ФЕДЕРАЛЬНОЕ АГЕНТСТВО МОРСКОГО И РЕЧНОГО ТРАНСПОРТА

Федеральное государственное бюджетное образовательное учреждение высшего образования
«Государственный университет морского и речного флота имени адмирала С.О. Макарова»
(ФГБОУ ВО «ГУМРФ имени адмирала С.О. Макарова»)
Воронежский филиал

Кафедра математики, информационных систем и технологий
Направление подготовки 09.03.02 Информационные системы и технологии
(код, наименование направления подготовки/специальности)
Форма обучения очная

«К ЗАЩИТЕ ДОПУЩЕН(А)»
Заведующий кафедрой
_____
(подпись)
Черняева С. Н.
(ФИО)
_____ 2024

Выпускная квалификационная работа

Обучающегося Жуковского Ильи Сергеевича
(фамилия, имя, отчество)
Вид работы выпускная квалификационная работа бакалавра
(выпускная квалификационная работа бакалавра, специалиста, магистра)

Пояснительная записка

Тема Разработка Web-приложения «Календарь задач» для транспортной компании (на примере УО МР «Печора»).
(полное название темы квалификационной работы, в соответствии с приказом об утверждении тематики ВКР)

Руководитель работы Черняева С.Н., 21.06.2024
(должность, подпись, фамилия, инициалы, дата)
Консультант _____
(при наличии) (должность, подпись, фамилия, инициалы, дата)
Консультант _____
(должность, подпись, фамилия, инициалы, дата)
Обучающийся Жуковский И.С, 21.06.2024
(подпись, фамилия, инициалы, дата)

Воронеж
2024

ФЕДЕРАЛЬНОЕ АГЕНТСТВО МОРСКОГО И РЕЧНОГО ТРАНСПОРТА

Федеральное государственное бюджетное образовательное учреждение высшего образования
«Государственный университет морского и речного флота имени адмирала С.О. Макарова»
(ФГБОУ ВО «ГУМРФ имени адмирала С.О. Макарова»)
Воронежский филиал

Кафедра математики, информационных систем и технологий
Направление подготовки 09.03.02 Информационные системы и технологии
(код, наименование направления подготовки/специальности)
Форма обучения очная

УТВЕРЖДАЮ
Заведующий кафедрой
_____
(подпись)
Черняева С. Н.
(ФИО)
_____ 2024

Задание
на выпускную квалификационную работу

Вид работы ВКР бакалавра
(ВКР бакалавра, ВКР специалиста, ВКР магистра)

Обучающемуся Жуковскому Илье Сергеевичу
(фамилия, имя, отчество)

Тема Разработка Web-приложения «Календарь задач» для транспортной компании (на примере УО МР «Печора»).
Утверждена приказом ректора университета от _____ 20___, № _____
Срок сдачи законченной работы _____ 20___

Исходные данные (или цель ВКР):
Разработать Web-приложения «Календарь задач» для транспортной компании.
Перечень подлежащих исследованию, разработке, проектированию вопросов (краткое содержание ВКР):
(актуальность темы, цели и задачи ВКР; аналитический обзор литературных источников; постановка задачи исследования, разработки, проектирования; содержание процедуры исследования, разработки, проектирования; обсуждение результатов; дополнительные вопросы, подлежащие разработке; заключение – выводы по работе в целом, оценка степени решения поставленных задач, практические рекомендации; и др.)
– Введение. Актуальность выбранной темы, цель и задачи ВКР
(наименование вопроса, раздела и его краткое содержание)
– Исследовательский раздел. Характеристика организации УО МР «Печора», Разбор существующих решений на рынке, Анализ технологий и инструментов разработки web-приложений, Архитектура клиент-сервер
(наименование вопроса, раздела и его краткое содержание)

– Проектный раздел. Основные стадии процессов разработки приложения и их взаимодействие, Техническое задание, Проектирование веб-приложения, Разработка и тестирование веб-приложения, Инструкция по эксплуатации и рекомендации по дальнейшему развитию веб-приложения

(наименование вопроса, раздела и его краткое содержание)

– Заключение. Выводы по работе в целом. Оценка степени решения поставленных задач

(наименование вопроса, раздела и его краткое содержание)

Практические рекомендации: Добавление фильтров, создание системы чата, рассылки уведомлений

Перечень графического материала (или презентационного материала):

1. Титульный лист
2. Цель и задачи ВКР
3. Существующие аналоги решений на рынке
4. Обоснование выбора технологий и инструментов, позволяющих реализовать web-приложение
5. Требования к функционалу и интерфейсу
6. Проектирование архитектуры: бизнес-логика
7. Проектирование архитектуры: Диаграмма вариантов использования взаимодействия пользователя с задачами
8. Проектирование архитектуры: Схема Базы данных
9. Разработка: Интерфейс окна авторизации
10. Разработка: Интерфейс главной страницы
11. Рекомендации по дальнейшему развитию web-приложения
12. Результаты ВКР

Консультанты по разделам ВКР (при наличии):

1. _____

(наименование раздела, ученая степень, ученое звание и должность, ФИО консультанта)

2. _____

(наименование раздела, ученая степень, ученое звание и должность, ФИО консультанта)

3. _____

(наименование раздела, ученая степень, ученое звание и должность, ФИО консультанта)

Дата выдачи задания: _____ 20___

Задание согласовано и принято к исполнению: _____ 20___

Руководитель ВКР: доцент к.ф.-м.н. Черняева С.Н. _____

(должность, ученая степень, ученое звание, ФИО) (подпись)

Обучающийся: ИТ-4 Жуковский Илья Сергеевич _____

(учебная группа, ФИО) (подпись)

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В условиях динамичного развития разных отраслей, а также в взаимодействии между ними и конкуренции на рынке, в том числе и для транспортной отрасли, эффективное управление временем и задачами играет ключевую роль в обеспечении успеха и эффективной работы транспортной компании.

Сотрудникам компании необходимо иметь высокий уровень организованности, планирования и пунктуальности для решения поставленных целей, таких как:

— Сложность логистических процессов;

— Нестабильность графиков;

— Эффективное распределение времени для решения рабочих задач.

Исходя из этого, каждая компания пытается найти способы, позволяющие решить данные проблемы и повысить свою конкурентоспособность на рынке, а также увеличить прибыль.

В настоящее время всё больше компаний пользуются приложениями, которые позволяют распределять время, повышая работоспособность сотрудника. Однако, имеется много организаций, где сотрудники продолжают пользоваться такими средствами распределения, как заметки на телефоне или бумажным блокнотом. Поэтому автоматизация этого, крайне важного процесса, позволит решить множество проблем, связанных с распределением времени.

Исходя из этого, была поставлена цель выполнения выпускной квалификационной работы: Разработать Web-приложение «Календарь задач» для транспортной компании (на примере УО МР «Печора»). Для реализации необходимо поставить цели, которые должны быть выполнены в данной работе:

— Изучить существующие аналоги решений на рынке;

— Проанализировать технологии и инструменты, позволяющие разработать поставленное web-приложение;

— Спроектировать структуру системы;

— Разработать web-приложение.

В задачи, которые нужно решить, входит подробный разбор способа реализации web-приложения, изучение технологии реализации и выбор наиболее подходящие и надежные, что позволит эффективно и быстро разработать программу. Проанализировав существующие аналоги будущего web-приложения, можно изучить и разработать структурную составляющую создаваемого приложения. Тестирование всех элементов поможет выявить и исправить ошибки, увеличив надежность и безопасность веб-приложения. Изучение системы распределения сетевой нагрузки поможет в уменьшении количества затрачиваемых ресурсов устройств, увеличив производительность веб-приложения. Рекомендации по дальнейшему развитию веб-приложения позволят увеличить качество и функционал программы.

ИССЛЕДОВАТЕЛЬСКИЙ РАЗДЕЛ

1.1. Характеристика организации УО МР «Печора»

1.1.1. Организационная структура и описание деятельности отдела

Название подразделения: Ит-отдел.

Фамилия, имя, отчество начальника: Челов Михаил Александрович

Основная цель Ит-отдела: Разработка приложений, автоматизация процессов, повышение эффективности выполнения задач.

Основные функции Ит-отдела: Разработка приложений, поддержка существующего ПО.

Кадровый состав Ит-отдела: 8 программиста, 2 сисадмин, проектировщик.

Документы, поступающие в Ит-отдел из других подразделений: Техническое задание, отчёты об эксплуатации ПО.

Физическое представление, время и частота поступления, вид обработки и требования к безопасности каждого из этих документов: проектная документация, цифровые копии, обновление правил доступа.

Продолжительность хранения каждого из документов и условия перевода в архив: Копии документов – полгода, оригиналы документов – 7 лет цифровом формате неограниченное количество времени.

Документы, формирующиеся в Ит-отдел: Документация ПО, документ о правилах использования ПО.

Физическое представление время, частота и вид обработки и требования к безопасности каждого из этих документов: цифровые копии, обновление правил доступа.

Продолжительность хранения каждого из документов и условия перевода в архив: не передаются в архив.

Документы, передающиеся в другие подразделения: Документация ПО, документ о правилах использования ПО – передаются сотрудникам организации;

отчеты о работе и эксплуатации ПО – директору организации/заместителю директора.

Продолжительность хранения каждого из документов и условия перевода в архив: 2 месяца.

Информация, поступающая в отдел продаж из внешних к Управление Образования МР «Печора». (банк, заказчик, налоговые органы и т.д.): Информация не поступает.

Продолжительность хранения каждого из документов и условия перевода в архив: Данные документы хранятся в цифровом формате неограниченное количество времени.

Информация, передающаяся из Ит-отдела во внешние органы: информация не поступает.

Описание информационной инфраструктуры (техническое оснащение и программные продукты) Ит-отдела: 15 ноутбуков; 3 принтера, 1 сервер.

Основные недостатки бизнес-процессов и существующей системы обработки документов в отделе продаж: из-за сильной загруженности отдела возникают сложности с распределением времени при решении задач.

Предложения по реорганизации бизнес-процессов и существующей системы обработки документов в отделе продаж: разработать приложение для планирования времени и эффективного решения задач.

1.1.2. Описание аппаратного и программного обеспечения в отделе

Аппаратное обеспечение:

— Высокопроизводительный сервер обеспечивает безопасное хранение и эффективную работу с данных. Большое количество оперативной памяти обеспечивает одновременную поддержку множества пользователей.

— В отделе расположены 15 ноутбуков, как со стандартными для повседневной работы комплектующими, так и с наиболее дорогими и мощными.

Используются в основном для обработки документов, работы с презентации, анализа и проектирования;

— Принтеры: Установлено три принтера, которые могут ксерокопировать документы, печатать в черно-белом и цветном виде;

— Моноблоки: В организации установлено два моноблока;

— Системные блоки: 17 системных блоков имеют различную конфигурацию. На большинстве блоков стоит Intel Celeron, на котором решают в основном офисные задачи, а более мощные AMD Ryzen 3 задействованы в более сложных задача, где требуется более высокая производительность. Различные модели оперативной памяти, в основном DDR2, DDR3.

Программное обеспечение:

— Основной операционной системой является система Windows. Практически на всех устройствах стоит Windows 10, поскольку она является наиболее производительной и функциональной. Также на некоторых устройствах стоит Windows 8.1;

— Офисные приложения, такие как офисный пакет Microsoft Office 2013, стоит на всех устройствах и предназначен для работы с таблицами, документами и электронными почтами. Офисный пакет включает в себя: Word, Excel, PowerPoint и Outlook;

— В отделе используются утилиты для работы с файлами, такими как: 7zip для архивирования документов, Acrobat Reader используется для чтения файлов в формате PDF. Используются разные браузеры, основным является Yandex браузер, Opera. На некоторых машинах установлен языки программирования Python 3.11 и JavaScript и различные IDE (интегрированная среда разработки) для работы с ними, такие как Visual Studio Code, Sublime Text, PyCharm;

— Средства защиты: в организации серьезно относятся к безопасности как данных и систем самой организации, так и личных данных сотрудников. Для этого используются следующие программы: КриптоПро CSP выполняет много

функций, таких как работу с электронными подписями, а именно создает, проверяет и защищает от подмены, также выполняет функцию шифрования текстов и файлов. Технология VipNet, благодаря своей устойчивости используется для развёртывания защищённых виртуальных частных сетей. Secret Net находится на некоторых устройствах и служит для защиты информации и контроля доступа к данным;

— Платформа ESXi используется для создания и управления виртуальными машинами (VM) на серверах. Данная платформа эффективно управляет ресурсами компьютера, благодаря чему есть возможность запускать сразу несколько виртуальные машины на одном сервере. Из-за этого упрощается администрирование инфраструктуры.

1.2. Разбор существующих решений на рынке

Для того чтобы начать разработку приложения, необходимо изучить существующие программы, которые решают поставленные задачи. В современном мире большинство сотрудников компаний рано или поздно, с увеличением количества решаемых задач, начинают испытывать трудности с распределением собственного времени. Тогда компании создают свои собственные приложения, для решения этой проблемы или начинают пользоваться уже существующими. Сами приложения, которые решают проблему распределения времени с каждым годом начинают пользоваться большим спросом. Некоторые компании создают приложение для собственной корпоративной сети, другие создают приложения с открытым исходным кодом и выкладывают их в общую сеть. Разбор существующих программ поможет в проектировании собственной системы, в заимствовании некоторых функций для web-приложения.

1) LeaderTask – российское приложение для планирования задач и таск-менеджер по выполнению поручений. Приложение доступно на всех платформах: Windows, Linux, macOS, Android, IOS.

Приложение обладает широким перечнем функций, которые помогут в планировании и распределении времени (рис. 1.1).

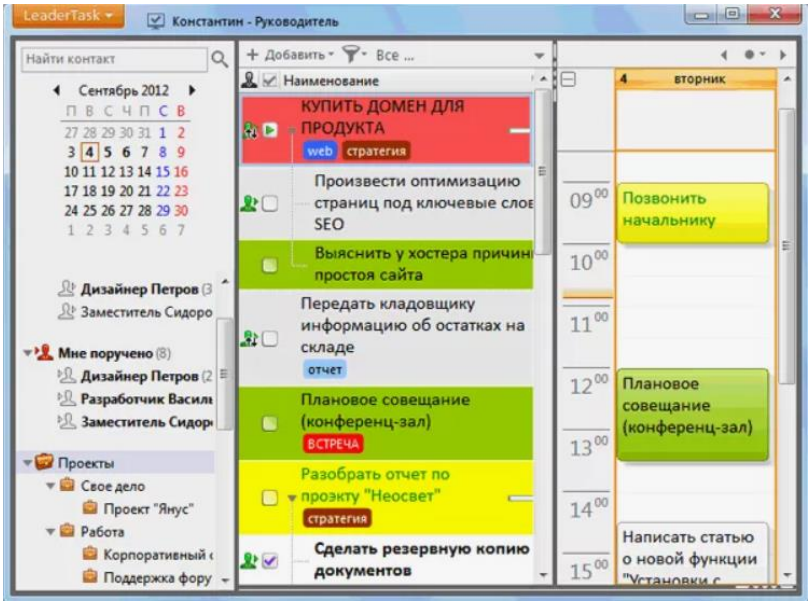


Рисунок 1.1 – Приложение LeaderTask.

Приложение позволяет добавлять чек-листы и файлы, выделять важные задачи, устанавливать им сроки выполнения. Имеется функция уведомления пользователя. К плюсам также можно отнести: широкую кастомизацию, что позволяет настроить приложение под себя, возможность работы в группах, что повышает навыки коммуникации в команде и помогает более эффективно решать поставленные цели.

К минусам можно отнести то, что приложение является платным. Хотя и существует бесплатная версия, но она лишена некоторых функций. Также разница в функционале отличается в зависимости от того, на какой платформе запущено приложение.

2) Weeek – приложение позволяет эффективно работать в команде. Задачи группируются по типу, например встреча с кем-то, или выполнение какого-либо действия. Также задачи возможно группировать по приоритету и дате.

Авторизацию в приложение можно пройти через аккаунты социальных сетей или браузеров (Google аккаунт, Yandex аккаунт, Microsoft Edge аккаунт).

Приложение является многопользовательским, через браузер ПК или с помощью телефона (рис.1.2).

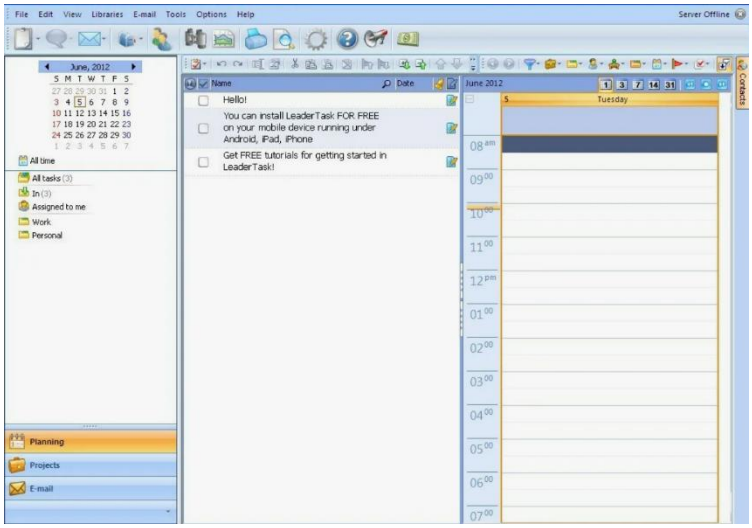


Рисунок 1.2 – Приложение Weeek.

Одним из плюсов данного приложения является то, что оно считает прогресс выполнения задачи в процентах, что существенно помогает в ориентировании и распределении времени. Имеется функция уведомлений, которая присылает их на аккаунт почты, в самом приложении, социальные сети, таких как Telegram и Vk.

Присутствует предварительная оценка времени, которое анализирует и показывает тот временной промежуток, которой необходим для решения задачи. Добавление тегов позволяет быстрее ориентироваться в приложении и в самих задачах. В приложении присутствует функция прикрепления файлов из различных облачных сервисов или с локальных папок.

Приложение является платным, но имеется бесплатный аналог, в котором присутствует недостаток. Он заключается в ограничении добавления проектов и участников. Так как приложение ориентировано на командную работу, это ограничение может существенно сказаться на качестве выполнения задач.

3) MyLifeOrganized – данная программа позволяет разбивать задачи на подзадачи, подробно настраивать каждую из них. Приложение является многофункциональным, в нём можно настраивать разные шаблоны под различные задачи (рис. 1.3).

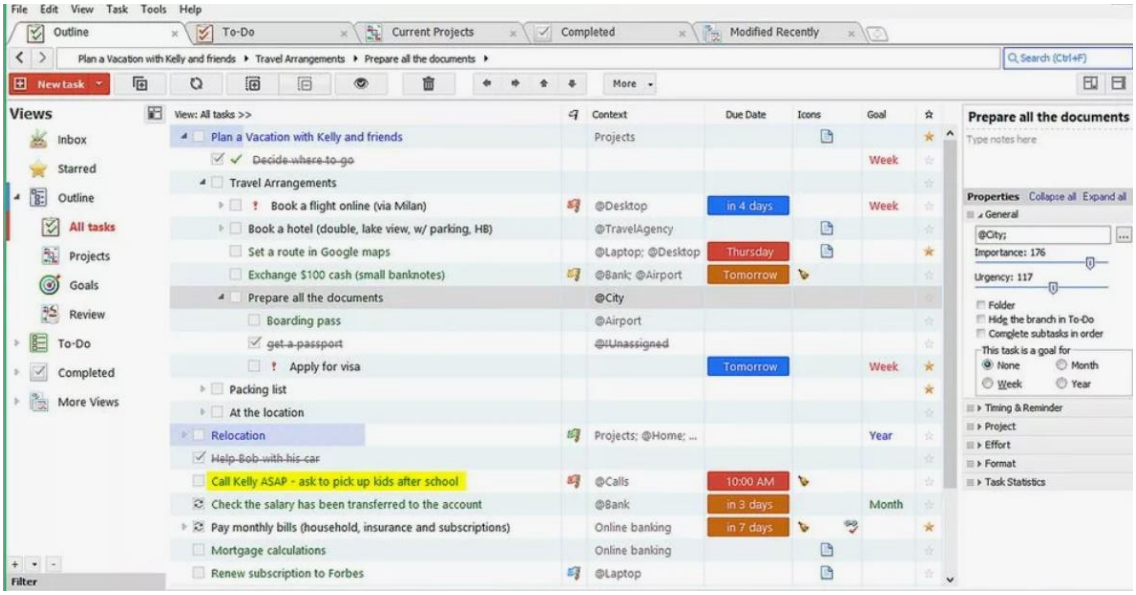


Рисунок 1.3 – Приложение MyLifeOrganized.

Имеется возможность разбивать на задачи на фильтры, устанавливать их важность, виды решаемых задач, добавлять теги к определенным задачам, позволяющих быстрее ориентироваться в них.

Присутствует функция добавления геолокации. Графический календарь позволяет удобно отслеживать созданные задачи, сроки их завершения и статус. С учётом того, что в этот приложении присутствует гибкая и многофункционально настраиваемая иерархия древа задач, календарь является необходимым модулем для ориентирования в приложении.

Имеется бесплатная и платная версия. В бесплатной версии сильно урезан функционал приложения.

4) Google Календарь – сервис, разработанный компанией Google. Является самым популярным сервисом, которым пользуются огромное количество компаний по всему миру. Google календарь имеет обширный функционал, позволяющий эффективно работать с задачи и распределять время. Сервис

позволяет создавать и управлять задачами, организовывать время встречи, добавлять других участников через электронную почту путём отправления уведомления. Присутствуют функции напоминания, которые присылаются по почте или в аккаунты социальных сетей, или через Push-уведомления. Имеются функции группировки задач по их видам, важности (рис. 1.4)

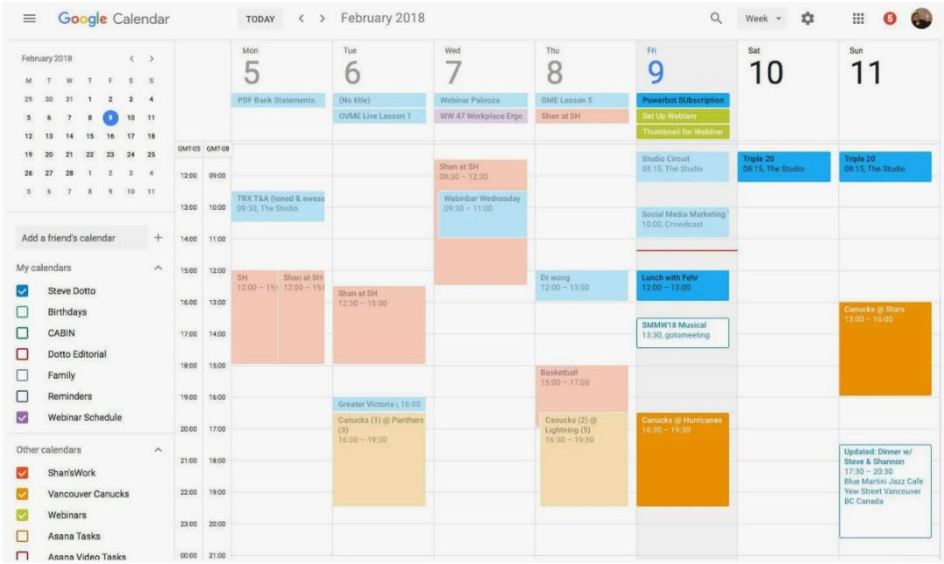


Рисунок 1.4 – Google календарь.

Работа с календарем происходит через веб-интерфейс, который можно открыть в любом браузере. Данные хранятся на серверах Google, что обеспечивает надежность и сохранность данных, а также бесперебойную работу. Для работы с календарем нужно подключение к интернету. Google Календарь работает на всех системах и устройствах, таких как Android, Windows, Linux, webOS, iOS. Благодаря поддержки на всех системах и огромном количестве языков, сервис имеет большую область применения, ведь им можно пользоваться не только на работе и дома, сидя за персональным компьютером, но в тех местах, где отсутствует возможность работы на ПК или ноутбуке.

Присутствует возможность управления задачами при помощи использования горячие клавиши, поддержка одновременной работы с несколькими созданными календарями в аккаунте, а также выдачи прав другим пользователям сервиса на просмотр и редактирование календарей, что позволяет

пользователям выбрать наиболее подходящий под его цели и задачи вид приложения.

Благодаря технологии Google Sync, добавлена функция синхронизации с различными мобильными устройствами с помощью бесплатного приложения по работе с электронными письмами и календарями Microsoft Outlook.

Разобрав приложения и сервисы выше, а также изучив технологии и инструменты, на которым они были разработаны, можно составить представление о том, каким должно быть разрабатываемое web-приложение для его эффективного использования в транспортной отрасли, как в плане графического интерфейса, так и в плане функционала.

1.3. Анализ технологий и инструментов разработки web-приложений

Чтобы разработать web-приложение, которое будет отвечать всем поставленным требованиям, необходимо изучить и проанализировать, какие технологии и инструменты разработки существуют на данный момент. Их изучение, в том числе анализ всех достоинств и недостатков, а также особенностей реализации, позволит выбрать наиболее подходящие технологии для реализации проекта.

Главной технологией разработки как web-приложения, так и любых других программ, сайтов, является язык программирования. Созданное приложение можно разделить на три составляющие:

— Backend (серверная часть);

— Frontend (клиентская часть);

— База данных.

Backend называется тот код приложения, который отвечает за её бизнес логику. Это работа с данными внутри программы, их валидация, перемещение, изменение и расчёты. Эти процессы обычный пользователь не видит.

Frontend отвечает за интерфейс приложения, с которым пользователь взаимодействует с приложением.

В базе данных хранятся различные данные и переменные, например данные о пользователях.

Взаимодействие компонентов происходит следующим образом: Пользователь, через графический интерфейс, делает запрос к веб-приложению. Frontend посылает созданный запросов в серверную часть приложения (Backend).

Серверная часть, тесно взаимодействую с базой данных, обрабатывает полученные запрос, сохраняет, удаляет или предоставляет из базы данных необходимые данные и передает их в Frontend часть приложения, которая, обновляя интерфейс, предоставляет пользователю необходимые данные.

Для разработки Backend составляющей web-календаря нужно рассмотреть следующие языки и их веб-фреймворки:

Python – это высокоуровневый и интерпретируемый язык программирования, являющийся самым популярным языком за счет огромного количества плюсов, по сравнению с другими языками. Он отличается простым синтаксисом, благодаря чему, порог вхождения в этот язык программирования довольно низок(рис. 1.5).

Рисунок 1.5 – Язык программирования Python.

Из-за огромного выбора библиотек, каждая из которых решает определенную задачу, на Python можно написать практически любую программу. Чтобы написать web-приложение на языке Python необходимо использовать web-фреймворк Django.

Django является высокоуровневым web-фреймворком на Python, который предоставляет полный набор инструментов по созданию Backend составляющей проекта и обеспечению эффективными методами безопасности от CSRF, XSS и SQL-инъекции.

Этот фреймворк обладает объемной документацией, где описаны все методы решения тех или иных задач, варианты подключения сторонних библиотек. Также у Django имеются встроенные методы аутентификации и маршрутизации [1,12]

К недостаткам web-фреймворка можно отнести:

— Медленную производительность;

— Переменную сложность в изучение определенных аспектов.

На Django было разработаны ряд популярных приложений:

— Instagram;

— YouTube (некоторая часть);

— Mozilla;

— Dropbox.

Ruby – высокоуровневый и объектно-ориентированный язык программирования, обладающий простым синтаксисом, хорошей гибкостью, заключающейся в быстром изменении классов и методов. Также язык обладает активным сообществом, большим количеством библиотек, что значительно упрощает изучение и разработку приложений. На языке пишут скрипты, системы безопасности. Однако, разработка на этом языке требует больших системных ресурсов, поэтому на устройствах, обладающих малой мощностью, разработка на Ruby будет невозможна. Отсутствие совместимости версии может вызвать

серьезные проблемы с разработкой приложения при выходе нового обновления (рис.1.6)

Рисунок 1.6 – Язык программирования Ruby.

Web-фреймворк Ruby on Rails, использующийся для написания web-приложений, позволяет проводить разработку более быстрее, чем на многих других языках и фреймворках, за счёт большого количества готовых решений и библиотек. Благодаря хорошей автоматизации в приложении встречается меньше повторяющего кода, что помогает в ориентировании и перемещении в коде. Сокращенное количество настроек конфигурации позволяет разработчикам больше внимания концентрировать на разработке бизнес-логики приложения [4].

Недостатками Ruby on Rails является низкая производительность по сравнению с другими похожими фреймворками, что сказывается на системах с высокой нагрузкой, а также требование больших технических и вычислительных ресурсах. Большое количество абстракций могут вызвать сложности в изучении и реализации определенных методов и библиотек.

На Ruby on Rails написаны:

— GitHub;

— Shopify;

— Basecamp.

C# - объектно-ориентированный язык программирования, который в комбинации фреймворком .NET или .NET Core используется для создания серверной части веб-приложений. Язык является типизированным, что означает снижение количества ошибок в коде, что в совокупности с высокой производительностью позволяет быстро реализовывать код. (рис. 1.7)

Рисунок 1.7 – Язык программирование C#.

IDE, в которой чаще всего пишут код на языке C#, а именно Visual Studio, позволяет быстро тестировать и находить ошибки в коде, что в сочетании с типизацией самого языка, позволяет сосредоточиться на разработке. К достоинствам данного языка можно также отнести современные технологии поддержки асинхронного программирования, ускоряющее разработку и запуск приложения.

Недостатками является большое потребление ресурсов в некоторых случаях, и преимущественная разработка на платформе Windows.

Популярные разработанные приложения и сервисы:

— Dell;

— Stack Overflow;

— Walmart;

— Microsoft Teams.

Изучив необходимые технологии для разработки Backend части web-приложения, нужно проанализировать способы реализации Frontend части. Для большинства web-приложений используют определенный стек технологий: JavaScript, Html, Css.

Язык программирования JavaScript – динамический, объектно-ориентированный язык программирование. Благодаря сочетанию JavaScript с Html и Css на графический интерфейс можно добавить интерактивные элементы и анимации. JavaScript не компилируемый язык, поэтому весь код выполняется в браузере или на сервере. Большим преимуществом является быстрая обработка операций и событий (рис. 1.8).

Рисунок 1.8 – Язык программирования JavaScript.

Имеет множество фреймворков и библиотек, самыми часто используемыми являются: React, Angular, Vue.js. Благодаря таким инструментам, как async и await, позволяет выполнять задачи в фоне, что отображается на более высокой эффективности работы с данными, такими как чтение, запись, а из-за того, что язык работает и ведет себя одинаково на разных платформах, то есть является кроссплатформенным, JavaScript является основным языком программирования для работы с веб-интерфейсом приложения, обеспечивающий надёжное создание динамических, а также интерактивных графических интерфейсов [8].

Недостатки JavaScript:

— Проблема с безопасностью: XSS и CSRF являются наибольшей угрозой утечки данных или заражением устройства пользователя. Причина

этому, что JavaScript – клиентский язык, а это означает, что язык имеет доступ к данным пользователя;

— Слабая типизация: из-за слабой типизации языка часто возникают ошибки, которые тяжело обнаружить и исправить.

Для разработки пользовательского интерфейса необходимо использовать языки разметки, Html и Css.

Язык разметки Html отвечает за разработку структуры веб-приложений и сайтов. Все заголовки, текст, поля, кнопки, блоки, и все остальные элементы, что видит пользователь на экране, написано на этом языке разметки. Синтаксис языка состоит из тегов, которые должны быть расположены в строгой иерархии, например, между тегами <body> и </body> содержится весь контент страницы, между тегов <title> и </title> пишется название страницы.

Css является языком стилей, служит для описания внешнего вида html. При помощи css можно стилизировать различные блоки, текст, ссылки по своему усмотрению. Кроме стилизации и использовании таблицы цветов и шрифтов, на этом языке можно менять расположение элементов относительно друг друга, что помогает создать именно такой интерфейс, который задумывали разработчики.

На языке Css можно писать в файле Html, но в некоторых случаях это может вызвать затруднение при перемещении и ориентировании в коде. Поэтому, обычно создают отдельный Css-документ, который затем подключается к документу Html с помощью тега <link>. Благодаря этому тему также можно подключить стили не только из локальной среды, а также из общей сети, достаточно в теге <link> вставить ссылку из интернета на необходимый стиль.

Также можно написать стиль для определенного элемента прямо в теге этого элемента, например .

Проанализировав технологии для разработки Frontend составляющей веб-приложения, необходимо рассмотреть базы данных, в которых будут хранится все переменные и данные, используемые приложением.

СУБД – система управления базами данных. От правильного выбора базы данных зависит эффективность обработки и хранения самих данных.

Базы данных бывает двух типов: реляционные и нереляционные.

Реляционными базами данных называют такие базы, в которых существующие таблицы имеют связи друг с другом (рис. 1.9)

Рисунок 1.9 – Реляционная база данных.

Таблицы в этих СУБД связаны между собой с помощью ключей. Из-за того, что данные в них предоставлены в понятном табличном виде, что обеспечивают гибкость и эффективность использования, реляционные базы данных используются в большинстве сервисов и приложений.

Самыми распространенными реляционными СУБД являются:

— MySQL;

— PostgreSQL;

— SQLite;

— Oracle Database;

— Microsoft SQL Server.

Нереляционными базами данных называют базы, которые используют модель хранения данных, при которой под каждое значение используется его конкретное требование хранения. Таким образом, данные обычно организованы в виде документов, графов, столбцов или ключ-значение.

Нереляционные базы данных бывают двух типов: сетевые и иерархические (рис. 1.10).

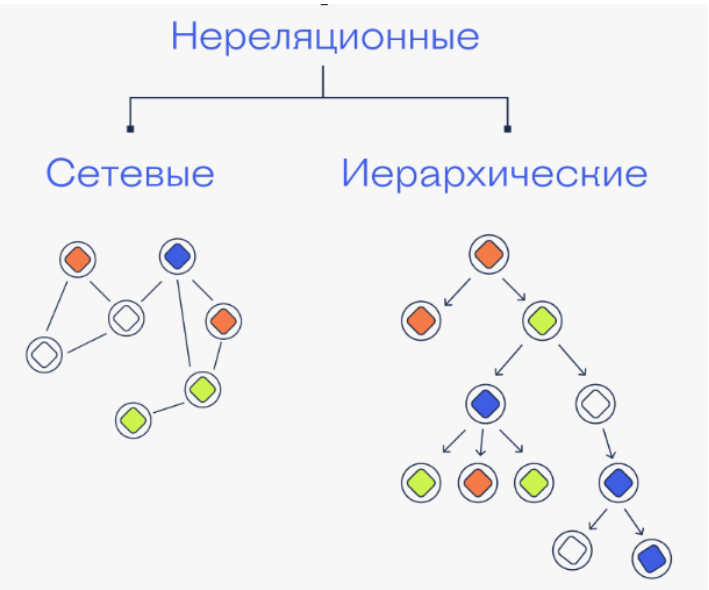


Рисунок 1.10 – Типы нереляционных баз данных.

Иерархические базы данных – это базы, которые моделируют данные в виде древовидной иерархии. Эта структура состоит в наследовании, при котором имеется родительская запись, от которой наследуются другие записи, их называют дочерними записями (один ко многим). Благодаря иерархичному виду базы, возможна быстрая и простая навигация по базе данных, а также легкий доступ к данным. Однако, из-за того же иерархичного вида возникают проблемы с изменением самой структуры данных, поэтому, при попытке изменения, придется переделывать большую часть иерархии. Также эта база не подходит для тех приложений, где данные имеют более сложные связи друг с другом, например, база не будет эффективна для тех приложений, где отношения между данными имеют структуру многие ко многим [6].

Обычно иерархическую модель используют для электронных каталогов и систем реестров, например Windows Registry.

Сетевые базы данных решают проблему с созданием более сложных связей между данными, благодаря чему у каждой записи могут быть несколько родительских и дочерних записей.

Однако из-за сложных структур возникают трудности в навигации и работе с данными.

Сетевые базы данных обычно используют в системах управления проектами и структурами.

Для написания кода необходимы инструменты разработки, IDE, то есть интегрированные среды разработки. IDE – программные пакеты, предоставляющие все необходимые инструменты для работы с кодом.

IDE включает:

— Редактор кода;

— Компилятор или интерпретатор;

— Систему контроля версий;

— Инструменты для отладки;

— Другие функции.

PyCharm — это мощная IDE, в которой разрабатывают приложения на языке Python. Благодаря специализации на одном языке программирования, PyCharm имеет большой функционал, позволяющих настраивать среду разработки и упрощает саму разработку приложений. Используя интеграцию с веб-фреймворками, такими как Django, является хорошим выбором для реализации веб-приложений.

Отличный редактор кода и пошаговая отладка позволяют быстро исправлять синтаксические и логические ошибки в коде. Поддержка таких библиотек, как NumPy, SciPy, Matplotlib, позволяет производить приложения, требующие сложных математических вычислений. Поддержка баз данных, PostgreSQL, MySQL, SQLite, а также фреймворков для тестирования (unittest, pytest) позволяет разрабатывать полноценные приложения на Python, которые будут отличаться высокой надежностью и безопасностью [11,17].

Visual Studio – интегрированная среда разработки, предназначенная для разработки приложений на множестве языков программирования, таких как: Python, JavaScript, C++, C, C#, Ruby, Java. Данная IDE служит для написания различных программ и сервисов, игр, мобильных приложений и веб-

приложений, поскольку предлагает множество методов и инструментов разработки.

Встроенная поддержка Git, хороший текстовый редактор, позволяющий быстро находит и исправлять синтаксические и логические ошибки в коде, интеллектуальный рефакторинг и мультиплатформенность делают этот IDE одним из наиболее популярных и продвинутых инструментов разработки приложений. Благодаря продвинутой системе отладки, включающую в себе контроль точек остановки и просмотр памяти, эта среда разработки является хорошим выбором для написания программ и ПО на C++, поскольку на этом языке присутствуют значительные сложности с управлением памятью, что может привести к ошибкам в работе приложения.

У этой IDE есть недостатки, такие как высокие системные требования, что исключает использование программы для работы на слабых устройствах. Для неопытных разработчиков могут возникать сложности в ориентировании по IDE, поскольку она предоставляет возможность писать множество проектов различного типа на разных языках программирования, существует большое количество функций и методов.

Анализ существующих технологий разработки веб-приложений позволит точечно определиться с инструментами и технологиями разработки для создания веб-приложения. Тщательное изучение интегрированных сред разработки позволит сократить время на разработку приложения и предоставить инструменты тестирования и отладки веб-приложения.

1.4. Архитектура клиент-сервер

Для разработки веб-приложения необходимо определиться с архитектурой сетевого распределения нагрузок. Выбор архитектуры позволяет правильно распределить нагрузку между приложением и устройствами, что значительно скажется на производительности и безопасности. Наиболее эффективной

архитектурой, для разработки и функционирование веб-приложения, а также для правильного распределение вычислительных мощностей является архитектура клиент-сервер.

В данной архитектуре процесс распределения происходит между сервером клиентом. Процесс происходит по формуле запрос-ответ, когда клиент делает запрос на сервер, который обрабатывает этот запрос и возвращает его обратно. Сервер может получать запросы от множества разных клиентов за определенный промежуток времени (рис. 1.11).

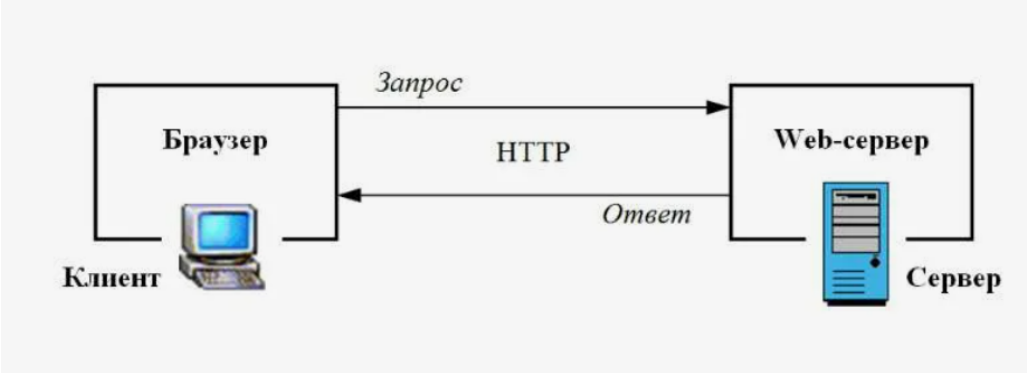


Рисунок 1.11 – Архитектура клиент-сервер.

Клиент может быть трех типов: сильным, слабым, распределенным. При сильном клиенте вычисления происходят на устройстве клиента, такое распределение называется сильный клиент- слабый сервер. При слабом клиенте, на его устройство поступает только конечная информация, а сами вычислительные процессы происходят на стороне сервера. При распределенных клиентах, сами клиенты взаимодействуют с разными серверами, что обеспечивает более эффективную надежность и безопасность [2].

Преимуществом данной архитектуры является наибольшая защищенность данных, по сравнению с клиентской частью. Также снижается потребление ресурсов у клиента, поскольку большинство вычислений происходят на сервере.

Из-за того, что большая часть вычислений производится сервером, стоимость и поддержка оборудования может быть довольно высокой. Если сервер будет неработоспособен, то это может привести к неработоспособности

всей сети. Чтобы решить эту проблему, существует несколько уровней архитектуры:

— Одноуровневая архитектура клиент-сервер. Операции при этом уровне выполняются на стороне клиента, не используя никаких узлов, что означает нахождение и клиента и сервера в одном пространстве;

— Двухуровневая архитектура клиент-сервер. В этом уровне присутствует сетевой узел, куда клиент отправляет запрос. Сам сервер работает с бизнес-логикой приложения и с БД.

— Трехуровневая архитектура клиент-сервер. Разбивается на три уровня, уровень клиента, уровень бизнес-логики и уровень баз данных. Данное распределение обеспечивает дополнительную безопасность данных.

— Многоуровневая архитектура клиент-сервер. Несколько уровней как клиента, так и сервера, каждый из них отвечает за определение конкретных функций. Распределяет нагрузку между серверами.

ПРОЕКТНЫЙ РАЗДЕЛ

2.1. Основные стадии процессов разработки приложения и их взаимодействие

Для эффективного выполнения задачи, а именно разработки Web-приложения «Календарь задач» необходимо определиться с основными стадиями разработки приложения, а также установить взаимосвязь между ними.

Создание диаграммы IDEF3 позволяет предоставить план разработки Web-приложения (рис. 2.1)

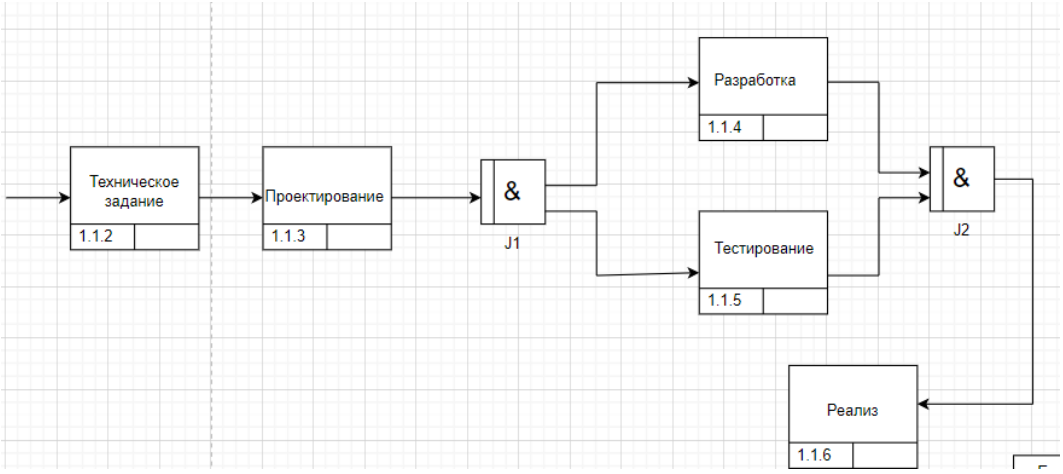


Рисунок 2.1 - Диаграмма IDEF3.

Анализ требований:

Анализ требований будущего приложения является очень важным этапом, исходя из которого будет проектироваться и разрабатываться приложение, проводиться тестирование.

1) Анализ ТЗ;

2) Выбор аппаратных и программных ресурсов;

Проектирование системы:

Проектирование системы – это очень важный этап разработки, на который необходимо упираться при реализации web-приложения. Это позволяет избежать многих ошибок и эффективно использовать время и ресурсы, а также

распределять задачи между разработчиками, согласовывать графика выполнения подзадач:

1) Проектирование архитектуры приложения;

2) Проектирование интерфейса и базы данных.

Разработка и тестирование приложения:

Разработка и тестирование приложения является самым важным этапом, в котором на основе анализа требований и проектирования будет разработано приложение.

Тестирование будет проводиться во время разработки приложения, а также по её окончанию, благодаря чему будут выявлены и устранены ошибки в работе.

1)Разработка backend(бизнес-логики) приложения, подключение базы данных;

2)Создание интерфейса приложения;

3)Соединение интерфейса приложения с backend;

4) Тестирование каждой части приложения на корректность работы;

5) Выявление и исправление ошибок.

Мониторинг и поддержка:

Данный этап будет поддерживать жизненный цикл программы. Мониторинг позволяет выявлять и исправлять недочеты и выпускать обновления для продления жизненного цикла приложения [18].

Обратная связь позволяет анализировать желания пользователей, а также точечно исправлять ошибки.

1)Мониторинг и анализ работы системы, трафика;

2)Поддержка и обновление приложения, выявление и устранения ошибок.

2.2. Техническое задание

Техническое задание направлено на разработку Web-приложения «Календарь задач», которое будет применяться в транспортной отрасли для

эффективной регулировки и распределению времени сотрудников организации. Анализ требований позволит точно определиться с требованиями к разработке программы.

1. Наименование разрабатываемого программного продукта:

"МуТ"

2. Назначение разрабатываемого программного продукта: Web-приложение предназначено для планирования и отслеживания задач, что способствует поддержанию высокой эффективности и пунктуальности сотрудников организации, а также личному и профессиональному росту.

3.Основные функции web-приложения:

— возможность регистрации и авторизации пользователей;

— добавление задач с установкой срока выполнения;

— отслеживание созданных задач, а также функции изменение статуса выполнения и удаления;

— визуализация задач путём создания формы графического календаря, в котором будут отображаться созданные задачи по дате.

4. Требования к программному продукту:

4.1. Технические требования [5]:

— операционные системы Windows, Linux;

— веб-браузер с подключенным интернетом.

4.2. Требования к безопасности приложения:

— резервное копирование данных;

— защита данных от несанкционированного доступа.

4.3. Требования к интерфейсу:

— простой и понятный для большинства пользователей.

5. Требования к документации:

— руководство пользователя;

— политика конфиденциальности.

6.Стадии и этапы разработки:

— Техническое задание;
— Проектирование web-приложения;
— Разработка;
— Тестирование;
— Внедрение и сопровождение web-приложения.

2.3. Проектирование веб-приложения

2.3.1 Обоснование выбранных инструментов и технологий для разработки

Основываясь на анализе разобранных приложений и описании их архитектуры и функциональности, а также разборе существующих технологий разработки, необходимо выбрать технологии и инструменты, которые будут использоваться при реализации приложения.

Django будет использован для создания бизнес-логики приложения. Сам язык, поскольку предназначен для создания веб-приложений, имеет обширные методы и инструменты разработки [3].

По причине того, что разработка будет осуществляться на язык Django, была выбрана интегрированная среда разработки PyCharm. Поскольку эта IDE предназначена для создания приложений на языке Python и предоставляет огромный функционал как разработки, тестирования и отладки кода, делает её наилучшим выбором среди других IDE [17,20].

Чтобы сделать интерактивным как сам календарь, так и блоки с кнопками, в которых будет происходить работа с задачами, необходимо написать скрипты, реализующие эти функции. Для этого будет использован JavaScript. Без использования этого языка невозможно создание интерактивного веб-приложения [9,10].

SQLite — система управления базами данных, которая предоставляет все необходимые возможности для локального хранения данных. Подходит для разработки, поскольку реализуемое веб-приложение не требует высоких

нагрузок. Через модуль sqlite3 будет происходить интеграция с Django, благодаря чему упрощается взаимодействие с базой данных.

2.3.2 Разработка архитектуры системы

В клиентской части приложения должен быть возможность просмотра и отслеживание по времени задач, а также работа с ними в формате понятного и простого интерфейса. Нужно предоставить пользователю возможности по добавлению, изменению и удалению задач.

Чтобы добавить возможность работы с задачами, необходимо разобраться с тем, как будет работать функционал внутри приложения. Для этого необходимо нарисовать диаграмму последовательности.

Добавление задачи будет происходить следующим образом (рис. 2.2):

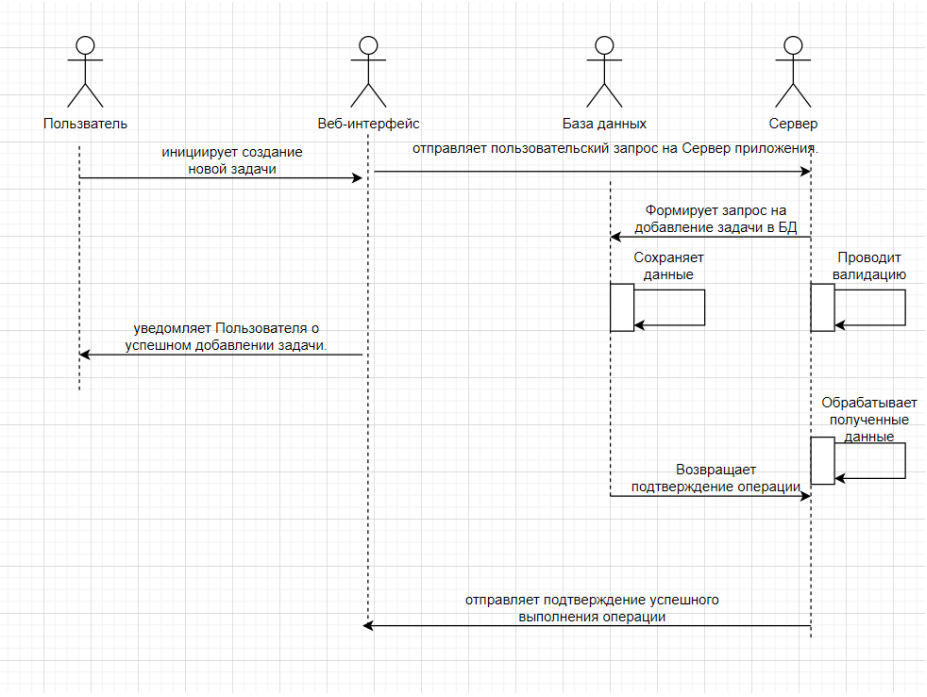


Рисунок 2.2 – Диаграмма последовательности.

1) Пользователь инициализирует создание новой задачи, указывая имя задачи и дату;

2) Веб-интерфейс отправляет этот запрос на сервер, где он обрабатывается и формирует запрос на добавление задачи в БД;

3) БД сохраняет данные и отправляет их обратно на сервер;

4) Сервер отправляет подтверждение выполнения операции веб-интерфейсу;

5) Веб-интерфейс уведомляет пользователя об успешном добавлении задачи.

Далее необходимо разработать структура взаимодействия пользователя с задачами. Эти действия осуществляются следующим образом: (рис. 2.3):

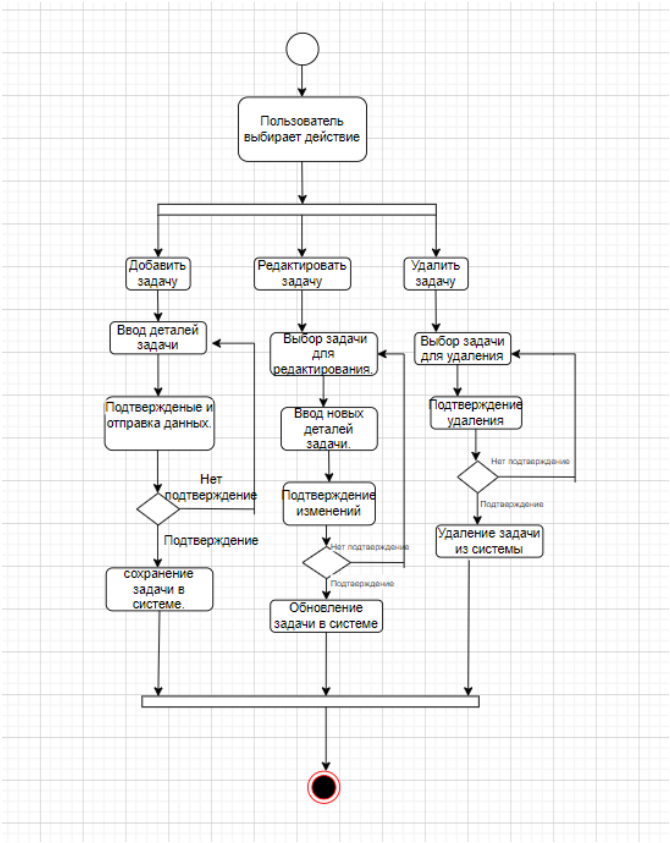


Рисунок 2.3 – Диаграмма деятельности.

Добавление задачи:

— Пользователь вводит данные о задаче (название, дату);

— Нажимает кнопку Добавить, тем самым подтверждает операцию и отправляет данные на сервер.

Изменение задачи:

— Пользователь выбирает нужную задачу;

— Изменяет результат выполнения задачи (Завершено/Не завершено);

Удаление задачи:

— Пользователь выбирает нужную задачу;

— Удаляет выбранную задачу с помощью кнопки.

Основываясь на диаграммах деятельности и последовательности можно определить, как будет взаимодействовать Web-приложение внутри себя, а также с пользователем [7].

Также необходимо добавить аутентификацию пользователя, при прохождении которой, пользователя переведет в основную часть Web-приложения «Календарь задач» для реализации этой функции будет использован стандартный метод, который будет создан при создании проекта на Django.

2.3.3 Проектирование интерфейса

Необходимо разработать основные интерфейсы экрана для приложения. Для решения этой задачи необходимо использовать Html и Css.

Используя Html необходимо произвести разметку блоков и определиться с их размещением на экране главного интерфейса приложения.

На главном интерфейсе будет находится:

1) Календарь. С помощью библиотеки FullCalendar будет создан интерактивный календарь с динамическим отображением задач, находится он будет в верхней части экрана. Также будут добавлены кнопки, функция которых состоит в перемещении между месяцами, они будут расположены ниже самого календаря;

2) Далее будет расположен блок с добавлением задач. В нём будут расположены поля: название задачи и дата, а также поле для дополнительной информации. В этом блоке необходимо добавить кнопку, при нажатии на которую, созданная задача будет сохраняться и добавляться в блок созданных задач и в календарь;

3) Блок с созданными задачами, в котором пользователь сможет увидеть существующие задачи, а также иметь возможность изменить статус задачи, а также удалить выбранную задачу как из этого блока, так и из календаря.

4) Нужно добавить кнопку выхода из аккаунта, которая будет располагаться в самой верхней части экрана.

Для реализации задуманного, помимо Html и Css, необходимо использовать язык программирования JavaScript для написания скриптов, которые и обеспечат данный функционал.

Чтобы пользователь смог перейти на главный интерфейс, необходимо создать окно авторизации. Которые будет состоять из:

1) Названия Web-Приложения;

2) Двух полей, в котором пользователю нужно ввести логин и пароль;

3)Кнопки, которая, в случае успешной аутентификации переведет пользователя с интерфейса аутентификации на главный интерфейс Web-приложения «Календарь задач».

Основываясь на этих данных, можно создать диаграмму, которая будет отображать, как пользователь сможет взаимодействовать с Web-приложением через интерфейс (рис. 2.4):

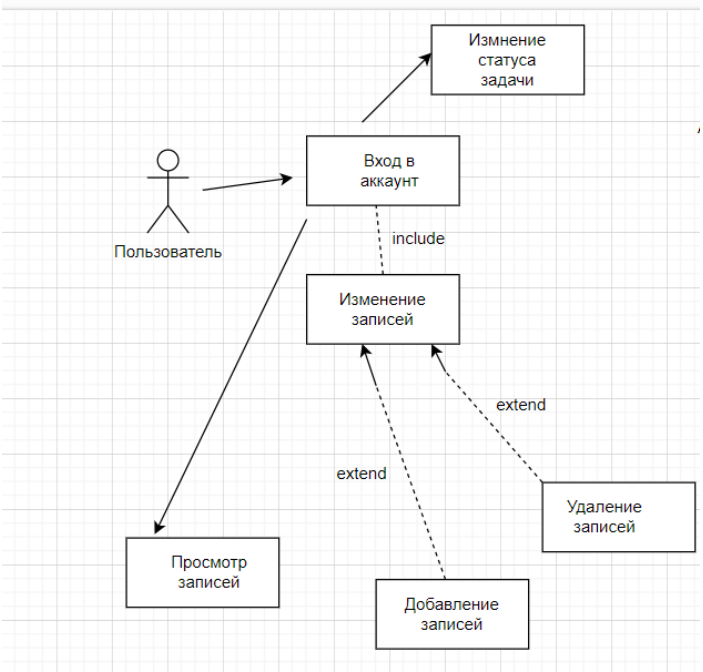
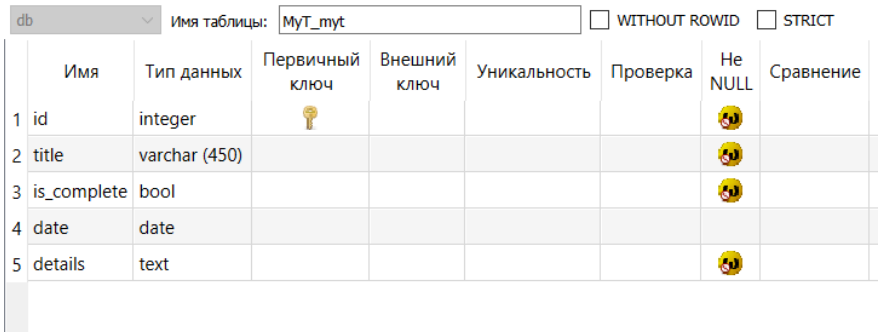


Рисунок 2.4 – Диаграмма вариантов использования.

2.3.4 Анализ и проектирование структуры базы данных

База данных будет автоматически создана при создании проекта на Django. Для открытия и прочтения базы данных будет использован SQLite. В этой базе будут созданы автоматически несколько таблиц. Для реализации Web-Приложения понадобятся две таблицы. Первая таблица будет создана в ручную называться именем проекта и содержать в себе данные по задачам, вторая, созданная автоматически, будет содержать логины и пароли.

В таблице данных будут следующие параметры (рис.2.5)


db Имя таблицы: MyT_myt WITHOUT ROWID STRICT

	Имя	Тип данных	Первичный ключ	Внешний ключ	Уникальность	Проверка	Не NULL	Сравнение
1	id	integer						
2	title	varchar (450)						
3	is_complete	bool						
4	date	date						
5	details	text						

Рисунок 2.5 – Таблица данных.

ID (Уникальный ключ) тип данных integer – будет использован для присвоение уникального ключа для каждой задачи;

title (Название задачи) тип данных varchar (450) – используется для хранения названия задачи;

is_complete (Задача завершена/ не завершена) тип данных bool – эта ячейка имеет bool тип данных, означающее только значение True или False. Используется для обозначения статуса созданной задачи;

data (дата) тип данных date – в этой ячейке хранится дата, которая указываются при создании задачи;

details (Дополнительная информация) тип данных text – будет хранится дополнительная информация о задаче.

Таблица сотрудников будет содержать следующие поля:

— ID (Уникальный ключ) тип данных integer;

— login тип данных varchar (450);

— password тип данных varchar (150).

2.3.5 Методы тестирования

Этап, при котором происходит проверка на наличие ошибок в Web-приложении для обеспечения качественного функционирования программы. Тестирование необходимо производить во время разработки самого приложения, так и после завершения, чтобы выявить и устранить все ошибки перед реализмом приложения.

Для обеспечения наилучшего результата будут проведены тестирования разного вида, которые затронут проверку как определенных методов и компонентов, так и процессы их взаимодействия. Такой метод позволит более точечно выявлять и исправлять ошибки. При разработке Web-приложений основными ошибками будут являться [13]:

1) Синтаксические ошибки. Возникают при неправильном написании кода, программа может полностью не запуститься, или же не запустится одна или несколько модулей приложения.

2)Ошибки валидации данных. Возникают при обработке базой данных различных переменных. База данных может не вывести на интерфейс нужные данные, или вывести их неправильно.

3) Ошибки конфигурации. Возникают при неправильном указании ссылок к файлам или библиотекам. А также полное их отсутствие, что приведет к тому, что приложение не будет видеть часть кода, в котором эти ссылки или библиотеки используются.

4) Ошибки интеграции. Возникают при объедении модулей web-приложения, из-за чего на пользовательском интерфейсе данные и блоки могут выводиться неправильно.

Чтобы избежать данных ошибок, будет проведены следующие тестирования: Юнит-тестирование, интеграционное тестирование, тестирование пользовательского интерфейса (UI), отладочный режим в IDE.

1) Юнит-тестирование – происходит тестирование каждого модуля и проверку отдельных компонентов. Обеспечивает корректную работу каждого модуля перед его интеграцией [14].

2) Интеграционное тестирование – следующий шаг, который проверяет взаимодействие между модулями после их интеграции.

3) Тестирование пользовательского интерфейса (UI) – тестирование пользовательского интерфейса. Направлено на то, чтобы обеспечить корректность вывода данных на экран пользователя, как было задумано разработчиком, а также на функциональность и удобство для конечного пользователя.

4) Отладочный режим в IDE – это тестирование проводится в самой IDE, используется для обнаружения ошибок в самом коде. Благодаря отладке можно также проследить корректность пути следования и обработки данных до конечного назначения.

2.4. Разработка и тестирование веб-приложения

2.4.1 Разработка бизнес-логики web-приложения

Для начала необходимо установить фреймворк Django в PyCharm. Для этого в консоли прописывается -pip install Django.

После создается проект: -python manage.py startproject

Затем в проекте нужно создать приложение: -python manage.py startapp.

Для реализации проекта основными файлами будут служить:

— views.py;

— models.py;

— forms.py;

— setting.py

— два файла urls.py. Один находится в папке проекта, второй в папке приложения.

В файле models.py нужно создать класс, в котором указываются все основные переменные для задач (рис. 2.6):

Рисунок 2.6 – Создание models.py.

В переменной title хранится название задачи, в is_complete содержится bool значение, отображающее статус выполнения задачи, в date хранится дата, в details будет хранится дополнительная информация о задаче. Все эти значения будут хранится в базе данных в таблице MyT.

Затем в файле views.py будут реализована вся бизнес-логика, направленная на работу с переменными и базой данных. В этой файле необходимо реализовать основные методы, которые будут взаимодействовать с задачами, с данными для входа пользователя, сохранением их в базу данных (рис. 2.7).

Рисунок 2.7 – Основные методы.

Метод def index(request) является основной для главной страницы. Он извлекает данных из БД, создает словарь, который будет формировать список задач и с помощью функции render передает данные на главную страницу и производит её рендеринг.

Методы def add_task(request), def update_task(request, myt_id), def delete_task(request, myt_id), def manage_your_time_view(request) работают с задачами, а именно их добавлением, изменением и удалением.

Метод def profile_view(request) отвечает за авторизацию пользователя и его перехода после аутентификации в файл index.html.

В проекте используются два файла urls.py, эти файлы отвечают за последовательность загрузки html страниц.

Urls.py приложения сначала загружает файл, отвечающий за аутентификацию сотрудника, затем основную страницу.

Urlpatterns=[

path('profile', profile_view, name='profile'),

path('', views.index, name='index'),

path('add_task', views.add_task, name='add_task'),

path('update_task/<int:myt_id>/', views.update_task, name='update_task'),

path('delete_task/<int:myt_id>/', views.delete_task, name='delete_task'),

Файл forms.py создает форму в Django для работы с определенной моделью базы данных, устанавливает формат даты и добавляет ввод через календарь (рис. 2.8).

Рисунок 2.8 – Файл forms.py.

Здесь происходит настройка отображения для поля date:

— Говорит браузеру, что date должно отображаться с календарём для выбора даты.

— Устанавливает формат даты как 'год-месяц-день'

— Процесс создания формы дополнительно настраивает поле date, чтобы при вводе проверять, что дата введена в формате 'год-месяц-день'.

2.4.2 Разработка интерфейса главной страницы.

Необходимо создать несколько файлов html и один файл css.

Файле index.html наследуется от файла layout.html, который прописывает ссылки на определенные стили для страницы (рис. 2.9).

Рисунок 2.9 – Файл layout.css.

В index.html находится основная часть кода, которая используется для вывода интерфейса главной страницы. Для создания интерактивного календаря необходимо написать скрипт на языке JavaScript. Сам скрипт создает интерактивный календарь, в котором отображаются созданные задачи, а также добавляет возможность навигации между месяцами с помощью кнопок. Const DaysInMonth принимает текущую дату, а возвращает количество дней текущего месяца. const daysOfWeek представляет собой массив, куда записываются названия месяцев, а функция const renderCalendar отвечает за отображения календаря, путём вывода текущей даты, массива с днями недели. С помощью. daysOfWeek.forEach(day => {calendarElement.innerHTML += `<div class='day'>${day}</div>` выводятся названия недели. На рисунке 2.10 можно видеть реализацию этого скрипта (рис. 2.10).

```
const tasks = {{ tasks|safe }};
console.log("Tasks from server:", tasks); // Для отладки, чтобы увидеть передаваемые задачи

const daysInMonth = (year, month) => new Date(year, month + 1, 0).getDate();
const firstDay = (year, month) => new Date(year, month, 1).getDay();

const calendarElement = document.querySelector(".calendar");
const monthDisplayElement = document.querySelector(".month-display");
const currentTimeElement = document.querySelector(".current-time");

const daysOfWeek = ["Пон", "Втр", "Ср", "Чт", "Пт", "Суб", "Вос"];
let today = new Date();
let currentMonth = today.getMonth();
let currentYear = today.getFullYear();

const renderCalendar = (year, month) => {
    calendarElement.innerHTML = "";

    // Выводим название месяца и год
    monthDisplayElement.textContent = new Date(year, month).toLocaleString('default', {
        month: 'long',
        year: 'numeric'
    });
```

Рисунок 2.10 – Создание календаря на JavaScript.

Прежде чем добавить в скрипт функцию отображения созданных задач на календаре, необходимо создать формы для управления задачами (рис. 2.11).

```
<div style="margin: 5px 25px;padding: 25px;display: inline-block" class="ui container">
   <h1 class="ui center aligned header">Календарь задач MyT</h1>
   <h3 class="test_style">Управляй своим временем!</h3>
   <form class="ui form" method="post" action="{% url 'add_task' %}">
       {% csrf_token %}
       <div class="field">
           <label>Новая задача</label>
           <input type="text" name="title" placeholder="Добавьте названия задачи" required/>
           <input type="date" name="date" required />  <!-- Поле для выбора даты -->
       </div>
       <button class="ui teal basic button" type="submit">Добавить</button>
   </form>
</div>


% for myt in manage_your_time %}
<div style="margin: 0;weight:70px;" class="ui segment">
   <p class="ui big header">{{ myt.title }}</p>
   <div>
       <span>{{ myt.title }} - {{ myt.date }}</span>
       {% if myt.is_complete == False %}
       <span class="ui gray label">Не завершено</span>
       <a class="ui primary basic button" href="{% url 'update_task' myt.id %}">Изменить</a>
       {% else %}
       <span class="ui green label">Завершено</span>
       <a class="ui primary basic button" href="{% url 'update_task' myt.id %}">Изменить</a>
       {% endif %}
       <a class="ui negative basic button" href="{% url 'delete_task' myt.id %}">Удалить</a>
   </div>
```

Рисунок 2.11 – Создание блоков для работы с задачами.

В первом блоке кода прописана контейнер для добавления задач. Реализована эта функция по трём переменным, названию задачи, которая обозначается переменной title, времени, которая обозначается переменной date и для внесения дополнительной информации о задаче, обозначается переменной

details. Второй блок кода, обращаясь к базе данных, реализуют вывод созданных задач в специальную форму с указанием даты, с помощью тега {{ myt.title }} - {{ myt.date }} - - {{ myt.details }} .

При обращении к базе данных {% if myt.is_complete == False %}, сравнивается переменная myt.is_complete. В зависимости от состояния этой переменной, меняется статус завершения задачи, который можно изменить, нажав соответсвующую кнопку. Задачу также можно удалить из базы данных (рис. 2.12)

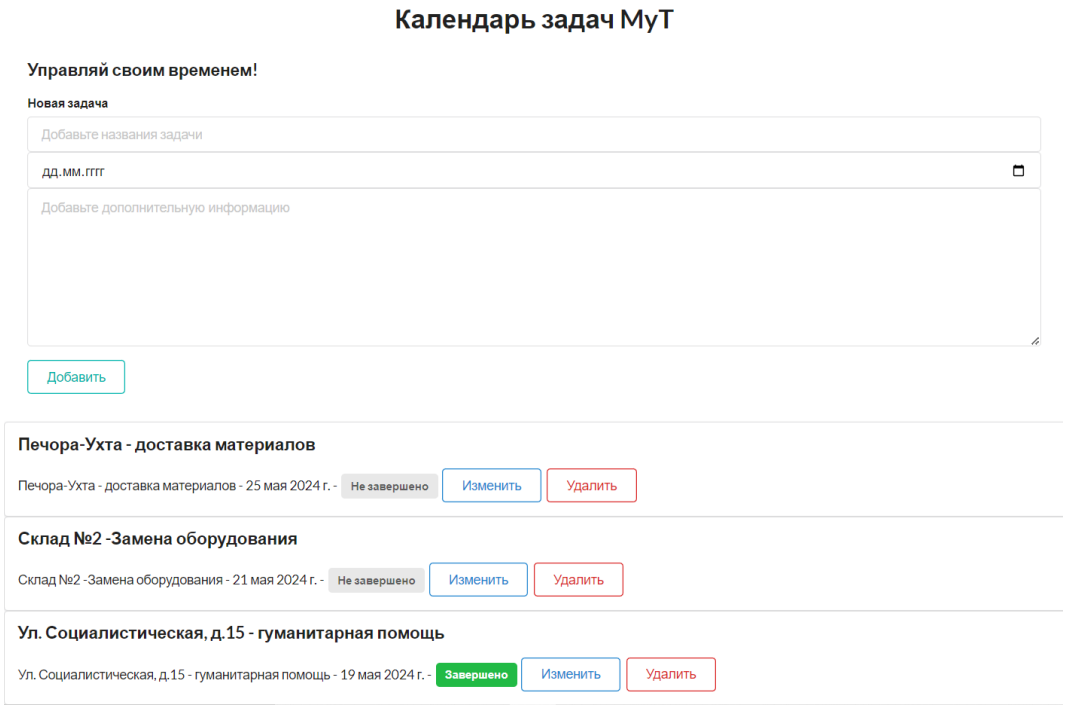


Рисунок 2.12 – Готовый интерфейс для работы с задачами.

Теперь, когда уже создан блок для работы с задачами и сам календарь, необходимо сделать так, чтобы созданные задачи отображались на календаре. Для реализации этой функции в созданный ранее скрипт необходимо добавить метод, при котором будет происходить процесс поиска по дате задачи и передачи её в календарь (рис. 2.13).

```
const taskForDay = tasks.find(task => task.date === formattedDate);
if (taskForDay) {
    const taskElement = document.createElement("div");
    taskElement.classList.add("task");
    taskElement.textContent = taskForDay.title;


    dayElement.appendChild(taskElement);
}

calendarElement.appendChild(dayElement);
```

Рисунок 2.13 – Реализация функции добавление задачи в календарь.

Переменная const taskForDay обращается к списку tasks и ищет задачу под конкретную дату. Затем, когда задача найдена, с помощью метода appendChild добавляет её ко дню, а последняя строчка скрипка на рисунке уже выводит задачу на календарь, отображая в конкретной ячейке (рис. 2.14)

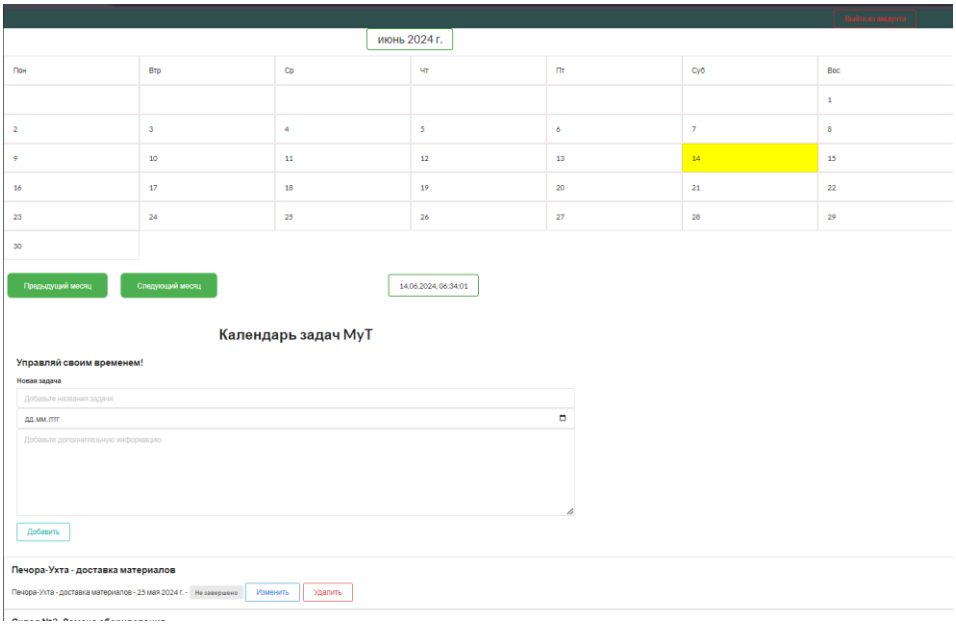


Рисунок 2.14 – Реализованный интерфейс приложения.

При создании проекта была автоматически создана база данных, которую можно открыть, с помощью программы SQLite и убедиться, что все данные сохраняются исправно. При открытии базы данных можно наблюдать большой список таблиц, автоматически сгенерированных Django (рис. 2.15):

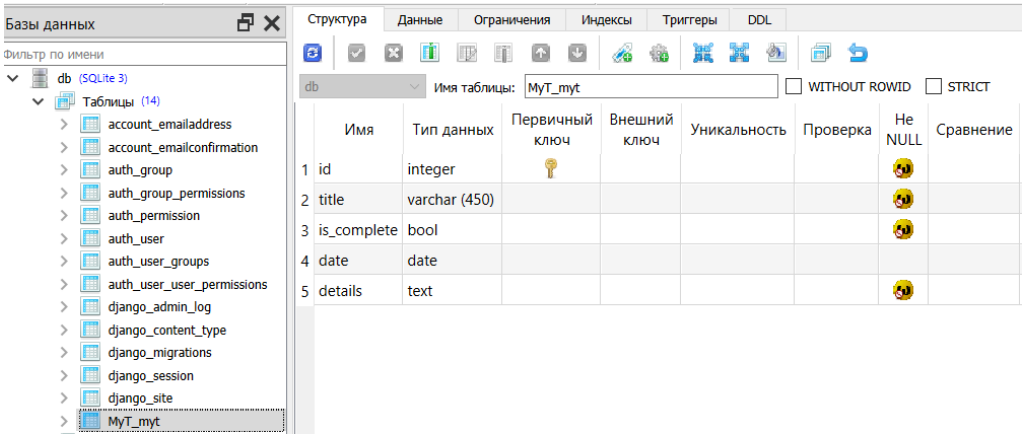


Рисунок 2.15 – База данных.

Таблица MyT_myt содержит в себе данные о созданных и сохраненных задача. Для того чтобы узнать, какие конкретно данные были сохранены, нужно перейти во вкладку «Данные» (рис. 2.16).

id	title	is_complete	date	details
115	Печора-Ухта - доставка материалов	0	2024-05-25	
116	Склад №2 -Замена оборудования	0	2024-05-21	
117	Ул. Социалистическая, д.15 - гуманитарная помощь	1	2024-05-19	

Рисунок 2.16 – Сохраненная информация о созданных задачах.

2.4.3 Разработка окна авторизации пользователя.

Для создания окна авторизации необходимо создать profile.html и profile.css. В html определяется содержание страницы, в css стиль. Далее необходимо добавить ссылки на этот html в файле setting.py.

Команды LOGIN_REDIRECT_URL = reverse_lazy("MyT:index") и LOGOUT_REDIRECT_URL = '/' создают настройку Url, которая используется при перенаправлении на другие страницы после входа или выхода. При авторизации пользователя перебросит на главную страницу.Чтобы избежать непреднамеренного входа в чужой аккаунт веб-приложения, в файле views.py нужно найти функцию «def index» и добавить декоратор «@require_http_methods(['POST'])», что означает, что эта функция, требующая логин пользователя. Чтобы декоратор заработал, нужно импортировать библиотеку django.contrib.auth.decorators.

После прописывания бизнес-логики, нужно создать интерфейс заголовка (рис.2.17).

```
    <script type="text/javascript" href="{% static 'MyT/css/profile.js' %}"></script>
    <title>Title</title>
</head>
<body>

    <div class="container">
        <!-- Контейнер для заголовка -->
        <div class="header-container">
            <div class="header-title">MyT</div>
            <div class="sub-title">Календарь задач</div>
        </div>

        <!-- Контейнер для форм -->
        <div class="form">
            <form class="login-form">
                <input type="text" placeholder="Логин"/>
                <input type="password" placeholder="Пароль"/>
                <button>Войти</button>
            </form>
        </div>
    </div>


</body>
```

Рисунок 2.17 – Profile.html.

Последним шагом будет добавление ссылок на форму авторизации в файлы urls.py.

После добавления ссылок в urls.py форма авторизации будет реализована (рис. 2.18):

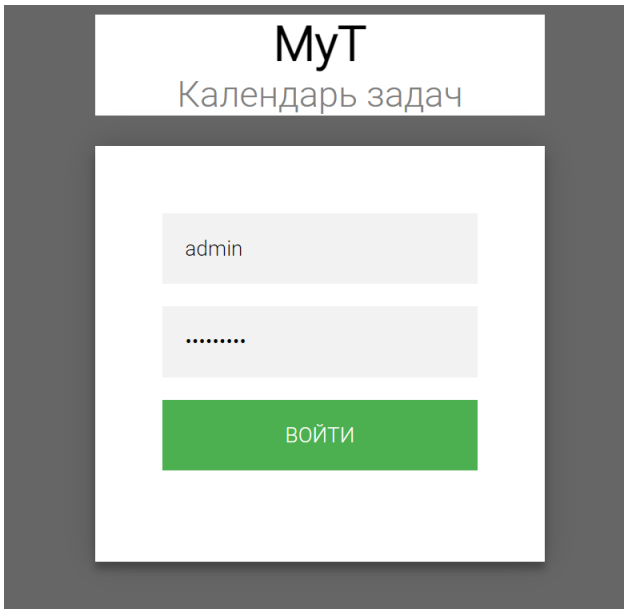


Рисунок 2.18 – Форма авторизации пользователя.

Чтобы проверить, сохраняются ли данные о пользователях, нужно открыть базу данных и найти таблицу autho_users, при переходе можно наблюдать созданных пользователей (рис. 2.19).

		last login	is superus	username
1	$nBbaMYmgdZiT4C7Py57trx$0wjyMWLZrcymQghSlBqV4dooOjRelBaeegvDjq3fkGI=	2024-06-12 23:08:40.634406	1	bipbip
2	$iqsZ9tQC4dF8o3bttrs3IM$gErxefRPi751f0ZHCUsqHHHV5HITB0HyWCkOwB0kUP8=	2024-06-09 13:10:41	0	Илья_Жуковский
3	$QBwMRTwd60vkUkQjB68IOI$9INEWzHQ67QJuk2e1b4U5fL4eeVp4/JHaq25MgRvzls=	2024-05-24 08:12:45	0	Хромов_Андрей

Рисунок 2.19 – Сохраненные пользователи.

2.4.4 Тестирование приложения

Для тестирования приложения необходимо задействовать несколько инструментов тестирования. Первым видом тестирования будет отладка в IDE, которую необходимо проводить во время разработки приложения. После будет проведены тестирование пользовательского интерфейса (UI) и интегрированные тесты, для выявления ошибок при авторизации сотрудников и проверке, при которой будет происходить правильном перенаправлении и корректного отображения страниц веб-приложения.

Чтобы протестировать, правильно ли данные с базы данных переносятся в календарь, в скрипт, созданный в файле index.html, была введена команда, благодаря которой можно отследить, какие данные проходят корректно, а какие нет (рис. 2.20).

```
<script>
    const tasks = {{ tasks|safe }};
    console.log("Tasks from server:", tasks);

    const daysInMonth = (year, month) => new D
    const firstDay = (year, month) => new Date
```

Рисунок 2.20 – Вставка команды в скрипт.

Результаты тестирования будут выводиться в консоль в реальном времени, тем самым в консоль будет выводится результат выполнения действий (рис. 2.21)

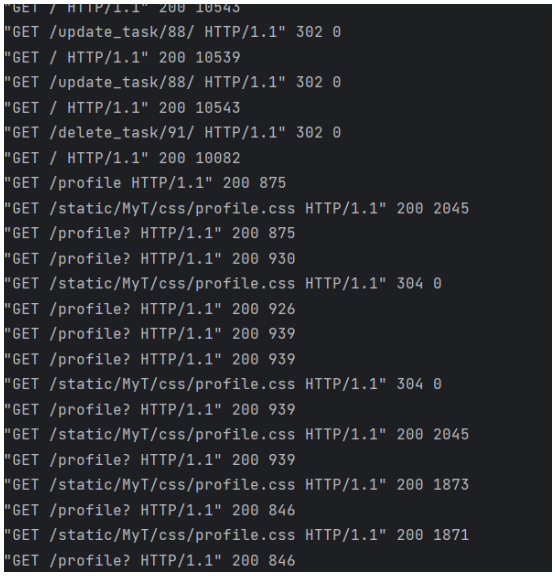


Рисунок 2.21 – Тестирование на корректность передачи данных.

В результате этого тестирования было выяснено, что все данные перемещаются и взаимодействуют друг с другом и с базой данных корректно.

Для Тестирование пользовательского интерфейса (UI) и интегрированного теста создается файл test.py, в котором будет реализован код для тестирования (рис. 2.22)

```
    yield driver
    driver.quit()


def test_login(browser):
    browser.get('http://127.0.0.1:8000/profile/')
    username_field = browser.find_element(By.NAME, 'login')
    password_field = browser.find_element(By.NAME, 'password')
    username_field.send_keys('bipbip')
    password_field.send_keys('40381972')

    password_field.send_keys(Keys.RETURN)
    profile_element = browser.find_element(By.ID, 'profile')
    assert profile_element.is_displayed(), "Логин не найден"
    browser.get('http://127.0.0.1:8000/profile/')
    index_link = browser.find_element(By.LINK_TEXT, 'Вход')
    index_link.click()
    assert "index.html" in browser.current_url
```

Рисунок 2.22 - Тестирование пользовательского интерфейса (UI) и интегрированных тестов.

В данном файле будет проводится тестирование на успешную авторизацию пользователя, проверка загрузку файла profile.html и переход на index.html путем обнаружения текущего URL страницы [16,19]. Результатом стало полное прохождения тестов Тестирование пользовательского интерфейса (UI) и интегрированных тестов (рис. 2.23).

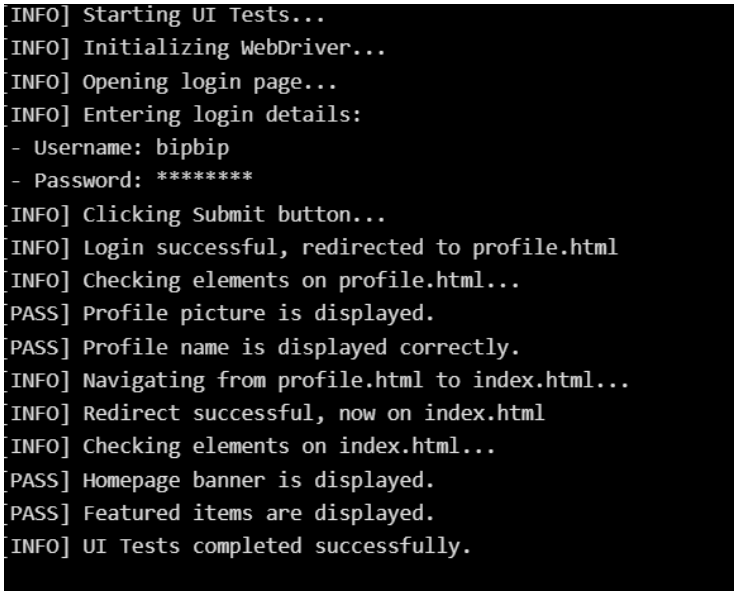


Рисунок 2.23 – Результаты тестов на интегрирование и интерфейс.

Тестирование системы прошло успешно. Пройденные виды тестов показали, что система успешно авторизирует пользователей, это свидетельствует о правильной работе всех компонента авторизации и прохождении интегрированных тестов. Проверка переходов между страницами также прошла успешно. После ввода логина и пароля произошла переадресация на главную страницу веб-приложения, что означает успешное прохождение Тестирование пользовательского интерфейса (UI).

Прохождение всех видов тестирование показывает, что веб-приложение «Календарь задач» удовлетворяет техническим требованиям, поставленным в техническом задании, обладает высокой надежностью и готов к внедрению в организацию.

2.4.5 Инструкция по эксплуатации и рекомендации по дальнейшему развитию веб-приложения

При открытии веб-приложения «Календарь задач» пользователя встречает окно авторизации. В данном окне необходимо ввести свой логин и пароль, выданный в организации, как показано на рисунке 2.24.

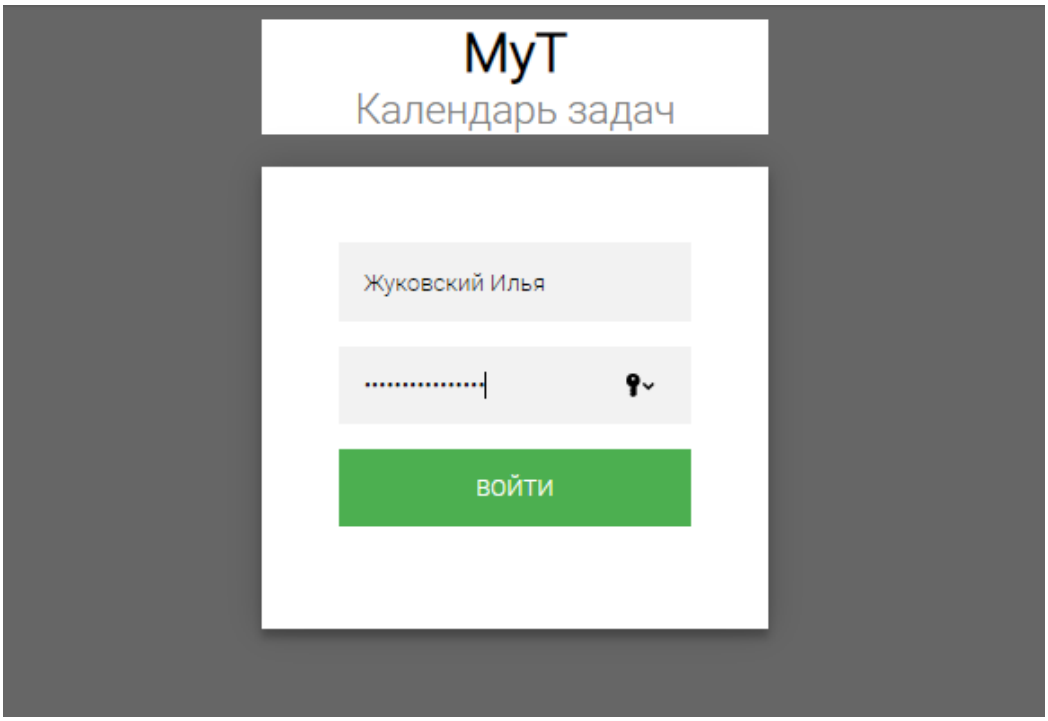


Рисунок 2.24 – Ввод личных данных в окно авторизации.

В случае успешной авторизации, пользователя переносит на главную страницу веб-приложения (рис.2.25).

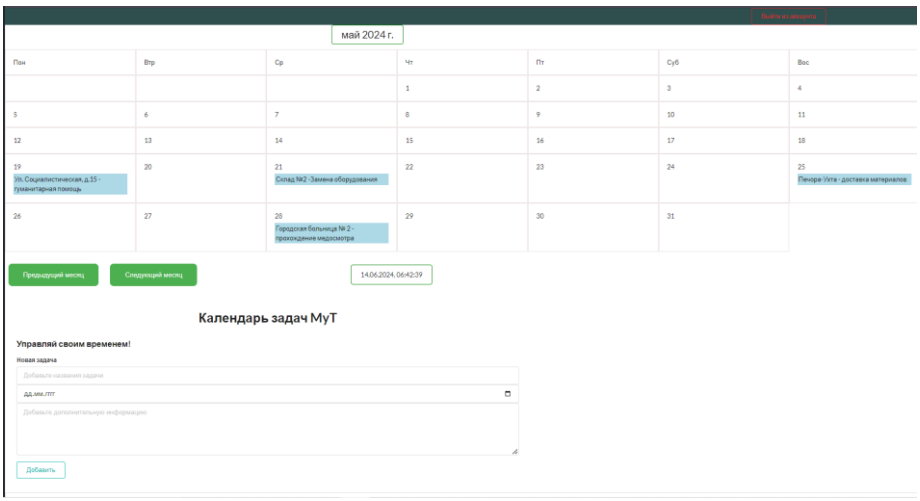


Рисунок 2.25 – Основной интерфейс веб-приложения

На верхней части страницы расположен интерактивный календарь, в верхнем левом углу экрана расположена кнопка «Выхода из аккаунта».

Ниже расположена форма добавления задач и список уже созданных. Для того чтобы добавить задачу, необходимо ввести название и дополнительную информацию и выбрать нужную дату (рис. 2.26)

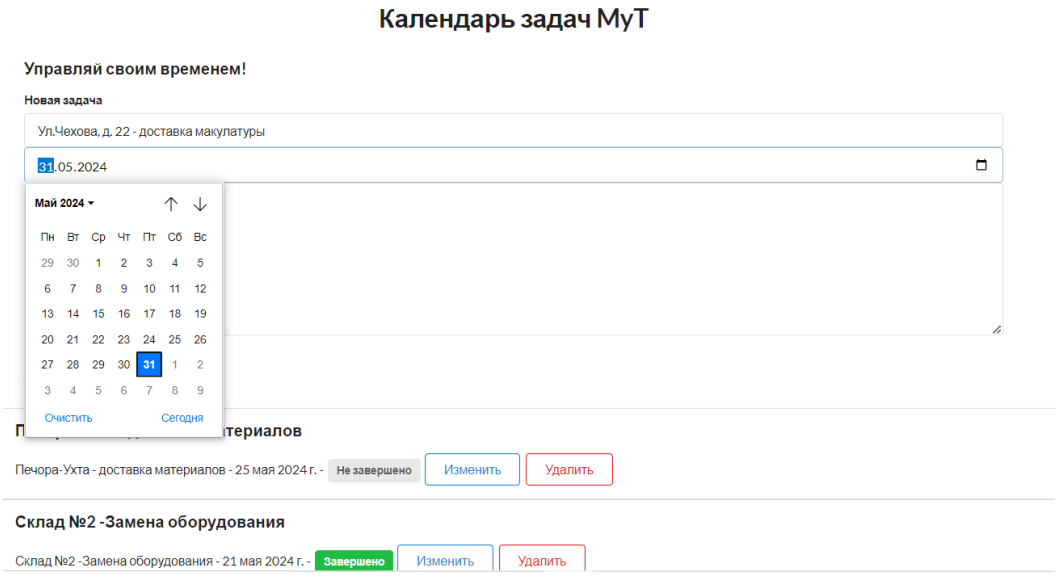


Рисунок 2.26 – Добавление задачи.

После указания названия и даты нужно выбрать кнопку добавить, и задача отобразится в списке созданных задач, а также на календаре, как показано на рисунке 2.27.

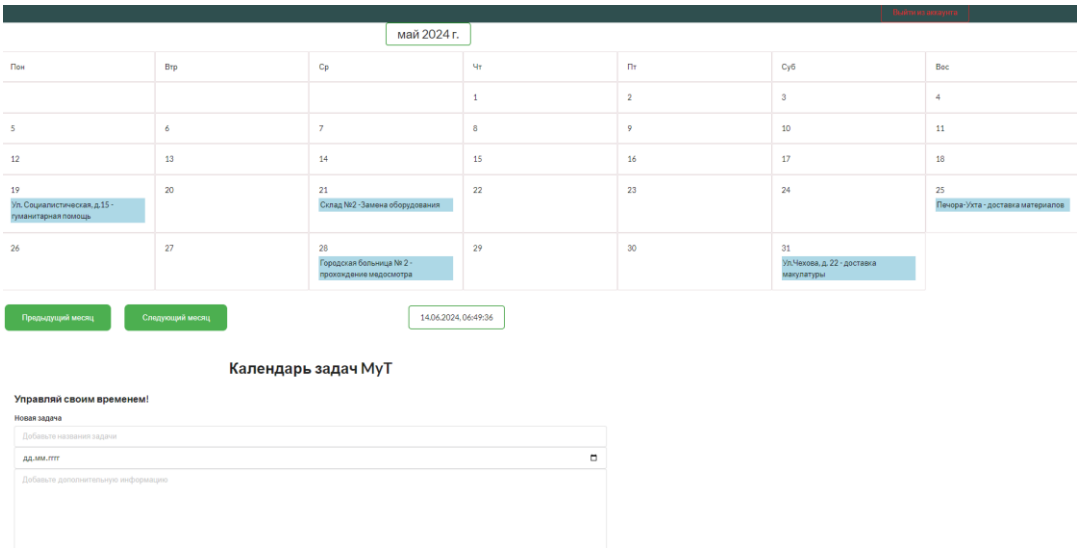


Рисунок 2.27 – Обновленный интерфейс.

Для просмотра и выполнения действий по изменению и удалению задач необходимо использовать форму списка созданных задач (рис.2.28).

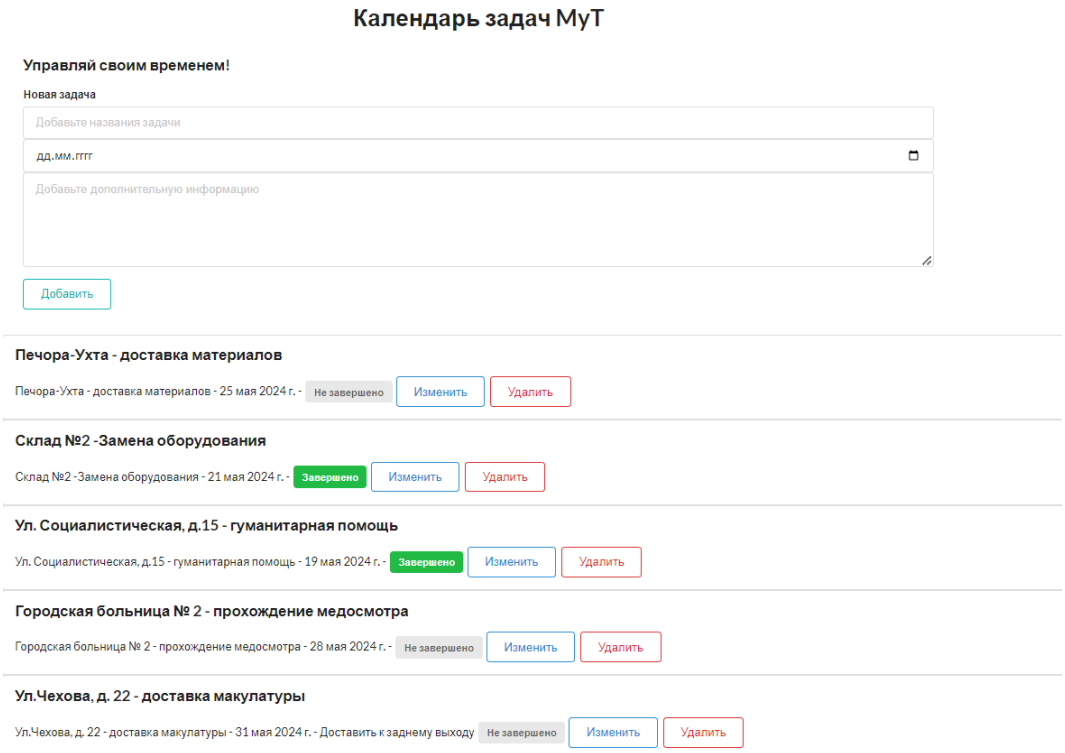


Рисунок 2.28 – Список созданных задач.

Список содержит следующие элементы:

— Информацию о задаче (названии, даты и дополнительной информации);

— Поле статуса выполнения;

— Кнопку для изменения статуса выполнения;

— Кнопку удаление задачи из веб-приложения.

При нажатии кнопки «Изменить» меняется статус завершения задачи, и имеет два состояния: завершено и не завершено (рис. 2.29)

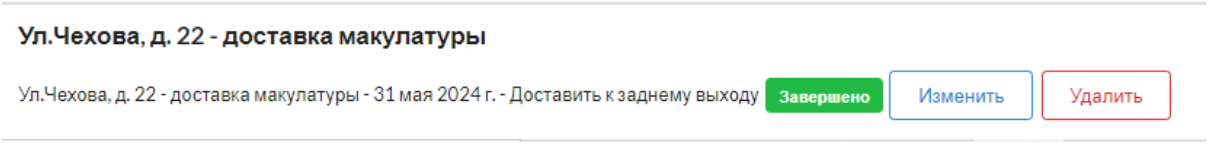


Рисунок 2.29 – Изменение результата выполнения задачи.

Для удаления задачи из списка, необходимо нажать на соответствующую кнопку. Также, как и из списка, задачу будет удалена из календаря и базы данных. (рис. 2.30).

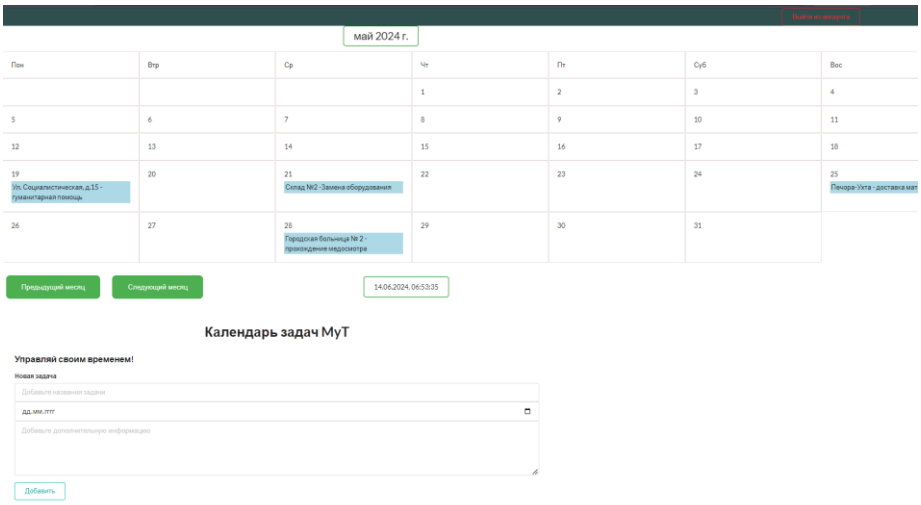


Рисунок 2.30 – Обновленный календарь.

Анализируя существующие приложения, решающие выявленную проблему распределения времени при выполнении рабочих задач, для развития функционала и достижения более высокой надежности веб-приложения, предлагаются следующие рекомендации по дальнейшему развитию проекта:

1) Добавление фильтров. Для более точной настройки и сортировки задач рекомендуется создать систему фильтров и папок, позволяющих сотрудникам разбивать задачи на категории, настраивать важность каждой задачи, благодаря добавления фильтров, сотрудники смогут правильно распределять своё время, уделяя больше времени и сил на решения важный и срочных задач;

2) Создание отчётов. Генерация отчётов о количестве решенных задач и сроков их выполнения позволит сотрудникам анализировать прогресс решенных ими задач. Подсчёт количества времени, затраченного на выполнение задачи, позволит сотрудникам разобрать и изменить подход процесса решения задач, что существенно снизит количество затраченного на выполнения времени и увеличит работоспособность сотрудников, а также даст начальству информацию,

которая может помочь для создания более эффективного подхода к решению поставленных сотрудникам задач;

3) Рассылка уведомлений. Добавление функции рассылки уведомлений на электронную почту позволит своевременно проинформировать пользователя о нерешенных задачах, сроки выполнения которых будут завершены в скором времени;

4) Разработка чата. Реализация функции по добавлению в созданное веб-приложение системы чата позволит сотрудникам коммуницировать между собой, что благотворно отразится на сокращении времени выполнения задач. Добавление в чат функции создания групп позволит эффективно распределять дату, необходимую для реализации рабочих задач, которое будет удобно для всех участников группы. Такой подход будет увеличить работоспособность сразу нескольких сотрудников одновременно.

ЗАКЛЮЧЕНИЕ

В ходе выполнения выпускной квалификационной работы была реализовано веб-приложение «Календарь задач». В условиях быстро растущей загруженности сотрудников компаний, реализация приложения, позволяющего распределять время и планировать как рабочие, так и личные дела, обусловлена актуальностью проблемы планирования времени.

Разработанное web-приложение является надежным способом решения проблемы распределения времени. Благодаря чему повышает эффективность выполнения рабочих задач. Тестирование приложения показало, что все компоненты системы надежно взаимодействуют друг с другом. Успешное тестирование аккаунтов пользователей позволило убедиться в надежности сохранения личных данных от несанкционированного доступа, а тестирование интерфейсов исключило риск возникновения ошибки некорректного отображения страниц.

В результате были выполнены все поставленные цели и задачи. Анализ существующих решений позволил составить архитектуру веб-приложения. Было выявлено, что от качественного и грамотного выбора технологий и инструментов разработки зависит конечное качество разрабатываемого продукта. Были даны рекомендации по добавлению функционала и улучшению качества приложения, позволяющие сотрудникам компании более точечно настраивать и управлять задачами, а также взаимодействовать с коллегами с помощью встроенного чата. Реализованное веб-приложение прошло проверку на надежность и работоспособность и готово к внедрению в компанию.

СПИСКО ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Документация Django. [Электронный ресурс]. // Официальный сайт Django. // Режим доступа: https://docs.djangoproject.com/ru/ (дата обращения 28.05.2024)

2. П.В. Терещенков, В.А. Астапчук, Архитектура корпоративных информационных систем, Новосибирск, 2015 г. – 75с.

3. Дронов В.А. , Django 3.0., Практика создания веб-сайтов на Python, БХВ-Петербург, 2021 г. – 704 с.

4. Шелли Пауэрс, Ruby on Rails. Быстрая веб-разработка, БХВ-Петербург, 2008 г. – 224 с.

5. Хоган Брайан, Уоррен Крис, Уэбер Майк, Джонсон Крис, Годин Аарон, Книга веб-программиста: секреты профессиональной разработки веб-сайтов [Электронный ресурс], "Питер", 2012 г. – 288 с. // Режим доступа: https://www.google.ru/books/edition/Книга_веб_программист/DN9o0S9y5W4C?hl=ru&gbpv=1&dq=веб-разработка+c%23&printsec=frontcover (дата обращения 01.06.2024)

6. Б. А. Новиков, Е. А. Горшкова, Н. Г. Графеева; под ред. Е. В. Рогова., Основы технологий баз данных: учеб. пособие, ДМК Пресс, 2020. — 582 с.

7. Одинцов Игорь Олегович , Профессиональное программирование. Системный подход, [Электронный ресурс], БХВ-Петербург, 2004 г. – 624 с. // Режим доступа: https://goo.su/FouTg. (дата обращения 03.06.2024)

8. Хавербеке Марейн, Выразительный JavaScript. Современное веб-программирование. 3-е издание, [Электронный ресурс]. Издательский дом , БХВ-Петербург, 2005 г. – 400 стр. // Режим допуска: https://www.google.ru/books/edition/Реляционные_базы_данн/yqW0D1U0Q9IC?hl=ru&gbpv=1 (дата обращения 27.05.2024)

9. Сэмми Пьюривал, Основы разработки веб-приложений, [Электронный ресурс]. // Режим доступа:

http://lib.jizpi.uz/pluginfile.php/6880/mod_resource/content/0/Pyurival_Semmi_-_Osnovy_razrabotki_veb-prilozheniy_Bestselle.pdf. (дата обращения 30.05.2024)

10. Е. Г. Сысолетин, С. Д. Ростунцев, Проектирование интернет-приложений [Электронный ресурс]. // URL: https://elar.urfu.ru/bitstream/10995/34785/1/978-5-7996-1503-1.pdf (дата обращения 29.05.2024)

11. Анатолий Тузовский, Проектирование и разработка web-приложений. Учебное пособие для СПО [Электронный ресурс]. // URL :https://www.google.ru/books/edition/Проектирование_и_разр/ (дата обращения 31.06.2024)

12. Постолит А., Python, PyCharm и Django для начинающих [Электронный ресурс], БХВ-Петербург, 2021 г. – 464 с. // Режим доступа: https://www.google.ru/books/edition/Python_PyCharm_и_Django_для_начи (дата обращения 02.06.2024)

13. Гарри Персиваль, Python. Разработка на основе тестирования [Электронный ресурс]. // URL :https://www.google.ru/books/edition/Python_Разработка_на_осн (дата обращения 09.06.2024)

14. Хориков Владимир , Принципы юнит-тестирования [Электронный ресурс], Издательский дом "Питер", 2020 г. – 320 с. //Режим доступа: https://www.google.ru/books/edition/Принципы_юнит_тестиро/8KceEAAAQBAJ?hl=ru&gbpv=1 (дата обращения 03.06.2024)

15. Хабрахабр - крупнейший ресурс для IT-специалистов. [Электронный ресурс]. (дата обращения 27.05.2024)

16. Савин Р.В., Тестирование программного обеспечения. Базовый курс., Питер, 2021 г. – 304 с.

17. Иванов А.Н., PyCharm: Профессиональная разработка приложений на Python, Диалектика, 2020 г. – 352 с.

18. Петров С.В., Проектирование веб-приложений: от идеи до реализации, O'Reilly, 2022 г. – 480 с.

19. Холл А., Тестирование веб-приложений. Тест-дизайн и автоматизация, Питер, 2022 г. – 416 с.

20. Лутц М., Изучаем Python. 5-е издание, ДМК Пресс, 2019 г. – 1600 с.

ПРИЛОЖЕНИЕ 1

Листинг программы.

Views.py:

```
from datetime import datetime
from django.shortcuts import render, redirect
from django.views.decorators.http import require_http_methods
from .forms import MyTForm
from .models import MyT
def index(request):
    mytos = MyT.objects.all()
    tasks = [{'date': myt.date.strftime('%Y-%m-%d'), 'title':
myt.title} for myt in mytos if myt.date is not None]
    return render(request, 'mytapp/index.html', {
        'manage_your_time': mytos,
        'title': 'Главная страница',
        'tasks': tasks
    })
@require_http_methods(['POST'])
def add_task(request):
    form = MyTForm(request.POST)
    if form.is_valid():
        form.save()
        return redirect('index')
    else:
        return render(request, 'mytapp/error.html', {'error':
'Invalid form data', 'form': form})
def update_task(request, myt_id):
    myt_update = MyT.objects.get(id=myt_id)
    myt_update.is_complete = not myt_update.is_complete
    myt_update.save()
    return redirect('index')
def delete_task(request, myt_id):
    myt_delete = MyT.objects.get(id=myt_id)
    myt_delete.delete()
    return redirect('index')
```

```
    def profile_view(request):
        return render(request, 'mytapp/profile.html')
    models.py:
    from django.db import models
    class MyT (models.Model):
        title = models.CharField('Название задания',
max_length=450)
        is_complete = models.BooleanField ('Завершено',
default=False)
        date = models.DateField('Дата', null=True, blank=True)
        class Meta:
            verbose_name = 'Задание'
            verbose_name_plural = 'Задания'
        def __str__(self):
            return self.title
    forms.py: from django import forms
    from .models import MyT
    class MyTForm(forms.ModelForm):
        class Meta:
            model = MyT
            fields = ['title', 'date']
            widgets = {
                'date': forms.DateInput(format='%Y-%m-%d',
attrs={'type': 'date'}),
            }
        def __init__(self, *args, **kwargs):
            super(MyTForm, self).__init__(*args, **kwargs)
            self.fields['date'].input_formats = ['%Y-%m-%d']
    profile.html:
    {% load static %}
    <html lang="en">
    <head>
        <meta charset="UTF-8">
        <link rel="stylesheet" href="{% static
'MyT/css/profile.css' %}">
```

```
        <script type="text/javascript" href="{% static
'MyT/css/profile.js' %}"></script>
        <title>Title</title>
    </head>
    <body>
        <div class="container">
            <!-- Контейнер для заголовка -->
            <div class="header-container">
                <div class="header-title">MyT</div>
                <div class="sub-title">Календарь задач</div>
            </div>
            <!-- Контейнер для форм -->
            <div class="form">
                </form>
                <form class="login-form">
                    <input type="text" placeholder="Логин"/>
                    <input type="password" placeholder="Пароль"/>
                    <button>Войти</button>
                </form>
            </div>
        </div>
    </body>
    </html>
    Index.html:
       </style>
    </head>
    <body>
        <div class="top-bar">
            {% if user.is_authenticated %}
                <a href="{% url 'logout' %}" class="logout-
button">Выйти из аккаунта</a>
            {% else %}
                <a href="{% url 'login' %}">Login</a> |
                <a href="{% url 'register' %}">Register</a>
            {% endif %}
```

```
        </div>
        <div class="month-display"></div>
        <div class="calendar"></div>
        <button class="button1" id="prev-month">Предыдущий
месяц</button>
        <button class="button2" id="next-month">Следующий
месяц</button>
        <div style="border: 1px solid green;border-radius:
5px;padding: 10px 20px; display: inline-block;text-align:justify;
        margin-left:320px" class="current-time"></div>
        <script>
            const tasks = {{ tasks|safe }};
            console.log("Tasks from server:", tasks); // Для
отладки, чтобы увидеть передаваемые задачи
            const daysInMonth = (year, month) => new Date(year,
month + 1, 0).getDate();
            const firstDay = (year, month) => new Date(year,
month, 1).getDay();
            const calendarElement =
document.querySelector(".calendar");
            const monthDisplayElement =
document.querySelector(".month-display");
            const currentTimeElement =
document.querySelector(".current-time");

            const daysOfWeek = ["Пон", "Втр", "Ср", "Чт", "Пт",
"Суб", "Вос"];
            let today = new Date();
            let currentMonth = today.getMonth();
            let currentYear = today.getFullYear();
            const renderCalendar = (year, month) => {
                calendarElement.innerHTML = "";
                // Выводим название месяца и год
                monthDisplayElement.textContent = new Date(year,
month).toLocaleString('default', {
```

```
                    month: 'long',
                    year: 'numeric'
                });
                // Выводим названия дней недели
                daysOfWeek.forEach(day => {
                    calendarElement.innerHTML += `<div
class='day'>${day}</div>`;
                });
                let firstDayIndex = firstDay(year, month);
                let totalDays = daysInMonth(year, month);
                // Заполняем пустые ячейки до начала месяца
                for (let i = 0; i < firstDayIndex; i++) {
                    calendarElement.innerHTML += "<div
class='day'></div>";
                }
                for (let day = 1; day <= totalDays; day++) {
                    const date = new Date(year, month, day);
                    const formattedDate = `${year}-${String(month
+ 1).padStart(2, '0')}-${String(day).padStart(2, '0')}`; //
Форматируем дату для сравнения
                    const dayElement =
document.createElement("div");
                    dayElement.classList.add("day");
                    if (date.toDateString() ===
today.toDateString()) {
                        dayElement.classList.add("current-day");
                    }
                    // Добавляем дни месяца
                    dayElement.innerHTML = day;
                    // Ищем задачу для конкретной даты
                    const taskForDay = tasks.find(task =>
task.date === formattedDate);
                    if (taskForDay) {
                        const taskElement =
document.createElement("div");
```

```
                        taskElement.classList.add("task");
                        taskElement.textContent =
taskForDay.title;
                        // Добавляем задачу к дню
                        dayElement.appendChild(taskElement);
                    }
                    // Добавляем в элемент календаря
                    calendarElement.appendChild(dayElement);
                }
            };
            document.querySelector("#prev-
month").addEventListener("click", () => {
                currentMonth--;
                if (currentMonth < 0) {
                    currentMonth = 11;
                    currentYear--;
                }
                renderCalendar(currentYear, currentMonth);
            });
            document.querySelector("#next-
month").addEventListener("click", () => {
                currentMonth++;
                if (currentMonth > 11) {
                    currentMonth = 0;
                    currentYear++;
                }
                renderCalendar(currentYear, currentMonth);
            });
            renderCalendar(currentYear, currentMonth);
            setInterval(() => {
                const now = new Date();
                currentTimeElement.textContent =
now.toLocaleString();
            }, 1000);
        </script>
```

```
    </body>
    </html>
    <div style="margin: 5px 25px;padding: 25px;display: inline-
block" class="ui container">
        <h1 class="ui center aligned header">Календарь задач
MyT</h1>
        <h3 class="test_style">Управляй своим временем!</h3>
        <form class="ui form" method="post" action="{% url
'add_task' %}">
            {% csrf_token %}
            <div class="field">
                <label>Новая задача</label>
                <input type="text" name="title"
placeholder="Добавьте названия задачи" required/>
                <input type="date" name="date" required />  <!--
Поле для выбора даты -->
            </div>
            <button class="ui teal basic button"
type="submit">Добавить</button>
        </form>
    </div>
    {% for myt in manage_your_time %}
    <div style="margin: 0;weight:70px;" class="ui segment">
        <p class="ui big header">{{ myt.title }}</p>
        <div>
            <span>{{ myt.title }} - {{ myt.date }}</span>
            {% if myt.is_complete == False %}
            <span class="ui gray label">Не завершено</span>
            <a class="ui primary basic button" href="{% url
'update_task' myt.id %}">Изменить</a>
            {% else %}
            <span class="ui green label">Завершено</span>
            <a class="ui primary basic button" href="{% url
'update_task' myt.id %}">Изменить</a>
            {% endif %}
```

```
            <a class="ui negative basic button" href="{% url
'delete_task' myt.id %}">Удалить</a>
        </div>
    </div>
    {% endfor %}
    {% endblock content %}
    Layout.html:
    <!DOCTYPE html>
    {% load static %}
    <html lang="ru">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport"
              content="width=device-width, user-scalable=no,
initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">
        <meta http-equiv="X-UA-Compatible" content="ie=edge">
        <title>{{ title }}</title>
        <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/semantic-
ui@2.4.2/dist/semantic.min.css">
        <link rel="stylesheet" href="{% static
'MyT/css/style.css' %}">
    </head>
    <body>
    {% block content %}
    {% endblock content %}
        <script src="https://cdn.jsdelivr.net/npm/semantic-
ui@2.4.2/dist/semantic.min.js"></script>
    </body>
    </html>
    Profile.css:
    @import
url(https://fonts.googleapis.com/css?family=Roboto:300);
    .login-page {
      width: 360px;
```

```
      padding: 8% 0 0;
      margin: auto;
    }
    .form {
      position: relative;
      z-index: 1;
      background: #FFFFFF;
      max-width: 360px;
      margin: 0 auto 100px;
      padding: 45px;
      text-align: center;
      box-shadow: 0 0 20px 0 rgba(0, 0, 0, 0.2), 0 5px 5px 0
rgba(0, 0, 0, 0.24);
    }
    .form input {
      font-family: "Roboto", sans-serif;
      outline: 0;
      background: #f2f2f2;
      width: 100%;
      border: 0;
      margin: 0 0 15px;
      padding: 15px;
      box-sizing: border-box;
      font-size: 14px;
    }
    .form button {
      font-family: "Roboto", sans-serif;
      text-transform: uppercase;
      outline: 0;
      background: #4CAF50;
      width: 100%;
      border: 0;
      padding: 15px;
      color: #FFFFFF;
      font-size: 14px;
```

```
  -webkit-transition: all 0.3 ease;
  transition: all 0.3 ease;
  cursor: pointer;
}
.form button:hover,.form button:active,.form button:focus {
  background: #43A047;
}
.form .message {
  margin: 15px 0 0;
  color: #b3b3b3;
  font-size: 12px;
}
.form .message a {
  color: #4CAF50;
  text-decoration: none;
}
.form .register-form {
  display: none;
}
.container {
  position: relative;
  z-index: 1;
  max-width: 300px;
  margin: 0 auto;
}
.container:before, .container:after {
  content: "";
  display: block;
  clear: both;
}
.container .info {
  margin: 50px auto;
  text-align: center;
}
.container .info h1 {
```

```
  margin: 0 0 15px;
  padding: 0;
  font-size: 36px;
  font-weight: 300;
  color: #1a1a1a;
}
.container .info span {
  color: #4d4d4d;
  font-size: 12px;
}
.container .info span a {
  color: #000000;
  text-decoration: none;
}
.container .info span .fa {
  color: #EF3B3A;
}
body {
  background: #666666;
  font-family: "Roboto", sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}
.header-container {
    text-align: center;
    margin-bottom: 20px;
    background: #FFFFFF;
    margin:#FFFFFF;
}

.header-title {
    font-size: 2em;
    font-weight: bold;
}
```

```
.sub-title {
    font-size: 1.5em;
    color: gray;
    background: #FFFFFF;
}
```