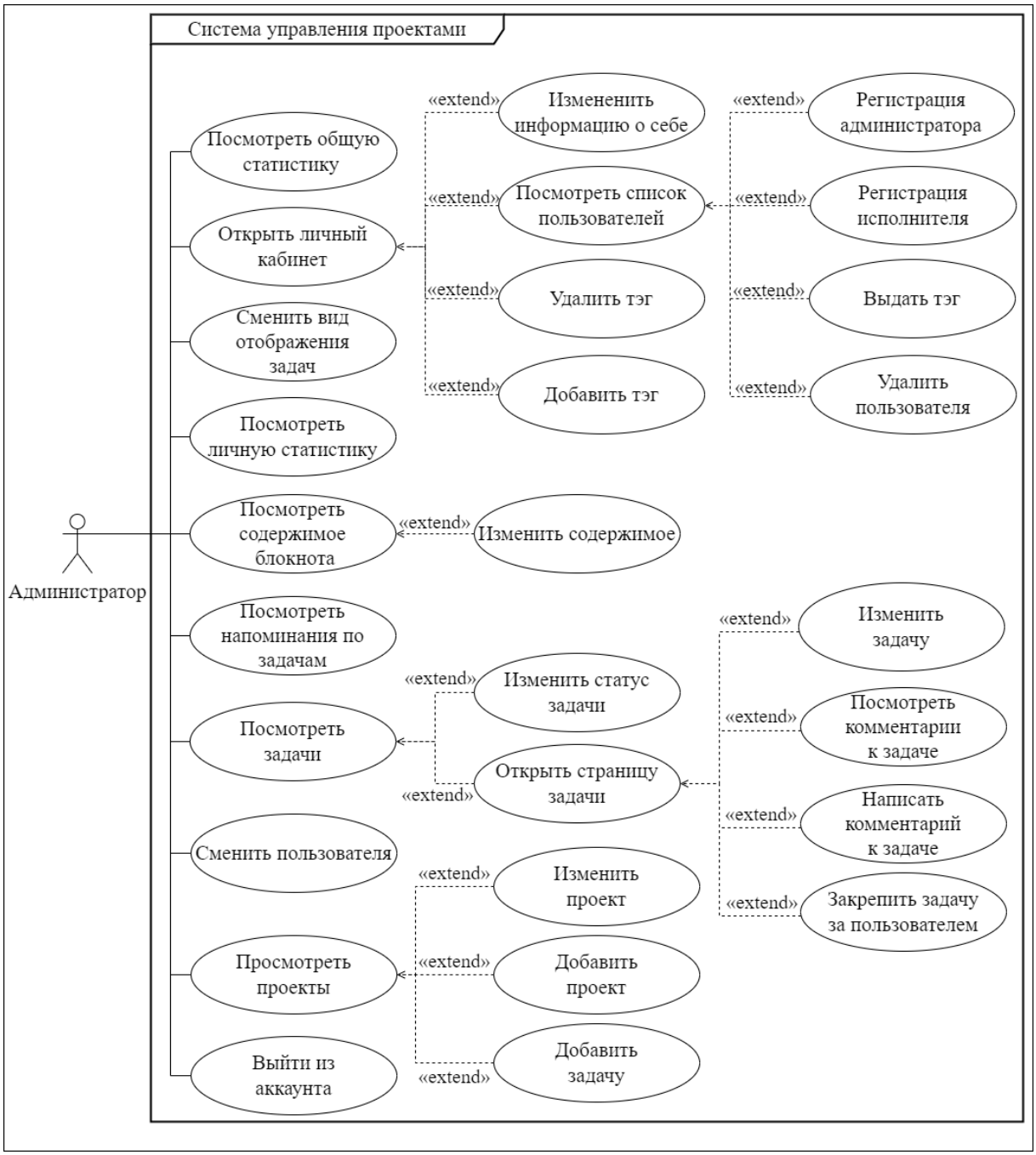

Рисунок 6 – Диаграмма вариантов использования «Администратор»

На приведенной диаграмме вариантов использования изображен актер «Администратор». Опишем данного актера.

«Администратор» – данный актер представляет собой пользователя, который имеет доступ к функциям управления проектами и администрирования.

Все актеры действуют в рамках одной системы. Многие варианты использования встречаются как у «Исполнителя», так и у «Администратора».

Краткое описание вариантов использования представлено ниже.

1. Отправка напоминания по задаче – данный вариант использования создает и отправляет уведомление пользователю системы о предстоящей задаче.
2. Авторизация – данный вариант использования осуществляет вход посетителей в систему.
3. Сменить пользователя – данный вариант использования позволяет авторизованным пользователям сменить аккаунт.
4. Выйти из аккаунта – данный вариант использования позволяет авторизованному пользователю выйти из аккаунта.
5. Открыть личный кабинет – данный вариант использования открывает страницу с информацией о пользователе.
6. Изменить информацию о себе – данный вариант использования изменяет логин, ФИО, пароль, номер телефона, e-mail и дату рождения.
7. Просмотр списка пользователей – данный вариант использования выводит список пользователей, существующих в системе.
8. Добавить тэг – данный вариант использования создает новый тэг.
9. Удалить тэг – данный вариант использования удаляет новый тэг.
10. Регистрация администратора – данный вариант использования позволяет администратору создать нового администратора в системе.
11. Регистрация исполнителя – данный вариант использования позволяет администратору создать аккаунт нового пользователя в системе.

12. Выдать тэг – данный вариант использования позволяет администратору выдать пользователю указанный тег.

13. Удалить пользователя – данный вариант использования позволяет администратору удалить пользователя из системы.

14. Посмотреть содержимое блокнота – данный вариант использования выводит блокнот конкретного пользователя.

15. Изменить содержимое – данный вариант использования позволяет изменить содержимое блокнота.

16. Посмотреть личную статистику – данный вариант использования отображает информацию по задачам пользователя.

17. Сменить вид отображения задач – данный вариант использования позволяет пользователю изменить вид отображения на таблицу или доску.

18. Посмотреть напоминания по задачам – данный вариант использования показывает пользователю напоминания по предстоящим задачам.

19. Посмотреть общую статистику – данный вариант использования показывает администратору статистику по всем задачам и проектам.

20. Посмотреть проекты – данный вариант использования выводит список проектов.

21. Добавить проект – данный вариант использования позволяет администратору создать новый проект.

22. Изменить проект – данный вариант использования позволяет администратору изменить информацию о существующем проекте.

23. Добавить задачу – данный вариант использования позволяет администратору создать новую задачу в существующем проекте.

24. Посмотреть задачи – данный вариант использования выводит список задач.

25. Изменить статус задачи – данный вариант использования позволяет пользователю изменить статус задачи.

26. Открыть страницу задачи – данный вариант использования позволяет пользователю открыть страницу с информацией по задаче.

27. Изменить ссылки к задаче – данный вариант использования позволяет исполнителю изменить ссылки к задаче.

28. Изменить задачу – данный вариант использования позволяет администратору изменить информацию о существующей задаче.

29. Закрепить задачу за пользователем – данный вариант использования позволяет администратору закрепить существующую задачу за исполнителем.

30. Посмотреть комментарии к задаче – данный вариант использования позволяет пользователям просматривать комментарии к выбранной задаче.

31. Написать комментарий к задаче – данный вариант использования позволяет пользователю оставить комментарий к выбранной задаче.

2.3. Общее описание архитектуры системы и ее компонентов

Архитектура Django похожа на «Модель-Представление-Контроллер» (MVC – Model-View-Controller):

- модель – бизнес-логика, то есть совокупности методов, правил и ограничений работы с данными;
- представление – компонент, отображающий пользователю данные в зависимости от изменения модели;
- контроллер – программный посредник, обрабатывающий действия пользователя и сообщающий модели, как она должна измениться.

Контроллер классической модели MVC примерно соответствует уровню, который в Django называется представление, а логика представления реализуется в Django уровнем Шаблонов. Из-за этого уровневую архитектуру Django часто называют «Модель-Шаблон-Представление» (MTV – Models-Templates-Views).

Схема архитектуры веб-приложения представлена на рисунке 7.

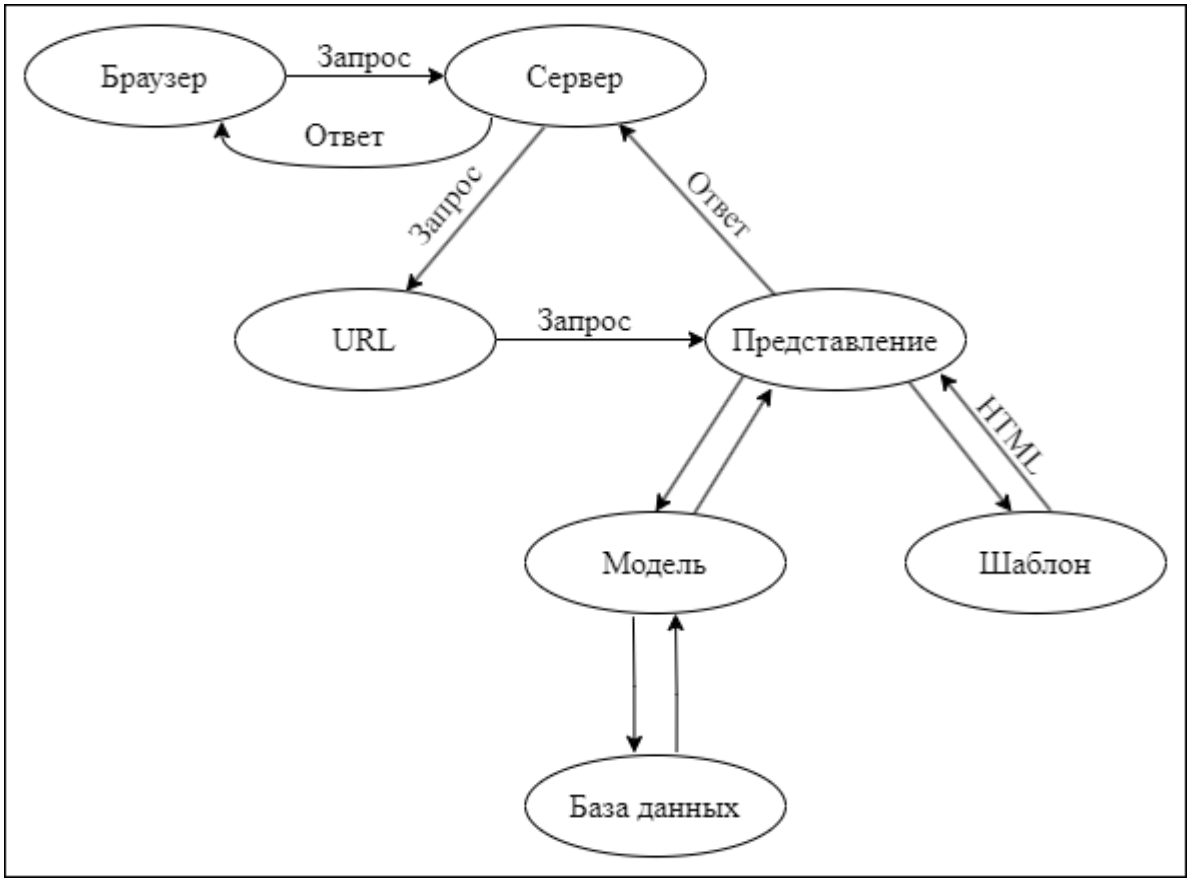

Рисунок 7 – Схема архитектуры веб-приложения

Архитектура Django состоит из следующих основных компонентов:

- модель – отвечает за работу с данными (доступ, обработку, проверку и т.д.);
- шаблоны – определяют, как будет отображаться информация;
- представление – описывает, какие именно данные будут показываться пользователю.

Принцип повторяет концепцию MVC, поэтому Django относится к MVC-совместимым платформам.

В соответствии с архитектурой веб-приложения, на рисунке 8 представлена диаграмма компонентов веб-приложения управления проектами.

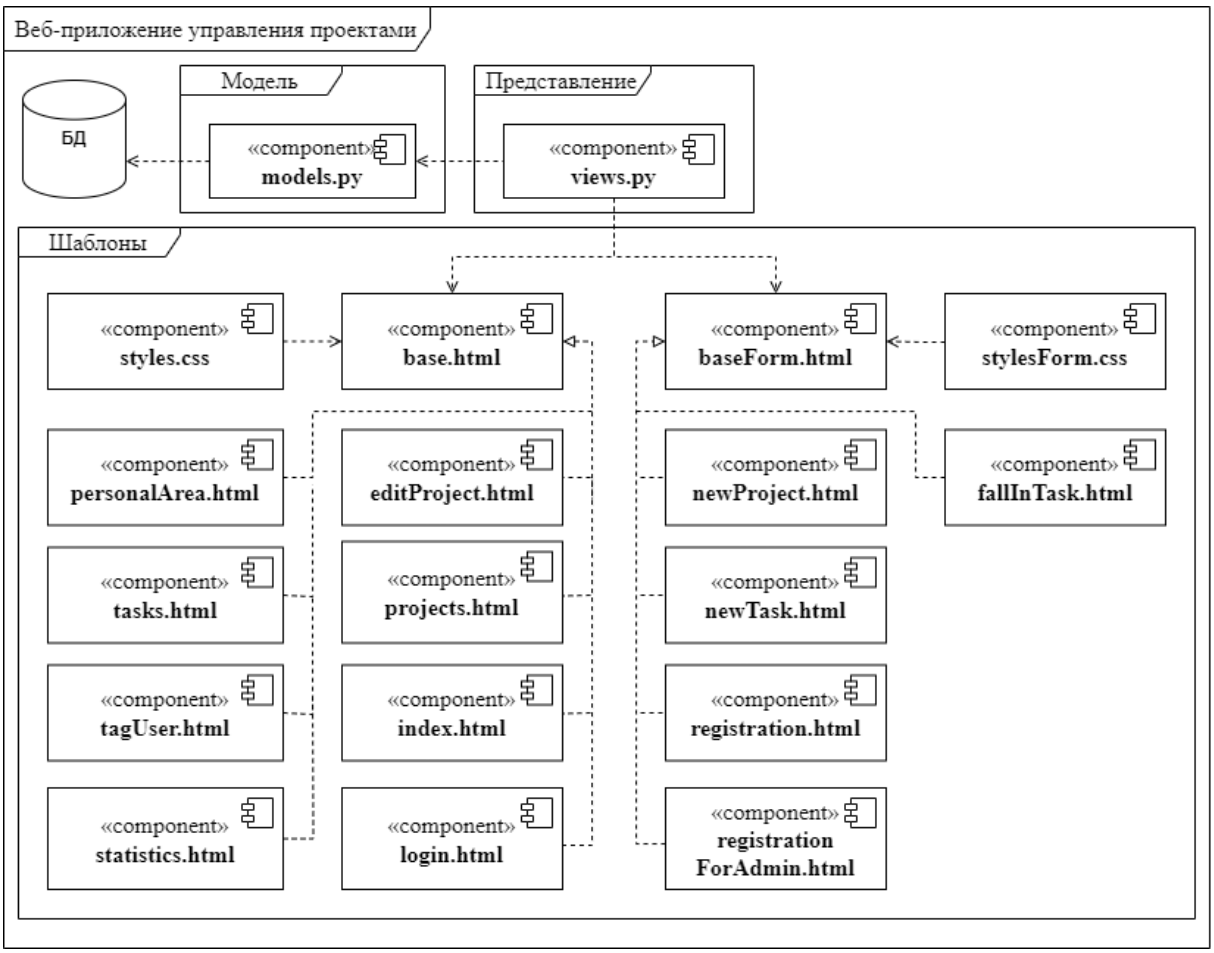

Рисунок 8 – Диаграмма компонентов веб-приложения управления проектами

Описание компонентов веб-приложения управления проектами:

– models.py представляет собой модель, которая является основным источником данных;

– views.py является представлением, которое описывает, какие именно данные и шаблоны будут показываться пользователю.

Все файлы с расширением .html представляют собой веб-страницы, на которых отображаются данные и с помощью которых пользователь взаимодействует с системой. Файлы с расширением CSS представляют собой таблицу стилей, определяющих позиционирование и отображение контента на веб-странице.

2.4. ER-диаграмма базы данных

В Django таблицы базы данных описываются в виде моделей классов. Модель содержит набор полей данных, которые хранятся в базе. Так как фреймворк следует принципу DRY (Don’t repeat yourself), то все модели определяются в одном месте. Django использует принцип ORM (Object-Relational Mapping), следовательно, поддерживаются такие принципы ООП как: наследование, полиморфизм и инкапсуляции и т.д.

На рисунке 9 изображена ER-диаграмма базы данных.

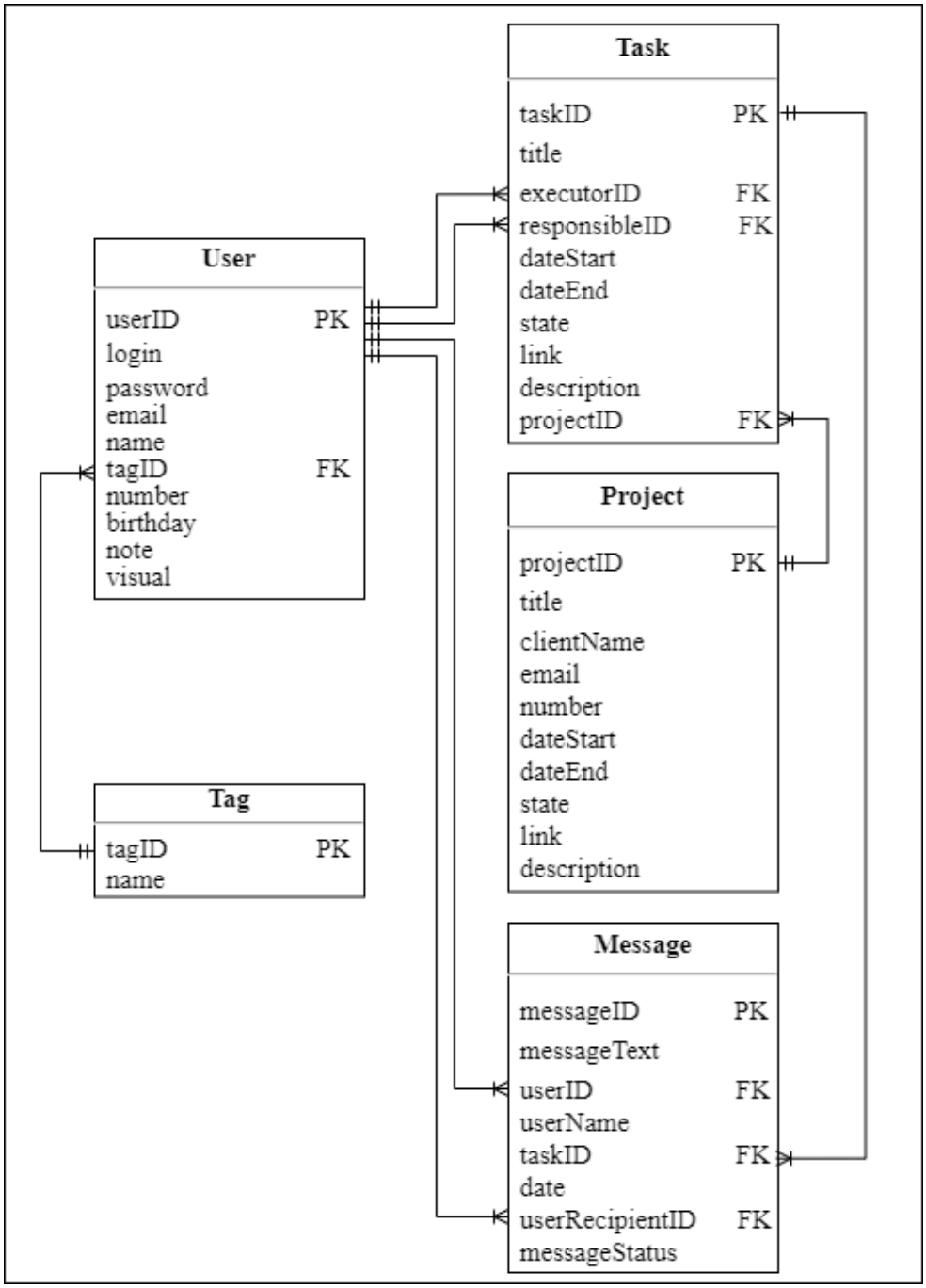

Рисунок 9 – ER-диаграмма базы данных

Для построения ER-диаграммы базы данных использовался инструмент Draw.io [13], который предназначен для создания диаграмм, блок-схем и др. Все сущности связаны между собой связью один ко многим и их внешний вид обусловлен использованием нотации Crowsfoot («Воронья лапка») в приложении.

Сущности, представленные на ER-диаграмме:

1) tag – сущность, представляющая собой тэг, который присваивается пользователям для закрепления за ними задач, соответствующих их обязанностям;

2) user – сущность, представляющая собой пользователя системы;

3) project – сущность, представляет собой проект, и служит для обобщения задач;

4) task – сущность, представляет собой задачу, которая закрепляется за проектом;

5) message – сущность, представляющая собой сообщение, которое пользователи оставляют к задаче.

Вывод по второй главе

В данной главе были составлены функциональные и нефункциональные требования к системе, разработаны диаграммы вариантов использования с полным описанием и спецификациями. Дано описание архитектуры системы и ее компонентов, а также составлена ER-диаграмма базы данных.

3. РЕАЛИЗАЦИЯ

В разделе 3.1 приводятся используемые технологии и инструменты, которые понадобятся при разработке. В разделе 3.2 приводится информация о создании проекта. В разделе 3.3 описана реализация компонента модель. В разделе 3.4 приводятся примеры реализации вариантов использования в компоненте представление. В разделе 3.5 представлена реализация компонента шаблон.

3.1. Средства разработки

Используемые технологии

Для реализации веб-приложения управления проектами были выбраны технологии, представленные ниже.

1. HTML [14] – стандартизированный язык разметки документов в сети интернет.
2. CSS [15] – формальный язык описания внешнего вида документа, написанного с использованием языка разметки.
3. Bootstrap 5 [16] – набор инструментов для работы с интерфейсом веб-сайтов. Включает в себя шаблоны оформления для HTML и CSS, а также таких вещей, как веб-формы и кнопки.
4. Python 3 [17] – высокоуровневый язык программирования общего назначения с динамической строгой типизацией и автоматическим управлением памятью, ориентированный на повышение производительности разработчика, читаемости кода и его качества, а также на обеспечение переносимости написанных на нем программ.
5. SQLite3 [18] – библиотека на языке C, которая реализует небольшой, быстрый, автономный, высоконадежный, полнофункциональный механизм базы данных SQL.

Для разработки веб-приложения с использованием языка программирования Python 3 был выбран набор библиотек и фреймворков, представленный ниже.

1. Django – это высокоуровневый Python веб-фреймворк, который позволяет быстро создавать безопасные и поддерживаемые веб-сайты.

2. Passlib [19] – это библиотека хеширования паролей для Python 3, которая обеспечивает реализацию 30 алгоритмов хеширования паролей.

3. Sqlparse [20] – это анализатор SQL для Python. Он обеспечивает поддержку для анализа, разделения и форматирования операторов SQL.

Инструменты разработки

Вся работа над веб-приложением производилась в среде разработки PyCharm 2023.2 [21]. Для работы с HTML и CSS использовался редактор кода Visual Studio Code [22].

3.2. Создание проекта

Для создания веб-приложения был установлен фреймворк Django. После установки, с помощью команды «`django-admin startproject server`», был создан проект. Затем создается приложение командой «`python manage.py startapp ProjectManagementApplication`». На рисунке 10 представлена структура проекта.

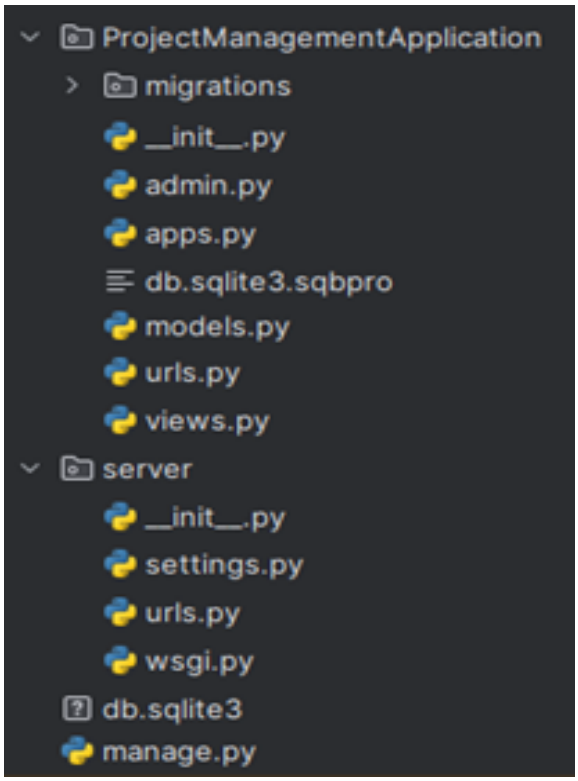

Рисунок 10 – Структура проекта Django

Назначение файлов:

- «server/__init__.py» – пустой файл, указывающий интерпретатору Python, что данная директория должна восприниматься в качестве пакета;
- «server/settings.py» – содержит конфигурацию проекта;
- «server/urls.py» – в данном файле объявляются URL;
- «server/wsgi.py» – файл, обеспечивающий работу с веб-сервером по протоколу WSGI;
- «manage.py» – файл, обеспечивающий взаимодействие с проектом;
- «projectManagementApplication/admin.py» – конфигурационный файл, связанный с панелью Администратора;
- «projectManagementApplication/apps.py» – файл с настройками приложения;
- «projectManagementApplication/models.py» – файл с определениями моделей приложения;
- «projectManagementApplication/urls.py» – в данном файле объявляются URL для приложения;
- «projectManagementApplication/views.py» – файл, содержащий функции, которые обрабатывают запросы пользователей.

Созданное приложения было добавлено в конфигурационном файле «settings.py» в списке «INSTALLED_APP».

3.3. Реализация компонента модель

Созданная ER-диаграмма базы данных описывается с помощью классов на языке Python. Каждый класс наследует базовый класс `Model`, который, в свою очередь, содержит методы по добавлению, удалению и изменению данных.

Исходя из спроектированной ER-диаграммы базы данных, в файл «models.py» были добавлены классы: `User`, `Tag`, `Project`, `Task` и `Message`.

В листинге 1 приведен код файла «models.py».

Листинг 1 – Код файла models.py

```
class User(models.Model):
    login = models.CharField(max_length=20)
    password = models.CharField(max_length=150)
    email = models.EmailField()
    name = models.CharField(max_length=100)
    tag = models.CharField(max_length=100)
    number = models.CharField(max_length=11)
    birthday = models.CharField(max_length=10)
    note = models.TextField()
    visual = models.IntegerField()

class Tag(models.Model):
    name = models.CharField(max_length=100)

class Project(models.Model):
    title = models.CharField(max_length=150)
    clientName = models.CharField(max_length=100)
    dateStart = models.CharField(max_length=10)
    email = models.EmailField()
    number = models.CharField(max_length=11)
    dateEnd = models.CharField(max_length=10)
    state = models.CharField(max_length=20)
    link = models.TextField()
    description = models.TextField()

class Task(models.Model):
    title = models.CharField(max_length=150)
    idExecutor = models.IntegerField()
    idResponsible = models.IntegerField()
    dateStart = models.CharField(max_length=10)
    dateEnd = models.CharField(max_length=10)
    link = models.TextField()
    description = models.TextField()
    idProject = models.IntegerField()
    state = models.CharField(max_length=20)

class Message(models.Model):
    idTask = models.IntegerField()
    messageText = models.TextField()
    date = models.TextField()
    idUser = models.IntegerField()
    userName = models.CharField(max_length=100)
    idUserRecipient = models.IntegerField()
    messageStatus = models.IntegerField()
```

После изменения кода файла «models.py» была выполнена миграция для файлов приложения с помощью команды «`python manage.py migrate`», а также миграция классов с помощью команды «`python manage.py makemigrations`». Созданная база данных в папке проекта представлена на рисунке 11.

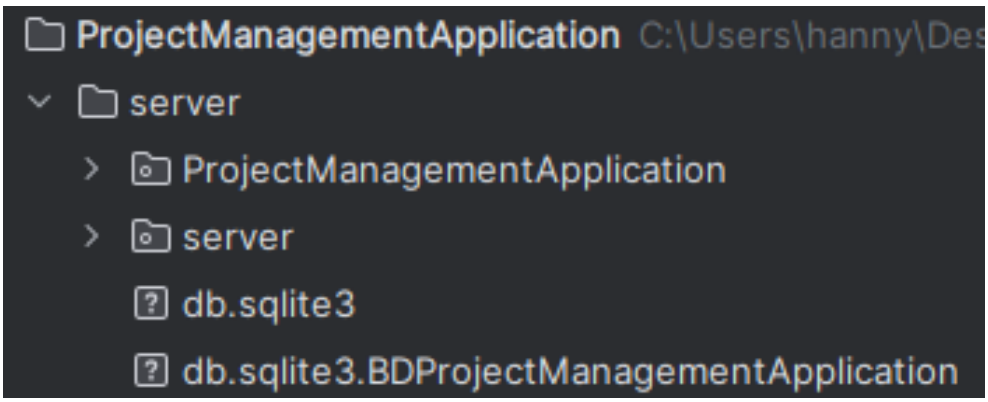

Рисунок 11 – Созданная база данных

Созданные таблицы базы данных представлены на рисунке 12.

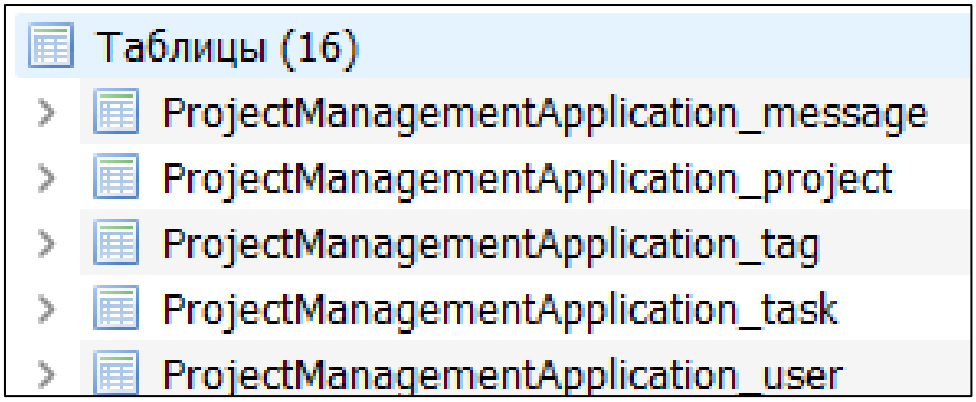

Рисунок 12 – Таблицы базы данных

3.4. Реализация компонента представление

Реализация компонента представления содержится в файле «views.py». То есть, «views.py» принимает HTTP-запросы от веб-клиентов и возвращает HTTP-ответы. Представление имеет доступ к базам данных и шаблонам. Поэтому в этом файле и будут прописаны все функции, которыми могут обладать пользователи веб-приложения.

Из-за большого количества вариантов использования в веб-приложении, было принято решение описать основные из них, с которыми пользователи будут работать постоянно. Их спецификации представлены в приложении.

Вариант использования «Добавить задачу»

Диаграмма последовательности для варианта использования «Добавить задачу» показана на рисунке 13.

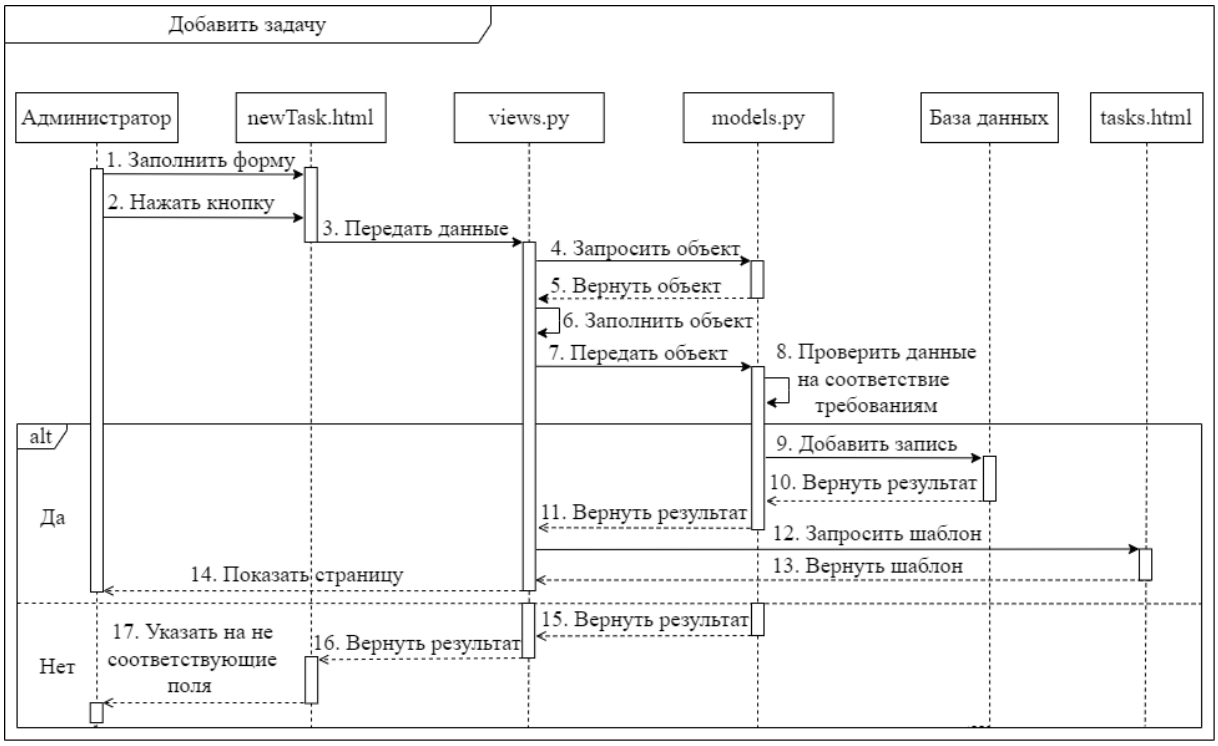

Рисунок 13 – Диаграмма последовательности для варианта использования «Добавить задачу»

Администратор, находясь на странице «newTask.html», заполняет форму, после чего нажимает кнопку «Сохранить». Данные передаются компоненту «views.py», после чего представление запрашивает объект у компонента «models.py». Модель возвращает объект представлению, которое заполняет его и передает обратно в модель, где происходит проверка на соответствие требованиям. Если данные соответствуют, то запись добавляется в базу данных. База данных возвращает результат о добавлении записи, после чего модель возвращает представлению результат об успешном добавлении записи. Компонент «views.py» запрашивает шаблон страницы «tasks.html», после чего администратор видит страницу. Если же данные не прошли проверку на соответствие требованиям, то на странице «newTask.html» происходит указание на поля ввода, где данные не соответствуют требованиям и администратору предлагается их исправить.

В листинге 2 представлена реализация функции `newTaskForm`, которая содержит код реализации варианта использования «Добавить задачу».

Листинг 2 – Реализация функции `newTaskForm`

```
def newTaskForm(request, id):
  user=User.objects.get(login=request.session['login'])
  messageList = list()
  msgState = 1
  messages = Message.objects.filter(idUserRecipient=user.id).exclude
(idUser=user.id)
  for msg in messages:
    if (msg.messageStatus == 0):
      messageList.append(msg)
  if (len(messageList) == 0):
    msgState = 0
  if request.method == "GET":
    form = NewTaskForm()
    return render(request, "newTask.html", {"form": form, "tag": re-
quest.session['tag'],"name": request.session['name'],  "login": re-
quest.session['login'], "messageList": messageList, "msgState": msgState,})
  if request.method == "POST":
    form = NewTaskForm(request.POST)
    project = Project.objects.get(id=id)
    user = User.objects.get(name=request.session['name'])
    task = Task()
    task.title = request.POST.get("title")
    task.executorName = request.POST.get("Исполнитель")
    task.dateStart = request.POST.get("dateStart")
    task.dateEnd = request.POST.get("dateEnd")
    task.titleProject = project.title
    task.state = "В планах"
    task.link = request.POST.get("link")
    task.description = request.POST.get("description")
    executor = User.objects.get(name=request.POST.get("Исполнитель"))
    task.idResponsible=user.id
    task.responsibleName = request.session['name']
    task.idExecutor=executor.id
    task.idProject=id
    task.save()
  return HttpResponseRedirect("/projects/tasks/"+str(task.idProject)+"/")
```

Вариант использования «Изменить статус задачи»

Одной из возможностей как исполнителя, так и администратора является изменение статуса задачи на главной странице. Диаграмму деятельности варианта использования «Изменить статус задачи» можно увидеть на рисунке 14.

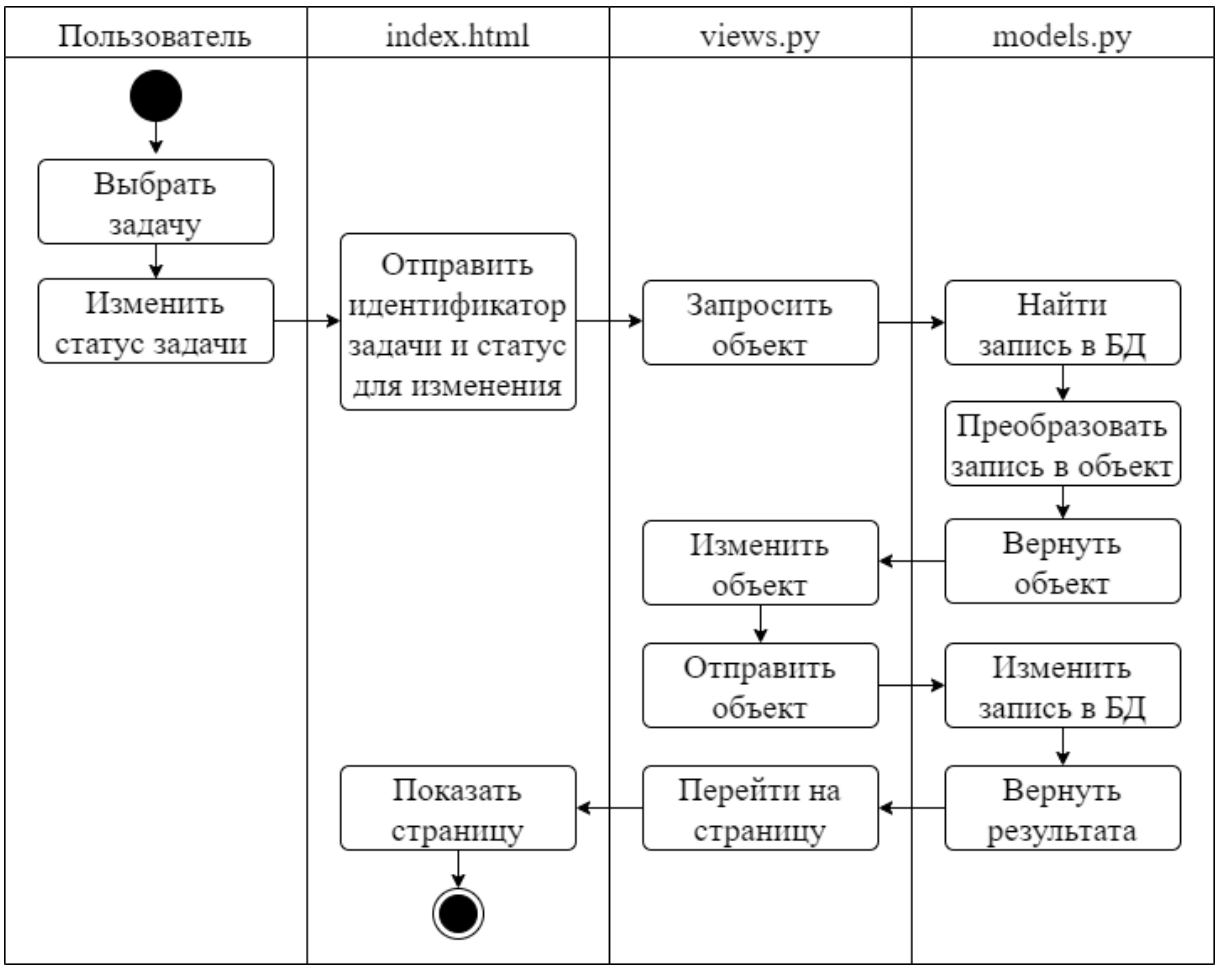

Рисунок 14 – Диаграмма деятельности для варианта использования «Изменить статус задачи»

Пользователь, находясь на главной странице «index.html», выбирает задачу и изменяет ее статус. Данные со страницы передаются компоненту «views.py», который обращается к компоненту «models.py» для получения объекта. Модель находит запись в базе данных, после чего объект возвращается в представление. Представление изменяет объект и передает его в модель, которая изменяет запись в базе данных. После выполнения изменения возвращается результат. Представление обновляет главную страницу, где пользователь видит задачу с измененным статусом.

В листинге 3 представлена реализация функции `mainUserSaveEditTask1`, которая содержит код реализации варианта использования «Изменить статус задачи».

Листинг 3 – Реализация функции `mainUserSaveEditTask1`

```
def mainUserSaveEditTask1 (request, id, state):
  try:
    task = Task.objects.get(id=id)
    task.state = state
    task.save()
    return HttpResponseRedirect("/")
  except Task.DoesNotExist:
    return HttpResponseNotFound("<h2>Задача не найдена</h2>")
```

3.5. Реализация компонента шаблоны

Шаблоны представляют из себя html страницы, которые имеют одинаковую структуру, но разное содержимое. Поэтому было принято решение создать два файла для хранения базовой структуры html файлов.

Для страниц, содержащих формы для заполнения, используется структура файла «baseForm.html», а для остальных страниц «base.html». Два этих файла отличаются только стилями, которые отвечают за внешний вид документа. На рисунке 15 представлена главная страница «index.html», которая наследует структуру файла «base.html».

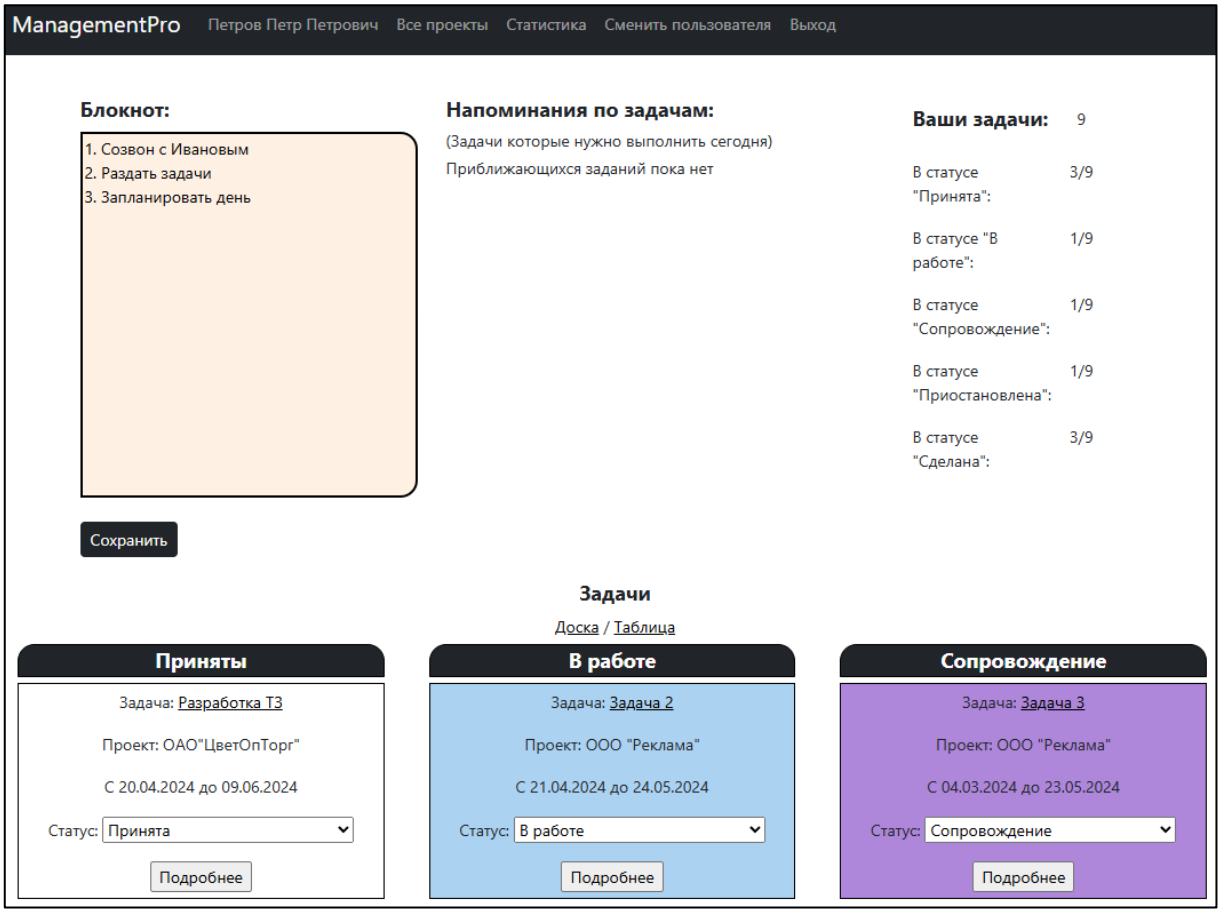

Рисунок 15 – Главная страница веб-приложения

На рисунке 16 представлена страница создания задачи «newTask.html», которая наследует структуру файла «baseForm.html».

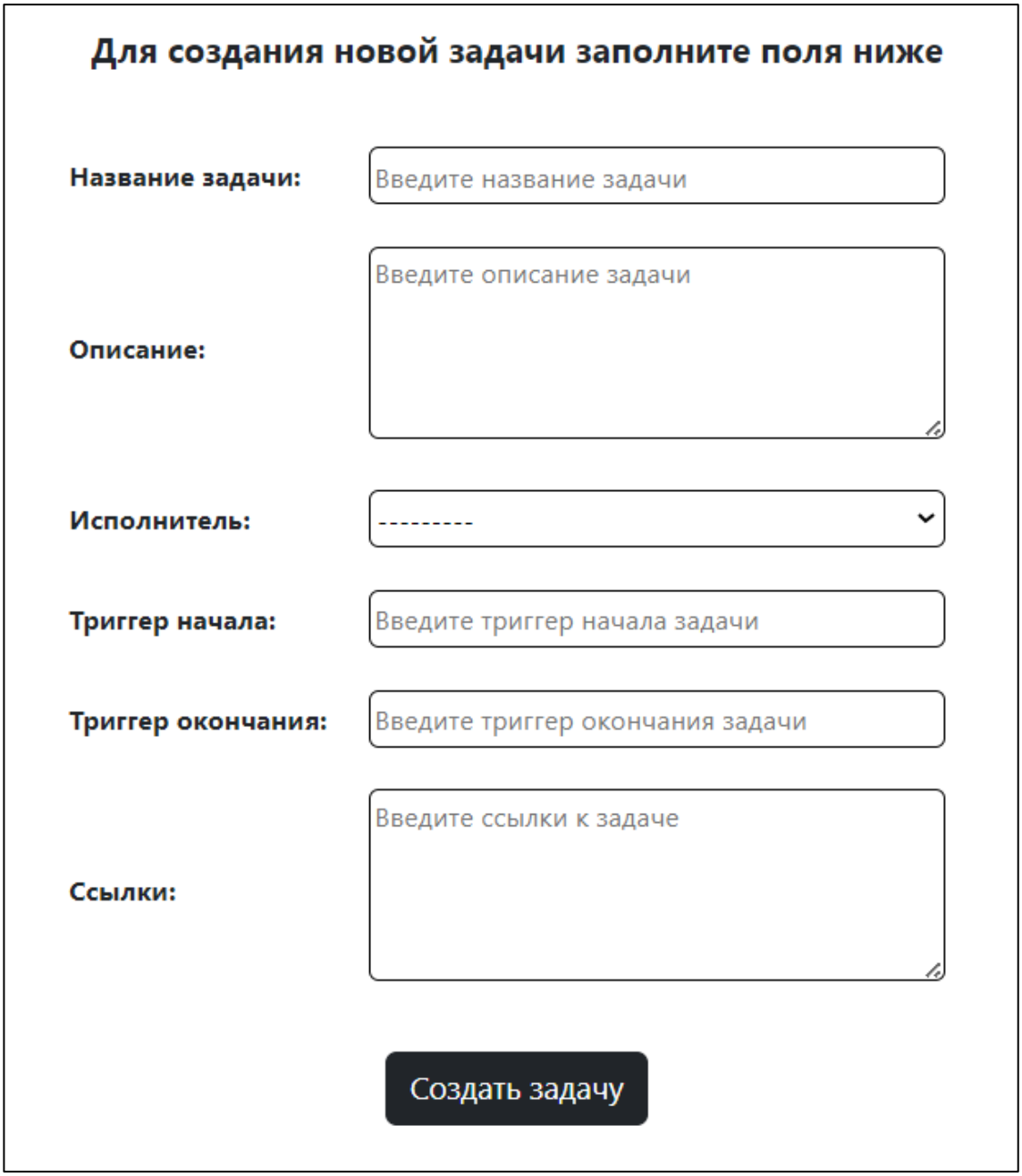

Рисунок 16 – Страница создания проекта в веб-приложении

Вывод по третьей главе

В данной главе были описаны средства разработки, процесс создания проекта и базы данных. Помимо этого, была описана реализация всех компонентов, определенных на стадии проектирования.

4. ТЕСТИРОВАНИЕ

В разделе 4.1 приводятся функциональное тестирование разработанного веб-приложения. В разделе 4.2 представлена проверка нефункциональных требований, определенных на этапе проектирования. В разделе 4.3 приводится юзабилити тестирование и тестирование верстки.

4.1. Функциональное тестирование

Функциональное тестирование предназначено для определения соответствия разработанного программного обеспечения исходным функциональным требованиям. Результаты функционального тестирования представлены в таблице 4.

Таблица 4 – Функциональное тестирование

№	Название теста	Действие	Ожидаемый результат	Результат теста
1	Просмотр контента	Исполнитель просматривает контент каждой веб-страницы	Актуальный контент отображается на веб-странице	Пройдено
2	Добавление проекта, изменения информации о нем	Администратор добавляет проект, а затем изменяет информацию о нем	Проект добавлен. Информация о проекте изменена	Пройдено
3	Добавление задачи, изменения информации о ней	Администратор добавляет задачу, а затем изменяет информацию о ней	Задача добавлена. Информация о задаче изменена	Пройдено
4	Закрепление задачи за исполнителем	Администратор закрепляет задачу за исполнителем	Задача закреплена за исполнителем	Пройдено
5	Изменение статуса задачи	Пользователь изменяет статус своей задачи	Статус задачи изменен	Пройдено
6	Просмотр напоминаний	Пользователь открывает главную страницу и находит пункт «Напоминания по задачам»	Задача, срок выполнения которой приближается, отображается в напоминании	Пройдено
7	Просмотр задач всех пользователей	Администратор открывает список задач для одного из проектов	Информация о задачах и исполнителях отображается на странице	Пройдено
8	Оставление комментариев	Пользователь оставляет комментарий к задаче	Комментарий отображается на странице задачи	Пройдено

Окончание таблицы 4

№	Название теста	Действие	Ожидаемый результат	Результат теста
9	Личная статистика	На главной странице пользователь находит пункт «Ваши задачи»	Отображается статистика выполнения задач пользователя	Пройдено
10	Добавление и удаление пользователя	Администратор добавляет пользователя, после чего переходит в список пользователей и удаляет данного пользователя	Пользователь был добавлен и отображается в списке пользователей. После удаления, пользователь пропадает из списка	Пройдено
11	Изменение информации о себе	Пользователь заходит в личный кабинет и меняет логин, имя, номер телефона и пароль	Данные о пользователе изменены	Пройдено
12	Создание и присвоение тэга	Администратор создает новый тэг, а затем присваивает его пользователю	Новый тэг был создан и присвоен пользователю.	Пройдено
13	Удаление тэга	Администратор удаляет тэг	Тэг удален и больше н отображается в списке	Пройдено

По результатам тестов можно считать, что функциональное тестирование пройдено успешно.

4.2. Проверка нефункциональных требований

Проверка нефункциональных требований была проведена для определения того, все ли требования к системе были соблюдены во время реализации. Результаты проверки нефункциональных требований приведены в таблице 5.

Таблица 5 – Результаты проверки нефункциональных требований

№	Требование	Результат
1	Система управления проектами должна быть написана на Python с использованием фреймворка Django.	Система написана на языке программирования Python 3.12 с использованием фреймворка Django 5.0.6
2	Система управления проектами должна быть защищена от несанкционированного доступа путем шифрования данных и аутентификации пользователей.	Для защиты от несанкционированного доступа используется библиотека Passlib, которая шифрует пароли и позволяет выполнять аутентификацию без дешифрования пароля.

Окончание таблицы 5

№	Требование	Результат
3	Система управления проектами должна быть защищена от несанкционированного доступа путем контроля допуска к информации для пользователей разных уровней.	Система имеет несколько выделенных ролей: администратор, исполнитель и посетитель. Разделение позволяет показывать только ту информацию, которую должен видеть пользователь, исходя из его роли.
4	Система управления проектами должна быть масштабируема.	Веб-приложение написано на языке программирования Python с использованием фреймворка Django. Язык программирования позволяет писать понятный код, а фреймворк дает возможность наращивать на проект новые возможности.
5	Система управления проектами должна иметь удобный интуитивно понятный интерфейс для пользователей.	Интерфейс приложения был разделен по соответствующим вкладкам, все кнопки и пункты имеют короткое и понятное название, что позволяет пользователю понимать, за что отвечает кнопка или определенный пункт в веб-приложении. Для проверки этого требования было проведено юзабилити тестирование.
6	Система управления проектами должна быть совместима с различными браузерами и операционными системами.	Для проверки этого требования было проведено тестирование верстки.

4.3. Юзабилити-тестирование и тестирование верстки

Юзабилити-тестирование

Юзабилити-тестирование – это исследование, выполняемое в целях определения, удобен ли некоторый искусственный объект для его предполагаемого применения. Целью тестирования была проверка удобства разработанного интерфейса веб-приложения. Для этого были сформированы следующие задачи:

- зарегистрировать пользователя;
- выполнить аутентификацию;
- изменить информацию о пользователе;
- открыть список всех пользователей;
- добавить тэг;
- присвоить тэг пользователю;
- изменить запись в блокноте;

– добавить проект;
– изменить информацию о нем;
– добавить задачу к проекту;
– изменить информацию о задаче;
– изменить исполнителя к задаче;
– изменить статус задачи;
– изменить вид отображения задач.

В проведенном юзабилити-тестировании приняли участие 5 человек, в ходе тестирования все задания были успешно выполнены.

Тестирование верстки

Тестирование верстки веб-приложения для различных разрешений было проведено в следующих браузерах: Firefox версии 113.0.2, Google Chrome версии 113.0.5672.127, Opera версии 99.0.4788.31, Yandex версии 23.5.0, Safari версии 16.5. Верстка на всех страницах выглядит идентично и работает правильно. Тестирование пройдено успешно.

Вывод по четвертой главе

В данном разделе было проведено тестирование функциональных требований веб-приложения. Проверка нефункциональных требований к системе. Проведено юзабилити-тестирование, в котором приняло участие 5 человек, а также проведено тестирование верстки в разных браузерах. Все тесты были успешно пройдены.

ЗАКЛЮЧЕНИЕ

Целью выпускной квалификационной работы являлась разработка веб-приложения для управления проектами на основе методологии Scrumban.

В ходе работы были выполнены следующие задачи:

1) проведен анализ предметной области;
2) определены требования и спроектировано веб-приложение;
3) реализовано веб-приложение;
4) протестировано веб-приложение.

В результате были решены все поставленные задачи и разработано веб-приложение для управления проектами на основе методологии Scrumban, таким образом, цель данной работы достигнута.

В дальнейшем планируется расширение функционала веб-приложения. Среди нововведений будет возможность добавления исполнителей в команды для совместной работы над задачами. Также планируется внедрение функции личных чатов для пользователей, которая позволит им обмениваться сообщениями в режиме реального времени и улучшит коммуникацию внутри команд. Кроме того, будет предоставлена более развернутая статистика по проектам и задачам, что поможет администраторам анализировать прогресс и принимать обоснованные решения для дальнейшего улучшения рабочих процессов. Эти улучшения направлены на повышение удобства использования приложения и увеличение его функциональных возможностей, что позволит командам работать более слаженно и эффективно.

ЛИТЕРАТУРА

1. Веб-приложение. [Электронный ресурс] URL: https://aws.amazon.com/ru/what–is/web–application/ (дата обращения: 04.02.2024 г.).

2. Халеева, Е. П. Разработка веб-сервиса для управления проектами. / Е. П. Халеева, А. Р. Гребенкин, М. А. Копцев. // Научные исследования и инновации: Сборник статей XI Международной научно-практической конференции, Саратов, 02 сентября 2021 года. – Саратов: Индивидуальный предприниматель Емельянов Николай Владимирович, 2021. – С. 96–100.

3. Scrumban: освоение двух методологий. [Электронный ресурс] URL: https://www.atlassian.com/ru/agile/project-management/scrumban (дата обращения: 04.02.2024 г.).

4. Коннолли Т., Бегг К. Базы данных. Проектирование, реализация и сопровождение. Теория и практика. // Вильямс, 2017. – 1440 с.

5. Сайт веб-приложения Trello. [Электронный ресурс] URL: https://trello.com/ru (дата обращения: 04.02.2024 г.).

6. Сайт веб-приложения Bitrix24. [Электронный ресурс] URL: https://www.bitrix24.ru/ (дата обращения: 04.02.2024 г.).

7. Сайт веб-приложения GanttPro. [Электронный ресурс] URL: https://ganttpro.com/ru/ (дата обращения: 04.02.2024 г.).

8. Васильченко А.Д. Сравнительный анализ фронтенд-фреймворков. // Современные проблемы лингвистики и методики преподавания русского языка в вузе и школе, 2022. – С. 1223–1231.

9. Документация Django. [Электронный ресурс] URL: https://www.djangoproject.com/ (дата обращения: 04.02.2024 г.).

10. Функциональные требования. [Электронный ресурс] URL: https://visuresolutions.com/ru/blog/functional–requirements/ (дата обращения: 04.02.2024 г.).

11. Нефункциональные требования. [Электронный ресурс] URL: https://visuresolutions.com/ru/blog/non–functional–requirements/ (дата обращения: 04.02.2024 г.).

12. Вейцман, В. М. Проектирование информационных систем: учебное пособие / В. М. Вейцман. – Санкт–Петербург: Лань, 2019. – 316 с.

13. Сайт приложения Draw.io. [Электронный ресурс] URL: https://www.drawio.com/ (дата обращения: 21.03.2024 г.).

14. Основы HTML. [Электронный ресурс] URL: https://developer.mozilla.org/ru/docs/Learn/Getting_started_with_the_web/HTML_basics (дата обращения: 21.03.2024 г.).

15. Основы CSS. [Электронный ресурс] URL: https://developer.mozilla.org/ru/docs/Learn/Getting_started_with_the_web/CSS_basics (дата обращения: 21.03.2024 г.).

16. Фреймворк Bootstrap. [Электронный ресурс] URL: https://bootstrap–5.ru/ (дата обращения: 21.03.2024 г.).

17. Язык программирования Python. [Электронный ресурс] URL: https://www.python.org/ (дата обращения: 21.03.2024 г.).

18. СУБД SQLite [Электронный ресурс] URL: https://www.sqlite.org/index.html (дата обращения: 21.03.2024 г.).

19. Библиотека passLib. [Электронный ресурс] URL: https://passlib.readthedocs.io/en/stable/ (дата обращения: 21.03.2024 г.).

20. Библиотека sqlparse. [Электронный ресурс] URL: https://pypi.org/project/sqlparse/ (дата обращения: 21.03.2024 г.).

21. Среда разработки PyCharm. [Электронный ресурс] URL: https://www.jetbrains.com/ru-ru/pycharm/ (дата обращения: 21.03.2024 г.).

22. Редактор кода Visual Studio Code. [Электронный ресурс] URL: https://code.visualstudio.com/ (дата обращения: 21.03.2024 г.).

ПРИЛОЖЕНИЕ. Спецификации вариантов использования

Спецификации вариантов использования системы управления проектами приведены в таблицах 1–3.

Таблица 1 – Спецификация ВИ «Добавить задачу»

Прецедент: Добавить задачу
ID: 1
Краткое описание: Создание новой задачи в существующем проекте
Главные актеры: Администратор
Второстепенные актеры: Нет
Предусловия: Администратор находится на странице добавления задач
Основной поток: 1. Прецедент начинается, когда администратор вводит информацию о задаче. 2. Администратор нажимает кнопку «Сохранить». 3. Система проверяет корректность введенных данных. 4. Система сохраняет новый проект.
Постусловия: Администратор видит в списке задач новую задачу
Альтернативные потоки: 1.1

Таблица 2 – Спецификация ВИ «Добавить задачу: данные не соответствуют требованиям»

Прецедент: Добавить задачу: данные не соответствуют требованиям
ID: 1.1
Краткое описание: Администратор некорректно заполняет поля ввода
Главные актеры: Администратор
Второстепенные актеры: Нет
Предусловия: Администратор добавлял новую задачу в существующий проект
Основной поток: 1. Прецедент начинается, когда система определила, что данные не соответствуют требованиям. 2. Система выводит администратору поля которые заполнены некорректно.
Постусловия: Информация остается без изменений

Таблица 3 – Спецификация ВИ «Изменить статус задачи»

Прецедент: Изменить статус задачи
ID: 2
Краткое описание: Пользователь изменяет статус задачи
Главные актеры: Исполнитель
Второстепенные актеры: Нет
Предусловия: Исполнитель находится на странице со списком задач
Основной поток: 1. Прецедент начинается, когда исполнитель изменяет статус задачи. 2. Система сохраняет изменения.
Постусловия: Пользователь видит задачу с измененным статусом
Альтернативные потоки: Нет