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

Факультет Компьютерных наук

Кафедра информационных систем и технологий

Онлайн-мессенджер «MesChat»

Курсовая работа по дисциплине
«Технологии программирования»

09.03.02 Информационные системы и технологии
Информационные технологии управления предприятием

Руководитель _____В.С. Тарасов, ст. преподаватель __._20__

Обучающийся _____Н.А. Антонян, 3 курс, д/о

Обучающийся _____А.А. Островерхов, 3 курс, д/о

Обучающийся _____А.К. Рассман, 3 курс, д/о

Руководитель _____К.В. Зенин, преподаватель

Воронеж 2023

Содержание

Введение

Современный мир невозможно представить без интернета и сопутствующих технологий. Интернет-технологии стали неотъемлемой частью нашей жизни, оказывая огромное влияние на все сферы деятельности: от образования и науки до бизнеса и межличностных отношений.

Социальные сети, как часть интернет-технологий, такие как Facebook, Instagram и ВКонтакте, доминировали в нашей культуре на протяжении последнего десятилетия. Однако, с ростом прогресса, мессенджеры уверенно выходят на первый план и становятся неотъемлемой частью нашей жизни, особенно в настоящее время. Большинство мессенджеров предоставляют нам возможность общаться с друзьями, любимыми и коллегами на расстоянии одним нажатием кнопки. Кроме того, они имеют множество других возможностей, такие как передача файлов, видеозвонки, распознавание речи, групповые чаты и многое другое.

В данной курсовой работе мы рассмотрим создание онлайн-мессенджера, который позволит пользователям общаться друг с другом, находясь на любых расстояниях, из любой точки мира.

1 Постановка задачи

Данный проект предназначен для обеспечения возможности пользователям общаться между собой в онлайн режиме.

Целью данного проекта является разработка сайта с возможностью общаться и вести диалоги людям, авторизованным в системе.

1.1 Требования к разрабатываемой системе

1.1.1 Функциональные требования

К разрабатываемому приложению выдвигаются следующие функциональные требования для пользователя:

— Функционирующий чат;

— Добавление чатов в избранные;

— Поиск пользователей.

1.1.2 Нефункциональные требования

К разрабатываемому приложению выдвигаются следующие нефункциональные требования:

— Поддержка основных браузеров;

— Система не должна давать доступ к изменению данных пользователя другим пользователям;

— Пароли пользователей должны храниться в зашифрованном виде;

— Требования к безопасности;

— Требования к интерфейсу;

— Оптимизация веб-приложения под изменение размеров экрана браузера;

— Оформление приложения должно быть выполнено в едином стиле.

1.2 Требования к архитектуре

Список требований к архитектуре выглядит следующим образом:

— Приложение должно быть построено на клиент-серверной архитектуре с использованием протоколов HTTP, а также протокола связи

WebSocket, который позволяет серверу и клиенту обмениваться данными в режиме реального времени без необходимости отправки запросов;

— Серверная часть приложения должна быть написана с использованием современных технологий back-end разработки, таких как Python и Django;

— Для хранения информации необходимо использовать базу данных, обеспечивающую высокую производительность и надежность. В качестве такой базы данных будет выступать PostgreSQL[4];

— Клиентская часть приложения должна быть написана с использованием front-end технологий, таких как HTML, CSS, а также фреймворка для JavaScript – React [5].

1.3 Задачи, решаемые в процессе разработки

В процессе разработки мессенджера будут решаться следующие задачи:

— Для определения функциональных требований к приложению и разработки соответствующей архитектуры необходимо провести анализ предметной области, изучив особенности работы мессенджеров и требования пользователей;

— Необходимо разработать структуру базы данных на основе требований для приложения, чтобы обеспечить эффективное взаимодействие веб-приложения с данными;

— На данном этапе необходимо разработать серверную часть приложения, которая будет обрабатывать запросы клиента и взаимодействовать с базой данных, используя фреймворк Django;

— Для создания удобного и понятного интерфейса пользователя, необходимо на данном этапе разработать клиентскую часть приложения с использованием технологий front-end разработки, таких как HTML, CSS и фреймворка языка JavaScript React [5];

— Для проверки соответствия приложения требованиям, определенным в начале проекта, необходимо провести тестирование и отладку приложения, чтобы убедиться в его корректной работе.

2 Анализ предметной области

2.1 Терминология (глоссарий) предметной области

Веб-приложение — клиент-серверное приложение, в котором клиент взаимодействует с веб-сервером при помощи браузера.

Мессенджер — система мгновенного обмена сообщениями.

Проект — это задача с определенными исходными данными и требуемыми результатами (целями), обуславливающими способ ее решения.

Гость — это человек, который посетил ресурс или совершил на нем какое-либо действие.

Учетная запись — хранимая в компьютерной системе совокупность данных о пользователе, необходимая для его опознавания и предоставления доступа к его личным данным и настройкам.

Пользователь — это человек, который имеет учетную запись.

Сервер — это компьютер, принимающий HTTP-запросы от клиентов, обычно веб-браузеров, и выдающий им HTTP-ответы, как правило, вместе с HTML-страницей, изображением, файлом, медиа-потоком или другими данными.

Клиент — это устройство, основным приложением которого (с точки зрения разработчика устройства или маркетолога) является браузер.

Github — это облачная платформа для совместной разработки IT-проектов.

Исполнимый модуль - это разновидность файла, содержимое которого является готовой к непосредственному исполнению компьютерной программой.

2.2 Обзор аналогов

Существует большое количество мессенджеров, которые имеют свои преимущества и недостатки. Наиболее используемыми являются Telegram, WhatsApp, Viber, особенности которых необходимо рассмотреть более подробно.

2.2.1 Telegram

Telegram — мессенджер с функциями обмена текстовыми, голосовыми и видеосообщениями, стикерами и фотографиями, файлами многих форматов Интерфейс приложения представлен на Рисунке 1 [6].

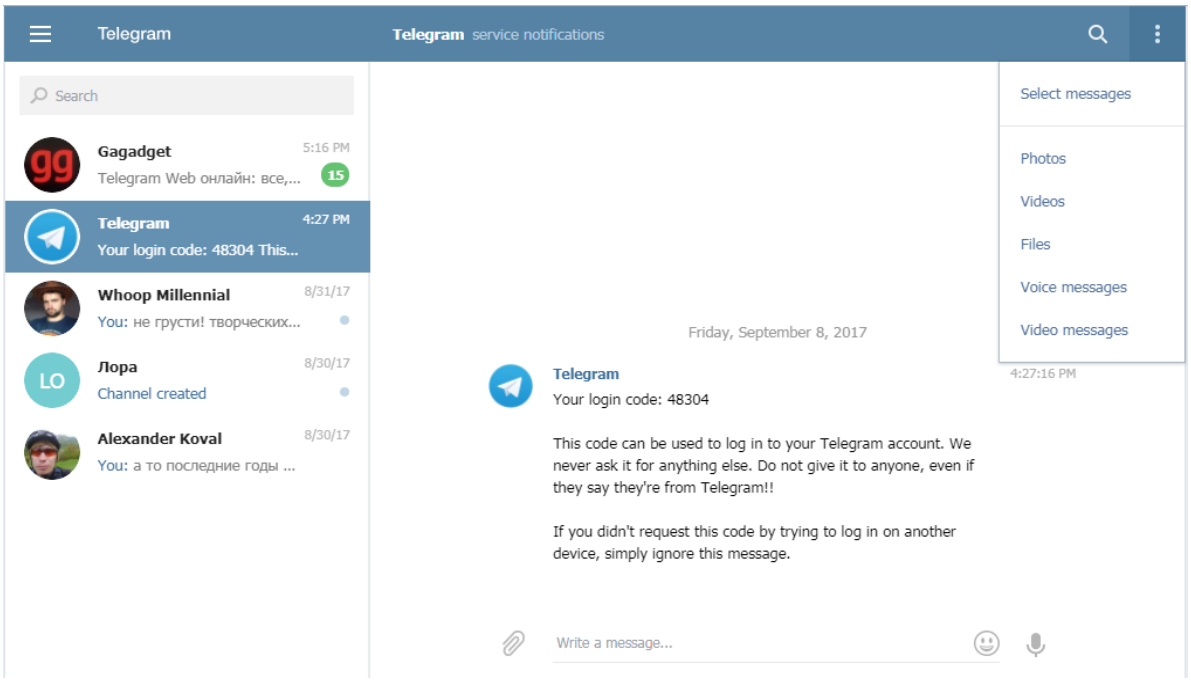

Рисунок 1 - Внешний вид Telegram

Telegram обладает следующим рядом преимуществ:

— Надежная защита от несанкционированного доступа к сообщениям сторонних особ, благодаря MTProto протоколу;

— Функция отправки пользователям файлов больших размеров (до 2 ГБ);

— Таймер автоуничтожения сообщений с выставленным промежутком времени.

И в свою очередь несколькими недостатками:

— Привязка к телефону и отправка ваших контактов на сервер;

— Неанонимный: Хотя Телеграм предоставляет некоторую степень безопасности, он не является анонимным, и все сообщения сохраняются на серверах. Это может быть проблемой для пользователей, которые ищут полную конфиденциальность;

— При добавлении нового аватара — старый не удаляется. Его необходимо удалять вручную.

2.2.2 WhatsApp

WhatsApp — американский бесплатный сервис обмена мгновенными сообщениями и голосовой связи по IP, принадлежащий компании Meta. Интерфейс приложения представлен на Рисунке 2 [7].

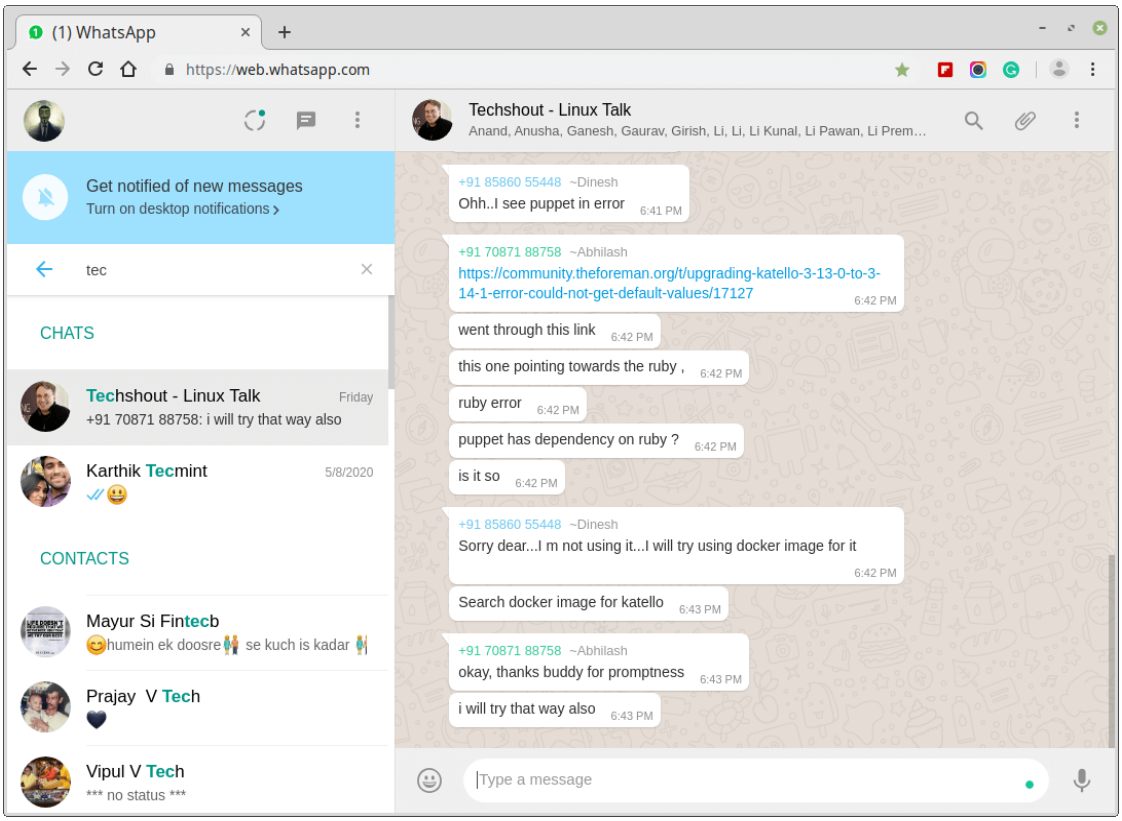

Рисунок 2 - Внешний вид WhatsApp

WhatsApp обладает следующим рядом преимуществ:

— Можно отправлять файлы формата GIF, при чем по умолчанию существует хорошая база файлов данного формата;

— Можно настроить внешний вид приложения под себя (поставить себе статус на 24 часа, аватар, изменить фон чата и т.д.);

— Есть возможность совершать видео- и аудио звонки, отправлять медиа-файлы.

И в свою очередь несколькими недостатками:

— WhatsApp имеет ограничение на размер файлов, которые можно отправлять (до 100 МБ), что может быть неудобно для пользователей, которые часто обмениваются большими файлами;

— Нет возможности отправки анонимных сообщений: WhatsApp не позволяет отправлять анонимные сообщения, и все сообщения сохраняются на серверах. Это может быть проблемой для пользователей, которые ищут полную конфиденциальность.

2.2.3 Viber

Viber — приложение-мессенджер, которое позволяет отправлять сообщения, совершать видео- и голосовые VoIP-звонки через интернет. Интерфейс приложения представлен на Рисунке 3 [8].

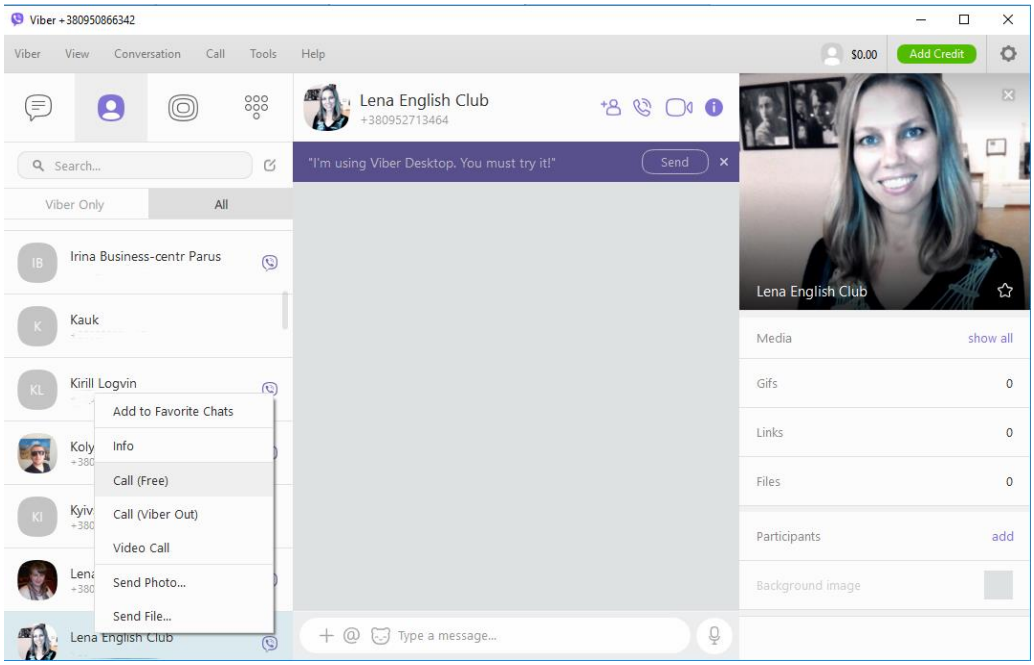

Рисунок 3 - Внешний вид Viber

Viber обладает следующим рядом преимуществ:

— Бесплатный мессенджер;

— Простой и элегантный дизайн;

— Скорость обмена файлами.

И в свою очередь следующим рядом недостатков:

— Ограниченность аудитории: Viber не настолько популярен, как, например, WhatsApp или Telegram, поэтому может быть трудно найти всех своих контактов на этой платформе;
— Viber может работать как фоновое приложение. Если забыть закрыть мессенджер, то он будет продолжать показывать, что пользователь «в Сети». А ещё статус и другие настройки конфиденциальности программа позволяет менять только один раз в 24 часа;
— Возможность взлома. Фонд электронных рубежей (EFF) поставил приложению всего 2 балла из 7. Специалисты фонда отметили, что главные минусы мессенджера - отсутствие тестов на безопасность (публичных), а также закрытый код.

3 Диаграммы, иллюстрирующие работу системы

3.1 Диаграмма прецендентов (Use case)

Диаграмма прецендентов (Use case) представлена на Рисунке 4 и на Рисунке 5. В данной системе имеют место быть два актора: неавторизированный пользователь и авторизированный пользователь.

Неавторизированный пользователь может:

— Регистрироваться;

— Авторизоваться.

Авторизированный пользователь может:

— Редактировать свой профиль;

— Писать сообщения;

— Просматривать чат;

— Просматривать профиль собеседника;

— Выходить из профиля.

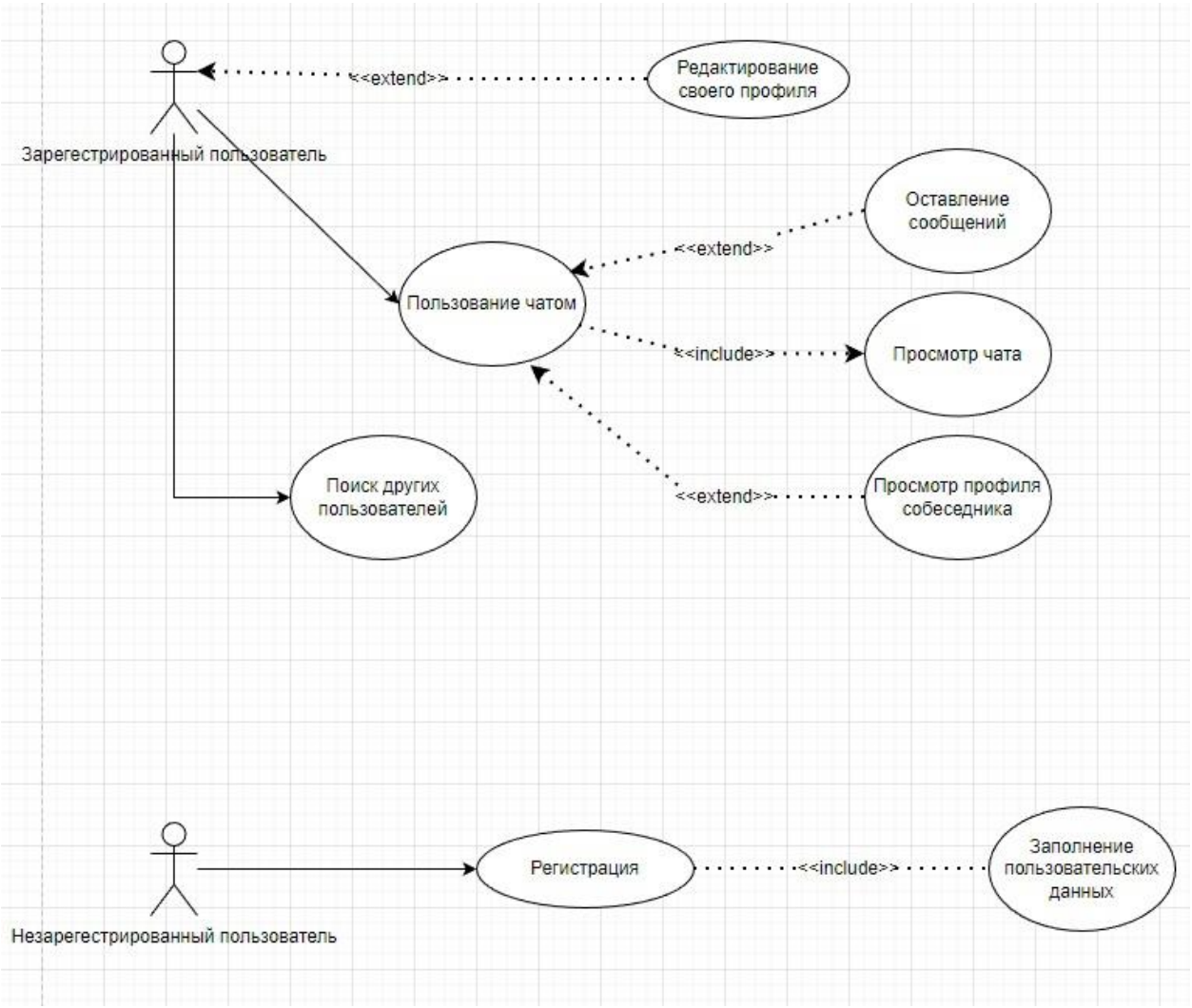

Рисунок 4 - Диаграмма прецедентов (Use case) для авторизованного пользователя

Рисунок 5 - Диаграмма прецедентов (Use case) для незарегистрированного пользователя

3.2 Диаграмма состояний (Statechart diagram)

Диаграмма состояний (Рисунок 6) отражает внутренние состояния объекта в течение его жизненного цикла от момента создания до разрушения

[2]. На данной диаграмме рассмотрены состояния от момента входа в систему до полного выхода из нее.

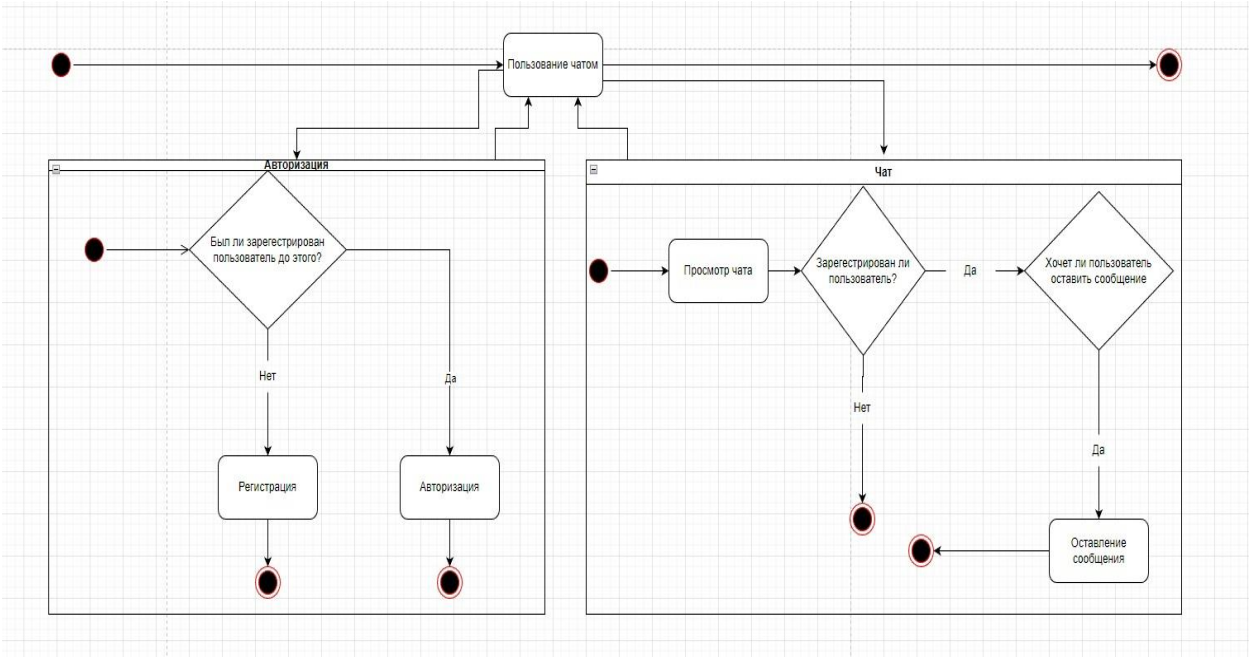

Рисунок 6 - Диаграмма состояний

3.3 Диаграмма активностей (Activity diagram)

Диаграмма деятельности (Рисунок 7) представляет собой диаграмму, на которой показаны действия, состояния которых описаны на диаграмме состояний. Она описывает действия системы или людей, выполняющих действия, и последовательный поток этих действий [2].

В данном случае рассмотрен путь действий пользователя. Диаграмма показывает, что пользователь, находясь в неавторизованной зоне системы не может заходить в мессенджер. Чтобы получить возможность использовать функционал, ему необходимо авторизоваться или зарегистрироваться.

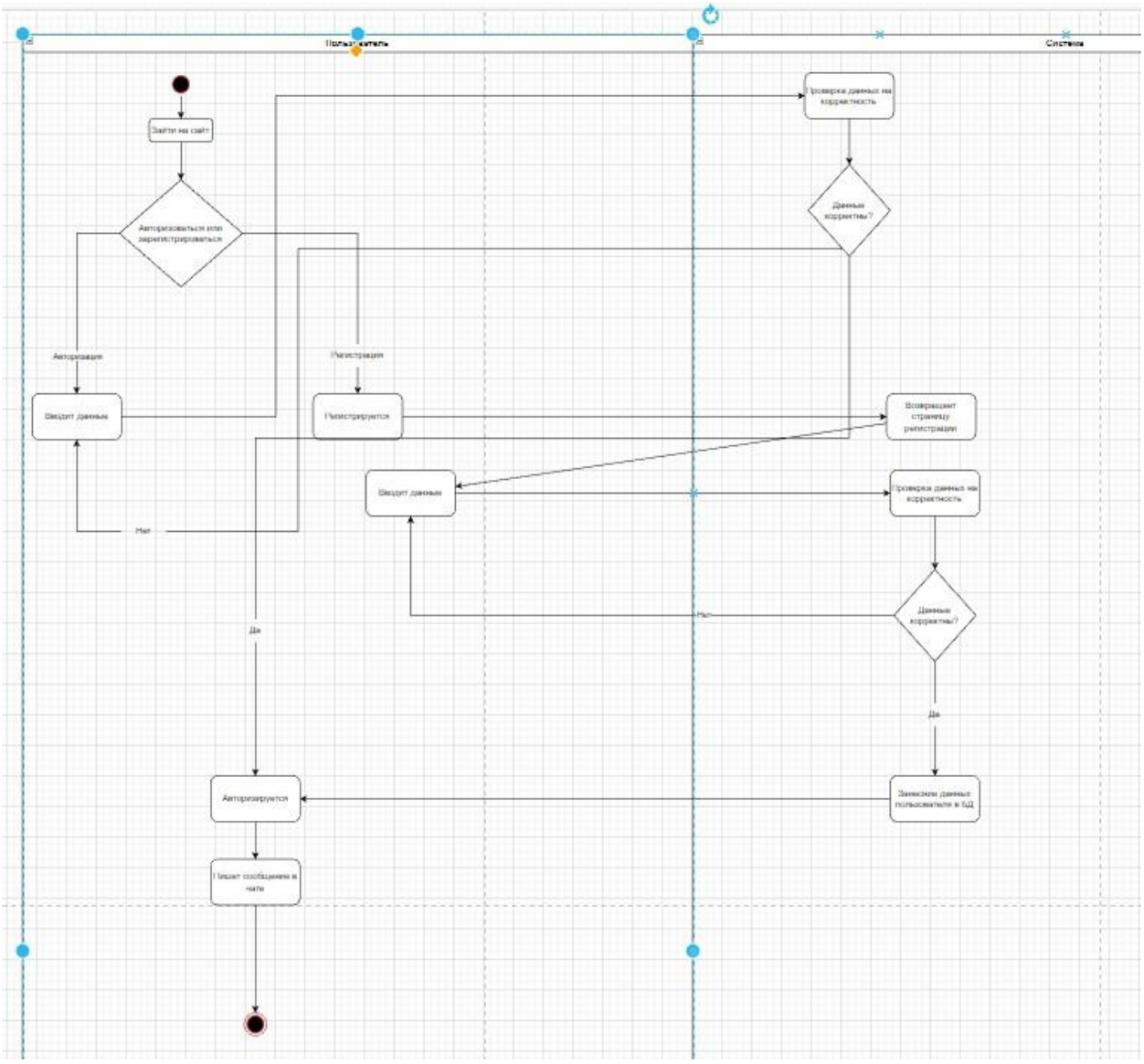

Рисунок 7 - Диаграмма активности (деятельности)

3.4 Диаграмма классов (Class diagram)

Диаграмма классов (Рисунок 8) демонстрирует общую структуру иерархии классов системы, их кооперацией, атрибутов, методов, интерфейсов и взаимосвязей между ними. В данной системе рассмотрены следующие классы:

— Класс «Пользователь»;

— Класс «Чат»;

— Класс «Сообщение».

У каждого из классов существуют свои атрибуты.

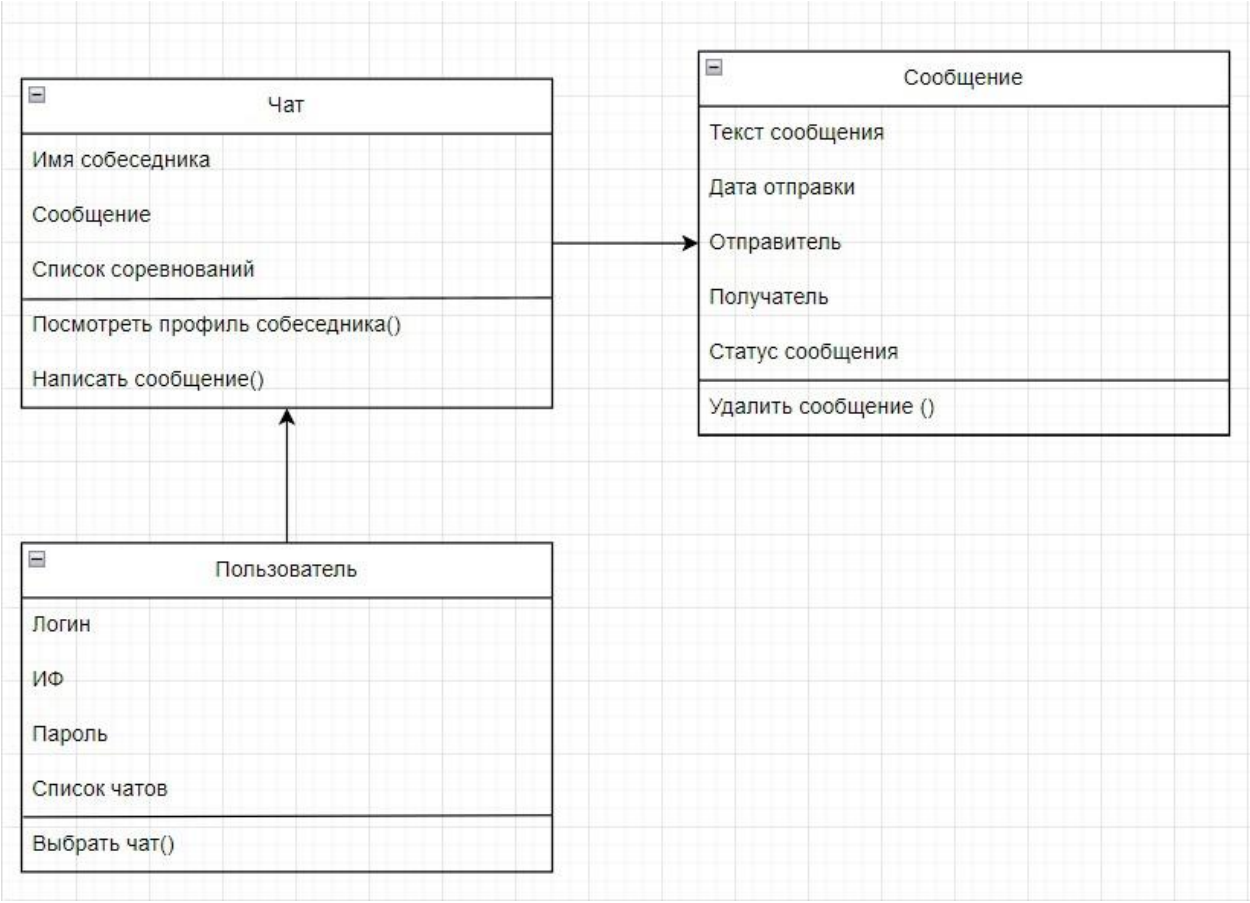

Рисунок 8 - Диаграмма классов

3.5 Диаграмма объектов (Object diagram)

По подобию диаграммы классов (Рисунок 8) была выполнена диаграмма объектов (Рисунок 9).

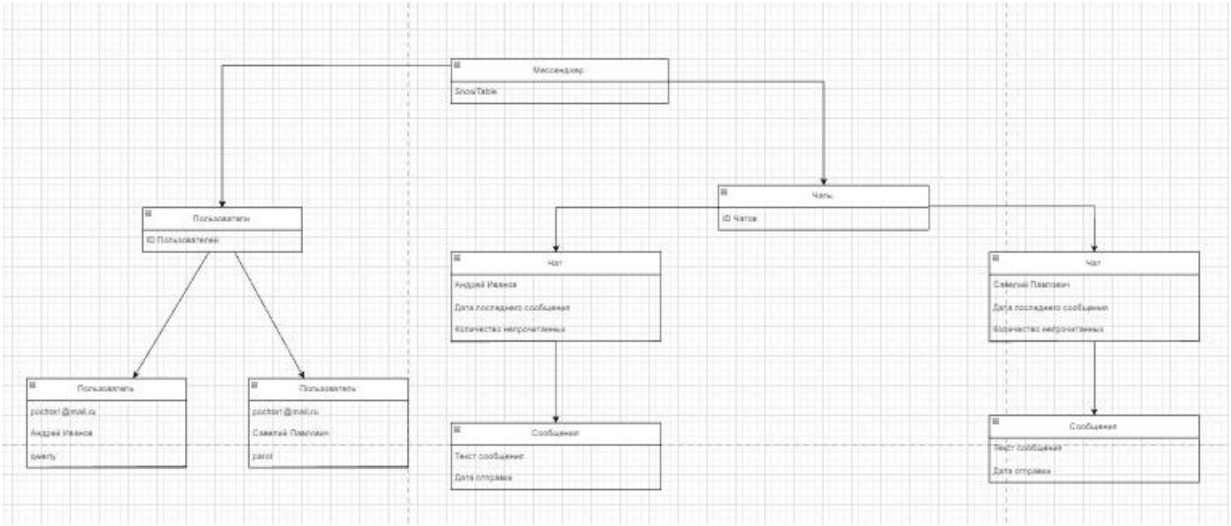

Рисунок 9 - Диаграмма объектов

3.6 Диаграмма сотрудничества (Collaboration diagram)

Диаграмма сотрудничества (Рисунок 10) — это вид диаграммы взаимодействия, в котором основное внимание сосредоточено на структуре взаимосвязей объектов, принимающих и отправляющих сообщения.

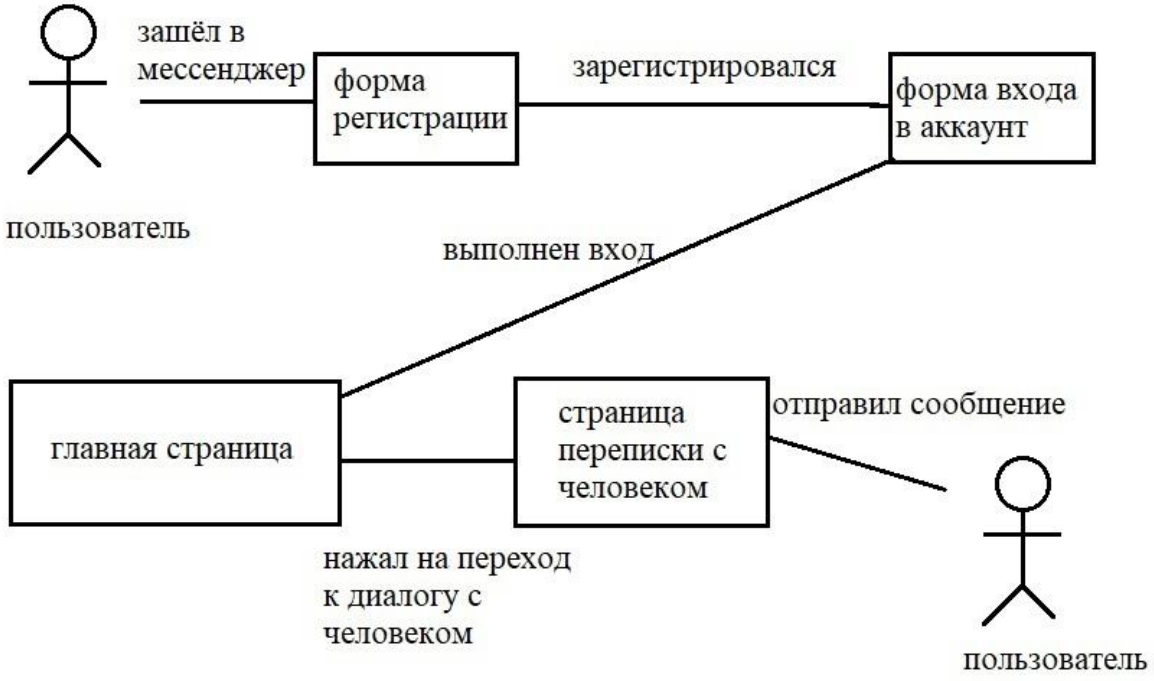

Рисунок 10 - Диаграмма сотрудничества

3.7 Диаграмма развертывания (Deployment diagram)

Диаграмма развертывания (Рисунок 11) предназначена для представления общей конфигурации или топологии распределенной программной системы.

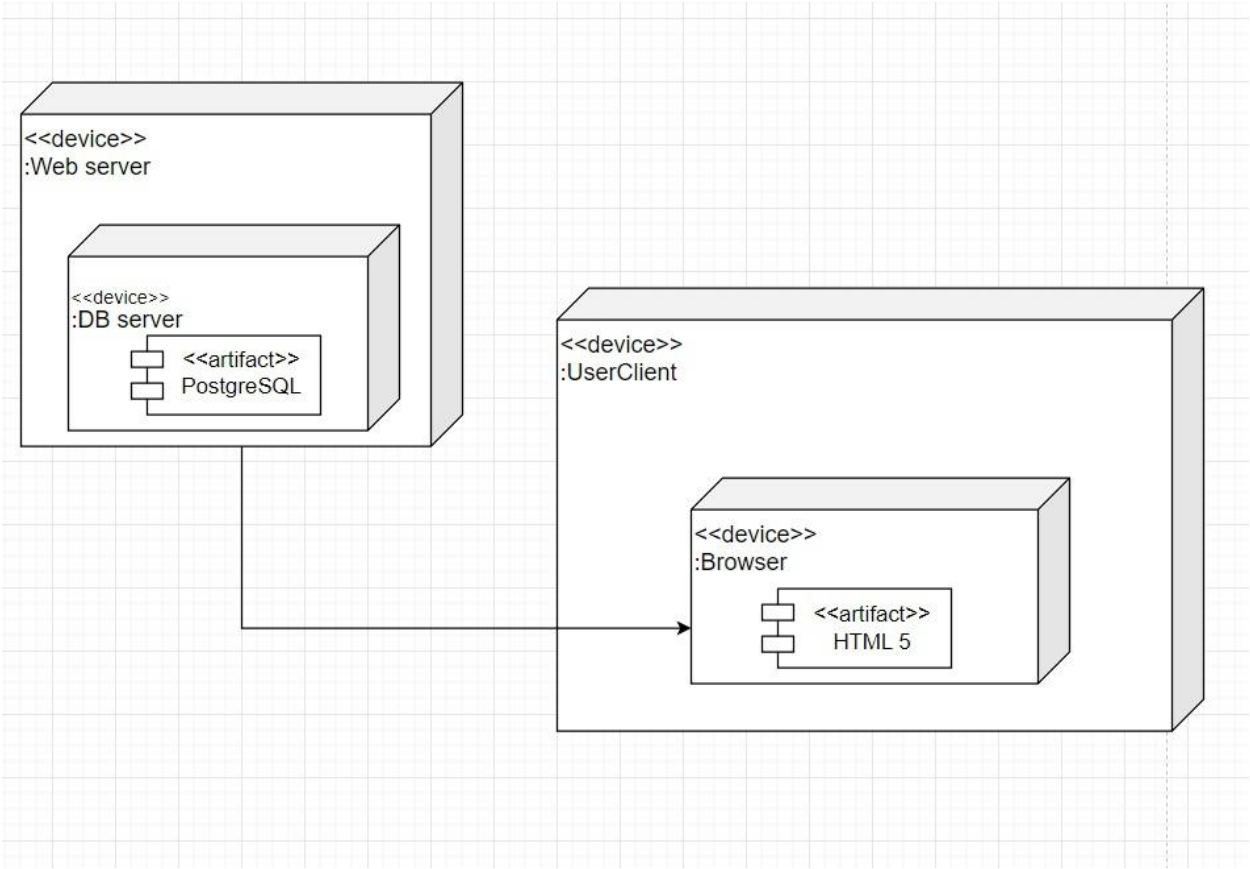

Рисунок 11 - Диаграмма развертывания

4 Реализация

4.1 Средства реализации

Веб-приложение использует клиент-серверную архитектуру и разделяется на две основные части: backend и frontend. Обмен данных между ними осуществляется посредством REST API [1]. В процессе разработки был использован Git-flow для организации работы. В нашем проекте эта модель ветвления включает две основные ветки: main и develop, а также временные ветки для добавления новых функций.

Для обеспечения реализации backend-составлящей приложения был выбран высокоуровневый веб-фреймворк языка Python Django, который позволяет быстро создавать безопасные и поддерживаемые веб-сайты.

Этот выбор объясняется следующими пунктами:

— Широкий набор функций: Django имеет множество встроенных функций, таких как автоматическая административная панель, обработка форм, аутентификация пользователей и многое другое, что упрощает разработку веб-приложений;

— Стандартизированный подход: Django использует стандартизированный подход к разработке веб-приложений, что облегчает создание качественных приложений с четкой структурой и легким сопровождением в дальнейшем;

— Безопасность: Django имеет встроенные функции для обеспечения безопасности веб-приложений, такие как защита от CSRF-атак, секретное хеширование паролей пользователей и многие другие;

— Расширяемость: Django предоставляет множество расширяемых компонентов, которые можно легко добавлять в проект, такие как библиотеки для работы с базами данных, асинхронное программирование, создание API [3] и многое другое;

— Кроссплатформенность: Django поддерживает работу на разных платформах, что позволяет создавать веб-приложения, которые будут работать на разных устройствах и операционных системах.

На основе данных пунктов был сделан выбор в пользу веб-фреймворка Django.

Для обеспечения реализации frontend-составлящей приложения был выбран фреймворк языка JavaScript React, и вот почему [5]:

— Простота интеграции: React легко интегрируется с другими фреймворками и библиотеками, что делает его удобным инструментом для создания комплексных приложений;

— Широкий набор инструментов: React работает во множестве сред разработки и поддерживает множество инструментов и библиотек для улучшения и оптимизации кода;

— Открытый и широко используемый фреймворк: React имеет большое сообщество разработчиков и обширную документацию, что облегчает изучение фреймворка и нахождение решений на проблемы;

— Эффективность: React использует виртуальный DOM, что делает обновление и отображение изменений на странице более эффективным и быстрым;

— Инициализация (__init__.py) — это специальный файл в Python, определяющий пакет. В Django он используется для определения пакета для приложения;

— Административная панель (admin.py) — это административная панель Django, которая обеспечивает автоматическую генерацию форм и предоставляет удобный интерфейс для работы с данными;

— Конфигурационный файл (apps.py) — это Python файл, который определяет основные настройки для приложения. Он содержит метаданные, относящиеся к данному приложению, и позволяет

настроить приложение в соответствии с требованиями и лучше контролировать его работу;

— Модели данных (models.py) — это классы Python, которые определяют структуру базы данных и могут быть использованы для создания или обновления схемы базы данных. Модели могут содержать поля для хранения данных (текстовые, числовые, даты, файлы и др.), а также методы для работы с этими данными;

— Сериализация (serializers.py) — файл, обеспечивающий сериализацию и десериализацию данных, передаваемых через приложение. Он используется в Django для работы с данными, передаваемыми через HTTP в API-модулях [3]. Файл содержит классы сериализаторов, которые определяют, какие поля модели должны быть преобразованы в JSON, XML и т.д. Они также могут обеспечивать валидацию данных во время десериализации;

— Тесты (test.py) — это файл в приложении, который содержит модульные тесты для этого приложения. Эти тесты используются, чтобы убедиться, что функциональные возможности приложения работают должным образом, и выявить любые ошибки или ошибки до того, как приложение будет развернуто в рабочей среде;

— URL-адресация (urls.py) — это механизм маршрутизации запросов на определенные представления. Django использует файл urls.py для определения соответствующей представлению URL-адреса;

— Представления (views.py) — это функции Python, которые обрабатывают запросы от клиента и возвращают HTTP-ответы. Представления могут включать в себя логику приложения, обработку данных из модели, взаимодействие с другими системами.

Swagger – инструмент для документирования и тестирования API [3]. Он позволяет создавать интерактивную документацию для вебсервисов, что упрощает их использование и интеграцию. Swagger автоматически генерирует

документацию на основе аннотаций в коде, что позволяет разработчикам сосредоточиться на написании логики приложения, а не на создании и поддержке документации. Благодаря Swagger, разработчики могут изучить доступные эндпоинты, параметры, модели данных и примеры запросов и ответов.

4.2 Реализация Backend

Серверная (backend) часть приложения была написана на языке Phyton с использованием фреймворка Django. Django подразумевает разделение приложения на модули (прослойки), которые будут описаны далее. Каждый модуль организован в виде отдельного пакета.

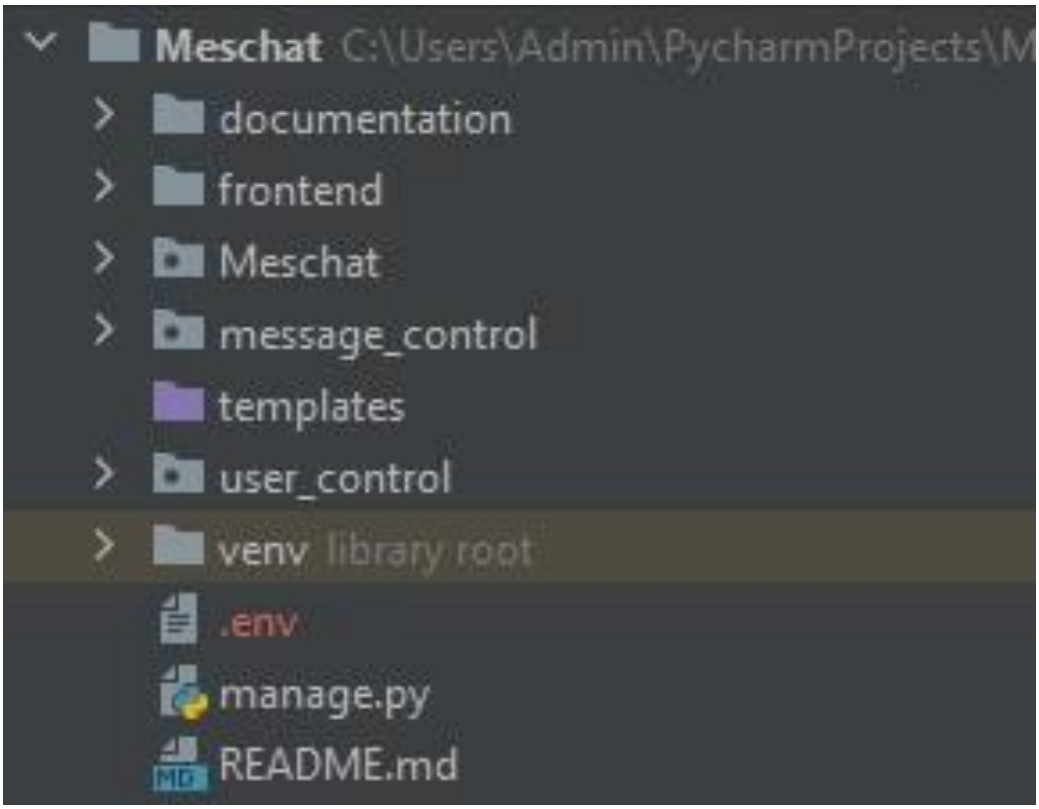

Рисунок 12 - Структура проекта

В корневой папке (Рисунок 13) особое внимание стоит обратить на файл settings.py, который содержит настройки проекта. Он содержит множество параметров для настройки работоспособности приложения:

— DEBUG — параметр, отвечающий за включение/выключение режима отладки.

— SECRET_KEY — секретный ключ приложения, который применяется для генерации токенов, а также для шифрования паролей пользователей.

— DATABASES — настройки подключения к базе данных, в которой хранятся данные приложения.

— INSTALLED_APPS — список приложений, установленных в проекте. Здесь указываются все приложения, которые будут использованы в проекте.

— MIDDLEWARE — список всех промежуточных компонентов, используемых в Django для обработки запросов и ответов между сервером и клиентом.

— TEMPLATES — настройки для генерации HTML-шаблонов, используемых для визуализации данных пользователя в БД.

— AUTH_PASSWORD_VALIDATORS — список компонентов, используемых для проверки безопасности паролей пользователей.

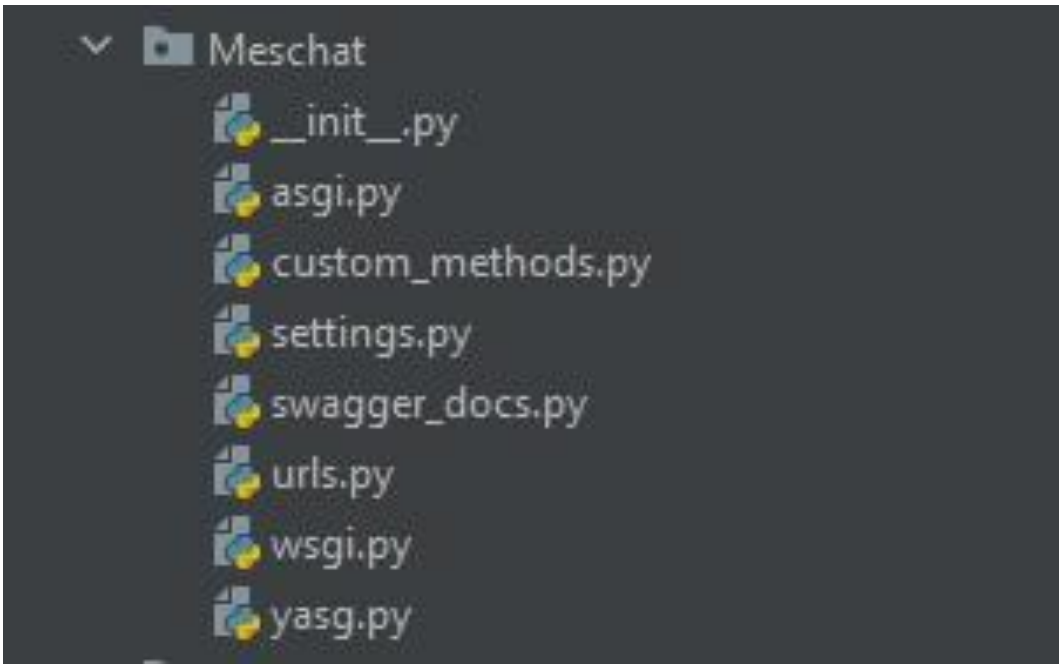

Рисунок 13 - Структура корневой папки проекта

При этом фреймворк Django подразумевает разделение приложения на несколько основных компонентов (Рисунок 14)

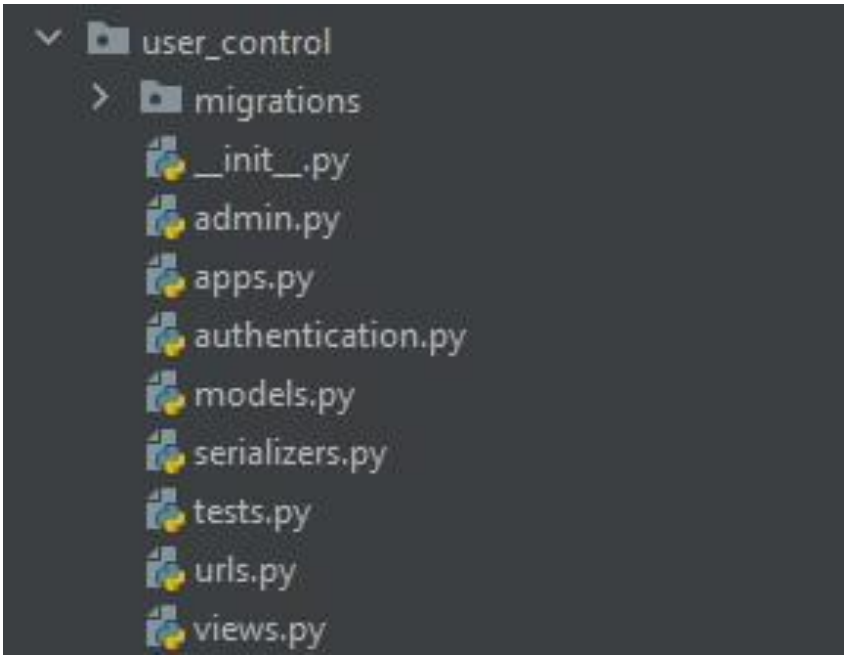

Рисунок 14 - Основные компоненты Django на примере user_control

Серверная часть приложения Django также может использовать множество дополнительных библиотек и компонентов.

База данных была развернута на виртуальной машине с помощью системы контейнеризации Docker. Такие данные как хост, порт, имя базы данных, имя пользователя и пароль задавались при помощи файла dockercompose.yaml. Эти данные будут использоваться для настройки подключения к базе данных из приложения.

Сервер был развернут также на виртуальной машине с помощью системы контейнеризации Docker. Этот процесс был выполнен с использования системы контроля версий Git. После загрузки потребовалось настроить переменные окружения, представляющие собой конфиденциальные настройки, такие как данные аутентификации для базы данных. Клиентская часть была загружена в другом Docker-контейнере.

Для описания спецификации API использовался Swagger [3]. Для того чтобы интегрировать его в проект, нужно было внести изменения в код,

включая добавление необходимой зависимости (drf_yasg), добавить ее в INSTALLED_APPS и описать спецификации всех методов.

4.3 Реализация Frontend

Клиентская часть приложения (frontend) разработана на языке JavaScript с использованием фреймворка React.js.

Одной из особенностей разработки приложения на React является декомпозиция приложения на независимые компоненты. Каждый компонент отвечает за определенный функциональный блок приложения и представляет собой функцию, которая возвращает HTML-код. Кроме того, компоненты могут содержать переменные и другие функции, что позволяет дополнительно упростить код и повысить его читабельность. Приложение загружается на страницу по мере необходимости, что уменьшает время загрузки страницы и повышает производительность приложения в целом.

Основной HTML файл называется index.html, именно на него загружаются компоненты, которые будут выведены в браузере. В нём вызывается index.js, в котором находится точка входа React-приложения, и вызывается основной компонент App.js, который будет меняться в зависимости от действий пользователя. Все остальные компоненты вызываются по мере необходимости.

Структура проекта на React включает в себя определенные основные элементы (Рисунок 15).

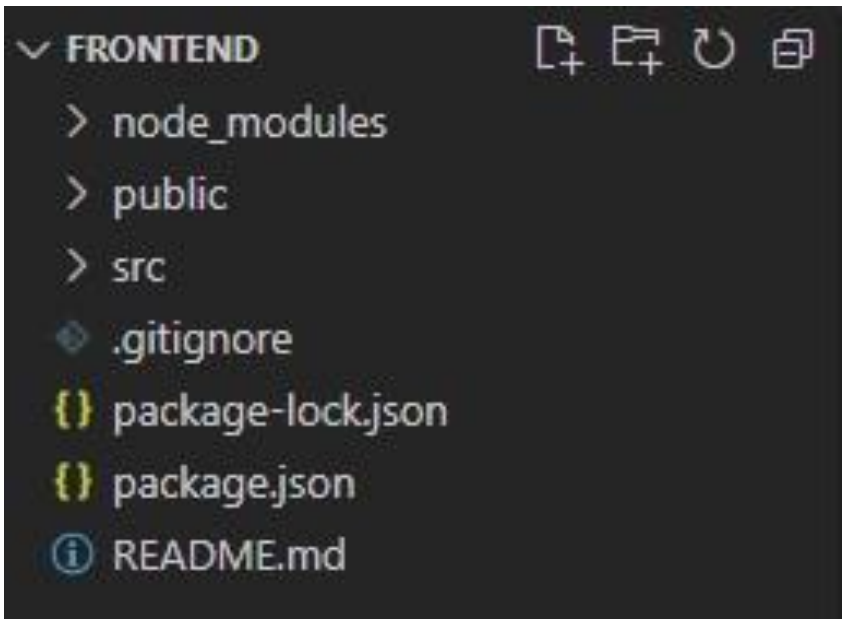

Рисунок 15 - Основные элементы React-проекта

— Основная папка проекта src/, содержащая исходный код;

— Папка public/, содержащая файлы, которые будут доступны публично;

— Файл package.json, содержащий информацию о проекте, а также список зависимостей для установки;

— Файл package-lock.json с информацией о текущей версии зависимостей.

В свою очередь в папке src/ (Рисунок 16) хранятся следующие разделы приложения.

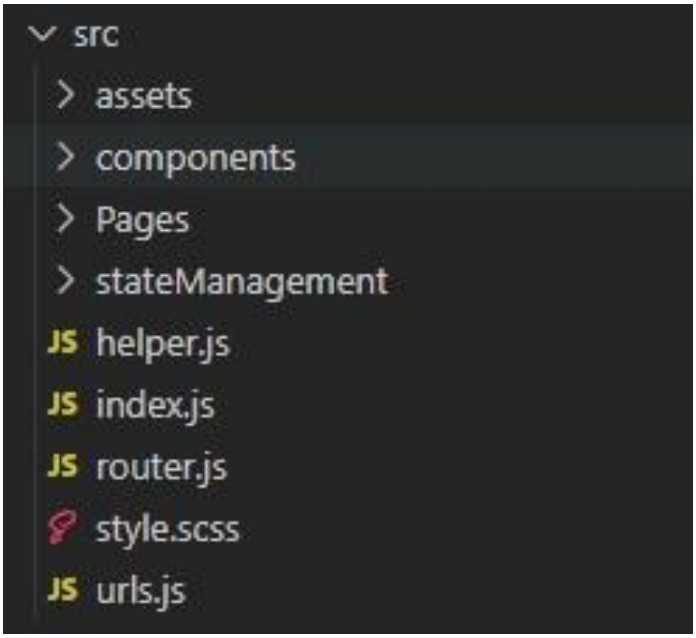

Рисунок 16 - Структура папки src/ проекта

— Папка assets/, содержащая все иконки, используемые для дизайна мессенджера;
— Папка components/, содержащая файл loader.js, который является компонентой для анимации прогрузки чата, веб-страницы;
— Папка Pages/, содержащая все необходимые react-компоненты для работы приложения;
— Папка stateManagement/, содержащая все файлы, используемые для описания логики различных состояний;
— Файл helper.js, содержащий обработчик отправки запросов;
— Файл index.js, который является точкой входа приложения;
— Файл router.js, который используется для настройки маршрутизации в приложении;
— Файл style.scss, содержащий все стили приложения;
— Файл url.js, содержащий ссылки для обращения к API.

В папке Pages/ содержатся все основные компоненты приложения (Рисунок 17).

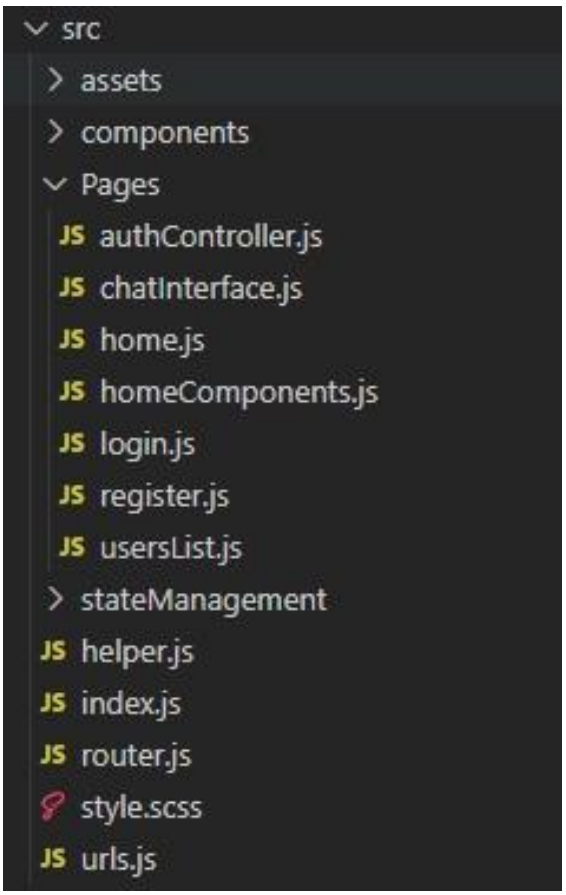

Рисунок 17 - Компоненты приложения

— Файл authController.js, отвечающий за контроль аутентификации пользователя;
— Файл chatInterface.js, который предназначен для отображения интерфейса чата;
— Файл home.js, который отвечает за представление основного интерфейса домашней страницы приложения;
— Файл homeComponents.js, который предназначен для отображения основной информации о пользователе, сообщений и его аватара;
— Файл login.js, отвечающий за отображение и обработку формы входа и аутентификации;
— Файл register.js, который отвечает за отображение и обработку формы регистрации пользователей;

— Файл usersList.js, который отвечает за отображение списка пользователей и выполнение некоторых операций связанных с этим списком.

4.4 Тестирование

Для тестирования разработанной системы были проведены тесты основных типов:

— Модульное тестирование (unit-тестирование);

— Тесты, связанные с изменениями (дымовое тестирование);

— Функциональные тесты (GUI-тестирование).

4.4.1 Модульное тестирование

Unit-тесты являются частью процесса разработки программного обеспечения и используются для проверки отдельных модулей или компонентов программы. Они предназначены для тестирования функциональности отдельных единиц кода, таких как отдельные классы, методы или функции.

Целью unit-тестов является проверка, что каждая единица программного кода работает правильно в изоляции от других компонентов системы. Unit-тесты проверяют, что методы возвращают ожидаемые результаты при заданных входных данных, а также что ошибочные ситуации обрабатываются корректно.

```python
def test_post_message(self):

    payload = {
        "sender_id": self.sender.id,
        "receiver_id": self.receiver.id,
        "message": "test message",

    }

    response = self.client.post(
        self.message_url, data=payload, **self.bearer)
    result = response.json()

    self.assertEqual(response.status_code, 201)
    self.assertEqual(result["message"], "test message")
    self.assertEqual(result["sender"]["user"]["username"], "sender")
    self.assertEqual(result["receiver"]["user"]["username"], "receiver")
```

Рисунок 18 - Модульное тестирование для отправки сообщения

```python

def test_delete_message(self):

    payload = {
        "sender_id": self.sender.id,
        "receiver_id": self.receiver.id,
        "message": "test message",

    }
    self.client.post(self.message_url, data=payload, **self.bearer)

    response = self.client.delete(
        self.message_url+"/1", data=payload, **self.bearer)

    self.assertEqual(response.status_code, 204)

def test_get_message(self):

    response = self.client.get(
        self.message_url+f"?user_id={self.receiver.id}", **self.bearer)
    result = response.json()

    self.assertEqual(response.status_code, 200)
```

Рисунок 19 - Модульное тестирование для удаления и получения сообщения

Рисунок 20 - Модульное тестирование для отправки профиля пользователя

4.4.2 Дымовое тестирование

Дымовое тестирование — это тип тестирования, при котором производится кратковременная проверка основной функциональности системы, чтобы убедиться в отсутствии серьезных проблем, ошибок или сбоев.

Такой тип тестирования полезен, чтобы обнаружить крупные проблемы в функциональности системы до того, как будут проведены более подробные и насыщенные тесты. Если в процессе дымового тестирования найдены ошибка или проблемы, то это может стать причиной для проведения дополнительных и более детальных тестов, чтобы устранить недочеты и проблемы в работе программы.

В ходе дымового тестирования выполняются базовые операции или сценарии, которые предполагаются как наиболее важные и часто используемые пользователем. Далее представлены результаты дымового тестирования для основных сценариев неавторизованного и авторизованного пользователей (Таблица 1-2).

Таблица 1 - Результаты дымового тестирования для неавторизованного пользователя

Тестовый сценарий	Результат теста
Регистрация	Пройден
Авторизация	Пройден

Таблица 2 - Результаты дымового тестирования для авторизованного пользователя

Тестовый сценарий	Результат теста
Поиск пользователей	Пройден
Добавление пользователей в избранное	Пройден
Просмотр информации о собеседнике	Пройден
Редактирование информации о себе	Пройден
Отправка сообщений	Пройден

4.4.3 GUI-тестирование

Тестирование пользовательского интерфейса (GUI-тестирование) — это процесс тестирования элементов управления в приложении, который помогает убедиться, что интерфейс соответствует ожидаемой функциональности, включая проверку различных функций и элементов, таких как окна,

диалоговые окна, кнопки, переключатели, выпадающие списки, формы, меню и т.д.

Задача проведения GUI-тестов — убедиться, что в функциях пользовательского интерфейса отсутствуют дефекты. В Таблице 3 представлена часть результатов тестирования пользовательского интерфейса. В ней отражены тестовые сценарии для неавторизованного пользователя.

Таблица 3 - Результаты GUI-тестирования

Тестовый сценарий	Ожидаемый результат	Результат теста
Нажатие на кнопку «Регистрация»	Переход на страницу регистрации	Пройден
Нажатие на кнопку «Авторизация»	Переход на страницу авторизации	Пройден
Ввод текста в поле «Поиск собеседника»	Вывод пользователей на главной странице, соответстующих введённому тексту	Пройден
Нажатие на кнопку «Настройки» у пользователя	Вывод формы для редактирования профиля	Пройден
Нажатие на кнопку «Просмотреть профиль собеседника»	Вывод формы для просмотра профиля собеседника	Пройден
Нажатие на чат собеседника	Переход на страницу чата с выбранным собеседником	Пройден

Нажатие на кнопку «Отправить сообщение»	Отправка сообщения и вывод его в чате	Пройден
Нажатие на кнопку «Добавить в избранное»	Перенос выбранного собеседника в верх списка	Пройден
Нажатие на кнопку «Выйти»	Выход из аккаунта	Пройден

Заключение

В результате курсовой работы был разработан веб-мессенджер «MesChat», реализующий следующие функции:

— Организация обмена сообщениями;

— Возможность добавить важные чаты в раздел «Избранное»;

— Возможность поиска пользователей.

Разработанный мессенджер в полной мере соответствует требованиям, предъявленным на этапе постановки задачи. В ходе выполнения курсовой работы был проведён детальный анализ предметной области, а также технологий реализации и необходимых программных платформ, а также выполнены все этапы проектирования и разработки веб-мессенджера.

Список использованной литературы

1. Rest, что это такое? [Электронный ресурс]. – Режим доступа: https://habr.com/ru/articles/590679/. – Заглавие с экрана. – (Дата обращения: 24.04.2023).
2. UML.indd [Электронный ресурс]. – Режим доступа: https://csharpcooking.github.io/theory/UnifiedModelingLanguageUsereGuide.pdf. – Заглавие с экрана. – (Дата обращения: 27.04.2023).
3. Тестирование API с помощью Swagger: особенности и преимущества [Электронный ресурс]. – Режим доступа: https://blog.ithillel.ua/ru/articles/api/testing-with-swagger. – Заглавие с экрана. – (Дата обращения: 27.04.2023).
4. Система управления объектно-реляционными базами данных PostgreSQL [Электронный ресурс]. – Режим доступа: https://web-creator.ru/technologies/webdev/postgresql. – Заглавие с экрана. – (Дата обращения: 20.04.2023).
5. Introduction - React-Native [Электронный ресурс]. – Режим доступа: https://reactnative.dev/docs/getting-started. – Заглавие с экрана. – (Дата обращения: 27.05.2023).
6. Telegram [Электронный ресурс]. – Режим доступа: https://web.telegram.org/a/. – Заглавие с экрана. – (Дата обращения: 27.05.2023).
7. WhatsApp [Электронный ресурс]. – Режим доступа: https://web.whatsapp.com/. – Заглавие с экрана. – (Дата обращения: 27.05.2023).
8. Viber [Электронный ресурс]. – Режим доступа: https://www.viber.com/ru/. – Заглавие с экрана. – (Дата обращения: 27.05.2023).