МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Тольяттинский государственный университет»

Кафедра «Прикладная математика и информатика»
(наименование)

09.03.03 Прикладная информатика
(код и наименование направления подготовки / специальности)

Разработка социальных и экономических информационных систем
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему Разработка сервиса управления задачами

Обучающийся Э. Г. Иванян _____
(Инициалы Фамилия) (личная подпись)

Руководитель канд.пед.наук, доцент, Е.А. Ерофеева
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Консультант канд.пед.наук, доцент, А.В. Егорова
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2024

Аннотация

Тема выпускной квалификационной работы – «Разработка сервиса управления задачами» в ООО «Квартплата 24».

Актуальность работы заключается в эффективном управлении задачами, тем самым организовывать рабочий процесс.

Целью данной работы является разработка сервиса управления задачами, которая представляет собой совокупность разновидных задач (личные, проектные, приоритетные и т.д.), хранящие в единой системе.

Разработан сервис для управления задачами, который позволяет обмениваться задачами между пользователями с использованием асинхронного программирования и веб-сервисов. Решена проблема производительности, затрачиваемого времени и организованности.

Выпускная квалификационная работа состоит из 59 страниц текста, 30 рисунков, 4 таблиц и 20 источников.

Abstract

The title of the bachelor’s thesis is «Development of a task management service».

The relevance of the work lies in the effective task management, thereby organizing the work process.

The object of research is a task tracking services. Namely, ready-made sites that allow you to create and manage tasks.

The subject of the research is methods and technologies for developing a task tracking service.

The goal of the graduation qualification work is to development of a task management service, which is a collection of various types of tasks (personal, project, priority, etc.) stored in a single system.

A task management service has been developed that allows you to exchange tasks between users using asynchronous programming and web services. Solved the problem of productivity, time spent and organization.

The graduation qualification work consists of 59 pages of text, 30 figures, 4 tables and 20 sources.

Оглавление

Введение

Любая трудовая деятельность включает в себя выполнение различных задач, каждая из которых имеет свои подзадачи. Иногда бывает непросто правильно распределить свое рабочее время для достижения максимальной продуктивности. Кроме того, легко упустить из виду определенные задачи, которые требовалось выполнить.

У каждого человека есть свои рабочие планы и обязанности, но представьте себе, насколько сложно отслеживать выполнение задач каждого сотрудника в организации, где трудятся 25 или более сотрудников, у каждого из которых свои задачи, которые необходимо выполнять максимально эффективно, чтобы организация не застыла на месте, а продолжала развиваться.

Для улучшения эффективности и улучшения результатов компании необходимо разработать приложение, который позволит создавать и управлять любыми типами задач, где будет вся информации о созданной задачи, в том числе дата, до которой нужно выполнить созданную задачу. Такой сервис должен быть адаптивным, то есть возможность создавать любые типы задачи, такие как, бизнес-задачи, научные, образовательные и так далее.

Актуальность работы заключается в желании улучшить показатели компании, с помощью повышения эффективности и работоспособность каждого сотрудника, тем самым показатели компании увеличиться.

Объектом исследования являются сервисы отслеживания задач. А именно готовые сайты, где дают возможность создавать и управлять задачами.

Предметом исследования являются методы и технологии для разработки сервиса отслеживания задач.

Цель выпускной квалификационной работы – разработать сервис отслеживания задачами, которая представляет собой совокупность разновидных задач (личные, проектные, приоритетные и т.д.), хранящие в единой системе.

Для достижения поставленной цели необходимо решить следующие задачи:

— изучить существующие сервисы, которые представляют возможность управлять задачами.

— спроектировать архитектуру сервиса управления задачами.

— разработать алгоритмы и методы для реализации приложения для управления задачами, обеспечивающие адаптивность.

— произвести тестирование и оценку эффективности разработанного сервиса управления задачами.

Работа состоит из трех глав. В первой главе описывается экосистема сервисов в ООО «Квартплата 24», дается характеристика текущим решениям и обосновывается актуальность. Во второй главе обозначаются требования к разрабатываемому продукту, описывается процесс проектирования архитектуры сервисов, осуществляется выбор инструментов реализации и разработка диаграмм класса сервиса. В третьей главе описываются инструменты разработки и процесс разработки интерфейса унифицированного доступа.

Глава 1 Характеристика организации и анализ существующих сервисов

1.1 Характеристика и анализ сервисов организации

«Квартплата 24» - IT-компания, которая разработала отраслевую цифровую платформу для расчётов в сфере ЖКХ, то есть данная платформа нацелена на то, чтобы пользователям было удобнее решать следующие задачи:

— вносить показания счетчиков;

— оплачивать квитанции;

— хранить оплаченные документы;

— распечатывать квитанции и выписки;

— получать уведомления от УК;

— оставить онлайн-заявку для аварийно-диспетчерской службы;

— обратиться в службу поддержки.

Также, на их сайте можно найти ответы на часто задаваемые вопросы, данный раздел так и называется «Часто задаваемые вопросы». Благодаря такой возможности пользователи не тратят своё личное время на типичные вопросы, которые возникают вовремя пользованием сайтом, а это очень ценно в наше время.

Все системы можно разделить на две основные группы – внутренние, предназначенные для поддержания технологических процессов, и клиенто-ориентированные – те, с которыми клиенты непосредственно взаимодействуют.

Схематичное изображение основных сервисов экосистемы ООО «Квартплата 24» представлено на рисунке 1.

Ниже дано описание основных, наиболее крупных и важных сервисов, с которыми взаимодействуют клиенты пользователи:

— биллинговая система – осуществляет автоматизированный расчет платы за жилищно-коммунальные услуги, также отвечает за расщепление платежей без участия УО и ТСЖ на конечных получателей – РСО и СО

— личный кабинет жителя – предназначен для плательщиков, предоставляет своевременную информацию о начислениях по ЖКУ, возможность оплаты, внесения показаний, просмотр платежных документов и т.д.;

— сервис аналитики – предназначен для руководителей обслуживающих организаций, и предоставляет им информацию о начислениях, поступлении и расщеплении платежей по управляемому жилому фонду;

— сервис работы с должниками (он же СРД) – предназначен для проведение досудебной и судебной работы с должниками клиентов.

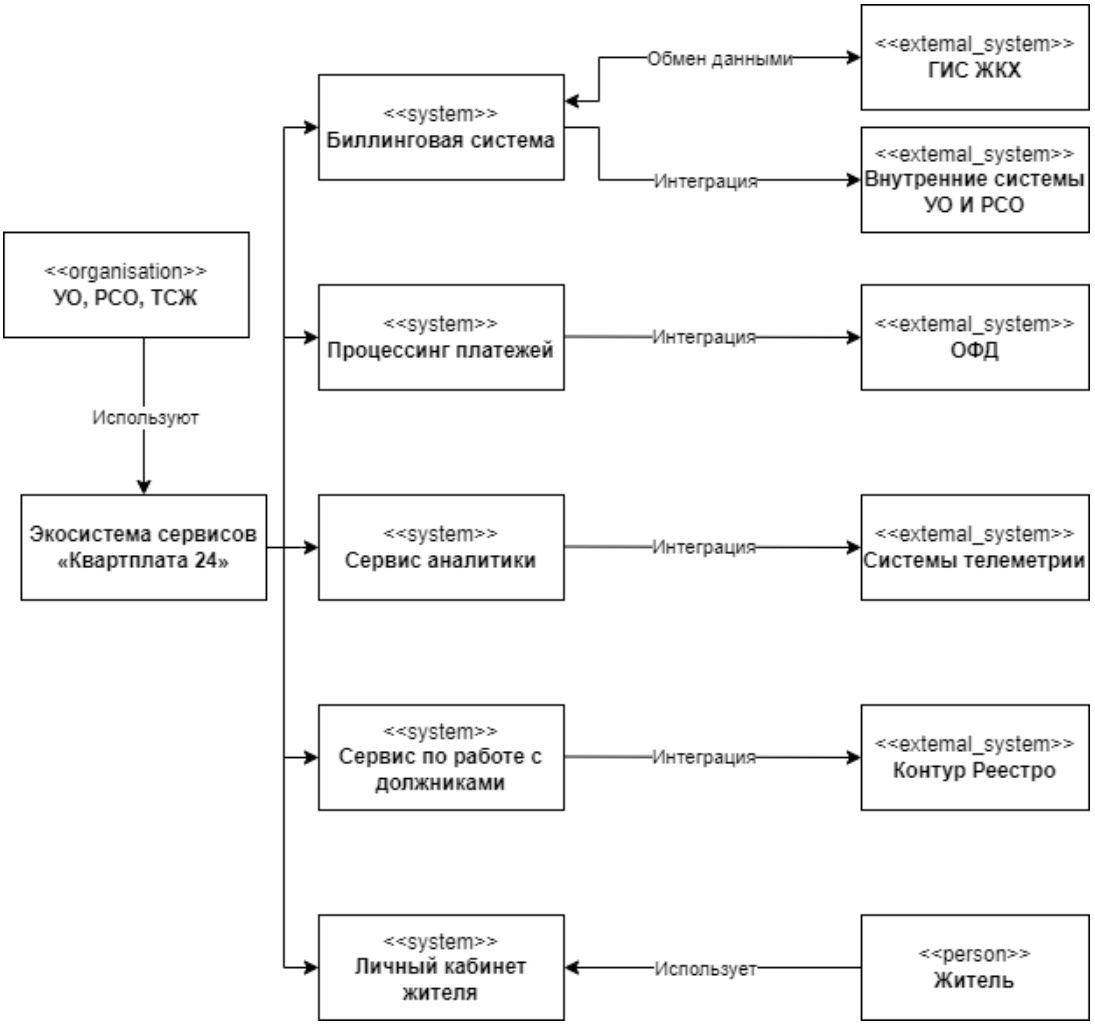


Рисунок 1 – Схема экосистемы сервисов ООО «Квартплата 24»

Помимо этих сервисов, также есть и те, что обеспечивают интеграцию с

другими внешними сервисами и/или отвечающие за внутренние процессы экосистемы:

— платежный сервис – специализируется на приеме платежей из более чем 40 банков и платежных систем. Данный сервис интегрирован с биллинговой системой;

— ОФД-шлюз – сервис фискализации платежей, обеспечивает интеграцию с онлайн-кассами, при оплате моментально формируя чеки и передает их в ФНС и плательщику;

— ГИС-шлюз – обеспечивает постоянный обмен данными с Государственной информационной системой жилищно-коммунального хозяйства (ГИС ЖКХ) в соответствии с законодательством;

— сервис телеметрии – обеспечивает интеграцию с системами телеметрии (например, Элдис);

— отдельные интеграции с внешними облачными сервисами и системами (такими как Суд РФ, ПО Контакт, Умное ЖКХ, Контур Реестро и т.д.).

Следует также отдельно выделить такой внутренний сервис, как Remote Access Point (RAP). Как следует из названия, он представляет собой единую точку доступа к сервисам экосистемы, через него проходят все запросы, адресованные сервисам, направленным на взаимодействие с клиентами.

На рисунке 2 представлена схема, демонстрирующая взаимодействие основных сервисов. Большая часть сервисов экосистемы представляет собой программное обеспечение, построенное по принципам монолитной архитектуры. Такая архитектура считается традиционной моделью программного обеспечения, в идеале работающего автономно от других приложений.

В компании "Квартплата 24" отсутствует собственный сервис управления задачами для своих сотрудников. Это может затруднять эффективную координацию рабочих процессов и повышать сложности в организации рабочих задач. Внедрение такого сервиса могло бы значительно упростить внутреннюю

коммуникацию и управление проектами в компании.

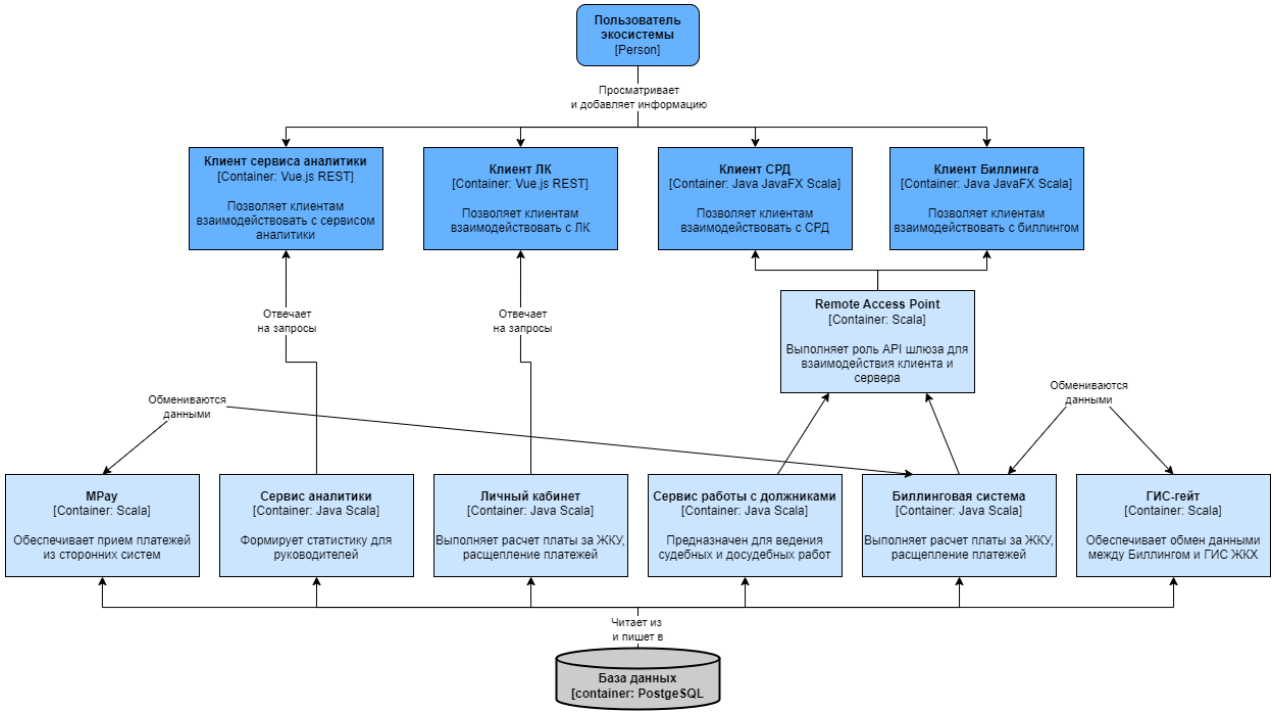


Рисунок 2 – Схема, отражающая взаимодействие основных сервисов ООО «Квартплата 24»

Рассмотрим модель «Как есть?», которая показана на рисунке 3, Данная модель помогает организовать текущие процессы и информационные ресурсы, выявляя слабые места в работе компании и взаимодействии бизнес-процессов, что позволяет определить необходимость изменений в текущей структуре.

Рассмотрим порядок бизнес-процессов внутри диаграммы:

– Процесс начинается с составления ТЗ для нового проекта;

– Далее проектный менеджер передает техническое задание руководителю практику;

– Если руководителю все понятно, он начинает составлять задания для своей команды, в ином случае уточняет у проектного менеджера подробности;

– Далее если участникам команды все понятно, они начинают работать над выполнением данных задач, в ином случае уточняют у руководителя команды

подробности;

– После того, как инициатор выполнил свою задачу, он отправляет её на проверку руководителю;

– Если руководитель одобряет решение инициатора, то принимает работу;

– Если все задачи были выполнены для реализации проекта, то готовый проект отправляется проектному менеджеру;

– Если проектный менеджер одобряет данную реализацию, то проект готов, иначе не принимает.

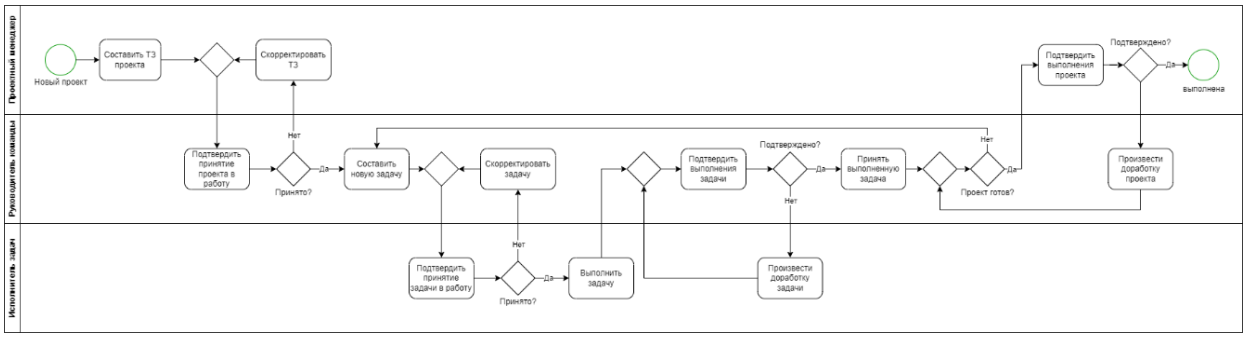


Рисунок 3 – Диаграмма бизнес-процесса «КАК ЕСТЬ»

Важно отметить, что все действия, которые описаны в модели выполняются вживую, из-за чего и тратиться много времени, особенно если у организации несколько корпусов.

1.2 Анализ сервисов управления задачами

Только с помощью правильных поставленных задач в команде, и своевременно выполнение их поможет сделать такую экосистему, которую предоставляет ООО «Квартплата 24» своим пользователям. Для начала рассмотрим, какими бывают задачи и для чего нужно их отслеживать.

В команде постановка задач – это процесс определения конкретной цели

или задания, которое требуется выполнить совместными усилиями. Этот этап планирования работы команды играет важную роль в установлении четких целей и ожиданий, определении ролей и обязанностей каждого участника, а также обеспечивает эффективное выполнение поставленной задачи.

Главное понимать, что достижение целей в команде тесно связанно с каждым участником, так как главная задача делится на подзадачи, которые в дальнейшем распределяются на всю команду. Поэтому очень важно чтоб каждый участник был проинформирован о своей задачи вовремя, а также чтоб он её выполнил в срок. Но порою, к сожалению, бывают такие моменты, когда исполнитель какой-либо задачи попросту не выполняет свою поставленную задачу, из-за разных причин, даже, бывает, банально из-за того, что его не оповестили. А если подзадачу не выполняет хотя бы один человек, то сама задача задерживается в своём исполнении, что сказывается на продуктивности выполнении всех задач компании.

Для решения данной проблемы, программисты разработали различные сайты, приложения для отслеживания задач. Где руководитель практики оперативно создаёт для каждого участника команды собственную задачу, указывая сроки выполнения, и подробно описывая саму задачу. Тем самым исполнителю очень удобно управлять своими задачами, а их бывает несколько, и более эффективно распланировать свой рабочий процесс.

Давайте для начала рассмотрим уже существующие сервисы по управлению задач на просторах интернета.

«Битрикс 24» – данный комплексный продукт способствует упрощению и оптимизации взаимодействия между различными отделами одной компании. Он включает в себя инструменты для планирования, делегирования задач, проведения аналитики, оценки результатов, а также обеспечивает эффективное деловое общение между сотрудниками и распространение важной информации по всему персоналу. В общем, все функции платформы Битрикс24 направлены на цифровизацию бизнес-процессов и ключевых направлений деятельности организации.

Продукт безусловно популярный и имеет много положительных сторон, но также и имеет немало отрицательных. Давайте рассмотрим их.

Плюсы:

- Обширный функционал: управление проектами, задачами, интеграция с другими сервисами;
- Облачное хранилище данных: доступ к информации с любой точки мира;
- Мобильное приложение: удобство работы на мобильных устройствах;
- Возможность настройки и адаптации под различные бизнес-процессы;

Минусы:

- Сервис платный: бесплатная версия имеет много ограничений, например, некоторые функции;
- Зависимость от интернет-соединения: данные хранятся в облаке, поэтому требуется стабильное интернет-соединение;
- Сложность настройки и изучения: потребуется немало времени для освоений всех возможностей системы, так как функционал обширный;

«Yandex Tracker» — инструмент, предназначенный для координации коллективной деятельности по выполнению задач и проектов. С его помощью можно эффективно организовать работу над различными задачами, от разработки программного обеспечения до создания службы технической поддержки или оптимизации работы отдела продаж.

Как и у любого сервиса, данная платформа имеет и плюсы, и минусы, давайте рассмотрим их.

Плюсы:

– Интеграция с другими сервисами Яндекса: это однозначно удобно для

пользователей, уже знакомых с другими продуктами компании;

- Возможность отслеживать задачи и проекты в реальном времени;
- Гибкие настройки уведомлений и оповещений для пользователя;
- Поддержка мобильных устройств для работы в любое время и из любой точки;
- Интеграция с календарем для удобного планирования задач;
- Поддержка командной работы и совместного доступа к задачам;

Минусы:

- Отсутствие расширенных функций управления проектами, которые могут быть необходимы для сложных проектов;
- Недостаток персонализации и гибких настроек для оптимизации рабочего процесса;
- Ограниченный набор функциональности по сравнению с некоторыми конкурентами;
- Платформа платная: ограничения в бесплатной версии и стоимость платных планов.
- Ограниченные возможности интеграции с другими популярными сервисами управления проектами.

Не смотря на удобное решение, которое предоставляет данный сервис для управления различными задачами, есть очень весомый минус – это цена. У этой платформы очень интересный расчёт стоимости, до 5 человек бесплатно, от 6 до 100 человек 440 рублей (с учётом первых 5), от 101 до 250 человек 400 рублей, от 251 человек стоимость сервиса будет уже 360 рублей. Давайте более подробно изучим ценовую политику сервиса «Yandex Tracker»

Данная платформа рассчитывает стоимость по максимальному количеству пользователей, которые в одно и тоже время имели доступ к сервису, например, в марте им пользовались:

- 12 дней: 247 пользователей;
- 10 дней: 235 пользователей;

– 8 дней: 285 пользователей.

Расчёт оплаты за март месяц будет следующим:

$$100 * 440 + 150 * 400 + 35 * 360 = 44000 + 60000 + 12600 = 116600\text{р.}$$

Давайте более подробно рассмотрим данный расчёт:

– первых 100 пользователей сервис рассчитает по цене 440 рублей за человека;
– дальше 150 человек будут рассчитаны по тарифу 400 рублей за человека;
– оставшиеся 35 пользователей, будут считаться по тарифу больше 250, то есть по цене 360 рублей за каждого.

Таким образом, цена за месяц выходит 116600 рублей за команду из 285 человек.

Ну и в качестве последнего сервиса, давайте рассмотрим довольно известную платформу «Kaiten». Платформа предназначена для управления задачами, проектами, коллективами и бизнес-процессами. С ее помощью можно эффективно организовать работу команды в соответствии с принципами методологий Kanban и Scrum.

Давайте так же рассмотрим плюсы и минусы данного сервиса.

Плюсы:

– Интуитивно понятный интерфейс и простота использования;
– Гибкие настройки и возможность адаптации под различные рабочие процессы;
– Удобное управление задачами и проектами, включая функции планирования и отслеживания сроков;
– Возможность интеграции с другими популярными сервисами и приложениями.
– Поддержка командной работы и совместного доступа к задачам

Минусы:

– Незначительные задержки и периодические проблемы с подключением к интернету, хотя глобальных сбоев не было замечено;
– Автоматическое прикрепление к карточке при передвижениях, что может

быть не всегда удобным для некоторых рабочих процессов;

- Общие метки для всех досок и пространств, что может вызвать неудобство при работе с несколькими задачами;
- Отсутствие уведомлений о перемещении карточки между колонками;
- Длительность ожидания экспорта данных из других сервисов, особенно если данных много, что иногда может вызвать ощущение зависания
- Цена, данный сервис так же является платным, много ограничении у бесплатной версии.

Также есть и иностранные аналоги, такие как «Trello», «Asana» и «Jira», но рассматривать их не целесообразно, так как отечественные платформы работают на отечественных серверах, что исключает внезапные сбои и блокировки, а также гарантирует стабильность работы.

Таким образом, мы рассмотрели три популярных сервисов, которые были разработаны для выстраивания более эффективного рабочего процесса, и в качестве одним из ключевых инструментов выступает система по отслеживанию задач.

1.3 Актуальность создания собственного сервиса по управлению задач

В пункте 1.2 мы рассмотрели три популярных платформы по управлению задачами, но также были перечислены их главные минусы, из-за которых и пришла идея сделать собственный сервис по управлению задачами. Рассмотрим более подробно актуальность данного решения.

Один из серьезных недостатков заключается в высокой стоимости, так как платформы такого типа далеко не бесплатные, а стоимость напрямую зависит от количества пользователей в компании. Если компания небольшая, где, например, 25 сотрудников, то тогда это не является большой проблемой. А если речь о крупной организации, где, допустим, работают около 300 сотрудников, это будет не выгодно. Давайте рассчитаем примерную стоимость каждой

платформы для малого, среднего и большого бизнеса.

Количество сотрудников в организации может существенно различаться в зависимости от размера и типа организации. Вот общие примеры среднего количества сотрудников в различных типах организаций:

- Маленькая компания обычно имеет менее 50 сотрудников. Это может быть небольшой стартап, семейное предприятие или небольшой магазин.
- Средняя компания может иметь от 50 до нескольких сотен сотрудников. Это могут быть средние компании в различных отраслях, такие как производство, услуги, IT и другие.
- Большие компании обычно имеют несколько сотен и тысяч сотрудников. Это могут быть крупные корпорации, международные компании, крупные банки, технологические гиганты и т.д.

Начнём для начала с определением количества пользователей. Это понадобится для расчёта средней стоимости. В малой организации не так много сотрудников, поэтому у нас будет 25 сотрудников. Для среднего бизнеса 25 человек будет уже маловато, так как в средней организации от 100 до 250 рабочих. Поэтому для расчета средней стоимости возьмем число 150. Ну а для более крупной организации возьмём число 300. Так же мы рассчитаем стоимость таких сервисов для компании ООО «Квартплата24», в которой на данный момент работают 70 сотрудников.

Таблица 1 – Месячная стоимость платформ.

Вид организации	Битрикс 24	Yandex Tracker	Kaiten
Малая организация	от 7000 до 35000 рублей	22000 рублей	от 21000 до 28000 и выше рублей
Средняя организация	от 15000 до 35000 рублей	64000 рублей	от 63000 до 84000 и выше рублей
Большая организация	от 35000 рублей	122000 рублей	от 126000 до 168000 и выше рублей
«Квартплата 24»	от 15000 до 35000 рублей	30800 рублей	от 29400 до 39200 и выше рублей

Смотря на месячные стоимости данных платформ, складывается мнение, что не особо то и дорого. А теперь, давайте проанализируем итоговый годовой расход на такие сервисы.

Таблица 2 – Годовая стоимость платформ.

Вид организации	Битрикс 24	Yandex Tracker	Kaiten
Малая организация	от 84000 до 420000 рублей	264000 рублей	от 252000 до 336000 и выше рублей
Средняя организация	от 180000 до 420000 рублей	768000 рублей	от 756000 до 1008000 и выше рублей
Большая организация	от 420000 рублей	1464000 рублей	от 1512000 до 2016000 и выше рублей
«Квартплата 24»	от 140000 до 420000 рублей	369600 рублей	от 352800 до 470400 и выше рублей

Смотря на таблицу 2, больше данные сервисы не кажутся такими дешевыми. Но как правило организации пользуются такими сервисами не один и два года, а гораздо более долгое время. А вторым минусом является функциональность платформ. Ведь не всем нужна, допустим, CRM-система или контакт-центр, когда есть много количество других бесплатных мессенджеров, где реализация звонков в какой-то степени будет более удобное и практичнее. Но, во всяком случае, благодаря этим ненужным возможностям придётся заплатить больше.

Обсудив эти два минуса, сделаем вывод, что невыгодно покупать тарифы на этих платформах, но и без такого сервиса тоже никак. Для этого было принято решение сделать свой сервис по отслеживание задач, так как это практичнее, удобнее и дешевле.

1.4 Постановка задачи на разработку

Исходя из того, что мы рассмотрели в пункте 1.3 сделали вывод, что это невыгодно с экономической точки зрения, так как у сервисов, без подписки, мало функционала, а платные версии стоят не дешево. Поэтому было решено создать свой сервис управления задачами, который будет аналогом готовых решений, но принадлежащий компании, тем самым организации сэкономит. Так же в интерфейсе приложения не будет ничего лишнего, что и является самым главным пунктом, если мы говорим о комфорте и удобстве сотрудника, который будет пользоваться данным сервисом.

Основной целью данной выпускной квалификационной работы является разработка сервиса управления задачами. Для этого предлагается интегрировать в наш сайт раздел с проектами, в котором мы уже и будем создавать различные задачи для участников данного проекта, а в дальнейшем уже и отслеживать данные задачи. Основным функционалом разрабатываемого приложения являются проекты и задачи, а точнее взаимосвязь между проектами и задачами. Кроме того, данный сервис будет отвечать за аутентификацию и авторизацию пользователей.

Для успешной разработки программного продукта необходимо:

- Выделить функциональные и нефункциональные требования;
- спроектировать архитектуру сервиса управления задачами;
- выбрать и описать инструменты реализации приложения, языка программирования и среды разработки;
- разработать сервис управления задачами;
- описать принципы работы разработанного сервиса;
- тестирование уже готовое программное решение.

В этой главе были рассмотрены основные проблемы и недостатки готовых сервисов по отслеживанию задач, и пришли к выводу что выгоднее купить один раз сервис управления задачами, чем каждый месяц платить за подписку. Поставлена задача на разработку сервиса отслеживания задач.

Глава 2 Проектирование веб-приложения для управления задачами

2.1 Требования к разрабатываемому сервису

На основании описанных в первой главе преимуществ было решено разработать собственный сервис управления задачами. А именно веб-приложение для создания и дальнейшего управления задачами. В нашем случае задачи будут создаваться и отслеживаться в проектах.

Основной функционал сервиса, разрабатываемого в ходе выполнения данной выпускной квалификационной работы, заключается в управлении задачами, а именно создании, редактировании и отслеживании. Для этого в качестве основного функционала также будут и проекты. Проекты можно будет добавлять и редактировать. Внутри проекта и будет реализовано создание, редактирование и отслеживание задач.

При разработке любого программного продукта необходимо сначала определить функциональные и нефункциональные требования разрабатываемого продукта, сервиса или системы.

Функциональные требования определяют, каким должно быть поведение продукта в тех или иных условиях. Они определяют, что разработчики должны создать, чтобы пользователи смогли выполнить свои задачи (пользовательские требования) [5].

Функциональные требования разрабатываемого сервиса:

– сервис должен предоставлять возможность создавать учетные записи новых пользователей;

– пользователи должны иметь возможность войти в систему с помощью учетных данных;

– пользователи должны иметь возможность создавать новый проект или новую задачу, указывая заголовок, описание, срок и выбирая исполнителя;

– пользователи должны иметь возможность просматривать список

своих задач, отмечать их как выполненные, редактировать и удалять;

– сервис должен предоставлять список созданных проектов и задач;

– пользователи должны иметь возможность назначать задачи и проекты другим пользователям и отслеживать их выполнение;

– система должна поддерживать экспорт и импорт задач в различные форматы для удобства обмена данными с другими приложениями;

– обеспечение структуры базы данных, позволяющей эффективное сохранение и доступ к данным;

– администратор должен иметь возможность добавления и удалении новых пользователей;

– администратор должен иметь список всех созданных проектов и задач;

– администратор должен иметь доступ ко всем проектам и задачам;

А теперь обратим внимание на нефункциональные требования.

Нефункциональные характеристики — это описания того, каким образом должен функционировать программный продукт и какими свойствами он должен обладать, чтобы обеспечить необходимую ценность системы в соответствии с её окружающей средой [5].

Нефункциональные требования:

– веб-приложение должно обеспечивать быстрое время отклика при выполнении различных операций, таких как загрузка списка проектов и задач;

– сервис должен обрабатывать большое количество одновременных запросов от пользователей без существенного снижения производительности;

– защита данных пользователей от несанкционированного доступа с помощью механизмов аутентификации;

– возможность масштабирования сервиса для обработки растущей нагрузки;

– минимизация вероятности сбоев и ошибок в работе сервиса;

– сервис должен предоставлять понятный и легко доступный пользовательский интерфейс;

– возможность расширения функциональности сервиса и добавления новых возможностей без существенного изменения базовой архитектуры;

– использование безопасных протоколов для передачи данных между клиентом и сервером

– проведение различных тестирований нашего веб-приложения для выявления соответствия нашим функциональным и нефункциональным требованиям;

– своевременное исправление ошибок и неполадок в работе.

Таким образом, были сформированы основные требования к разрабатываемому интерфейсу сервиса управления задачами.

2.2 Выбор технологии для разработки веб-приложения

В данный момент на рынке присутствует достаточно много различных инструментов, помогающих в реализации веб-приложения, поэтому одной из задач стала необходимость подобрать такой набор инструментов, который будет соответствовать всем требованиям.

Для разработки веб-приложения одним из важных инструментов является фреймворк. Фреймворк — это набор заранее определенных структур, библиотек и правил, способствующих упрощению разработки программного обеспечения. Обычно фреймворк определяет основную архитектуру приложения, предоставляет готовые компоненты для решения типовых задач и стандартизирует процесс разработки. Он предоставляет основу для создания приложений и ускоряет процесс разработки за счет предложения готовых решений для различных типов задач [10].

Использование фреймворка позволяет разработчикам сосредоточиться на бизнес-логике приложения, а не на создании базовых компонентов с нуля. Фреймворк предлагает готовые решения для распространенных задач, что упрощает жизнь разработчиков и снижает вероятность ошибок. Благодаря стандартизации процесса разработки, фреймворк способствует улучшению

качества кода и повышению производительности разработчиков [10].

Рассмотрим некоторые фреймворки, которые дают возможность для работы с базой данных, так как это является важным аспектом для нашего сервиса управления задачами.

Django (Python) – это фреймворк, написанный на языке Python, является полнофункциональным фреймворком для разработки веб-приложений на Python. Представляет набор инструментов и библиотек для создания мощных и масштабируемых веб-приложений, включая автоматическую генерацию административного интерфейса, систему маршрутизации URL, шаблонизацию HTML и многое другое. Он поставляется с встроенной ORM (Object-Relational Mapping), которая упрощает работу с базами данных, такими как PostgreSQL, MySQL, SQLite и другими. Django также предоставляет множество инструментов для работы с данными, включая механизмы администрирования, авторизацию и аутентификацию, а также мощные средства для создания RESTful API.

Flask (Python) – это фреймворк, написанный на языке Python, является легковесным и гибким микрофреймворком для разработки веб-приложений на Python. Представляет базовые инструменты для создания веб-приложений, такие как маршрутизация URL, управление сессиями и шаблонизация HTML, оставляя свободу выбора компонентов и библиотек для разработчика. Flask не включает в себя ORM «из коробки», но интегрируется с различными ORM, что позволяет работать с базами данных.

Ruby on Rails (Ruby) – это популярный фреймворк для разработки веб-приложений на языке Ruby. Обеспечивает полный стек инструментов для быстрой разработки веб-приложений. Он поставляется с встроенным ORM ActiveRecord, который обеспечивает удобную работу с базами данных, такими как MySQL, PostgreSQL и SQLite. RoR также предлагает мощные инструменты для создания и миграции баз данных, а также генерации структуры приложения.

Laravel (PHP) – фреймворк, написанный на языке PHP, является современным фреймворком для разработки веб-приложений на PHP.

Представляет множество инструментов и функций для быстрой и эффективной разработки веб-приложений. Он предоставляет ORM Eloquent для работы с базами данных, такими как MySQL, PostgreSQL и SQLite. Laravel также включает в себя инструменты для миграции баз данных, создания запросов и управления данными.

ASP.NET Core (C#) – это платформа для разработки веб-приложений на языке C#. Представляет мощные инструменты и библиотеки для создания высокопроизводительных и масштабируемых веб-приложений, включая маршрутизацию, систему внедрения зависимостей и многое другое. Она включает в себя Entity Framework Core, который является ORM для работы с базами данных, такими как SQL Server, MySQL, PostgreSQL и другими. ASP.NET Core также предоставляет средства для миграции баз данных и создания RESTful API.

На самом деле в области веб-разработки существует множество фреймворков для разных языков программирования. Но мы рассмотрели только самые популярные и удобные.

Для разработки нашего сервера нам нужно выбрать фреймворк, который лучше всего подойдёт для наших функциональных и нефункциональных требований. Сразу же исключим Flask, так как он уступает другим фреймворкам тем, что у него отсутствует встроенного ORM. А это существенный недостаток по сравнению с другими фреймворками, так как база данных – очень важный пункт для нашего сервиса управления задачами. Также исключим ASP.NET Core, в сравнении с другими фреймворками, такими как Django и Ruby on Rails, данная технология требует больше времени и усилий, чтобы овладеть основами и начать разработку с ASP.NET Core.

Проведём сравнение оставшихся фреймворков для веб-разработки в таблице 3.

Сравнивать будем по следующим критериям: сложность начала работы, гибкость и расширяемость, экосистема и сообщество, производительность и масштабируемость.

Таблица 3 – Сравнение выбранных фреймворков.

Критерии	Django	Ruby on Rails	Laravel
Сложность начала работы	Просто начать благодаря готовой структуре проекта и встроенных компонентов	Низкая, с использованием конвенция и готовой структуры проекта	Умеренная, но требует изучения концепций фреймворка
Гибкость и расширяемость	Менее гибок, но предоставляет множество инструментов и библиотек для разработки	Средняя, предоставляет множество встроенных компонентов, но требует соблюдения конвенций	Умеренная, но может потребоваться расширение для специфических потребностей
Экосистема и сообщество	Большое активное сообщество разработчиков и обширная документация	Большое и активное сообщество разработчиков с обширной документацией	Активное сообщество, но меньше по сравнению с Django и Ruby on Rails
Производительность и масштабируемость	Высокая, особенно подходит для крупных проектов	Высокая, подходит для крупных проектов	Хорошая, поддерживает крупные проекты

Выбор фреймворка остановился на Django. Данный фреймворк включает в себя большинство основных компоненты и функции, необходимые для создания веб-приложения. Это упрощает начало работы и сокращает время разработки. Также предоставляет полный стек инструментов, это позволяет быстро создавать и масштабировать веб-приложения любого уровня сложности. Поставляется с встроенным ORM, с помощью которой облегчается взаимодействие с базами данных, представляя удобный способ работы с

данными на основе объектно-ориентированной модели. Это упрощает создание, чтение, обновление и удаление данных без необходимости написания ручных SQL-запросов.

Для создания нашего веб-приложения также нужны следующие веб-технологии:

– JavaScript (JS) – это высокоуровневый интерпретируемый язык программирования, который широко используется для создания интерактивных элементов на веб-страницах. С помощью JavaScript веб-страницы становятся более динамическими, а точнее, реагировать на пользовательские действия, взаимодействовать с сервером и многое другое. JavaScript является одним из самых популярных языков программирования;

– HTML (HyperText Markup Language) – это стандартизированный язык разметки, используемый для создания структуры веб-страниц. С помощью HTML разработчики могут определять содержание страницы, такое как текст, изображения, ссылки и другие мультимедийные элементы. HTML использует специальные теги для указания различных элементов и их свойств. Браузеры интерпретируют HTML-код и отображают содержимое веб-страницы в соответствии с заданной разметкой. HTML является основным языком для создания веб-страниц и составляет основу для веб-разработки;

– CSS (Cascading Style Sheets) – это язык таблиц стилей, используемый для описания внешнего вида веб-страницы. С помощью CSS разработчики могут определять стиль, расположение и внешний вид элементов HTML на странице, таких как текст, изображения, фоны, рамки и многое другое. CSS позволяет управлять цветами, шрифтами, размерами и прочими визуальными аспектами веб-страницы, что делает ее более привлекательной и функциональной для пользователей. CSS использует селекторы и правила стиля для применения стилей к элементам HTML.

Таким образом, были выбраны основные веб-технологии, а именно Django, JavaScript, HTML и CSS. Веб-приложение будет реализовано с использованием этих технологий. Они являются основой и предоставляют необходимые

инструменты для создания пользовательского интерфейса, взаимодействия с пользователем и обработки данных на клиентской и северной сторонах.

2.3 План разработки веб-приложения

Прежде чем начинать реализацию нашего веб-приложения, нам нужен план. План – это некая упорядоченная последовательность шагов или действий, направленных на достижение определенной цели или результатов.

Важной схемой в планировки любого IT-проекта является диаграмма вариантов использования. Диаграмма вариантов использования (Use Case Diagram) – это вид диаграмм Unified Modeling Language (UML), который используется для визуализации функциональных требований системы с точки зрения взаимодействия между актерами (пользователями или внешними системами) и системой.

На диаграмме вариантов использования показываются различные варианты использования системы в виде прямоугольников, называемых «вариантами использования», а также актеры, которые взаимодействуют этими вариантами использования. Связи показывают, какие действия могут выполнять актеры в системе.

Этот вид диаграммы часто используется на этапе анализа и проектирования системы для определения ее функциональных требований, и он помогает установить, как пользователи будут взаимодействовать с системой и какие функции им будут доступны.

На рисунке 4 предоставлена диаграмма вариантов использования для нашего проекта.

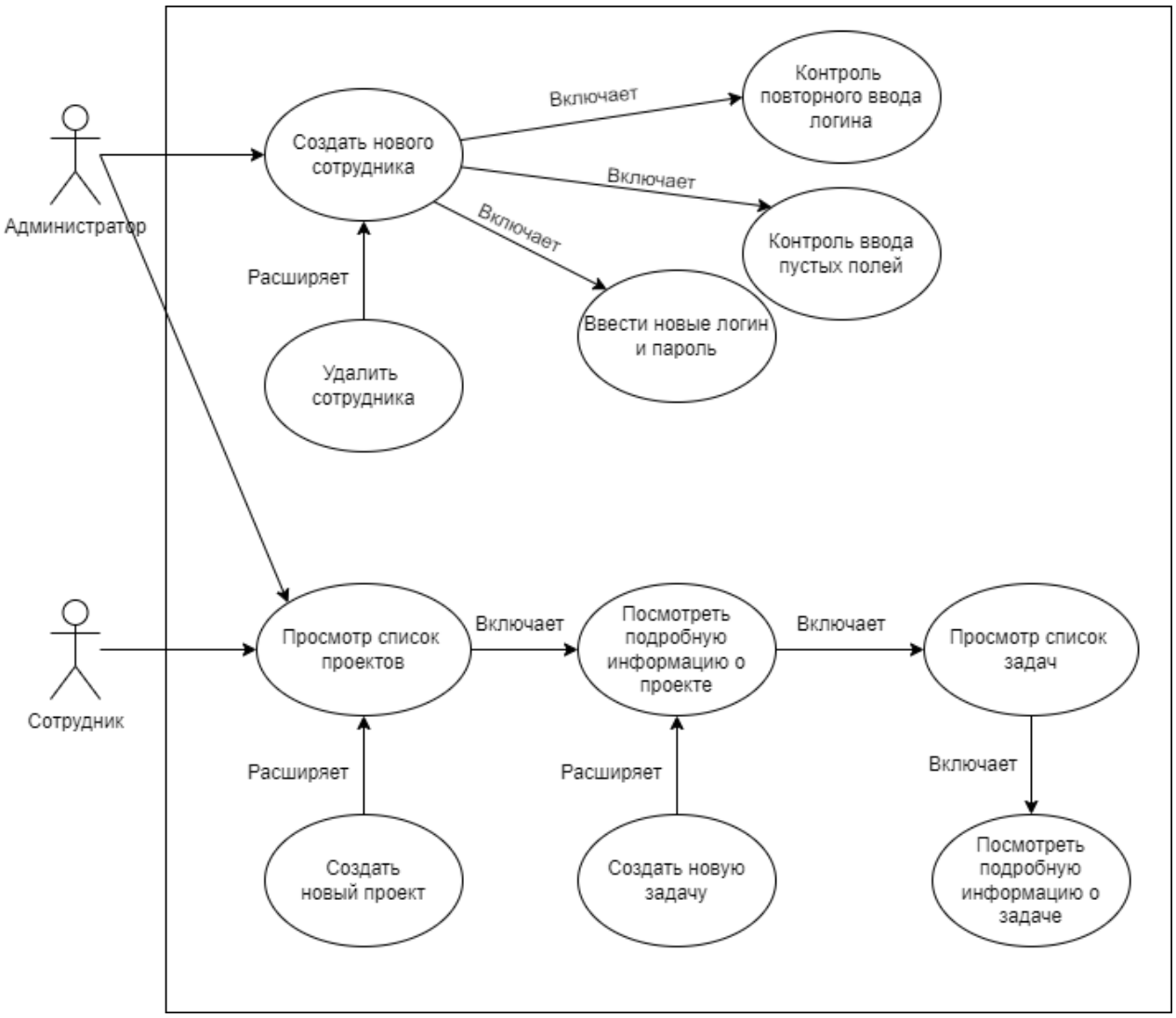


Рисунок 4 – Use Case Diagram

У нашего веб-сайта, как и у каждого веб-приложения, первая страница – это раздел авторизации, это важный шаг для начала работы нашего сайта, так как наш сервис будет доступен только для работников организации.

Пользователи нашего сервиса делятся на два типа. Администратор, а точнее, проектный менеджер, который будет иметь доступ ко всей нужной информации. У него будет доступ к информации обо всех проектах, задачах и сотрудниках, также и подробная информация обо всех проектах и задачах. Также именно администратор будет создавать новых сотрудников, данное решение обезопасит доступ посторонних лиц к нашему сервису. В качестве второго типа пользователя выступают обычные сотрудники, которые также

делятся на два типа, а именно на TeamLead, который является ответственным за конкретный проект, и Employee, который является обычным сотрудником, которому и будут даваться различные задачи.

Давайте рассмотрим все возможные функции для каждого типа пользователя.

Возможности администратора:

– создавать новых пользователей сервиса, указывая имя, логин, пароль и должность сотрудника в компании;

– доступ к списку всех проектов, а также к подробной информации каждого проекта;

– доступ к списку всех задач, а также к подробной информации каждой задачи;

– доступ к списку всех сотрудников;

Возможности сотрудников:

– доступ к списку своих проектов, а также к подробной информации каждого проекта;

– доступ к списку своих задач, а также к подробной информации каждой задачи;

Таким образом, структура нашего приложения будет разделена на три рабочих экрана.

– Экран авторизации.

– Рабочий экран администратора.

– Рабочий экран сотрудников.

Важно отметить, что рабочий экран сотрудников будет также доступен для администратора. Так некоторые функции, как просмотр подробной информации каждого проекта или задачи, пересекаются между пользователями.

2.4 Разработка пользовательского интерфейса веб-приложения

После того как определили типы пользователей и основные экраны каждого пользователя, нужно разработать пользовательский интерфейс для каждого рабочего экрана.

У пользовательского интерфейса также есть свои основные принципы, которых нужно придерживаться. Прежде чем начнем разработку пользовательского интерфейса, рассмотрим основные принципы, которые будут учтены при разработке нашего интерфейса.

Основные принципы пользовательского интерфейса:

– принцип ясности, то есть интерфейс должен быть понятным и легко читаемым;

– принцип минимализма, то есть дизайн должен быть понятным и фокусированным на ключевых функциях;

– принцип консистентности, то есть нужно сохранять единый стиль и поведение элементов интерфейса на всех страницах и экранах нашего веб-сайта. Это помогает пользователю быстрее ориентироваться и понимать, как взаимодействовать с приложением;

– принцип обратной связи, например, отображение сообщения об успешном выполнении операции или об ошибках ввода данных.

Благодаря этим основным принципам при разработке пользовательского интерфейса дизайн будет более удобный и интуитивно понятный. Теперь перейдём к проектированию шаблона каждого рабочего экрана.

Первым делом было спроектировано экран авторизации, на данном экране у нас имеется заголовок, два поля ввода для того, чтоб пользователь ввел свои данные, а именно логин и пароль, и кнопка для того, чтобы войти в сервис. Прежде чем пользователь получит доступ к сервису, данные проверяются с данными, которые хранятся в базе данных. Экран авторизации показан на рисунке 5.

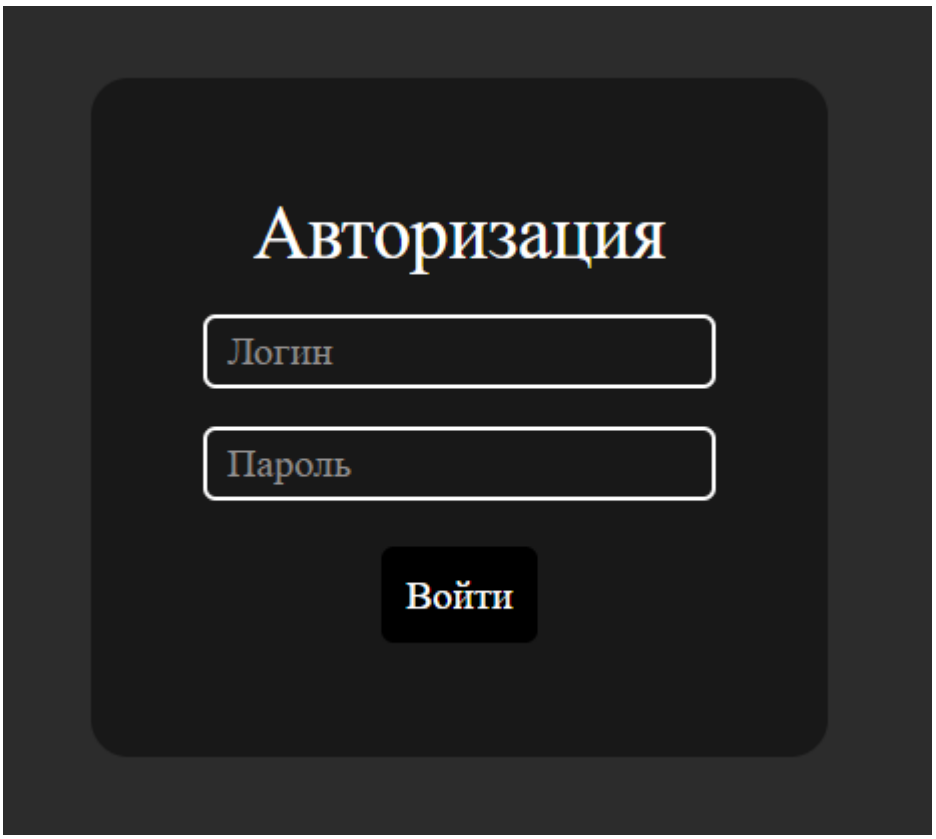


Рисунок 5 – Экран авторизации.

Если пользователь введет неверные или несуществующие данные, то на экран выведется надпись: «Пользователь не найден».

Второй рабочий экран будет уже только для администраторов. У администратора должна быть самая важная функция, а именно возможность создавать новых пользователей, а точнее сотрудников, указывая необходимые данные. Также у администратора должна быть вся информация о всех действиях, которые происходят на сайте, то есть информация о всех проектах, задачах и, конечно же, о всех сотрудников. Поэтому было принято решение поделить один контейнер на два важных контейнера. Рассмотрим контент каждого контейнера отдельно.

С помощью первого контейнера, который показан на рисунке 6, у администратора будет возможность добавить нового пользователя. Чтобы создать аккаунт для нового сотрудника, администратор заполняет необходимые поля, а именно имя, логин, пароль и выбирает должность сотрудника. После

нажатия на кнопку «Зарегистрировать» данные нового пользователя будут отправлены сервер и сохранены в базе данных. После чего новый сотрудник сможет авторизоваться на сайте, используя первый рабочий экран.

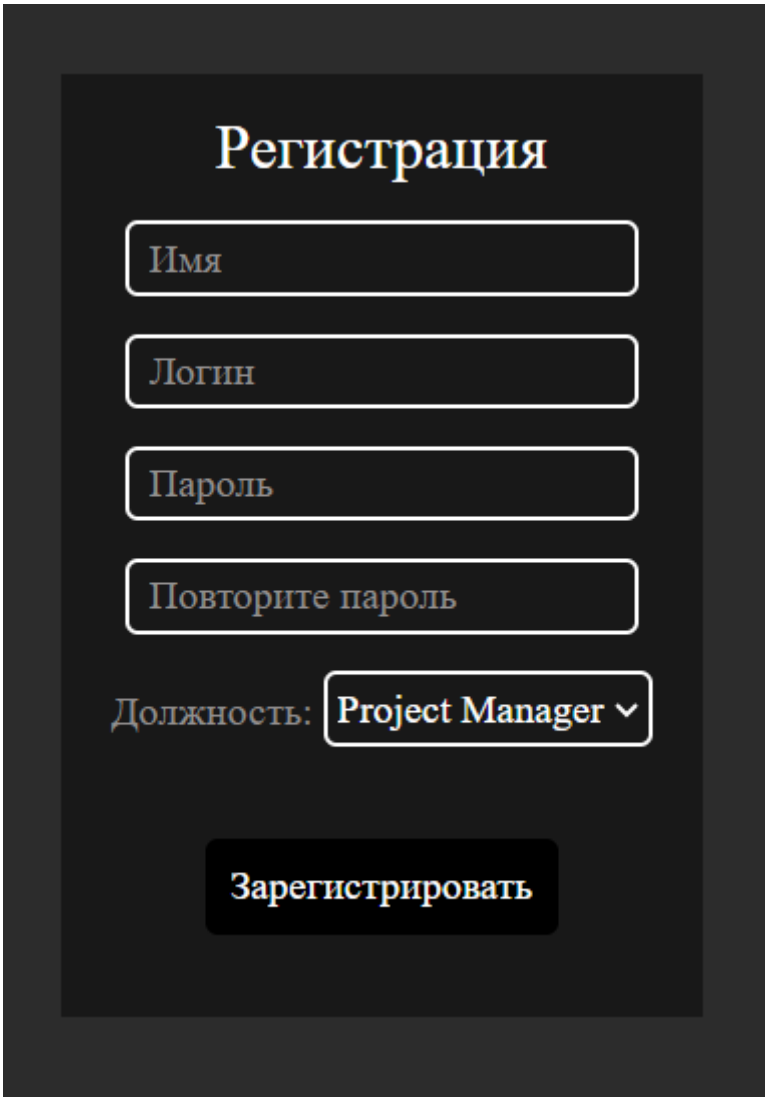


Рисунок 6 – Регистрация новых пользователей.

На втором контейнере, который продемонстрирован на рисунках 7, будет располагаться вся информация о проектах и задачах, а также о всех сотрудников, у которых есть доступ к данному сервису.

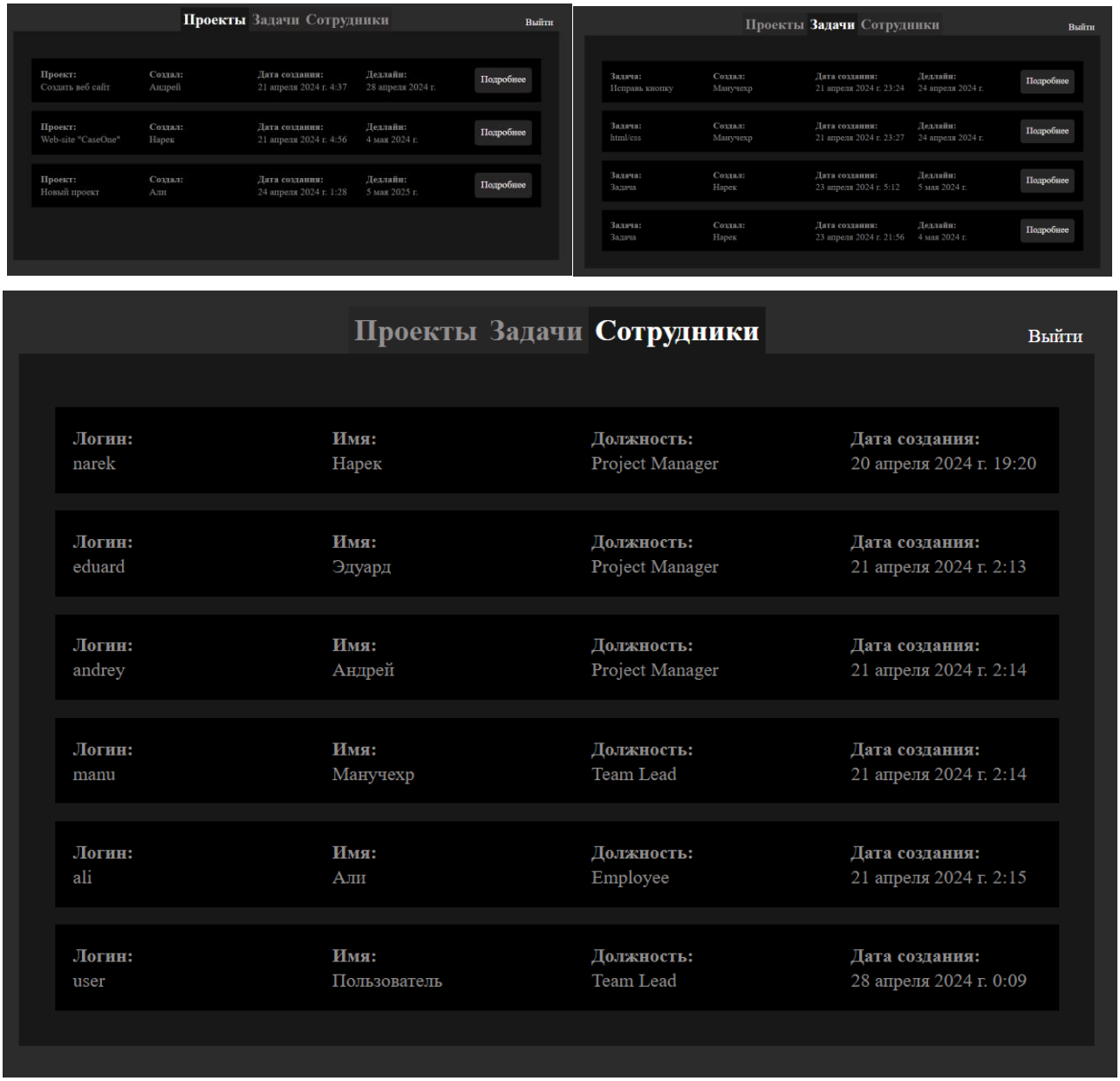


Рисунок 7 – Контейнер с разделами «Проекты», «Задачи» и «Сотрудники».

Данный контейнер состоит из трех разделов: «Проекты», «Задачи» и «Сотрудники». Разделы «Проекты» и «Задачи» рассмотрим более подробно, когда будем рассматривать третий рабочий экран для сотрудников, так как данный функционал у них идентичный.

В разделе «Сотрудники» администратор может посмотреть всех пользователей, которых он же и создал. В данном разделе про каждого сотрудника написана информация о логине, имени, должности и дата создания.

С помощью данного раздела администратор сможет посмотреть всех

сотрудников, после чего найти аккаунт сотрудника, который уже не является частью команды компании, что делает данный сервис более безопасным и менее доступным чужим пользователям, или же проверить, создавал ли администратор аккаунт нового сотрудника.

В правом верхнем углу расположилась кнопка под названием «Выйти», с помощью данной функции пользователи смогут выйти из своего авторизованного аккаунта.

Если администратор нажмет на кнопку «Подробнее» на вкладках «Проекты» или «Задачи», то он перейдет на рабочий экран для сотрудников, так как контент в данном случае у них пересекается. Чтобы обратно возвращаться в рабочий экран для администрации, необходимо просто нажать на кнопку «Админ», показанный на рисунке 8. После нажатии на эту кнопку администратор вернется в свой рабочий экран.

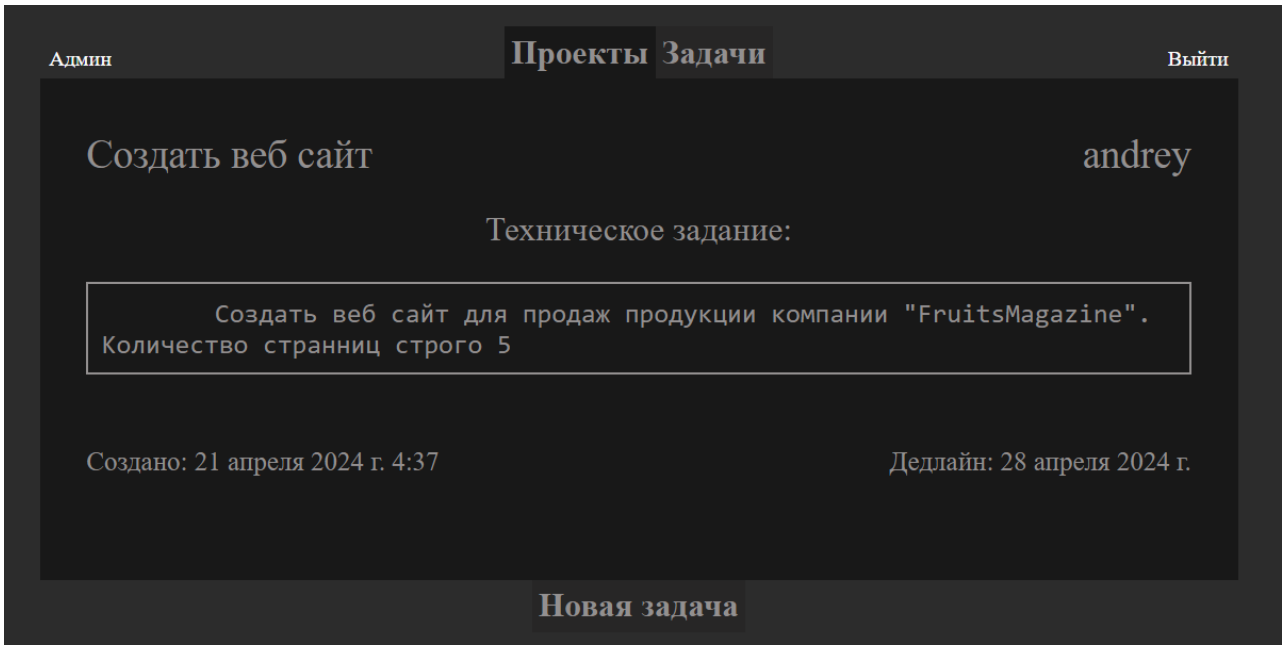


Рисунок 8 – Подробная информация о проекте.

На рисунке 9 показано, как данные два контейнера выглядят вместе на сайте.

Рисунок 9 – Основной экран администратора.

Последним основным экраном для нашего веб-приложения является основной экран для сотрудников, который вы можете увидеть на рисунке 10.

Рисунок 10 – веб-страница «Проекты».

Первым делом пользователь видит список проектов, в которых у него есть задачи. Также на данной веб-странице есть выпадающий раздел, который показан на рисунке 11, с помощью которого сотрудник может создать новый проект, указывая название проекта, техническое задание, крайний срок и, конечно же, выбирает ответственного за этот проект

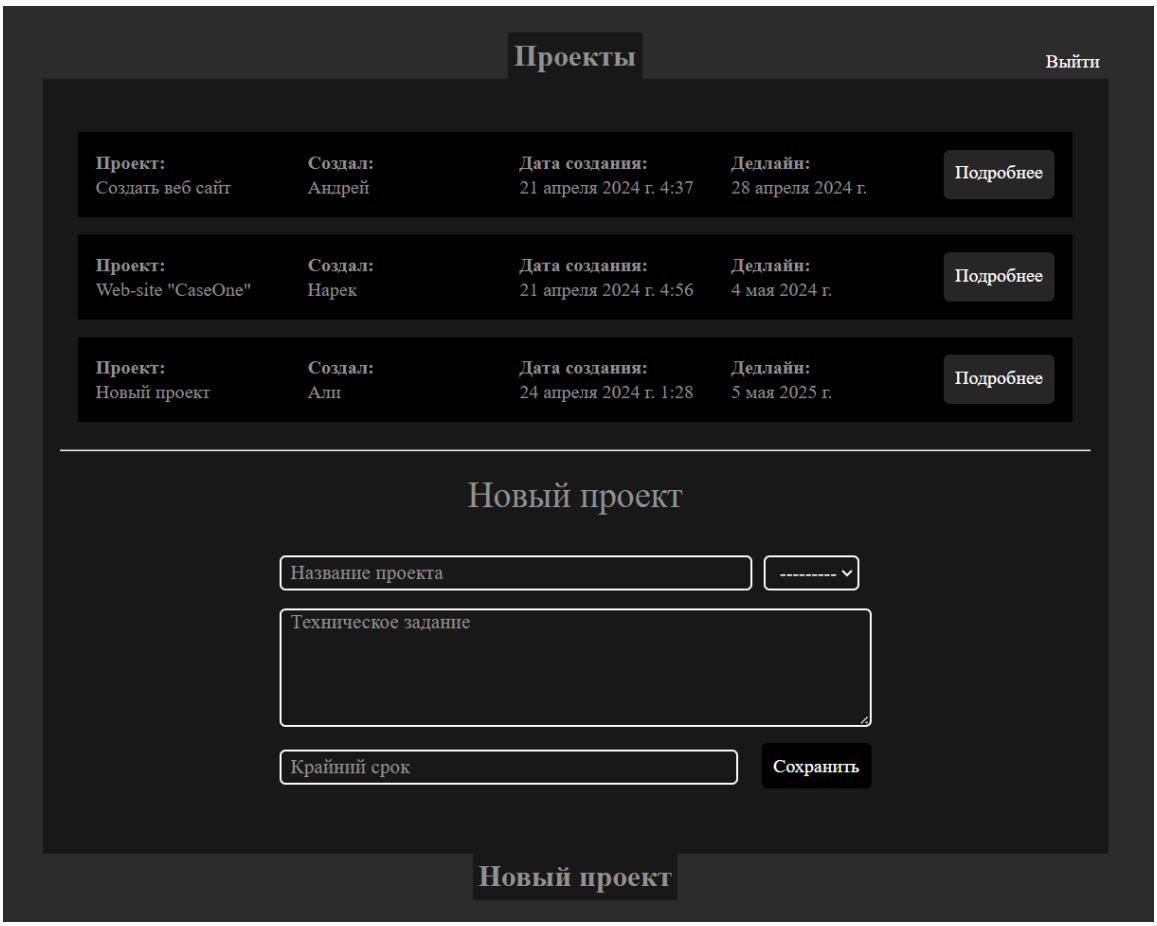


Рисунок 11 – веб-страница «Проекты» с выпадающим разделом «Новый проект».

После того как пользователь нажмет на кнопку «Подробнее», он перейдет на веб-страницу, где вся информация о проекте, которая вводилась при создании. Данную веб-страницу мы уже рассматривали на рисунке 8. Но у данной страницы также есть выпадающий раздел «Новая задача», который идентичен с разделом «Новый проект», рисунок 12.

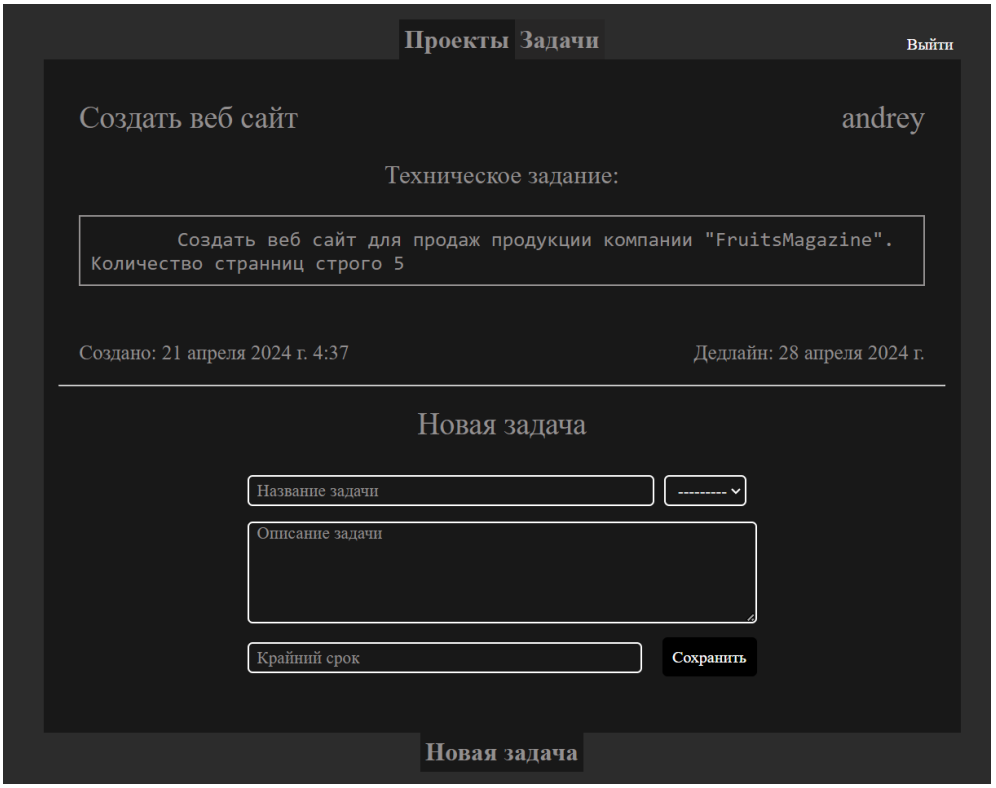


Рисунок 12 – Подробная информация о проекте с выпадающим разделом «Новая задача».

На данной странице также есть немаловажный раздел, который называется, как «Задачи», показанный на рисунке 13. Данный раздел на самом деле схож с разделом «Проекты», который мы уже рассмотрели.

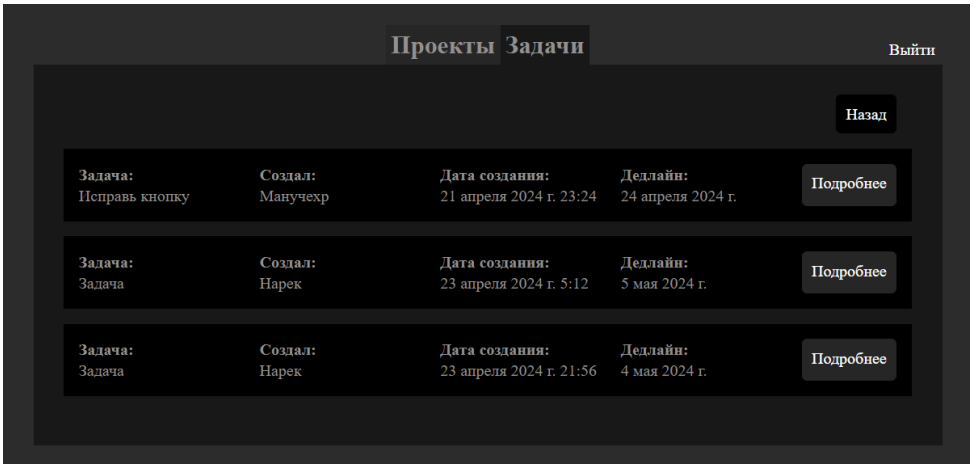


Рисунок 13 – Раздел «Задачи».

Если нажать на кнопку подробнее, то пользователь перейдет на следующую веб-страница, где будет подробная информация о задаче, а именно название, создатель, описание, когда было создано и до какого числа нужно сделать и, конечно же, статус выполнения задачи. Данная страница показана на рисунке 14.

Рисунок 14 – Подробная информация о задаче.

У данного раздела также есть выпадающий раздел под названием «Документ», который предоставлен на рисунке 15. В данном разделе исполняющий задачу сотрудник может загрузить документ, который является решением данной задачи. После того как пользователь нажмет на кнопку отправить, статус задачи автоматически поменяется с «создано» на «отправлено».

Рассмотрев все экраны и разделы нашего веб-сайта, можно отметить, что разработанный интерфейс полностью подходит под основные принципы.

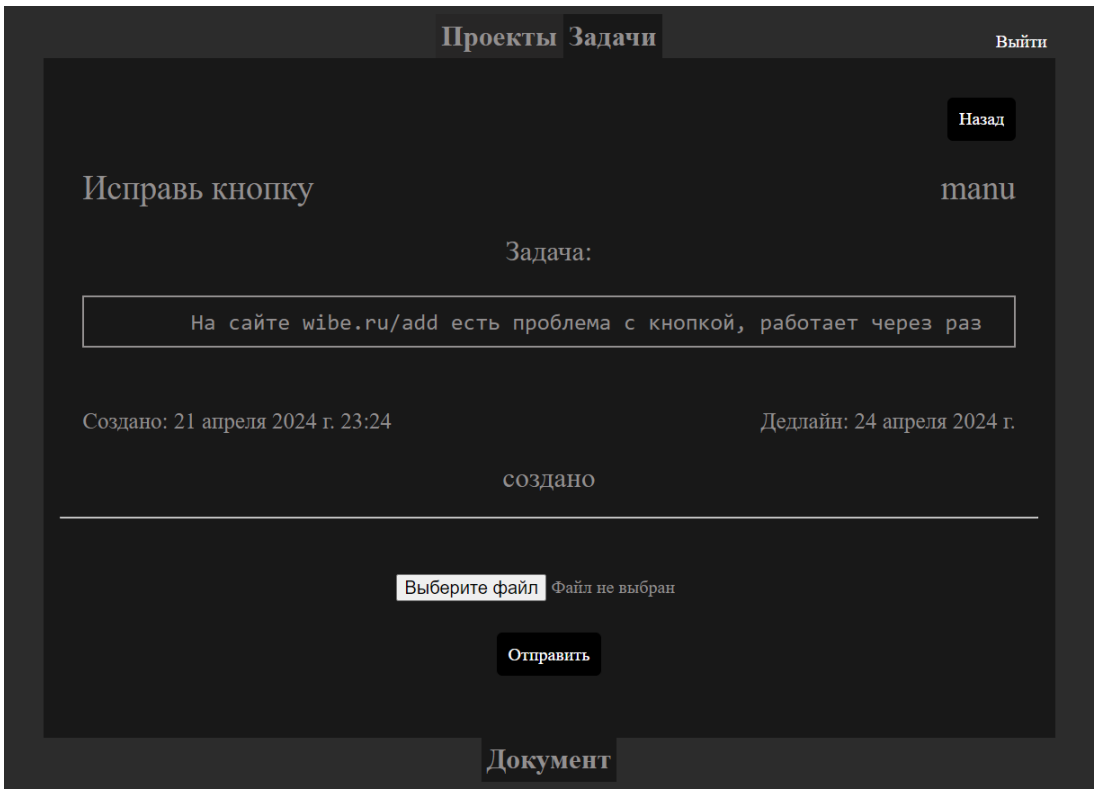


Рисунок 15 – Подробная информация о задаче с выпадающим разделом «Документ».

В этой главе были установлены основные требования к разрабатываемому сервису. Выбрали основные технологии для разработки нашего веб-приложения, также и план. Разработали пользовательский интерфейс приложения с соблюдением основных принципов.

Глава 3 Реализация сервиса управления задачами.

3.1 Выбор системы управления базы данных

Для того чтобы сохранять проекты и задачи, в нашей реализации будет задействована база данных. У фреймворка Django также есть встроенная ORM (Object-Relational Mapping) для взаимодействия с базой.

ORM (Object-Relational Mapping) – это техника программирования, которая связывает базы данных с объектно-ориентированными языками программирования, позволяя разработчикам работать с базой данных, используя объекты и методы, вместо SQL-запросов [15].

При разработке нашего сервиса, который требует эффективного управления данными, выбор подходящей системы управления базами данных (СУБД) играет ключевую роль в обеспечении производительности, надежности и масштабируемости приложения. Ниже рассмотрены наиболее известные и популярные БД, которые можно использовать для наших задач.

MySQL – это открытая реляционная система управления базами данных, которая широко используется в веб-разработке и других областях. Она обладает высокой производительностью, а также предлагает широкий спектр функциональных возможностей.

SQLite – это компактная и встраиваемая СУБД, которая не требует отдельного серверного процесса и работает непосредственно с файлами данных. Отличительной особенностью является ее уникальная архитектура, которая позволяет работать непосредственно с файлами данных, без необходимости запуска отдельного серверного процесса.

Microsoft SQL Server – это полнофункциональная реляционная система управления базами данных, разработанная Microsoft. Она предлагает широкий набор возможностей для работы с данными, включая транзакции, репликацию, аналитические функции и интеграцию с другими продуктами Microsoft. SQL Server доступен как вариант для установки на сервере, так и в облачной модели

(Azure SQL Database).

Сравним СУБД в таблице 4.

Таблица 4 – Сравнение СУБД.

Критерий	MySQL	SQLite	Microsoft SQL Server
Удобство эксплуатации	+	+	+
Гибкость настройки	+	+	+
Высокая производительность	+	+	+
Широкий функционал	+	+	+
Опыт работы	+	+	+
Простота реализации	-	+	-
Итого:	5	6	5

Проведя анализ известных и популярных БД, которые можно использовать для наших задач, приходим к выводу, что целесообразно использовать СУБД SQLite. Важно отметить, что SQLite предоставляется по умолчанию в фреймворке Django.

После того как определились с СУБД, которое будем использовать для реализации нашего сервиса, необходимо разработать физическую модель базы данных, на основе спроектированной ранее логической модели данных (рисунок). Физическая модель базы данных описывает, как данные фактически хранятся в базе данных. Это включает информацию о структуре данных, типах данных, индексах, ключах, ограничениях целостности и других аспектах, связанных с физическим расположением и организацией данных на уровне конкретной СУБД [20].

На рисунке 16 представлена физическая модель базы данных, которая включает в себя все таблицы, их поля и связи между ними. Каждая таблица представлена в виде прямоугольника, внутри которого указаны названия полей. Связи между таблицами обозначены линиями, указывающими на связанные

поля и тип связи.

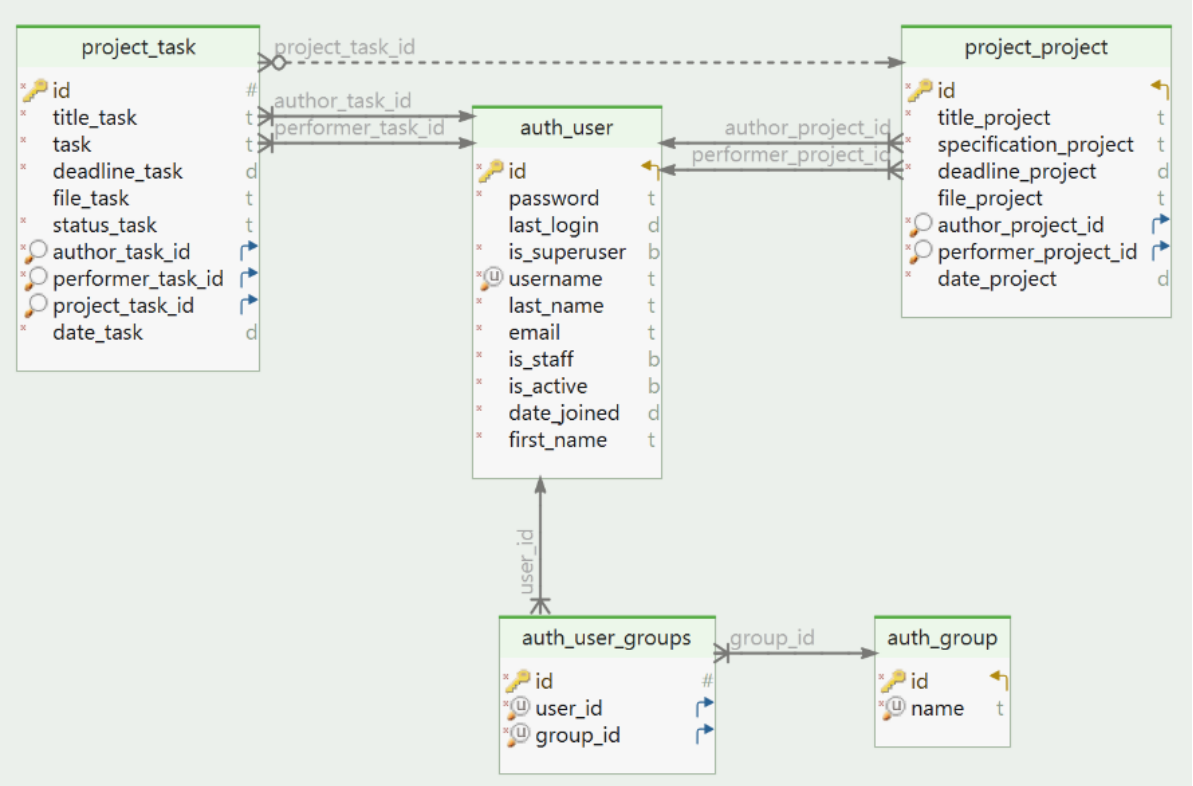


Рисунок 16 – Физическая модель базы данных

Таким образом, было выбрано СУБД SQLite, и разработана физическая модель базы данных.

3.2 Архитектура разрабатываемого веб-приложения при помощи фреймворка «Django»

При реализации веб-приложения с помощью фреймворка «Django» процесс разработки упрощается за счет организации кода на функциональные блоки [22]. Данные блоки представляют собой некий каркас проекта с готовой архитектурой, имеющий необходимые инструменты для реализации веб-приложения.

Более подробно осмотрим работу приложения на Django с помощью схемы, показанной на рисунке 17.

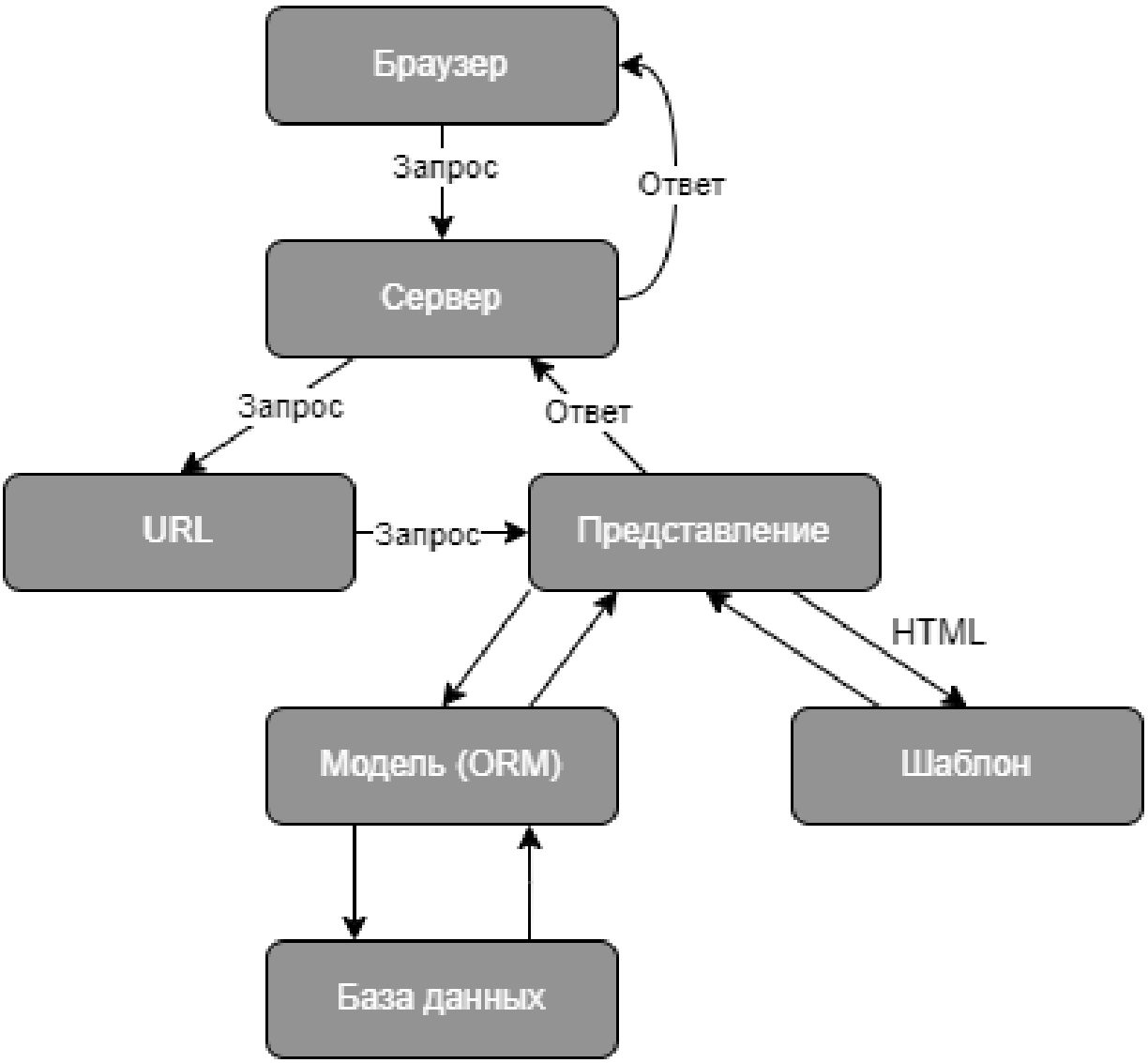


Рисунок 17 – Схема работы Django-приложения

Принцип работы Django-приложения основан на концепции MVC (Model-View-Controller) или ее варианте MVT (Model-View-Template), где модель отвечает за бизнес-логику и доступ к данным, представление отображает данные пользователю, а шаблоны определяют внешний вид страницы. Django обеспечивает мощные инструменты для быстрой и безопасной разработки веб-приложений.

Наш веб-сайт будет использовать концепцию MVT (Model-View-Template).

Model (модель) отвечает за представление данных и бизнес-логику нашего

приложения. Модель определяет структуру данных, хранит и извлекает информацию из базы данных, а также выполняет валидацию данных.

View (представление) отвечает за обработку запросов от пользователя, взаимодействие с моделью для получения необходимых данных и формирование ответа. Представления могут возвращать HTML-страницы, JSON-ответы или другие форматы данных, которые будут отображаться пользователю.

Template (шаблоны) отвечает за отображение данных пользователю. Шаблоны представляют собой HTML-файлы с вставками кода шаблонизатора Django, которые позволяют вставлять данные из представлений и моделей в HTML-разметку. Шаблоны облегчают разделение логики представления данных и их отображения, что делает код более чистым и поддерживаемым.

В целом, в MVT каждое слово имеет свою специфическую функцию. Модель управляет данными и бизнес-логикой, представление обрабатывает запросы и формирует ответы, а шаблон определяет внешний вид данных, отображаемых пользователю

Структура нашего проекта разработки веб-приложения представлена на рисунках 18.

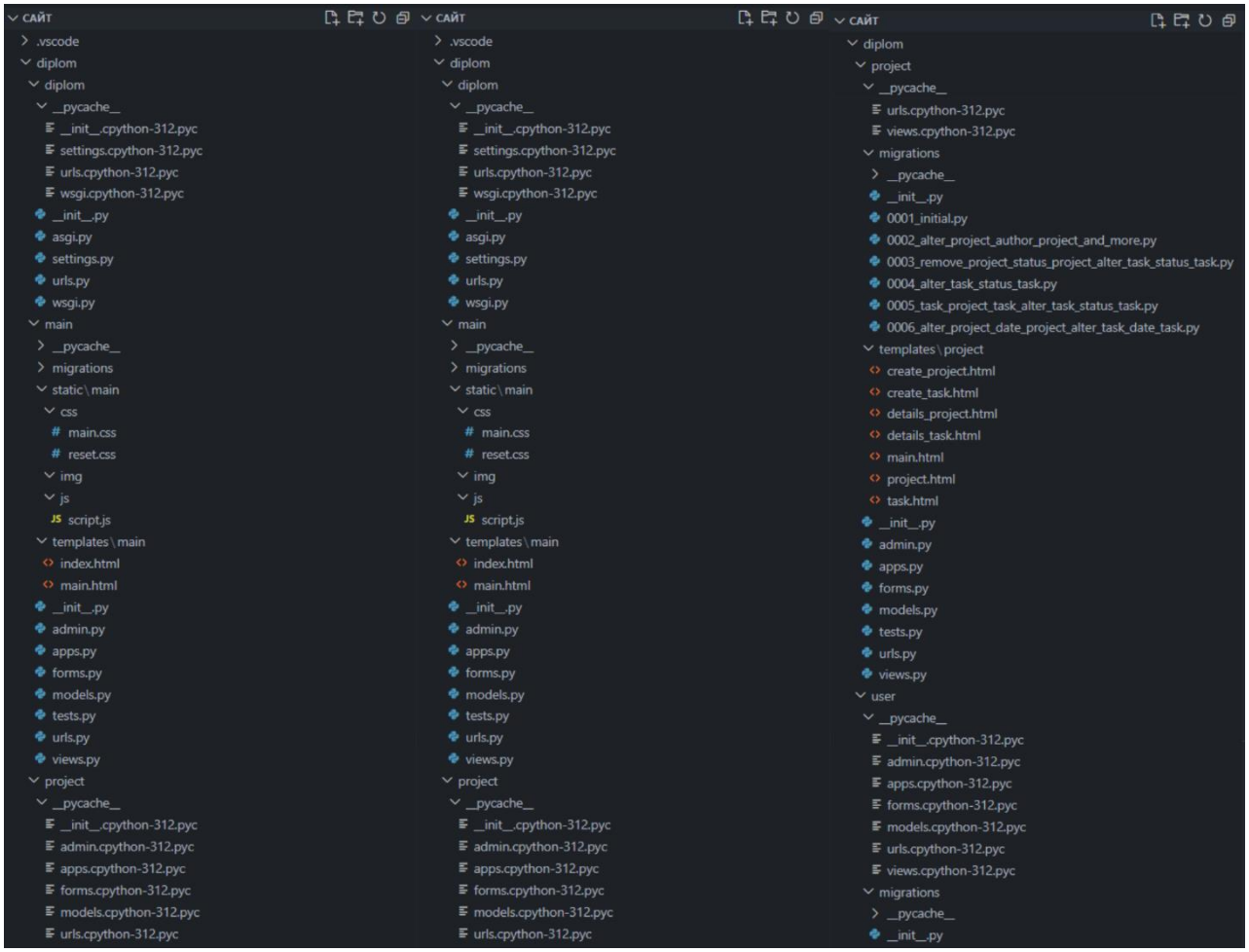
При создании проекта Django главными файлами являются следующие:

– settings.py, содержит основные настройки проекта Django, такие как базы данных, пути к статическим и медиафайлам, список установленных приложений, настройки безопасности и многое другое. Также можно определить переменные окружения, чтобы скрыть конфиденциальную информацию, такую как секретный ключ;

– urls.py, определяет маршрутизацию URL-адресов для вашего приложения. Он связывает URL-адреса с представлениями, которые обрабатывают запросы, и определяет, какие действия будут выполняться при запросе на определенный URL. Этот файл является основным для реализации маршрутизации в Django;

– wsgi.py и asgi.py, стандартные интерфейсы между веб-сервером и

вашим приложением Django. В этих файлах определяются приложения, которые будут обслуживаться сервером.



Рисунки 18 – Реализованная структура проекта.

При создании нового проекта с помощью команды «django-admin startproject <project_name>» эти файлы автоматически создаются в структуре проекта. Они находятся в основной папке проекта, которая имеет то же имя, что и проект.

Также можно заметить, что в нашей реализации имеются три папки, а именно «main», который отвечает за главную страницу авторизации, «project», который отвечает за все действия, связанные с проектами и задачами и «user», который отвечает за все, что связано с управлением пользователями и их аккаунтами.

У каждого приложения есть свои файлы, которые отвечают за его функциональность. Рассмотрим самые важные, которые есть во всех наших приложениях:

– Модели (models.py). В этом файле определяются модели данных, связанные с функциональностью приложения;

– Представления (views.py). В этом файле содержатся представления, которые обрабатывают запросы от пользователей и взаимодействуют с моделями для получения и обработки данных;

– URL-адреса (urls.py). В этом файле определяются маршруты URL, связанные с функциональностью приложения, и их соответствующие представления;

– Шаблоны (templates/). В этом каталоге хранятся HTML-шаблоны страниц, связанные с функциональностью приложения. После создании папки важно создать вторую, которая будет называет также, как и приложения, чтобы проект понимал, из какого именно приложения брать HTML-шаблон;

– Формы (forms.py). В этом файле определяются формы, используемые для ввода и обработки данных в приложении;

– Миграции базы данных (migrations/). В этом каталоге хранятся файлы миграций, которые описывают изменения структуры базы данных, связанные с моделями приложения.

Это самые важные файлы, которые присутствуют во всех трех приложениях нашего проекта. Однако важно понимать, что в других приложениях в зависимости от их функциональности, некоторые могут быть опущены или добавлены дополнительные файлы.

В нашем проекте есть и статические файлы, которые обычно используются для хранения ресурсов, необходимых для визуального оформления веб-страниц, таких как CSS, JavaScript, изображения и другие медиафайлы.

Таким образом, была рассмотрена структура нашего проекта и важные инструменты, с помощью которых и было реализовано веб-приложение.

3.3 Представление и тестирование приложения

Во второй главе были выдвинуты требования, учитывая которые и было разработано наше приложение управления данными. Протестируем каждую функцию последовательно, чтобы убедится в работоспособности каждой функции.

Во время представления конечного результата нашего проекта одновременно проведем проверку работоспособности разработанного интерфейса, а точнее, тестирование его основных функций. Было решено провести тестирование через запущенный сервер приложения сервиса управления задачами.

Любой пользователь, перед тем как начать пользоваться основным контентом, должен пройти авторизацию. После чего наш веб-сайт перенаправляет в зависимости от типа пользователя. Нашим сайтом будут пользоваться два типа пользователя, а именно сотрудник и администратор. У каждого типа пользователя свой рабочий экран, это также было решено во второй главе. Окно авторизации представлено на рисунке 19.

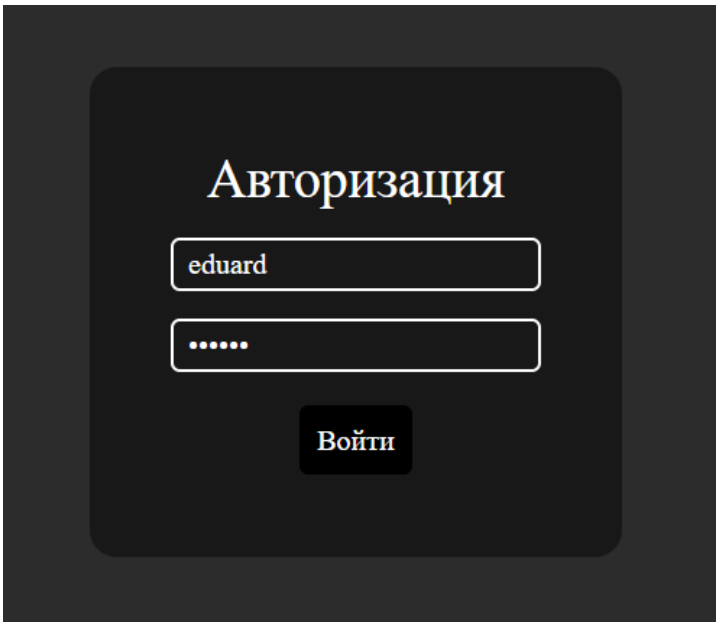


Рисунок 19 – Экран авторизации.

Для авторизации, как и на любых сайтах, нужно ввести свой логин и пароль, который вел администратор при создании аккаунта данного пользователя.

После авторизации браузер переадресовывает пользователя на другую страницу в зависимости от того, к какому типу он относится. Как можем заметить, функция авторизации полностью работоспособна. Давайте рассмотрим рабочий экран для сотрудников. Рабочий экран у сотрудников состоит из одного контейнера, а именно «Проекты». Рабочий экран для сотрудников показан на рисунке 20.

Рисунок 20 – Рабочий экран «Проекты» для сотрудников.

На данном экране пользователь может ознакомиться с подробной информацией любого из списка проекта. Подробная информация показана на рисунке 21.

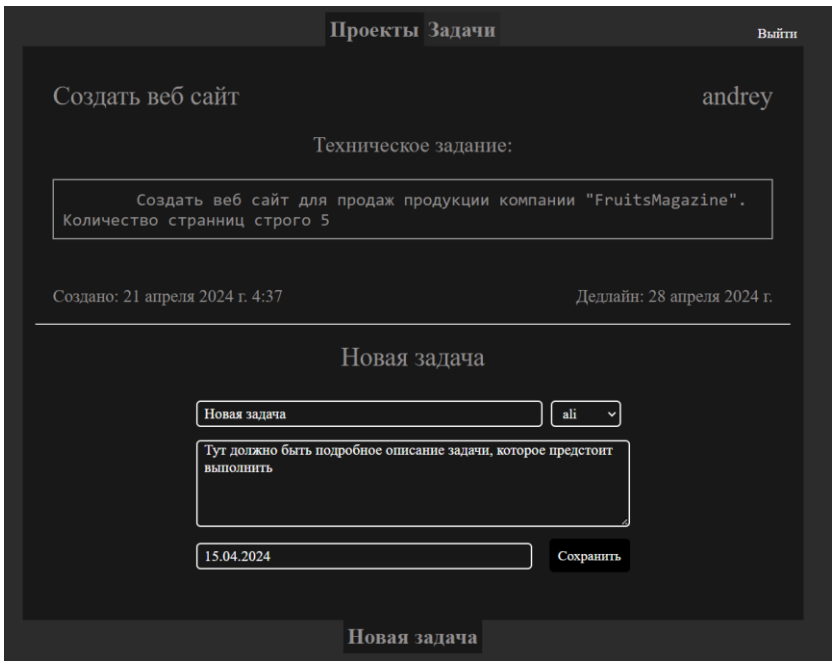


Рисунок 21 – Подробная информация о проекте с выпадающим разделом «Новая задача».

Также на рисунке 21 можно заметить выпадающее контейнер, которое появляется после нажатия на раздел «Новая задача», реализованное с помощью языка программирования JavaScript. Данный раздел доступен только пользователям, которые относятся к группе «Team Lead».

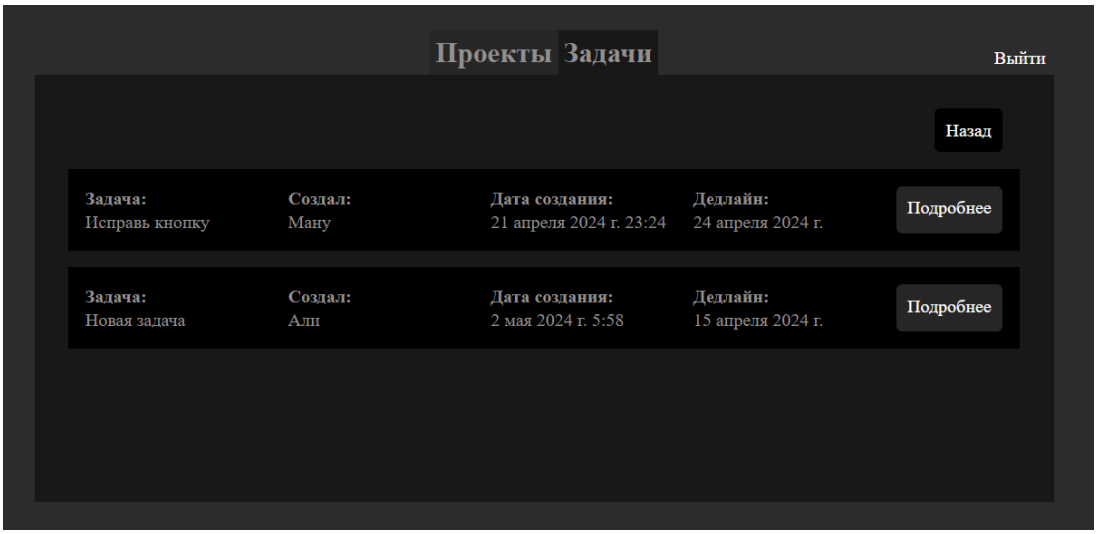


Рисунок 22 – Список задач до выполнения функции «Новая задача».

Создадим новую задачу для проверки работоспособности данной функции, для удобства в качестве исполнителя выбран зарегистрированный пользователь, чтобы могли сразу же посмотреть на результат выполнения данной функции. На рисунке 22 показан список задач до того, как создали новую, а на рисунке 23 – после того, как создали.

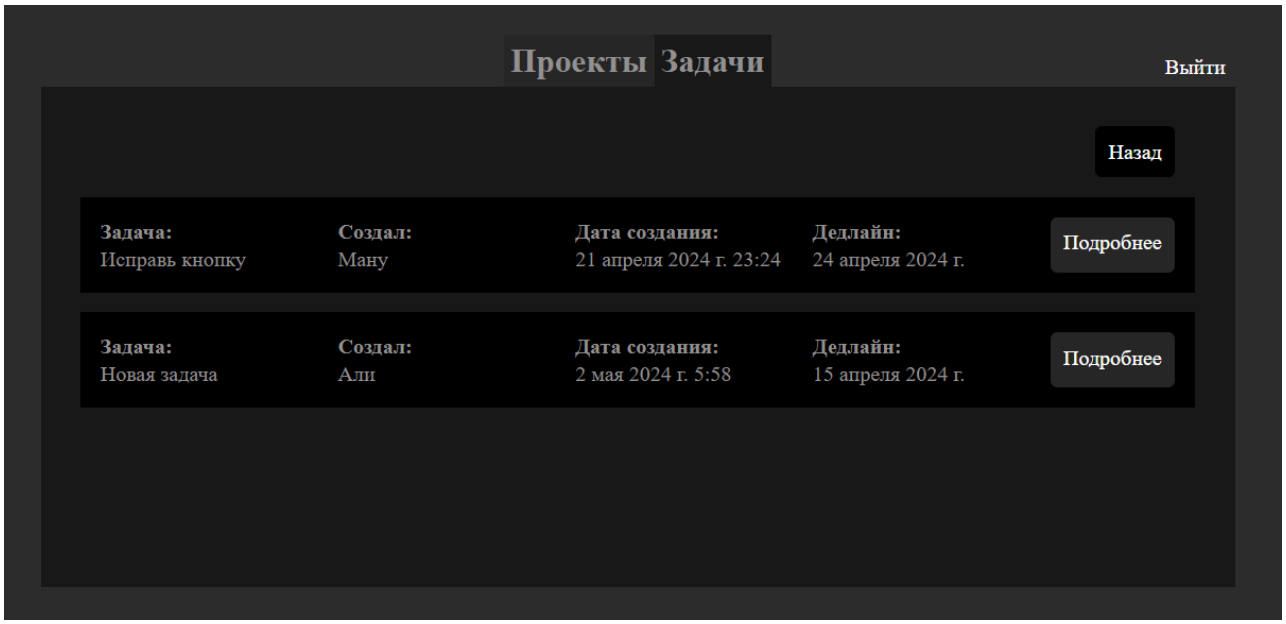


Рисунок 23 – Список задач после выполнения функции «Новая задача».

Функция сработала успешно, и создала нам новую задачу. Если нажать на кнопку подробнее, то откроется новая страница, где будет показана вся подробная информация о задаче с выпадающим разделом «Документ», как показано на рисунке 24.

Любая задача изначально создается со статусом «создано», но после просмотра исполнителем статус меняется на «просмотрено». На этом все экраны для сотрудников и весь функционал были просмотрены и протестированы. Перейдём к следующему типу пользователей, а точнее, администраторы (Project Manager).

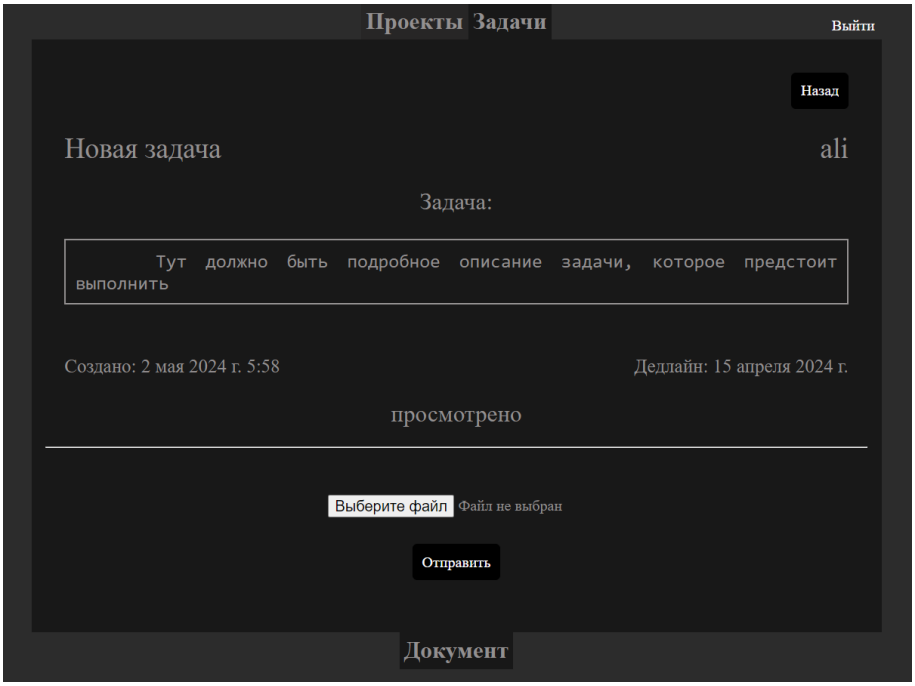


Рисунок 24 - Подробная информация о задаче.

Контент у сотрудников и администраторов пересекается, но начнём по порядку. После того как администратор авторизовался, веб-сайт переадресует его на самый главный экран для сотрудников Project Manager, который недоступен и защищен от обычных сотрудников. Данный рабочий экран предоставлен на рисунке 25.

Рисунок 25 – Главный рабочий экран для администраторов.

Данный экран состоит из двух контейнеров, рассмотрим каждый контейнер отдельно. Начнём с контейнера регистрации, где администратор может создать нового пользователя, который, в свою очередь, является сотрудником. Также важно отметить, что тут же можно создать и нового администратора (Project Manager). Для того чтобы создать нового пользователя, нужно заполнить необходимые поля, а именно поля «Имя», «Логин», «Пароль», «Повторите пароль» и самое главное – выбрать должность нового сотрудника. Для того чтобы проверить данную важную функцию, создадим нового пользователя, у которого будет должность «Employee» с именем «Пользователь», а остальные поля будут равны «user». Результат продемонстрирован на рисунке 26.

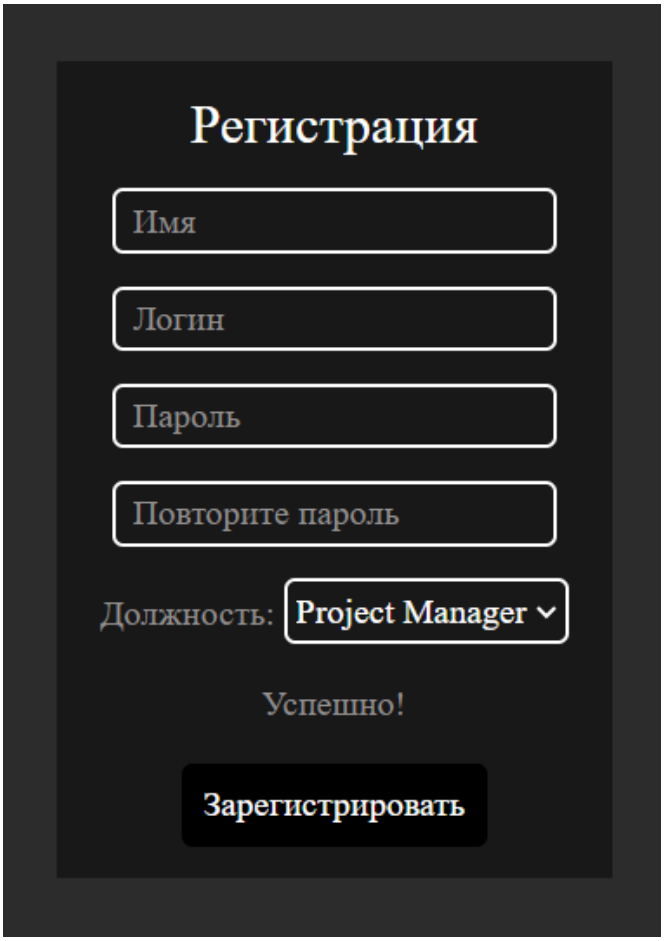
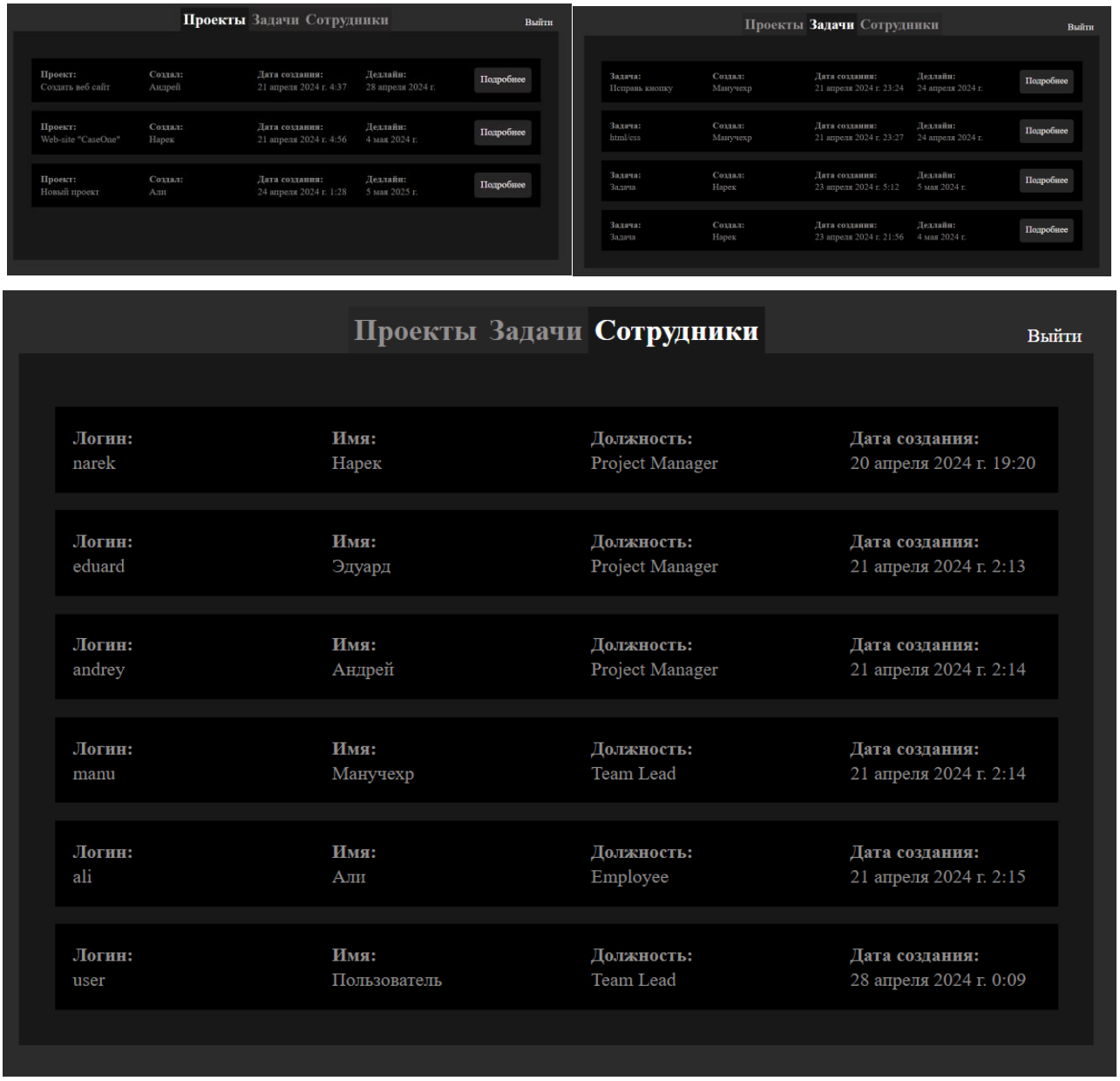


Рисунок 26 – Контейнер «Регистрация» после выполнения функции.

После того как нажали на кнопку «Зарегистрировать», функция вывела сообщение «Успешно!», которое говорит о том, что функция выполнила свою задачу без ошибок и новый сотрудник создан.

Перейдём ко второму контейнеру, который состоит из трех информативных разделов «Проекты», «Задачи» и «Сотрудники», на рисунках 27 более наглядно показан каждый раздел. Данный контейнер сделан динамическим с помощью языка программирования JavaScript.



Рисунки 27 – Контейнер с разделами «Проекты», «Задачи» и «Сотрудники».

Если нажать на кнопку подробнее в разделах «Проекты» и «Задачи», пользователя перенаправит на экраны, которые мы уже рассмотрели, когда обсуждали рабочие экраны для сотрудников. Но все равно осталась одна функция, которую мы так и не испытали, так как она доступна только проектным менеджерам (администраторам). Данная функция находится на экране «Проекты», а именно выпадающий раздел «Новый проект», который был реализован с помощью языка JavaScript. Рабочий экран для пользователей администраторов представлен на рисунке 28.

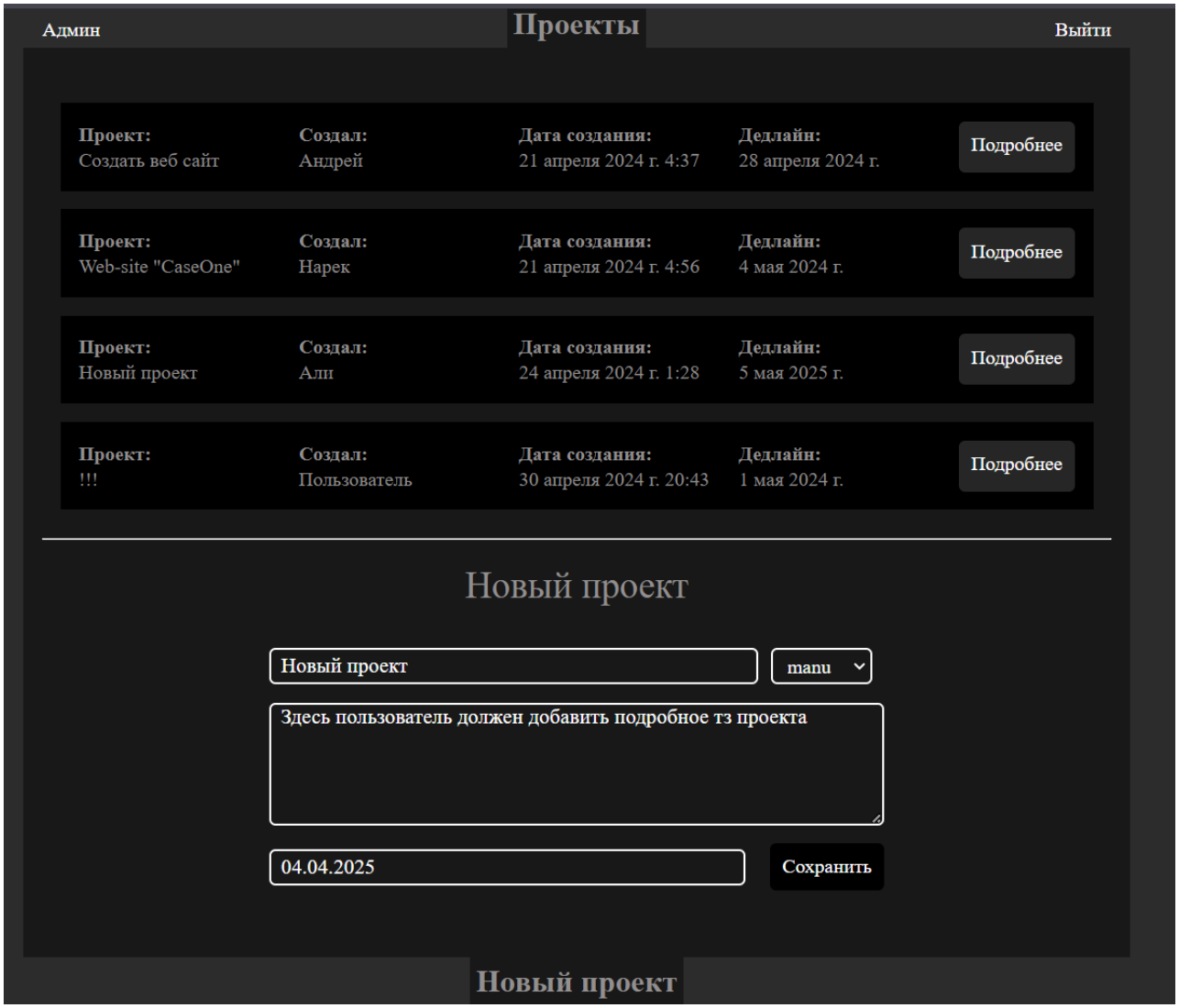


Рисунок 28 – Рабочий экран «Проекты» с выпадающим разделом «Новый проект».

Данный выпадающий раздел доступен только сотрудникам, относящимся к группе «Project Manager» (администратор). После создания нового проекта выпадающий раздел закроется, а новый проект будет добавлен к основному списку проектов, результат данного действия показан на рисунке 29.

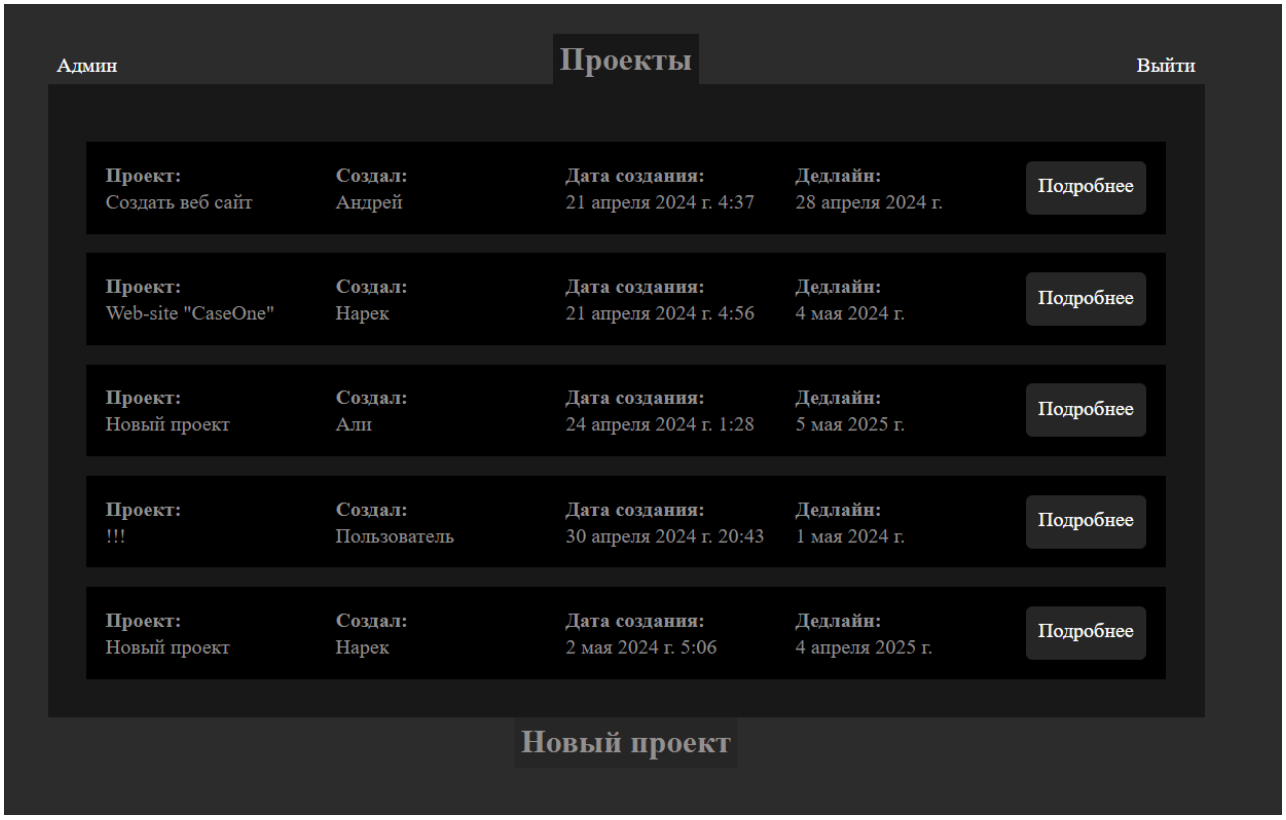


Рисунок 29 – Рабочий экран «Проекты» с новым проектом.

Если нажать на кнопку «Подробнее» данного проекта, то мы увидим те же данные, которые мы вводили при создании. Подробная информация показана на рисунке 30.

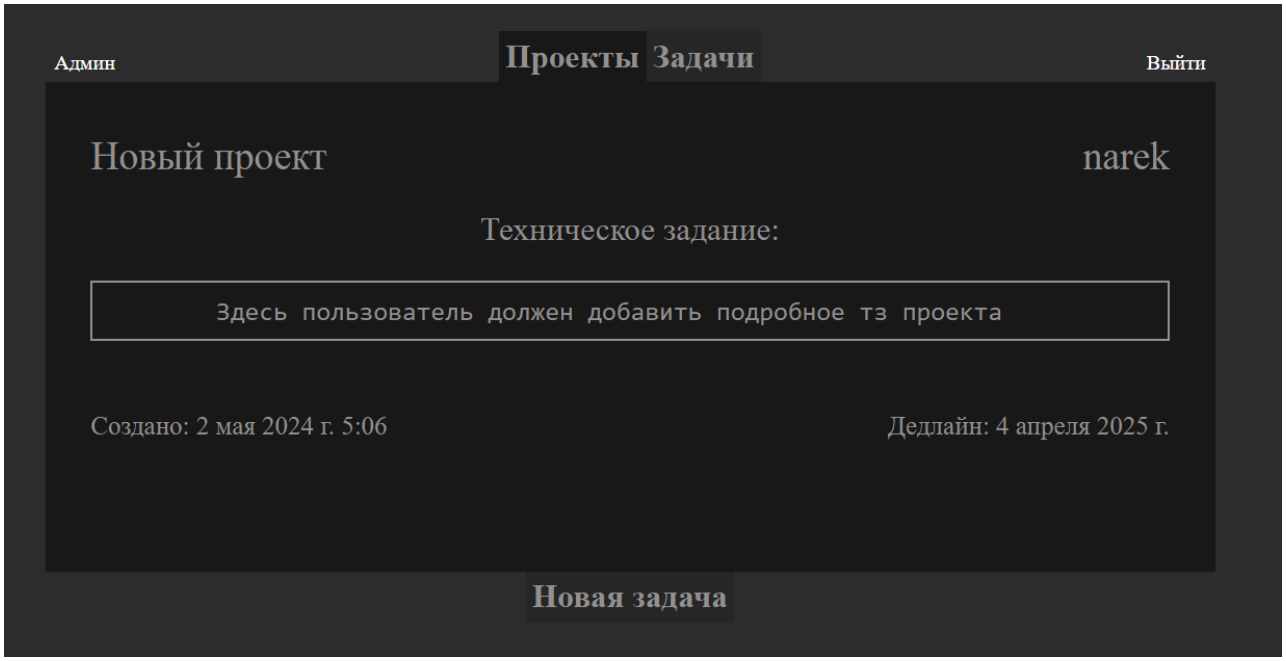


Рисунок 30 – Рабочий экран с подробной информацией о созданном проекте.

Таким образом, после тщательного тестирования всех функций веб-приложения, можно сделать вывод, что все функции работают достойно и без ошибок. А также выполняются все спроектированные в UML Use Case прецеденты.

Заключение

Данная выпускная квалификационная работа посвящена разработке сервиса управления задачами. В процессе выполнения выпускной квалификационной работы был выполнен ряд задач. Был произведен анализ и дана характеристика существующим сервисам по отслеживанию задач, у которых были как бесплатные функции и возможности, так и платные.

Подробно разобраны основные требования к разрабатываемому сервису, сформулированы функциональные и нефункциональные требования для нашего сервиса, а также разработаны такие диаграммы, как «Варианты использования» и «Физическая модель базы данных». Также были проанализированы популярные технологии для разработки веб-приложения, в том числе сравнивали подходящий СУБД, после чего был выбран наиболее подходящий. Описали план разработки и спланировали структуру разрабатываемого сервиса. Также был разработан пользовательский интерфейс с соблюдением всех основных принципов.

Основываясь на описанных требованиях и выбранных технологиях, был разработан гибкий и отказоустойчивый сервис, с помощью которого можно легко и удобно управлять задачами. Языком программирования для реализации сервиса был выбран Python, а точнее фреймворк Django. Тестирование веб-приложения подтвердило его функциональность и эффективность, показав соответствие ожиданиям, как в стандартных, так и в негативных сценариях использования. Тестирование выявило также и высокий уровень стабильности приложения, являющего важным фактором для дальнейшей эксплуатации и развития системы.

Таким образом, разработанный сервис предоставляет собой универсальное решение, которое может быть адаптировано под нужды различных организаций, включая ООО «Квартплата24».

Список используемой литературы

1. База Данных [Электронный ресурс]. URL: https://www.simplethread.com/youre-not-actually-building-microservices/ (дата обращения: 31.03.2024).
2. Вигерс К. Разработка требований к программному обеспечению // К. Вигерс, Д. Битти. – Спб:BHV,2014.-736 с.
3. Галимянов А.Ф., Галимянов Ф.А. Архитектура информационных систем. – Казань: Казан. ун-т, 2019. – 117 с.
4. Ездаков А.Л. Функциональное и логическое программирование. – М.: Бином. Лаборатория знаний, 2009. – 120 с.
5. Официальная документация SQLite [Электронный ресурс]. URL: sqlite.org (дата обращения: 02.05.2024).
6. Справочник по HTML, CSS [Электронный ресурс]. URL: https://htmlbase.ru/ (дата обращения: 10.04.2024).
7. Трутнев Д. Р. Архитектуры информационных систем. Основы проектирования: Учебное пособие. – СПб.: НИУ ИТМО, 2012. – 66 с.
8. "Software Requirements" by Karl Wiegers and Joy Beatty: The Object-Oriented Approach / B. Baesens, A. Backiel, S. Vanden Broucke. – 1st edition, Wrox, 2015.
9. Deitel, H. Java How to Program / H. Deitel, P. Deitel. – 9th edition, Prentice Hall, 2015.
10. Django Software Foundation. Django ORM. [Электронный ресурс]. URL: https://docs.djangoproject.com/en/stable/topics/db/models/ (дата обращения: 21.04.2024).
11. Django Documentation. [Электронный ресурс]. URL: https://docs.djangoproject.com/en/stable/ (дата обращения: 22.04.2024).
12. Hillar, Gaston C. Django RESTful Web Services: The Easiest Way to Build Python RESTful APIs and Web Services with Django. 2018. [Электронный ресурс]. URL:

https://books.google.com/books?hl=en&lr=&id=xNRJDwAAQBAJ&oi=fnd&pg=PP1&dq=Django+for+APIs&ots=afQobhjpO&sig=t0_pDX5mrnYtlIVj35qwQ3xAWAE (дата обращения: 19.02.2024).

13. Hudson O. Getting started with IntelliJ IDEA // O. Hudson, Birmingham: Packt Publishing, 2013. – 114p.

14. Hibernate ORM [Электронный ресурс]. URL: https://hibernate.org/orm/ (дата обращения: 22.04.2024).

15. JWT – как безопасный способ аутентификации и передачи данных [Электронный ресурс]. URL: https://vc.ru/dev/106534-jwt-kak-bezopasnyy-sposob-autentifikacii-i-peredachi-dannyh (дата обращения: 22.04.2024).

16. Krochmalski J. IntelliJ IDEA Essentials // J. Krochmalski. – Birmingham: Packt Publishing, 2014.-263p.

17. Kong API Gateway FAQ [Электронный ресурс]. URL: https://konghq.com/faqs#:~:text=Kong%20Gateway%20is%20the%20world%27s, for%20microservices%20and%20distributed%20architectures. (дата обращения: 22.04.2024).

18. MDN Web Docs (Mozilla Developer Network) [Электронный ресурс]. URL: https://developer.mozilla.org/en-US/docs/Web/HTML (дата обращения: 22.04.2024).

19. Web Development with Django Cookbook" by Aidas Bendoraitis [Электронный ресурс]. URL: https://openjfx.io/javadoc/18/ (дата обращения: 21.04.2024).

20. What is Nginx [Электронный ресурс]. URL: https://www.nginx.com/resources/glossary/nginx/#:~:text=NGINX%20is%20open%20source%20software,for%20maximum%20performance%20and%20stability.(дата обращения: 22.04.2024).