

МИНОБРНАУКИ РОССИИ  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ  
УЧРЕЖДЕНИЕ  
ВЫСШЕГО ОБРАЗОВАНИЯ  
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»  
(ФГБОУ ВО «ВГУ»)

Факультет компьютерных наук  
Кафедра информационных систем

Разработка веб-приложения для создания и управления личными или  
профессиональными целями Stride

Курсовая работа по дисциплине «Технологии программирования»

09.03.02 Информационные системы и технологии

6 семестр 2023/2024 учебного года

Зав. кафедрой _____	к. т. н., доцент Д.Н. Борисов
Обучающийся _____	ст. 3 курса оч. отд. И.С. Харламов
Обучающийся _____	ст. 3 курса оч. отд. Я.А. Березин
Обучающийся _____	ст. 3 курса оч. отд. А.В. Савенкова
Руководитель _____	ст. преподаватель В.С. Тарасов

Воронеж  
2024

## СОДЕРЖАНИЕ

СОДЕРЖАНИЕ .....	2
Введение.....	4
1 Постановка задачи .....	6
1.1 Требования к веб-приложению .....	6
1.1.1 Требования к функциям, выполняемым в веб-приложении .....	6
1.1.2 Требования к программному обеспечению веб-приложения ...	7
1.1.3 Требование к защите информации.....	7
1.1.4 Требования к оформлению страниц веб-приложения .....	7
1.2 Задачи, решаемые в процессе разработки .....	8
2 Анализ предметной области .....	10
2.1 Глоссарий.....	10
2.2 Обзор аналогов.....	11
2.2.1 Microsoft To Do .....	11
2.2.2 Todoist .....	13
2.2.3 Google Tasks .....	15
2.2.4 TickTick.....	17
2.2.5 Any.Do.....	19
3 Реализация .....	21
3.1 Средства реализации .....	21
3.2 Реализация базы данных .....	22
3.3 Реализация серверной части веб-приложения.....	23
3.4 Реализация клиентской части .....	25
3.4.1 Страница авторизации.....	27
3.4.2 Страница регистрации.....	27

3.4.3 Страница сброса пароля.....	28
3.4.4 Основная страница веб-приложения .....	29
3.4.5 Вкладка Today's Tasks.....	30
3.4.6 Вкладка Analytics.....	30
3.4.7 Вкладка Calendar.....	31
3.4.8 Вкладка Lists .....	32
3.5 Диаграммы, иллюстрирующие работу системы.....	34
3.5.1 Диаграмма последовательности.....	34
3.5.2 Диаграмма прецедентов.....	36
3.5.3 Диаграмма активности .....	37
3.5.4 Диаграмма классов .....	39
3.5.5 Диаграмма развертывания .....	41
Заключение .....	42
Список использованных источников .....	43

## Введение

В современном мире, насыщенном информацией и задачами, управление личными и профессиональными целями становится всё более важной составляющей успеха. Люди сталкиваются с необходимостью эффективного планирования, организации и отслеживания прогресса в достижении своих целей. В связи с этим возникает потребность в инструментах, которые помогут оптимизировать этот процесс. Веб-приложения для управления задачами и проектами становятся всё более популярными, предлагая пользователям широкий спектр возможностей для повышения продуктивности.

Данная курсовая работа посвящена разработке веб-приложения "Stride", предназначенного для создания и управления личными или профессиональными целями. Приложение "Stride" предоставляет пользователям гибкий инструмент для планирования, организации и контроля выполнения задач. С помощью "Stride" пользователи смогут создавать задачи и списки задач, задавать им сроки выполнения, приоритеты и периоды повторения. Для более детального отслеживания прогресса предусмотрена возможность создания подзадач. Также пользователи смогут добавлять к задачам описания и теги, что облегчит поиск и фильтрацию информации. Для удобства организации, приложение позволяет создавать теги с пользовательскими цветами. Фильтрация задач возможна как по тегам, так и по сроку выполнения. После выполнения задачи или подзадачи, пользователь может отметить их как завершённые. "Stride" также предоставляет возможность просмотра всех созданных задач в календарном виде, что позволяет визуализировать расписание и сроки выполнения задач.

Актуальность работы обусловлена отсутствием доступного и удобного, но при этом обширного по функционалу приложения для отслеживания задач, применимого как в профессиональной, так и в повседневной среде. Разработка веб-приложения "Stride" отвечает актуальной потребности в инструментах,

которые помогают людям организовывать свои цели, планировать задачи и отслеживать прогресс. Приложение "Stride" способствует повышению личной и профессиональной продуктивности, предоставляя пользователям удобный и функциональный инструмент для управления задачами.

## **1 Постановка задачи**

Задачей курсовой работы является разработка функционального веб-приложения "Stride" для создания и управления личными или профессиональными целями и обеспечивающий удобное взаимодействие с приложением, а также функционал для создания, редактирования, организации и отслеживания задач. Приложение "Stride" будет способствовать повышению эффективности пользователей в достижении своих целей.

### **1.1 Требования к веб-приложению**

#### **1.1.1 Требования к функциям, выполняемым в веб-приложении**

Веб-приложение должно обеспечить неавторизованному пользователю выполнение следующих функций:

- Создание пользовательского профиля;
- Вход в пользовательский профиль;
- Сброс пароля от пользовательского профиля.

Веб-приложение должно обеспечить авторизованному пользователю выполнение следующих функций:

- Выход из пользовательского профиля;
- Создание, редактирование, просмотр и удаление задач;
- Создание, редактирование, просмотр и удаление списков задач;
- Создание и удаление тегов;
- Просмотр аналитической информации по задачам;
- Просмотр календаря задач;
- Получение и просмотр напоминаний о ближайших задачах;
- Поиск задачи по их тегам и названиям;
- Добавление задачам подзадач.

### **1.1.2 Требования к программному обеспечению веб-приложения**

Веб-приложение должно иметь архитектуру, соответствующую модели Клиент-Серверного взаимодействия на основе REST API. Для реализации серверной части приложения будут использоваться следующие средства:

- Фреймворки Spring и Hibernate;
- СУБД PostgreSQL;
- Язык программирования Java.

Для реализации клиентской части приложения будут использоваться следующие средства:

- Язык гипертекстовой разметки HTML;
- Формальный язык описания внешнего вида документа CSS;
- Язык программирования JavaScript;
- Фреймворк React.

### **1.1.3 Требование к защите информации**

Веб-приложение должно обеспечить защиту личных данных пользователей путём хеширования паролей, хранящихся в базе данных, по алгоритму BCrypt в Spring Security.

Сервер должен быть защищён от SQL-инъекций путём применения параметризованных запросов при обращении к базе данных.

### **1.1.4 Требования к оформлению страниц веб-приложения**

Веб-приложение должно быть оформлено в одной цветовой палитре с использованием ограниченного набора шрифтов. У страниц веб-приложения должен быть единый стиль. В оформлении веб-приложения должно присутствовать разработанное название.

Основные цвета веб-приложения:

- Белый цвет (White - #FFFFFF) - используется как цвет навигационных панелей приложения;
- Серый цвет (#EFEFEF) – используется как цвет фона приложения и поисковой строки;
- Черный цвет (Black - #000000) - используется как цвет основного текста, рамок, иконок на навигационной панели.

Вторичные цвета приложения:

- Оттенок оранжевого цвета (#FABB1) - используется как цвет кнопок добавления задачи и иконки напоминаний;

Основной шрифт приложения – Arial (данный шрифт можно использовать в коммерческой и некоммерческой деятельности).

Необходимо корректное отображение веб-приложения в браузерах:

- Google Chrome 122.0.6261.89;
- Microsoft Edge 122.0.2365.66;
- Safari 16.5.2.

## **1.2 Задачи, решаемые в процессе разработки**

Были поставлены следующие задачи:

- Анализ предметной области;
- Обзор аналогов;
- Создание репозитория GitHub и досок в Trello и Miro;
- Утверждение требований: к приложению в целом, к функциям, к структуре, к программному обеспечению, к оформлению и верстке страниц, к защите информации;
- Создание диаграмм: прецедентов, состояний, активностей, последовательностей, классов, развертывания.
- Разработка дизайна приложения;
- Реализация интерфейса приложения;



- Реализация серверной части приложения;
- Развертывание приложения;
- Написание курсовой работы.

## 2 Анализ предметной области

### 2.1 Глоссарий

В настоящей работе используются следующие термины и сокращения с соответствующими определениями:

- frontend – это клиентская часть продукта (интерфейс, с которым взаимодействует пользователь);
- backend – программно-аппаратная часть приложения (логика приложения, скрытая от пользователя);
- серверная часть – это программа, которая обеспечивает взаимодействие клиента и сервера;
- сервер – это устройство, в частности компьютер, которое отвечает за предоставление услуг, программ и данных другим клиентам посредством использования сети;
- задача – основной объект, создаваемый пользователем в программе. Обладает параметрами, такими как статус выполнения, срок выполнения, теги и подзадачи;
- подзадача – дочерняя задача внутри родительской, которую необходимо выполнить для продвижения прогресса выполнения родительской задачи;
- список – объединённые пользователем задачи по какому-либо признаку, будь то срок выполнения, тематика или другое;
- тег – текстовая маркировка задач, позволяющая быстро сортировать и искать их. У задачи может быть несколько тегов или ни одного;
- NLP – обработка естественного языка (Natural language processing). Позволяет извлекать параметры задачи из текста, вводимого пользователем в названии задачи;
- календарное представление – календарь с распределением задач по дням текущего месяца.

## 2.2 Обзор аналогов

На рынке существует много приложений для отслеживания задач. Для определения функционала нашего веб-приложения, мы рассмотрели наиболее популярные по рейтингам и количеству пользователей решения. Далее мы выделили их достоинства и недостатки, чтобы определить, чего может не хватать пользователям существующих решений.

### 2.2.1 Microsoft To Do

Microsoft To Do – полностью бесплатное мультиплатформенное решение от Microsoft, предлагающее весь минимально необходимый функционал для отслеживания задач. Основными особенностями являются интеграция с другими сервисами Microsoft, такими как Outlook, рекомендации при создании задач и NLP.

Достоинства:

- Простой и удобный интерфейс;
- Полностью бесплатное решение;
- Поддержка всех актуальных платформ;
- Возможность прикрепить к задачам текст, фото и файлы;
- Экспорт задач в текстовый файл;
- Система рекомендаций, основанная на истории создания задач.

Недостатки:

- Отсутствие системы приоритетов задач;
- Отсутствие тегов для задач;
- NLP работает только на английском языке;
- Отсутствие календарного представления задач.

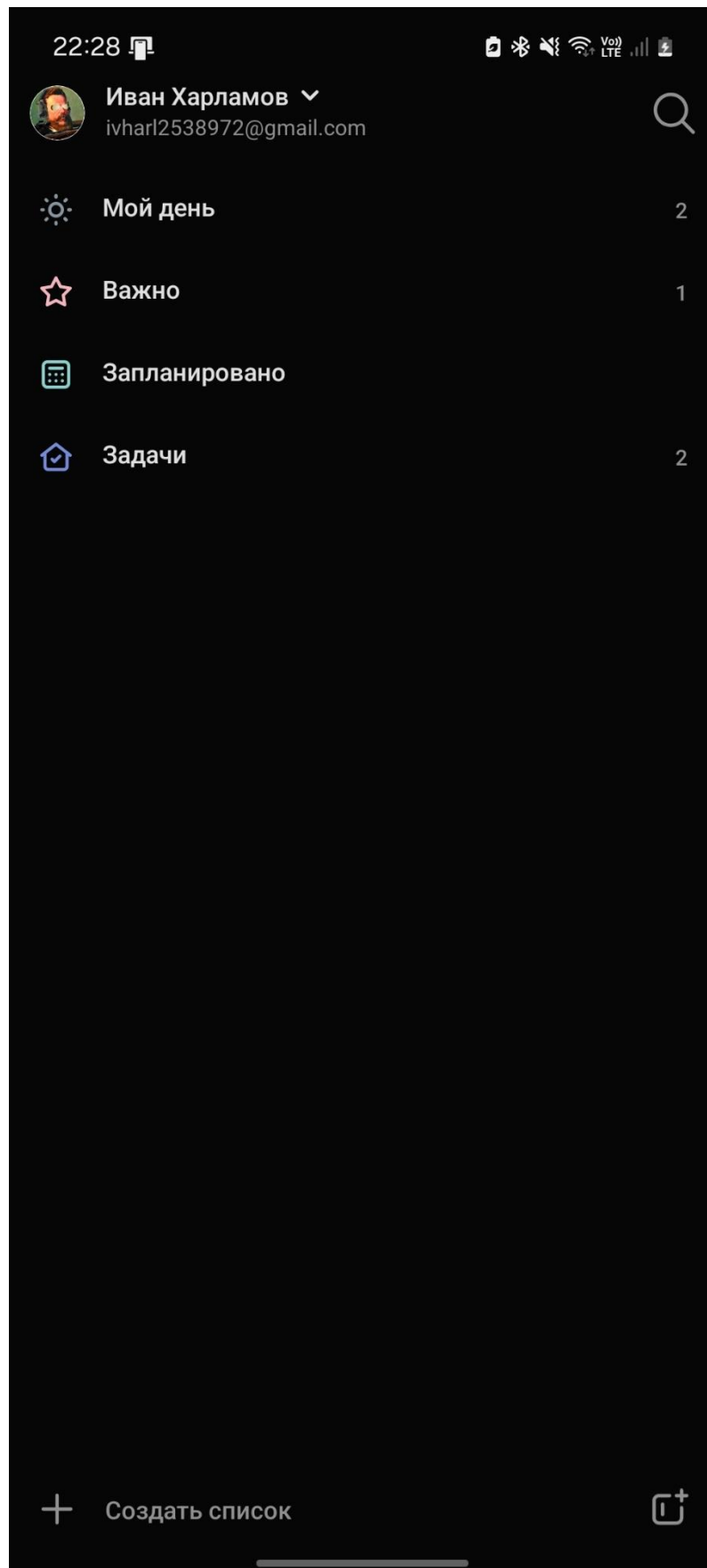


Рисунок 1 - Главная страница в приложении Microsoft To Do

### 2.2.2 Todoist

Todoist – одно из самых популярных и высоко оценённых мобильных приложений для отслеживания задач. Оно обладает крайне обширным функционалом, но посредственным интерфейсом. Бесплатно, но имеет подписку для доступа к продвинутым функциям. Выделяется поддержкой NLP на русском языке и системой приоритетов задач.

Достоинства:

- Весь базовый функционал доступен бесплатно;
- NLP на русском языке;
- Система приоритетов задач.

Недостатки:

- Не интуитивный и неудобный интерфейс;
- Самый главный отличительный функционал скрыт за подпиской: календарное представление задач, напоминания, прикрепление локации к задаче;
- Отсутствие возможности задать задаче период повторения.

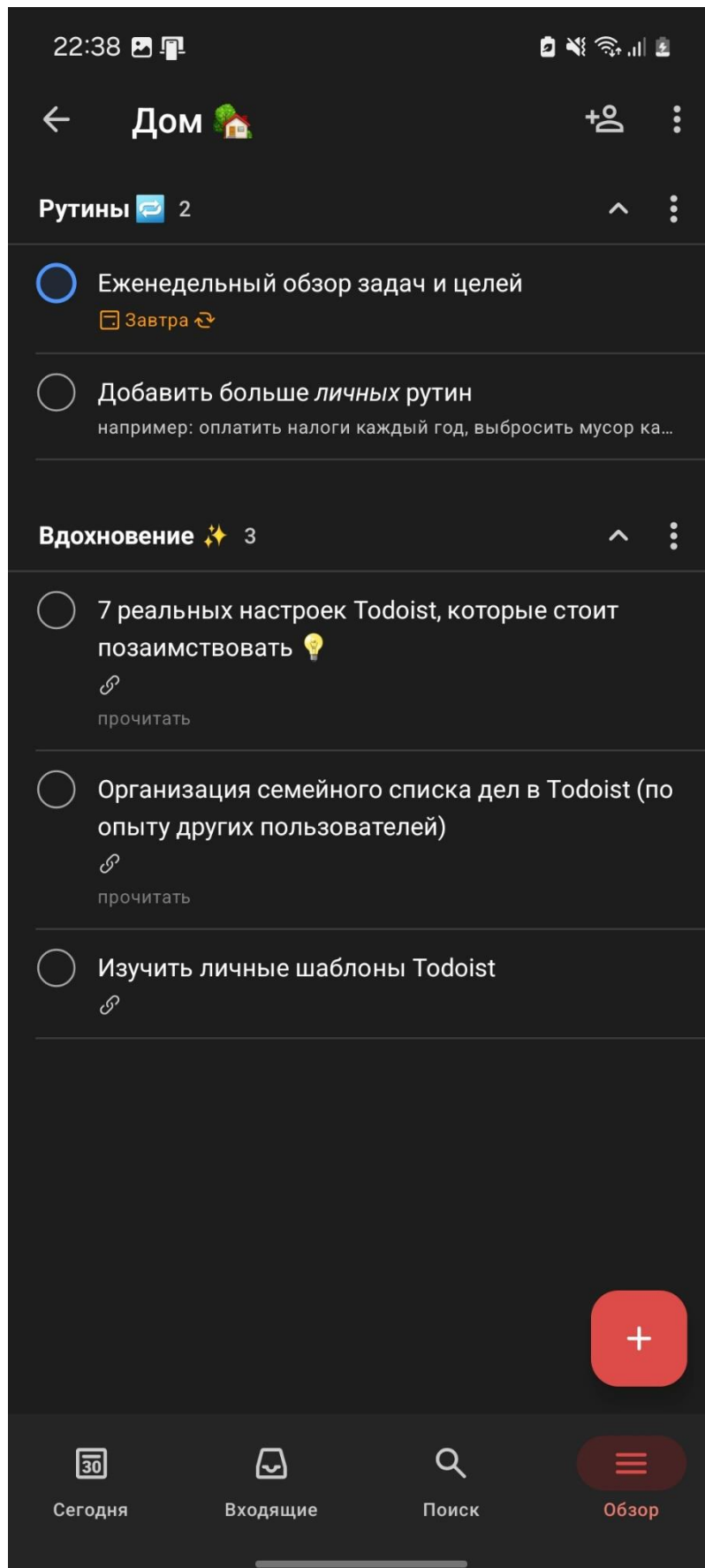


Рисунок 2 - Главный экран в приложении Todoist

### 2.2.3 Google Tasks

Google Tasks – полностью бесплатное решение от Google. Выделяется самым удобным интерфейсом из всех рассмотренных и интеграцией с другими сервисами корпорации, но не имеет даже некоторых базовых функций, необходимых для полноценного повседневного использования этого приложения.

Достоинства:

- Очень удобный интерфейс;
- Система повторяющихся задач;
- Интеграция с сервисами Google.

Недостатки:

- Отсутствие системы приоритетов;
- Отсутствие фильтров задач;
- Отсутствие тегов для задач;
- Отсутствие календарного представления задач.

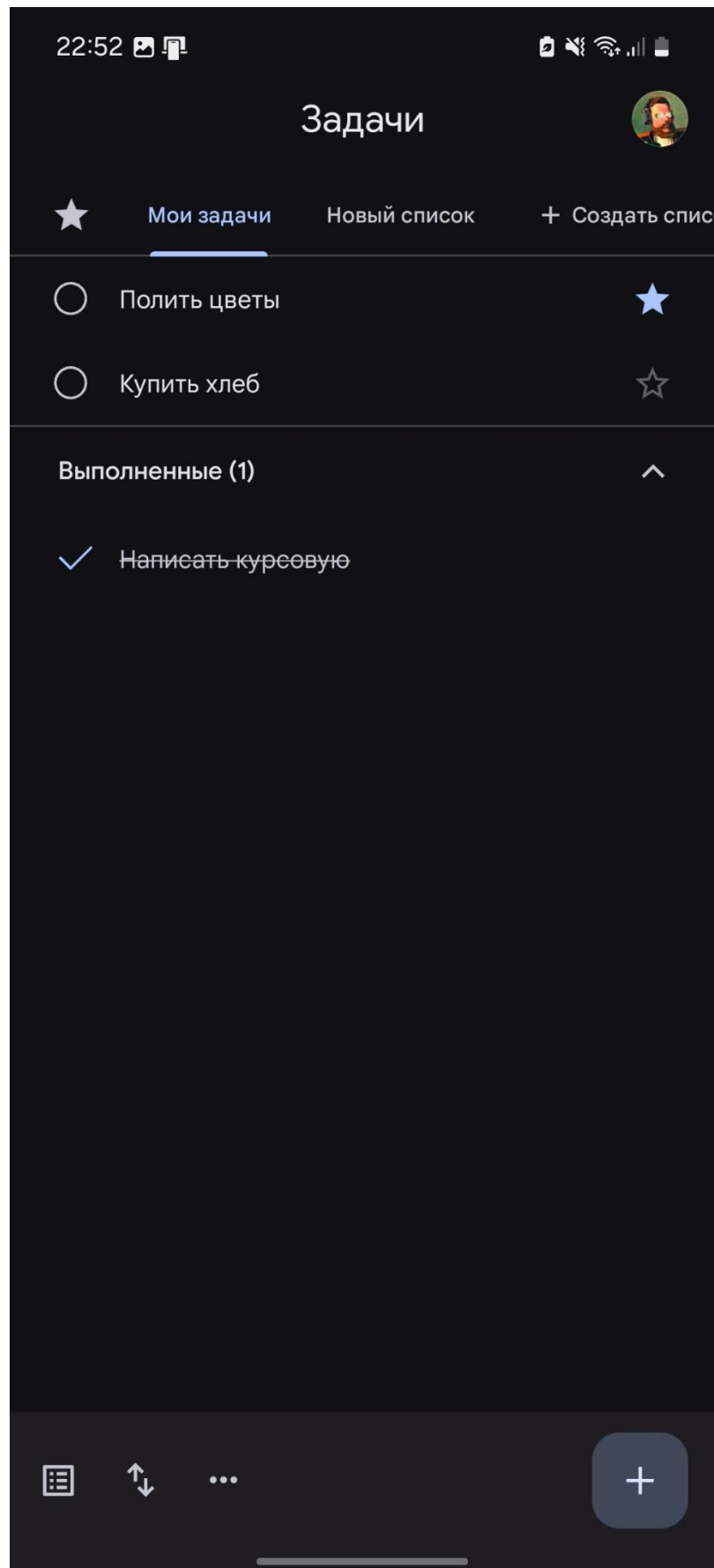


Рисунок 3 - Главный экран в приложении Google Tasks



## 2.2.4 TickTick

TickTick – приложение для отслеживания задач, обладающие самым обширным функционалом из всех рассмотренных. Бесплатно, но некоторые функции скрыты за подпиской. Имеет уникальный режим “Помидоро”, на время работы которого телефон блокируется, пока не будет выполнена поставленная задача. Так же есть система отслеживания привычек. Но интерфейс приложения не слишком удобен в использовании из-за затруднённого доступа к спискам задач.

Достоинства:

- Система приоритетов задач;
- Фильтры задач;
- Возможность делиться списками задач;
- Режим концентрации “Помидоро”;
- Теги для задач;
- Отслеживание привычек;
- NLP на русском языке.

Недостатки:

- За подпиской скрыты важные функции: напоминания о ближайших задачах и календарное представление;
- Неудобный интерфейс.

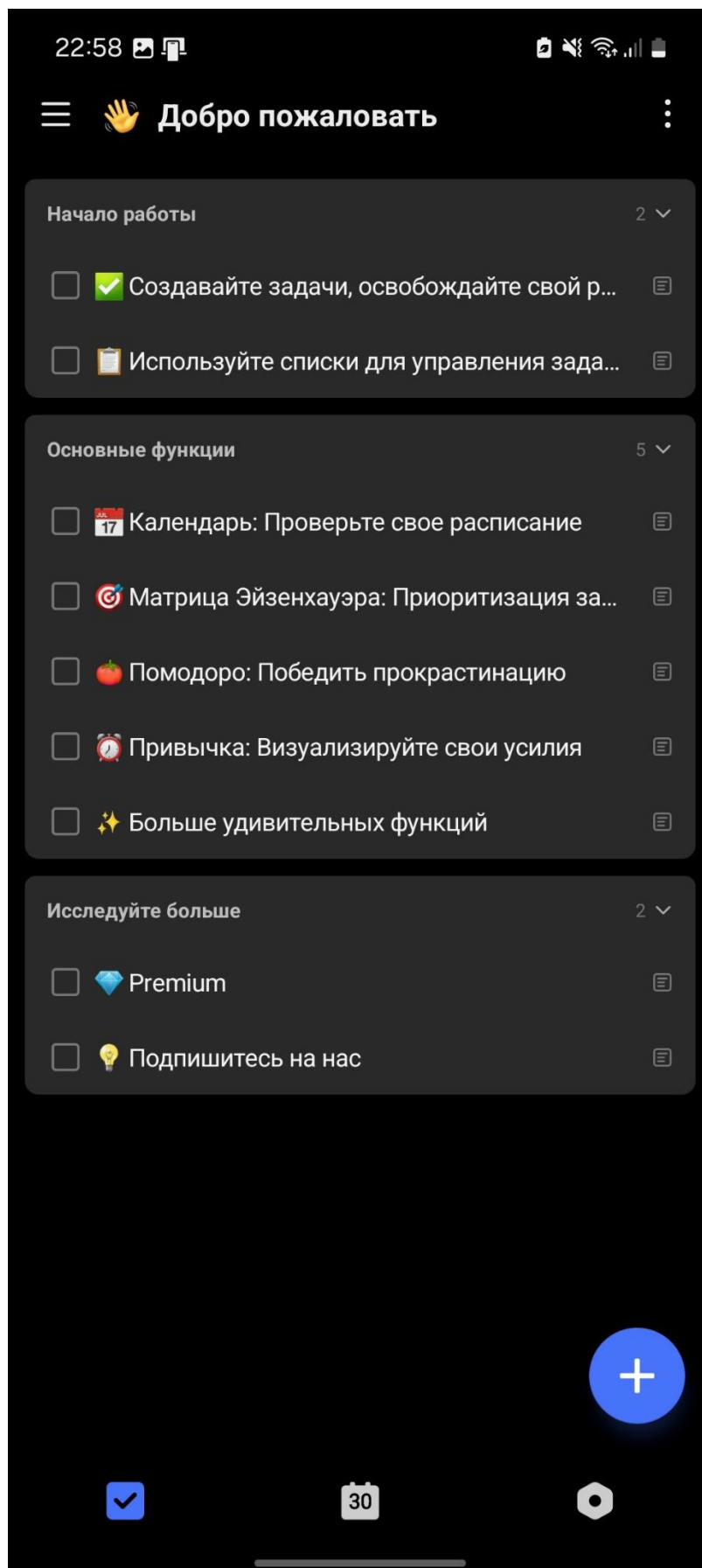


Рисунок 4 - Главный экран приложения TickTick

### 2.2.5 Any.Do

Any.Do – бесплатное приложение для отслеживания задач, но значимая часть функционала скрыта за подпиской. Интерфейс перегружен элементами и неудобен при использовании. Имеет уникальный режим планирования дня, при котором за один короткий сеанс сразу задаются задачи на целый день, а так же интеграцию с мессенджерами, позволяющую использовать текст из сообщений для автоматического добавления задач в приложение.

Достоинства:

- Режим планирования дня;
- Интеграция с мессенджерами;
- Экспорт задач в текстовый файл;
- Календарное представление задач;
- Напоминания о ближайших задачах.

Недостатки:

- Перегруженный элементами интерфейс;
- Значимая часть функционала требует подписки: система приоритетов задач, теги для задач, повторяющиеся задачи, рекомендации при создании задач, прикрепление локации к задаче, возможность делиться списками.

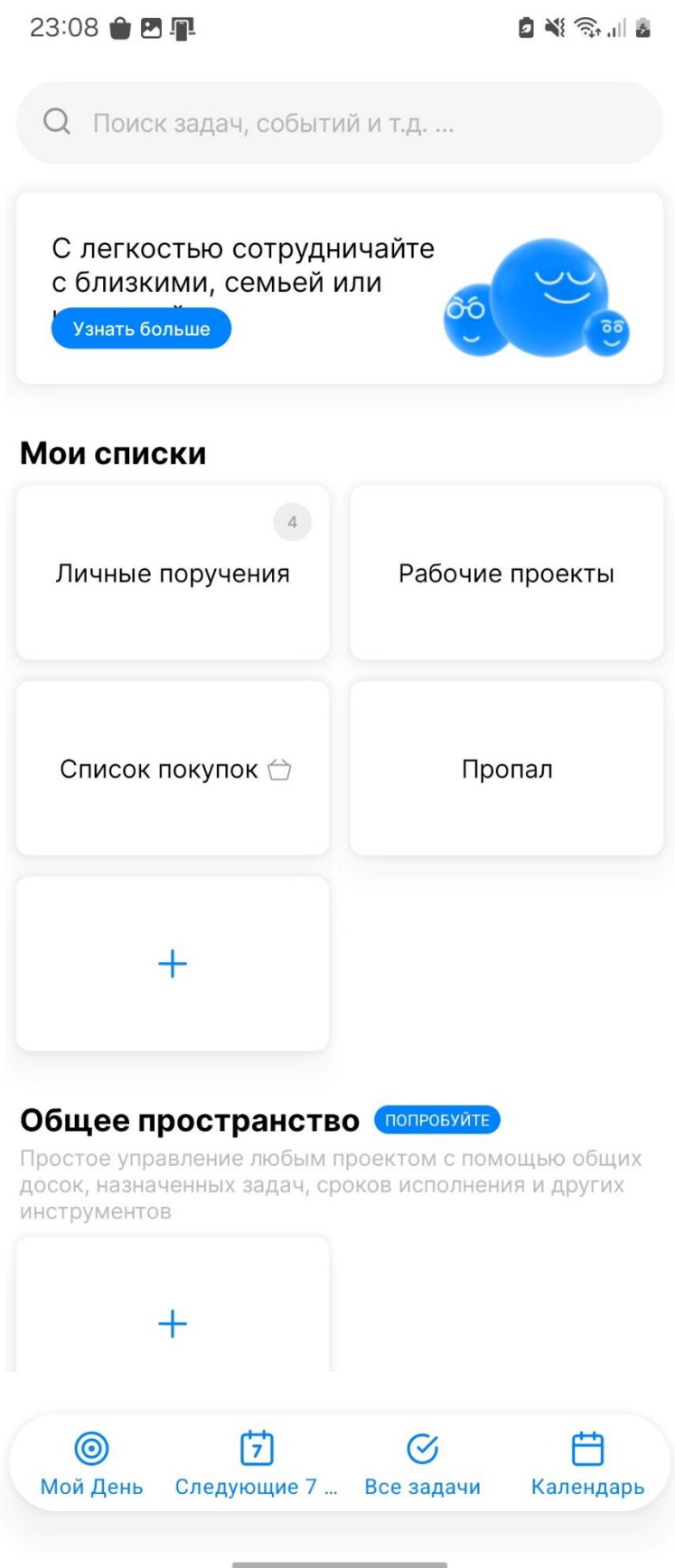


Рисунок 5 - Главный экран в приложении Any.Do

## 3 Реализация

### 3.1 Средства реализации

Веб-приложение имеет архитектуру, соответствующую шаблону клиент-серверного приложения, и разделяется на backend и frontend посредством REST API. Данная архитектура веб-приложения соотносится с основными требованиями к проекту, а именно:

- Создавать задачи и списки задач;
- Создавать подзадачи для детального отслеживания прогресса;
- Добавлять задачам описание и теги;
- Создавать теги и задавать им пользовательские цвета;
- Закрывать задачи и подзадачи после их выполнения;
- Просматривать задачи в календарном виде.

Для реализации серверной части были выбраны следующие технологии:

- Фреймворки Spring Boot 3.2.2 и Hibernate 6.4.8. Фреймворк Spring Boot обеспечивает лёгкость создания веб-приложений за счёт развитой системы аннотаций Java, а Hibernate обеспечивает надёжное и предсказуемое взаимодействие с базой данных;
- СУБД PostgreSQL 14. Она известна своей стабильной работой и хорошей производительностью. Система предоставляет ряд механизмов обеспечения целостности данных, что делает ее надежной и защищенной от ошибок. Она также имеет множество встроенных возможностей для будущего масштабирования;
- Язык программирования Java 21. Он обладает простым синтаксисом, обширной стандартной библиотекой для работы со структурами данных и сетью, а также удобными инструментами сборки, например, Maven, позволяющими быстро подключать необходимые зависимости.

Для реализации клиентской части были выбраны следующие технологии:

- Язык гипертекстовой разметки HTML 5;
- Формальный язык описания внешнего вида документа CSS 3;
- Язык программирования JavaScript ES2020;
- Фреймворк React 18.2.

### 3.2 Реализация базы данных

Данные приложения хранятся в реляционной базе данных PostgreSQL.

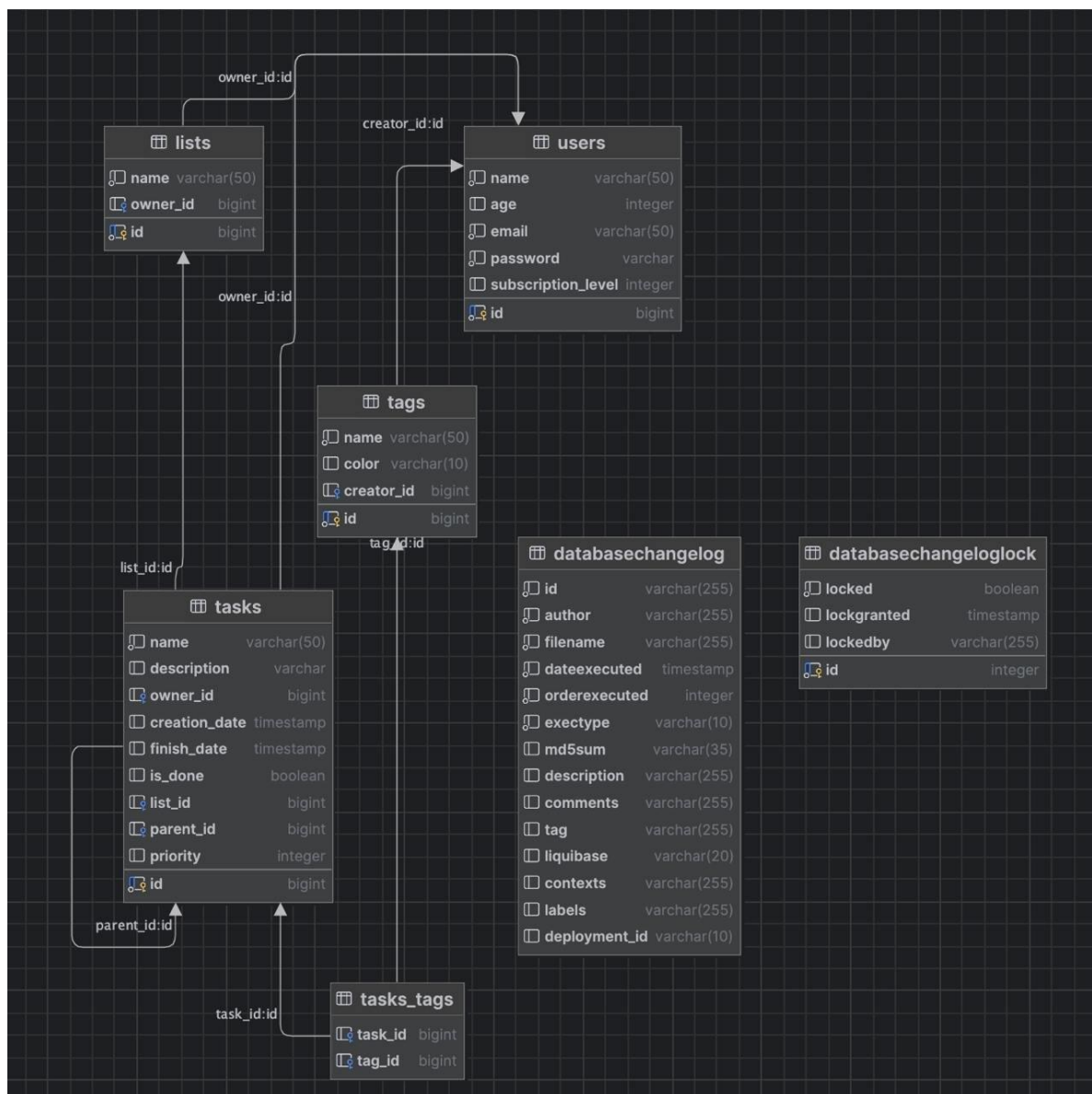


Рисунок 6 - Физическая модель базы данных

User – пользователь веб-приложения. Данные заполняются при регистрации, и пароль хранится в зашифрованном по алгоритму BCrypt виде.

Task – индивидуальные задачи в веб-приложении. Они принадлежат определённому пользователю (user), могут являться частью списка (list), иметь один или несколько тегов (tag), и являться подзадачей (связь parent\_id – task\_id).

Tag – теги, которые можно назначать задачам. Они принадлежат определённому пользователю (user) и связаны с таблицей task через таблицу task\_tag, так как у одной задачи может быть несколько тегов, а у одного тега – несколько задач.

List – списки задач. Принадлежат определённому пользователю (user).

### **3.3 Реализация серверной части веб-приложения**

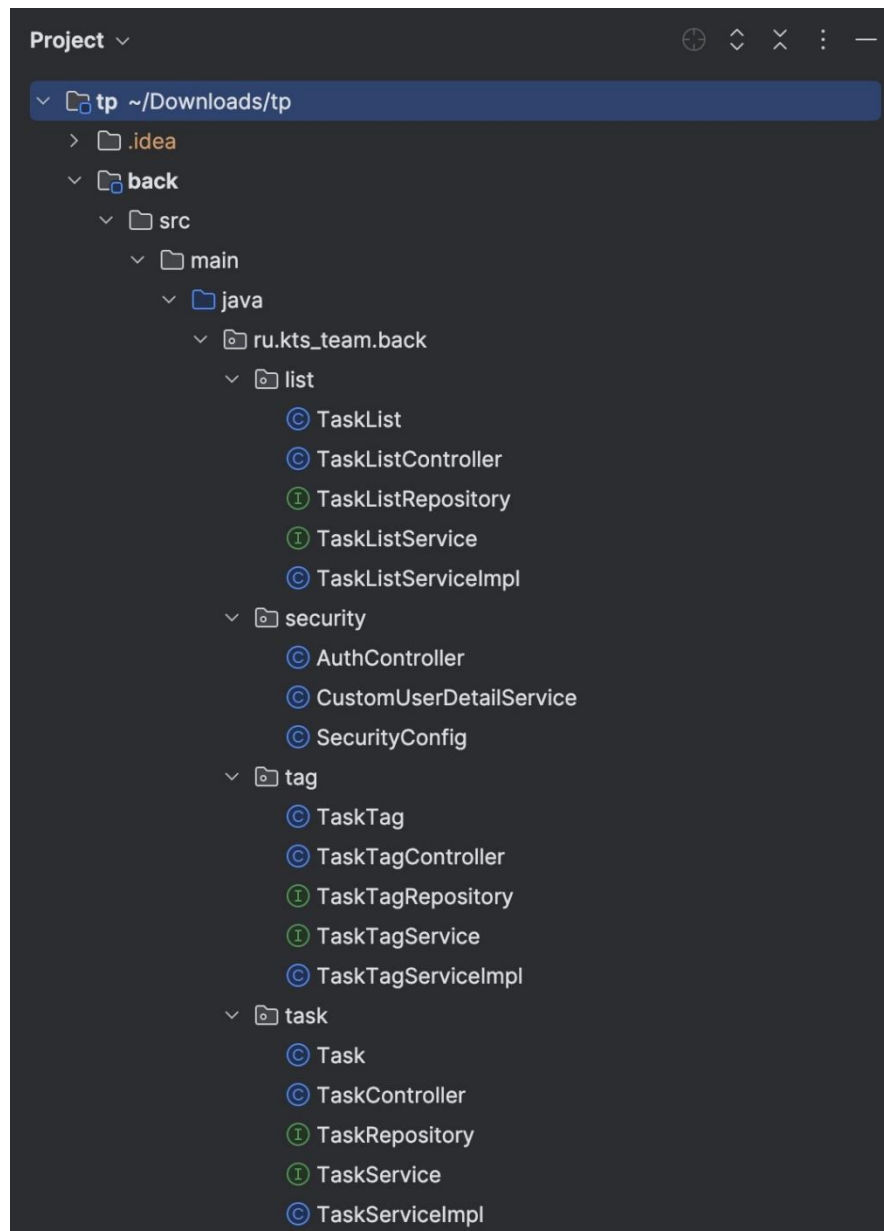
Серверная часть приложения реализована соответственно трехслойной архитектуре веб-приложения с API Rest с использованием фреймворка Spring Boot.

Presentation Layer - этот слой включает в себя контроллеры REST API, которые обрабатывают HTTP-запросы и возвращают данные клиенту в формате JSON. Контроллеры связаны с бизнес-логикой через слой сервисов.

Business Logic Layer - этот слой содержит сервисы, которые содержат бизнес-логику приложения. Сервисы служат прослойкой между контроллерами и слоем репозитория, обрабатывая запросы от контроллеров, выполняя необходимые операции и передавая данные назад в контроллеры.

Data Access Layer - этот слой содержит интерфейсы репозитория, которые обеспечивают доступ к базе данных. Репозитории позволяют получать и сохранять данные в хранилище данных.

Были реализованы классы, интерфейсы репозитория и контроллеры для всех необходимых объектов приложения по ER-диаграмме.



### Рисунок 7 - Структура backend



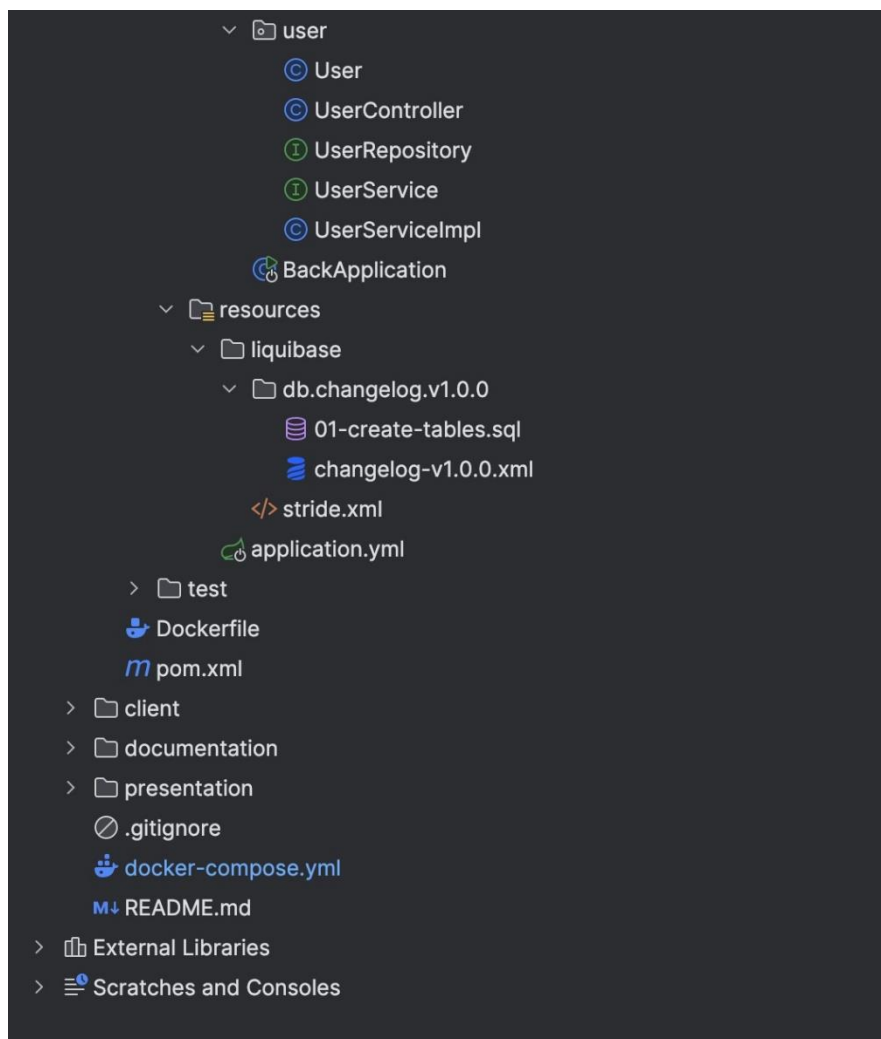


Рисунок 8 - Структура backend (продолжение)

### 3.4 Реализация клиентской части

Для реализации основных сценариев веб-приложения, клиентская часть разработки делится постранично. Каждая страница описывается языком программирования JavaScript, языком разметки HTML и благодаря использованию фреймворка React. Для реализации заранее осуществленного и утвержденного командой разработчиков дизайна используется язык стилей CSS.

Архитектура разработки была организована согласно бизнес-логике проекта на основании модульного подхода, по которому все компоненты и логика находятся рядом друг с другом, а благодаря модулю для работы с

файлами и их загрузкой через файл index.js экспортируется наружу все, что разрешено использовать.

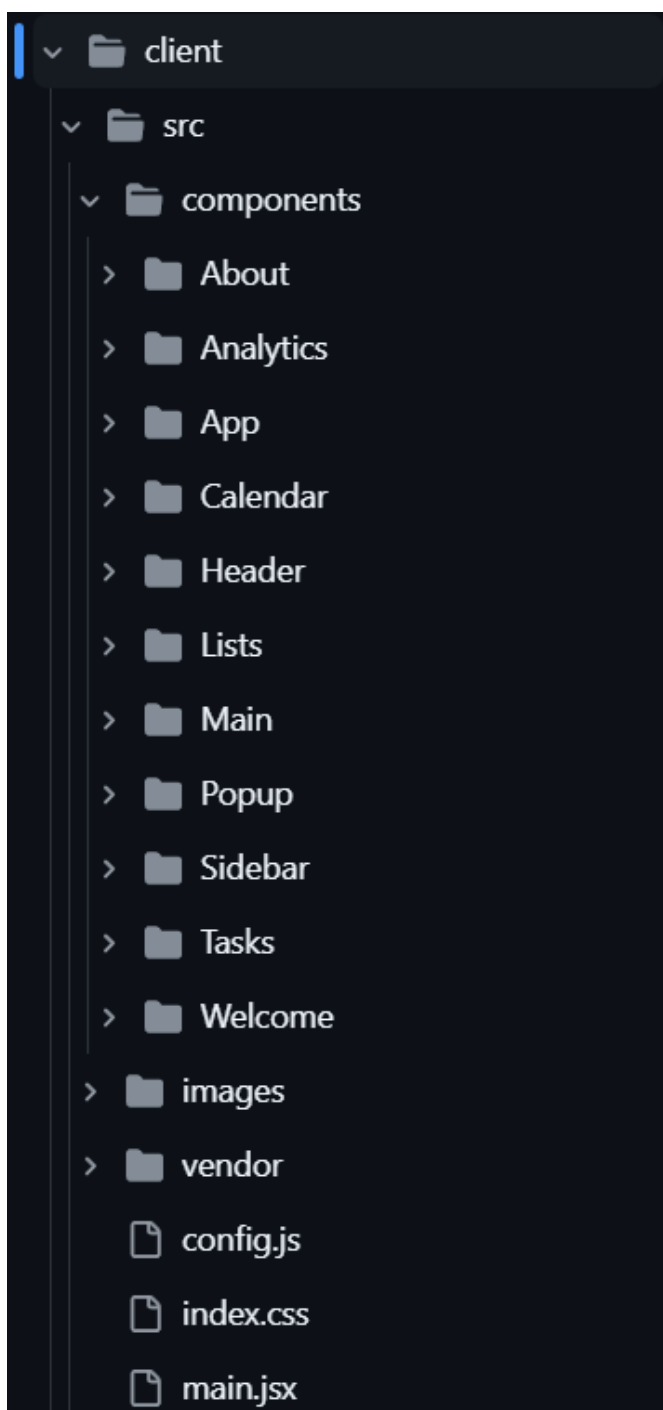
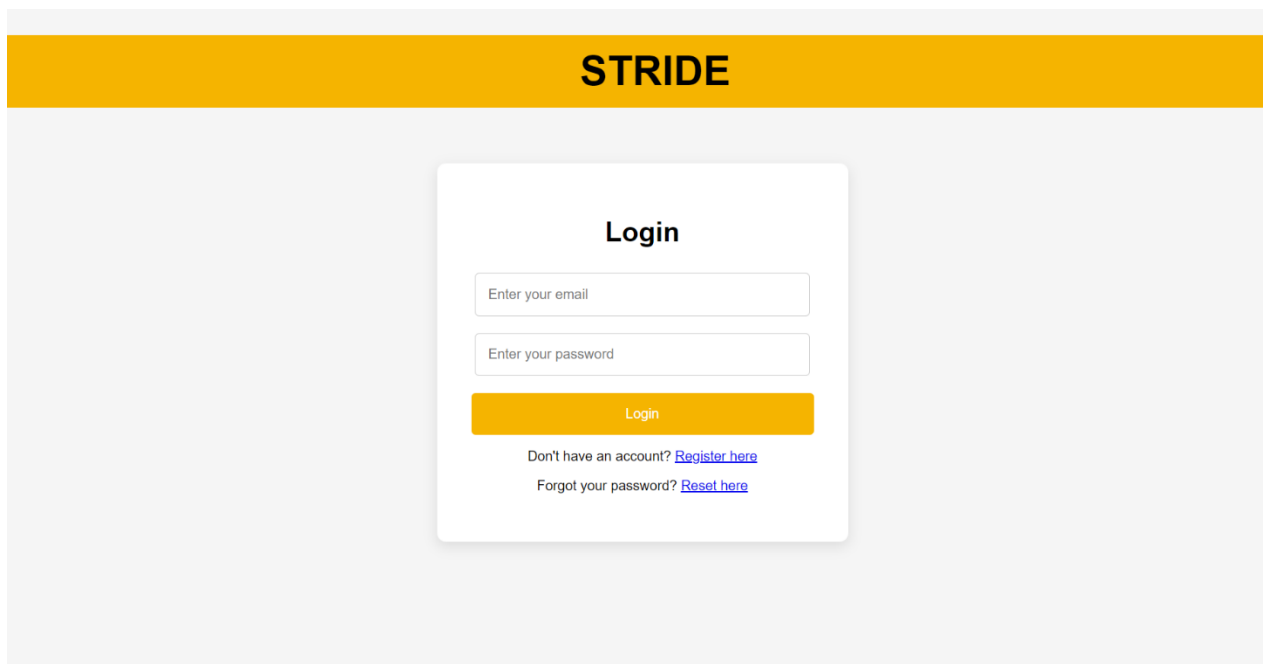


Рисунок 9 - Структура компонентов frontend

Все страницы веб-приложения были реализованы и представлены командой разработчиков в соответствии с заявленным дизайном и обозначенными требованиями к вёрстке.

### 3.4.1 Страница авторизации

Открывается по умолчанию у неавторизованного пользователя. Для дальнейшего использования функций веб-приложения необходима авторизация. Пользователь без аккаунта может зарегистрироваться.



The image shows a web application interface for login. At the top, there is a yellow horizontal bar with the word "STRIDE" in bold black capital letters. Below this bar, the background is a light gray. In the center, there is a white rounded rectangle representing the login form. Inside this form, the word "Login" is centered at the top. Below it are two input fields: the first is labeled "Enter your email" and the second is labeled "Enter your password". Below these fields is a yellow button with the text "Login" in black. At the bottom of the white form, there are two lines of text: "Don't have an account? [Register here](#)" and "Forgot your password? [Reset here](#)".

Рисунок 10 - Страница авторизации

### 3.4.2 Страница регистрации

На данной странице располагаются поля для ввода адреса электронной почты и пароля для регистрации нового аккаунта, а также кнопка для подтверждения регистрации.

**STRIDE**

**Registration**

Enter your username

Enter your email

Enter your password

Age

18

Confirm

Рисунок 11 - Страница регистрации

### 3.4.3 Страница сброса пароля

В случае если пользователь забыл пароль от своего аккаунта, он может восстановить доступ к нему, введя свой адрес электронной почты и название последней задачи, которую он создал, а также новый пароль. Если введенные данные оказались верными, то пользователь будет авторизован, и пароль от его аккаунта будет заменён на новый.

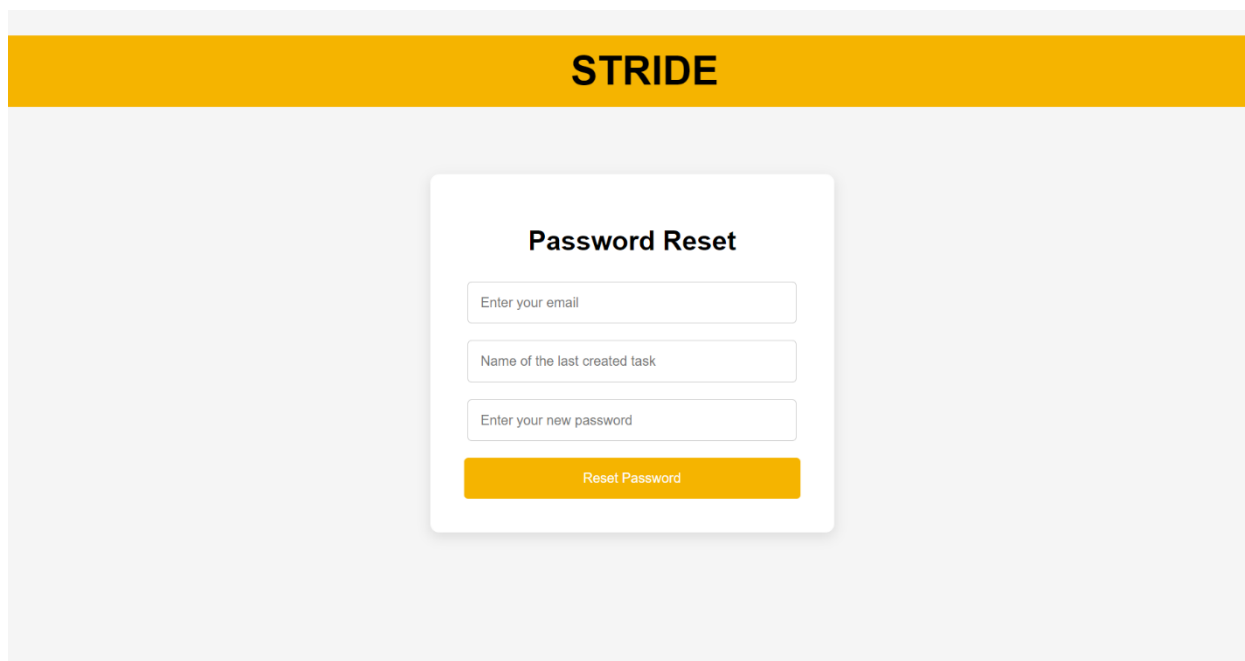


Рисунок 12 - Страница сброса пароля

#### **3.4.4 Основная страница веб-приложения**

После успешной авторизации пользователь оказывается на основной странице веб-приложения. Данная страница содержит боковое меню для перехода по основным окнам, навигационную строку с полем поиска задач, иконками напоминаний и профиля пользователя и доску задач, представляющую собой одну открытую вкладку из бокового меню (по умолчанию “Today’s Tasks”):

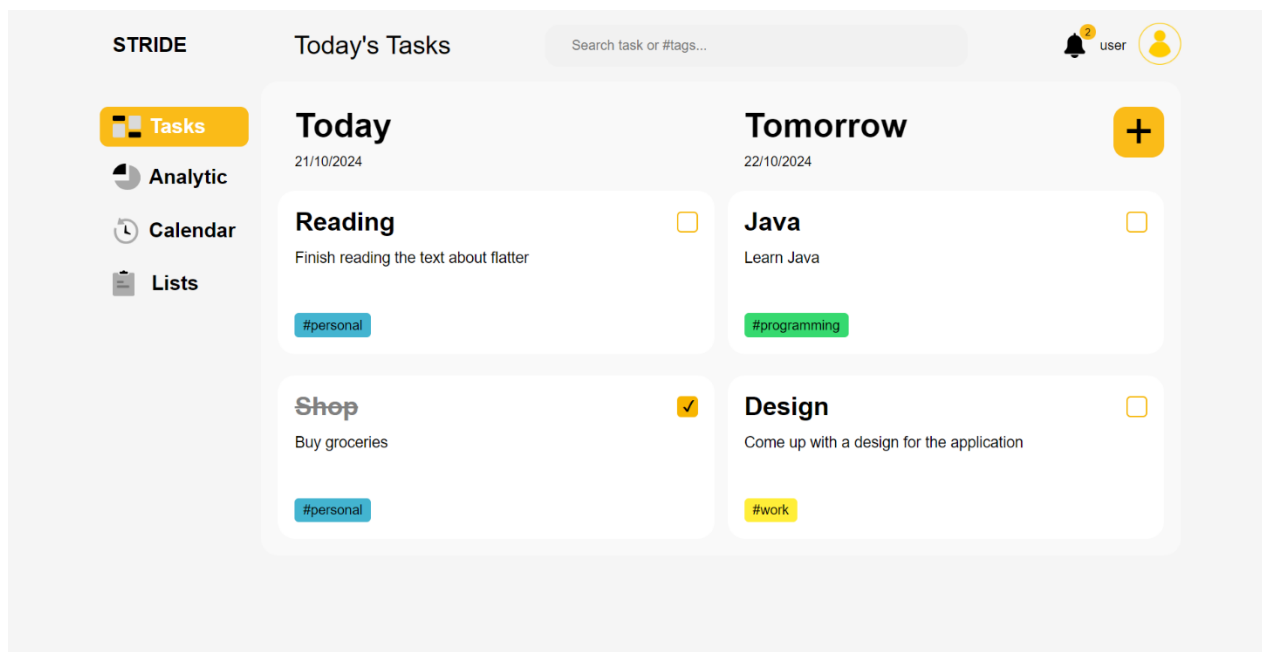


Рисунок 13 - Основная страница (Вкладка Tasks)

### 3.4.5 Вкладка Today's Tasks

На данной вкладке располагаются ближайшие по времени задачи. Пользователь может добавлять индивидуальные задачи, редактировать их параметры, такие как теги и подзадачи и удалять их.

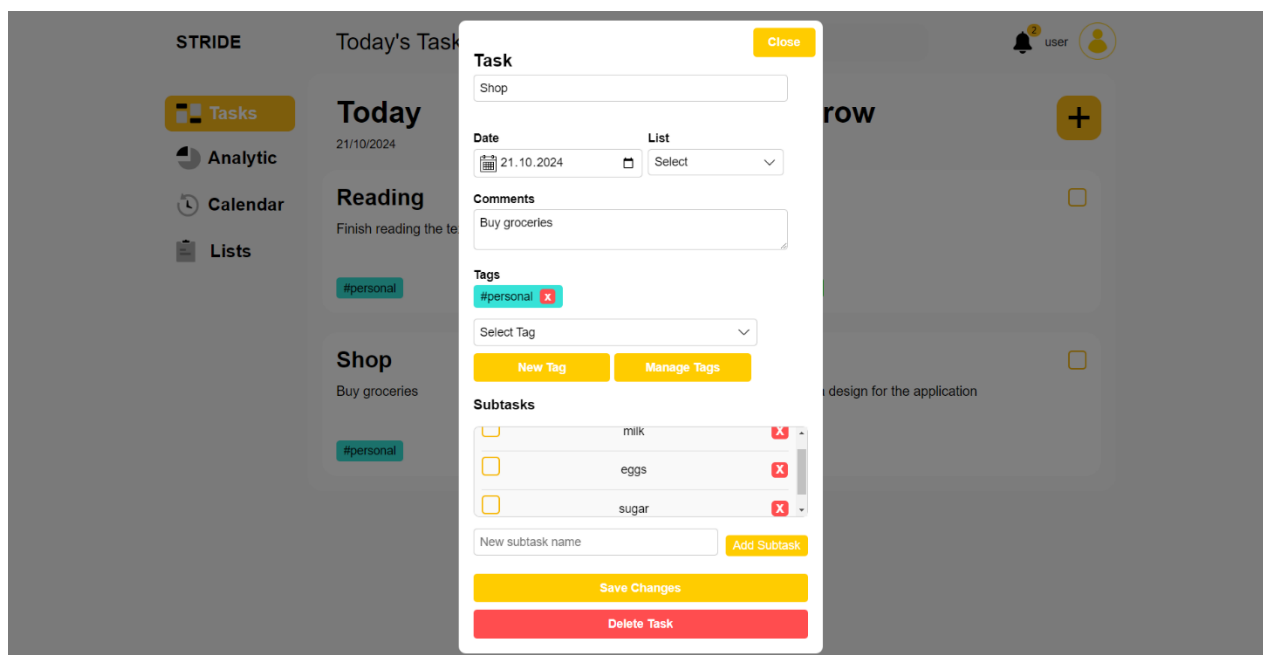


Рисунок 14 - Вкладка Tasks (конкретная задача)

### 3.4.6 Вкладка Analytics

На данной вкладке содержится аналитическая информация по количеству выполненных за промежуток времени задач.

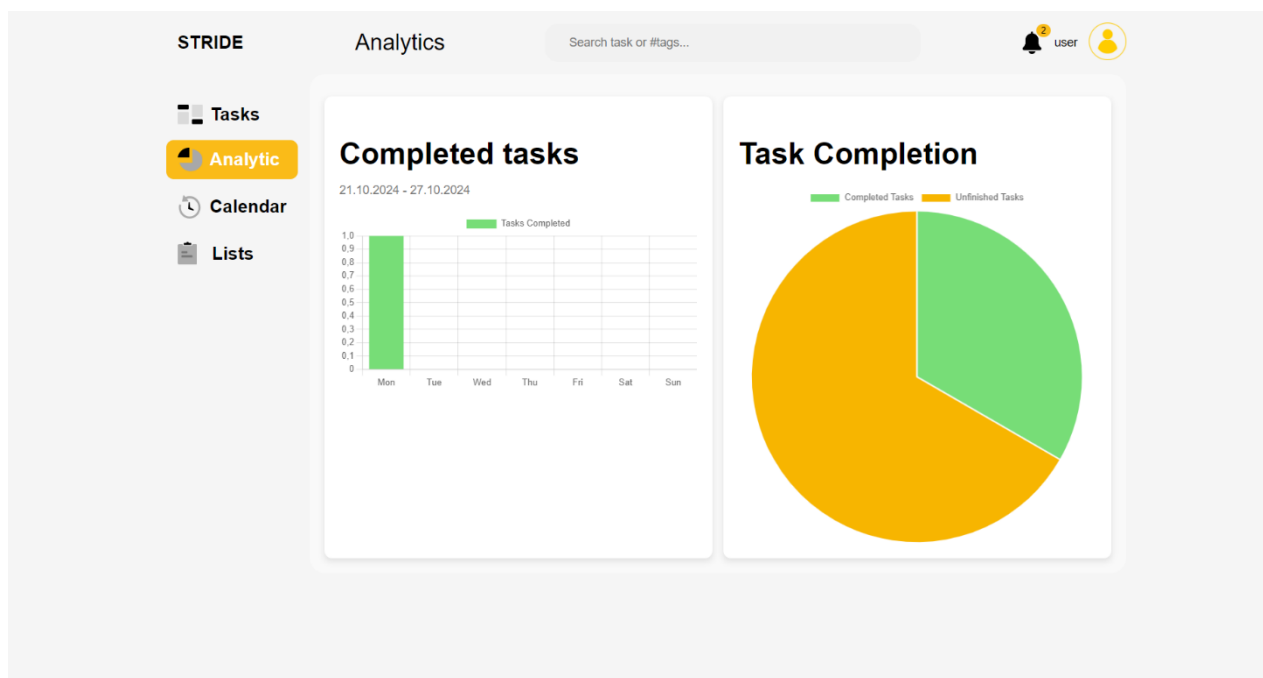


Рисунок 15 - Вкладка Analytics

### 3.4.7 Вкладка Calendar

Данная вкладка содержит календарь задач на месяц с распределением задач пользователя по дням по сроку их выполнения, с возможностью перехода к конкретной задаче.

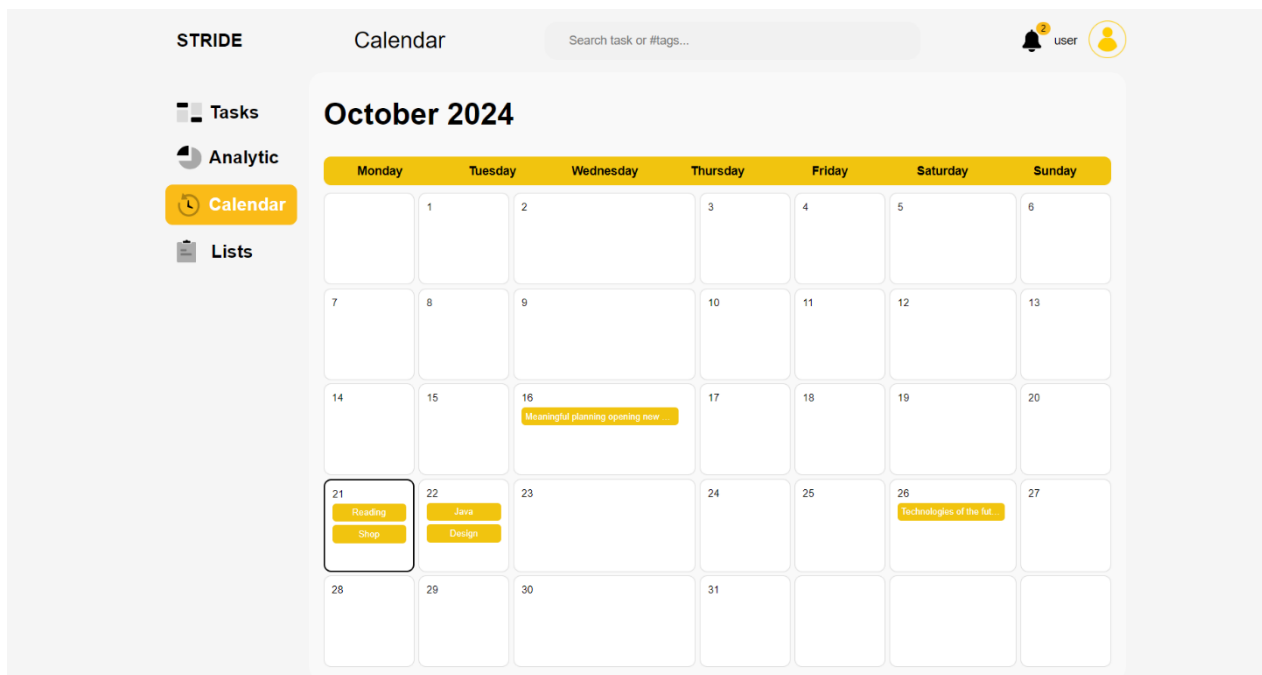


Рисунок 16 - Вкладка Calendar

### 3.4.8 Вкладка Lists

На данной вкладке пользователь может просмотреть все свои списки задач, а также создать и удалять их. Внутри списка можно добавлять и удалять индивидуальные задачи и изменять их статус.



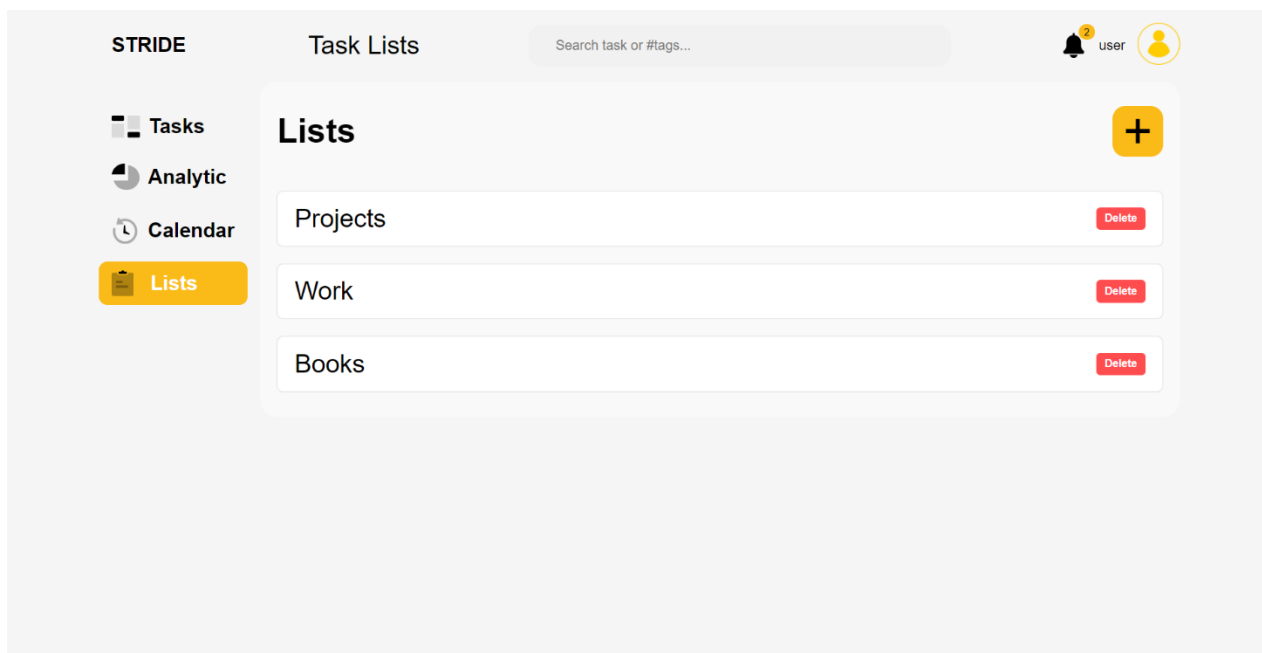


Рисунок 17 - Вкладка Lists (общий вид)

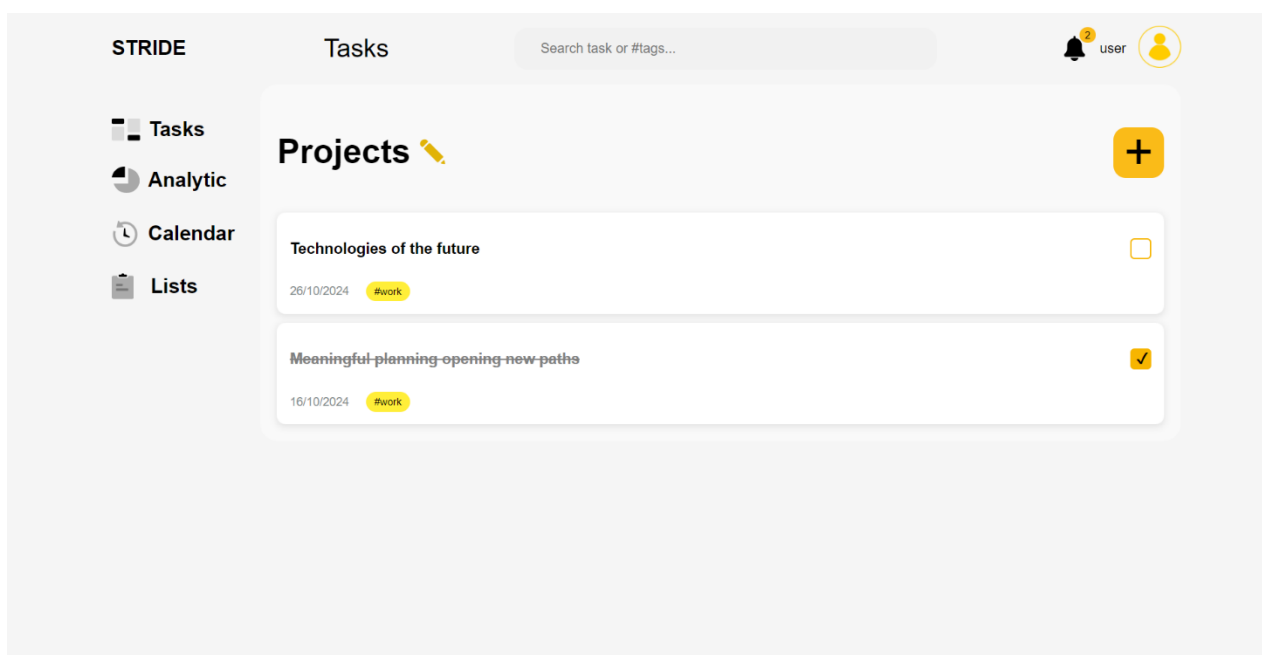


Рисунок 18 - Вкладка Lists (конкретный список)

### **3.5 Диаграммы, иллюстрирующие работу системы**

#### **3.5.1 Диаграмма последовательности**

Диаграммы последовательности отражают поток событий, происходящих в рамках варианта использования. На этих диаграммах изображаются только те объекты, которые непосредственно участвуют во взаимодействии т.к. ключевым моментом является именно динамика взаимодействия объектов во времени и не используются возможные статические ассоциации с другими объектами.

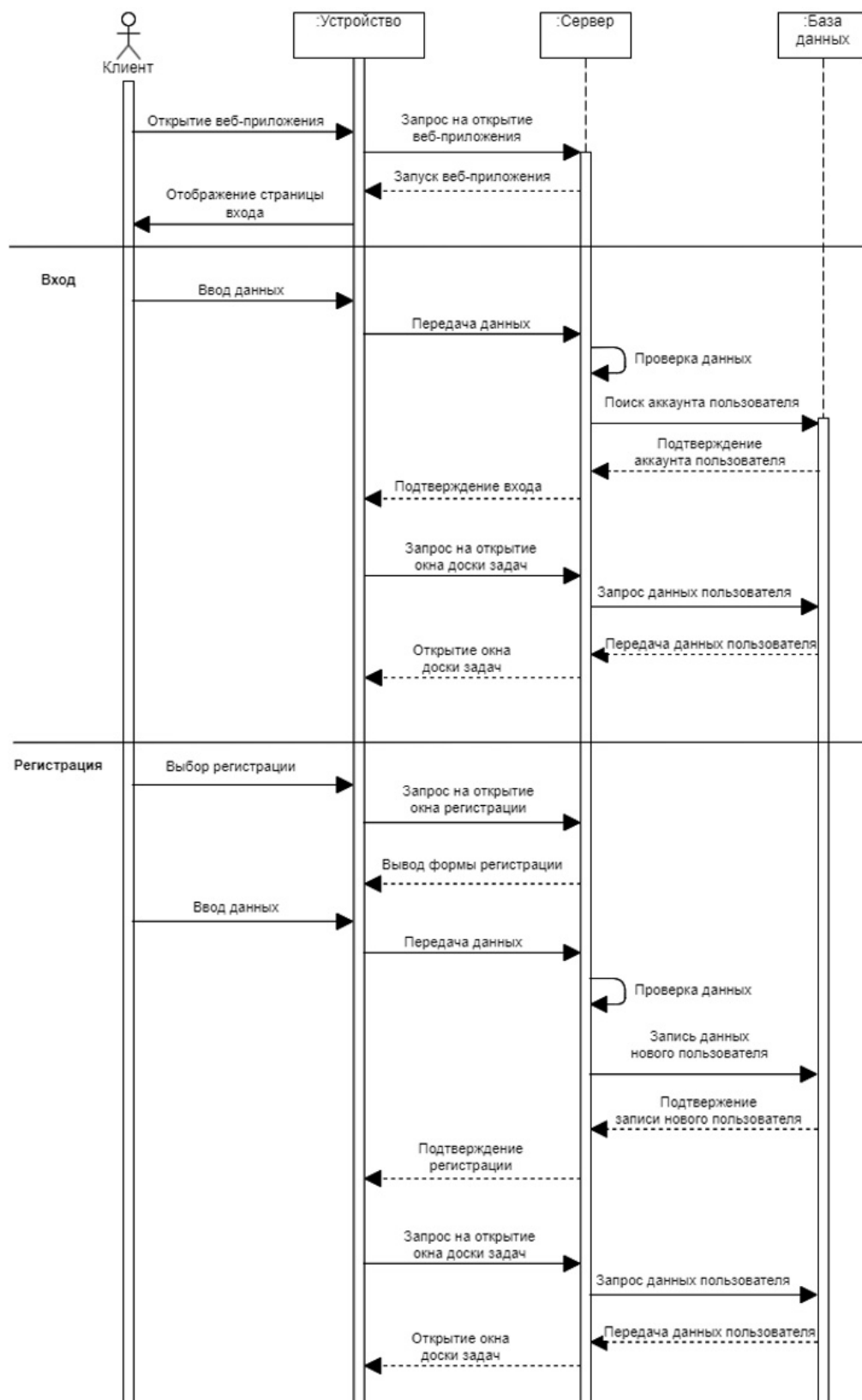


Рисунок 19 - Диаграмма последовательности

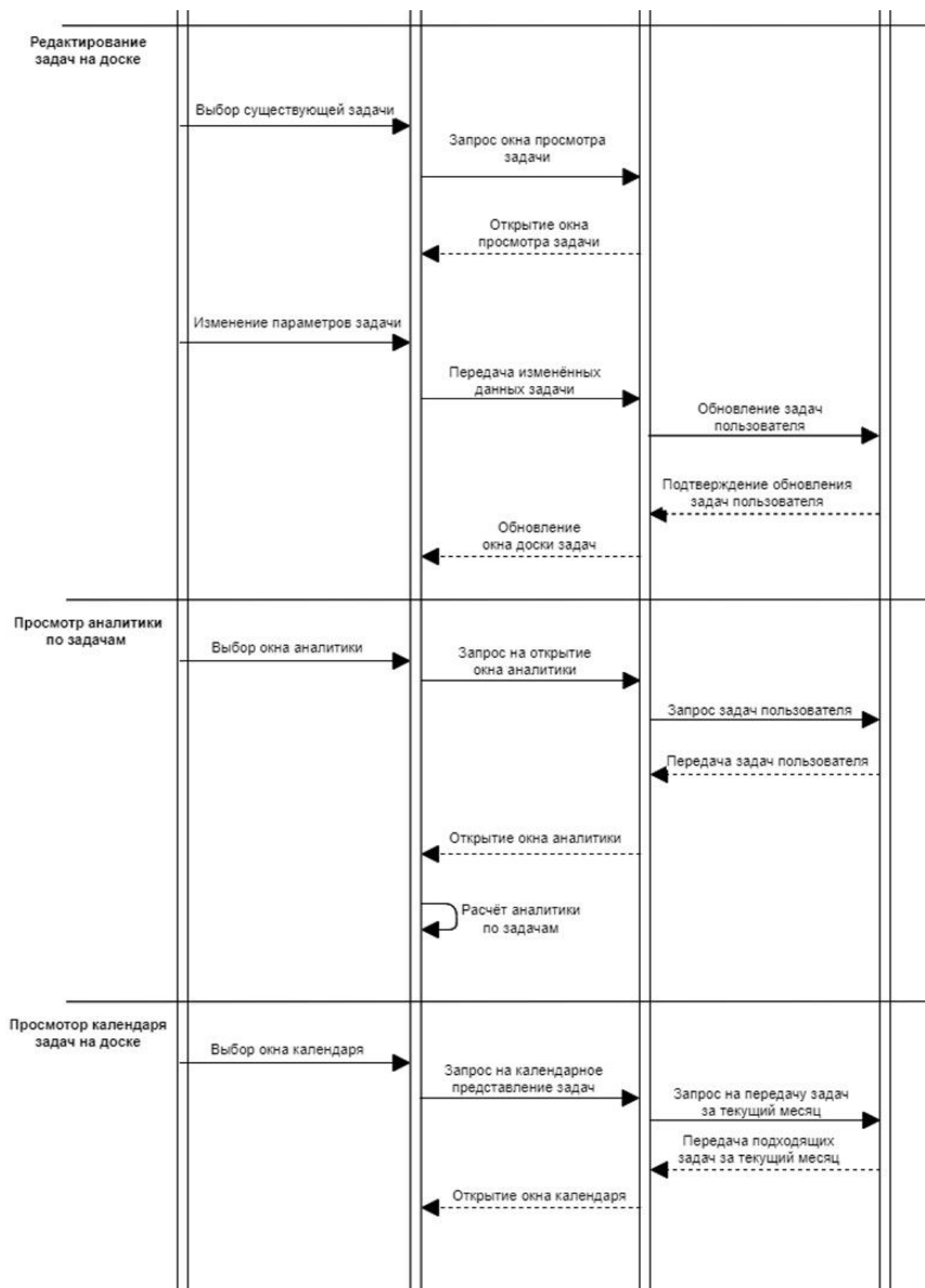


Рисунок 20 - Диаграмма последовательности (продолжение)

### 3.5.2 Диаграмма прецедентов

Диаграмма прецедентов представляет собой диаграмму, которая моделирует функциональность системы, показывая ее взаимодействие с актерами, внешними сущностями. Диаграмма прецедентов фокусируется на функциональных возможностях системы.



Рисунок 21 - Диаграмма прецедентов (неавторизованный пользователь)

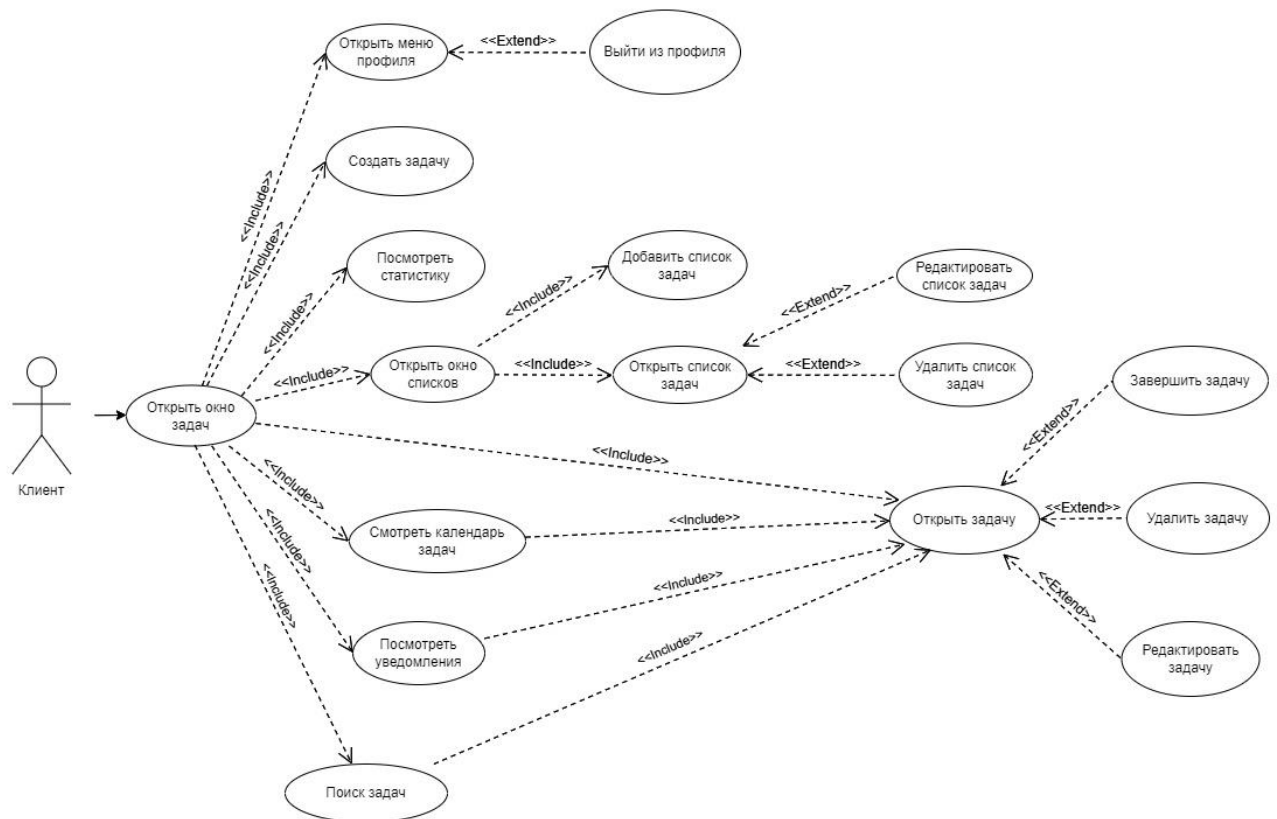


Рисунок 22 - Диаграмма прецедентов (клиент)

### 3.5.3 Диаграмма активности

Диаграммы активностей позволяют не только представить процесс изменения состояний анализируемой системы, но и детализировать особенности алгоритмической и логической реализации выполняемых системой операций.

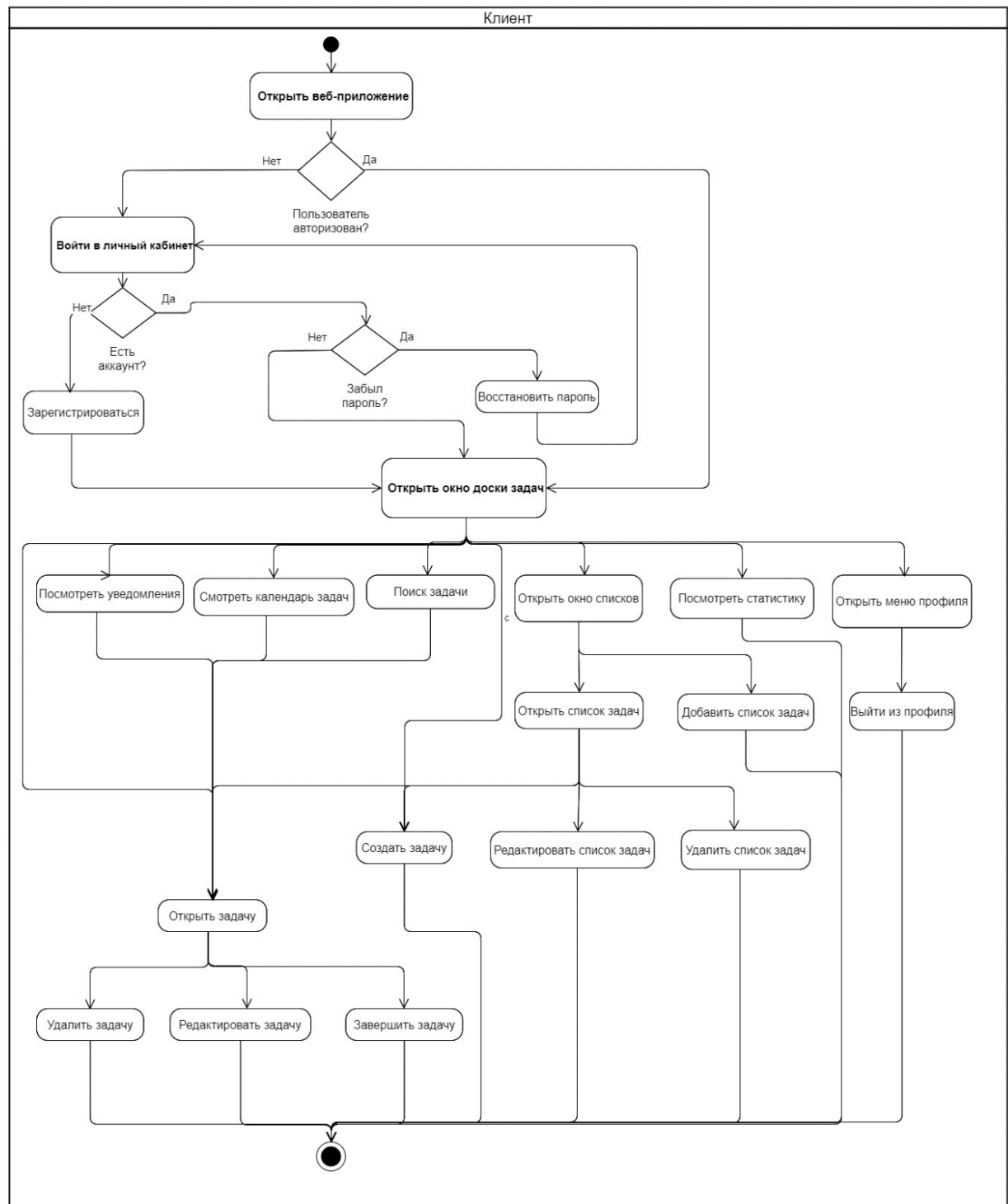


Рисунок 23 - Диаграмма активности

### 3.5.4 Диаграмма классов

Диаграмма классов служит для представления статической структуры модели системы в терминологии классов объектно-ориентированного программирования.

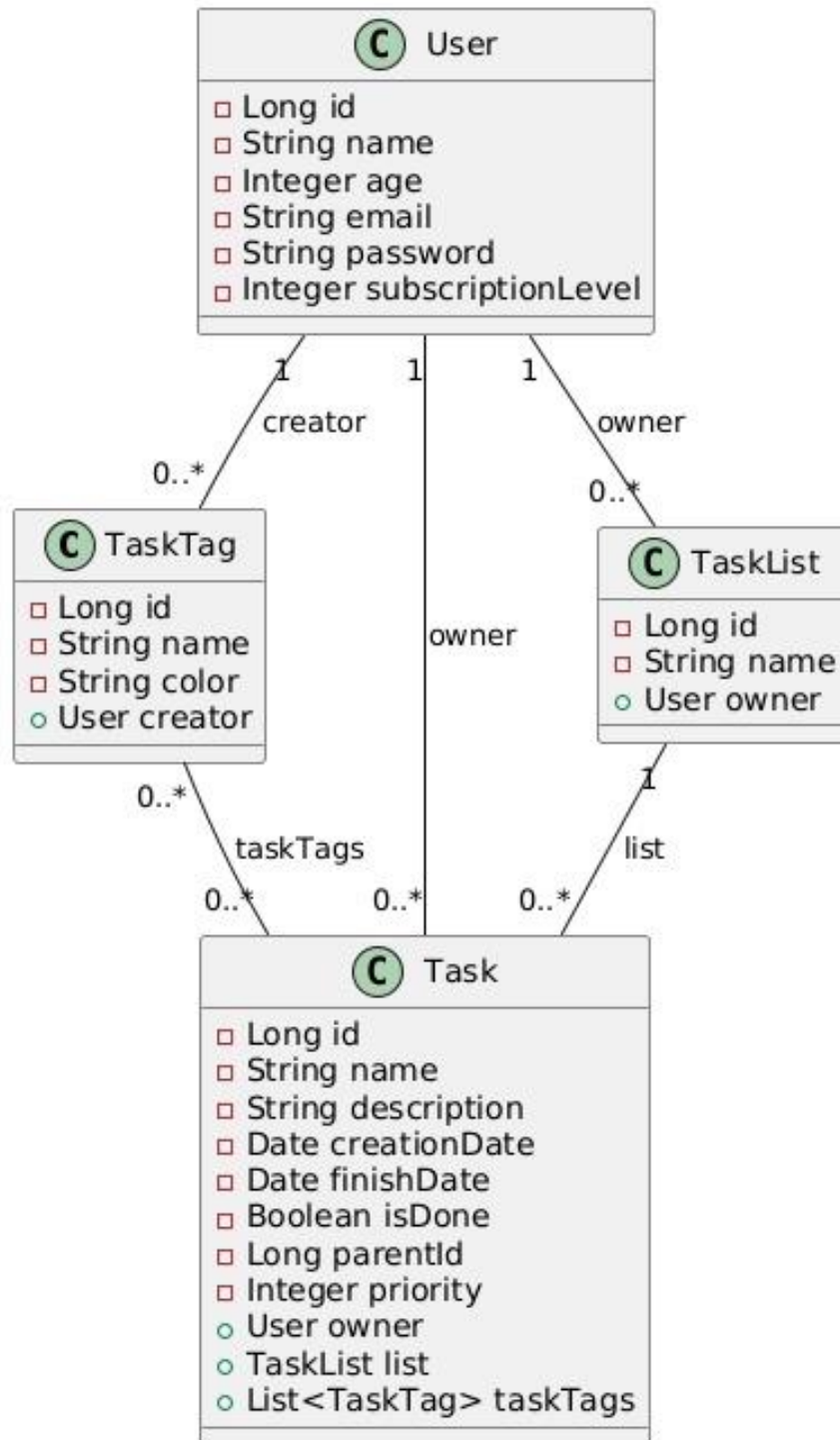


Рисунок 24 - Диаграмма классов

У класса User:

- id – уникальный идентификатор пользователя;
- name – имя пользователя;
- age – возраст пользователя;
- email – адрес электронной почты, указанный при регистрации;
- password – пароль пользователя;
- subscriptionLevel – уровень подписки пользователя.

У класса Task:

- id - уникальный идентификатор задачи;
- name – название задачи;
- description – текстовое описание задачи;
- owner – пользователь, создавший задачу;
- creation\_date – дата создания;
- finish\_date – крайний срок выполнения задачи;
- is\_done – флажок состояния задачи (завершена/нет);
- list – список, к которому принадлежит задача;
- parent\_id – идентификатор родительской задачи, для которой текущая является подзадачей (если такая есть);
- priority – приоритет задачи.

У класса Tag:

- id – уникальный идентификатор тега;
- name – название тега;
- color – цвет тега в интерфейсе веб-приложения;
- creator - пользователь, создавший тег.

У класса List:

- id – уникальный идентификатор списка;
- name – название списка;
- owner - пользователь, создавший список.



### 3.5.5 Диаграмма развертывания

Диаграмма развертывания позволяет определить требования к аппаратному обеспечению, планировать установку и настройку компонентов системы, а также оценивать ее производительность и масштабируемость.

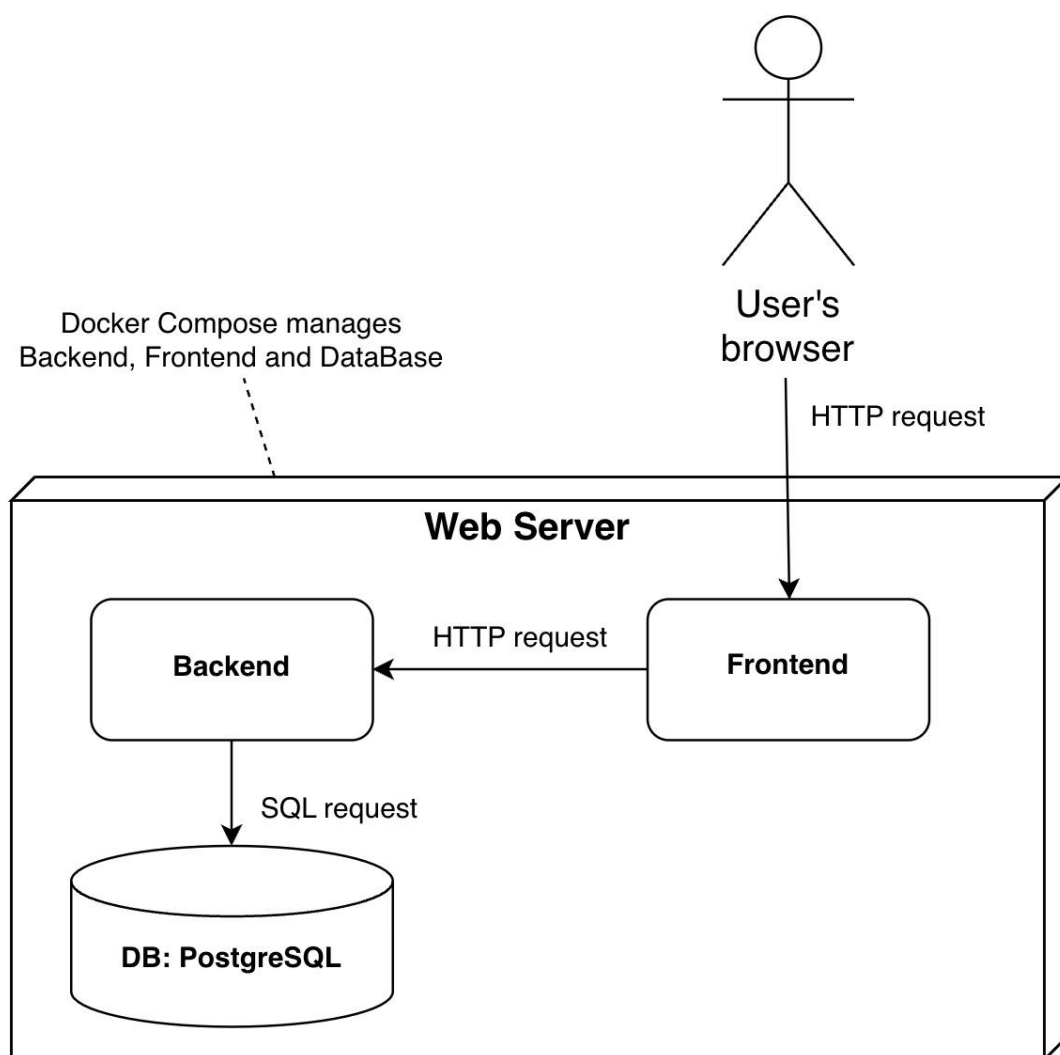


Рисунок 25 - Диаграмма развёртывания

## Заключение

В ходе выполнения курсового проекта командой было разработано веб-приложение для создания и управления личными и профессиональными целями, соответствующее поставленным перед проектом задачам.

В начале разработки был проведен анализ предметной области, определены основные требования к разрабатываемой системе.

В процессе работы были реализованы следующие задачи веб-приложения:

- Создание задач и списков задач;
- Создание подзадач для детального отслеживания прогресса;
- Добавление задачам описания и тегов;
- Создание тегов и установка им пользовательских цветов;
- Завершение задач и подзадач после их выполнения;
- Просмотр задач в календарном виде;
- Просмотр аналитической информации по выполненным задачам.

## Список использованных источников

1. Документация SpringBoot [Электронный ресурс]. – Режим доступа: URL: <https://docs.spring.io/spring-boot/docs/current/reference/html/>– Заглавие с экрана. — (Дата обращения 21.03.2024)
2. Система управления объектно-реляционными базами данных PostgreSQL [Электронный ресурс]. — Режим доступа URL: <https://webcreator.ru/technologies/webdev/postgresql/>— Заглавие с экрана. — (Дата обращения: 03.04.2024).
3. Общие сведения о React [Электронный ресурс]. — Режим доступа: <https://learn.microsoft.com/ru-ru/windows/dev-environment/javascript/reactoverview>. — Заглавие с экрана. — (Дата обращения: 20.04.2024).
4. Введение в UML от создателей языка/ Гради Буч, Джеймс Рамбо, Ивар Якобсон. — М.: ДМК Пресс, 2015. — 496 с.
5. An Introduction to Hibernate 6 [Электронный ресурс]. — Режим доступа: [https://docs.jboss.org/hibernate/orm/6.3/introduction/html\\_single/Hibernate\\_Introduction.html](https://docs.jboss.org/hibernate/orm/6.3/introduction/html_single/Hibernate_Introduction.html). — Заглавие с экрана. — (Дата обращения: 19.04.2024).