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 пользователей возможностью общаться с родными, близкими и не очень людьми, используя сеть Интернет.

Целью данного проекта является разработка сайта с возможностью общаться и вести диалоги людям, авторизованным в системе, а также использовать возможности искуственного интеллекта ChatGPT.

1.1 Требования к разрабатываемой системе

1.1.1 Функциональные требования

К разрабатываемому приложению выдвигаются следующие функциональные требования для пользователя:

— Работающий чат;

— Добавление чатов в избранные;

— Поиск пользователей;

— ИИ помощник.

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 Средства реализации

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 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:

— 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.

1.3 Требования к архитектуре

Список требований к архитектуре выглядит следующим образом:

— Приложение должно быть построено на клиент-серверной архитектуре с использованием протоколов HTTP/HTTPS, 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;

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

1.4 Задачи, решаемые в процессе разработки

В процессе разработки интернет-каталога кондитерских изделий 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.

— 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;

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

Проект — это некоторая 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, предназначенная для пользователей, не имеющих постоянной учетной записи на компьютере или в домене;

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 HTMLстраницей, 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.

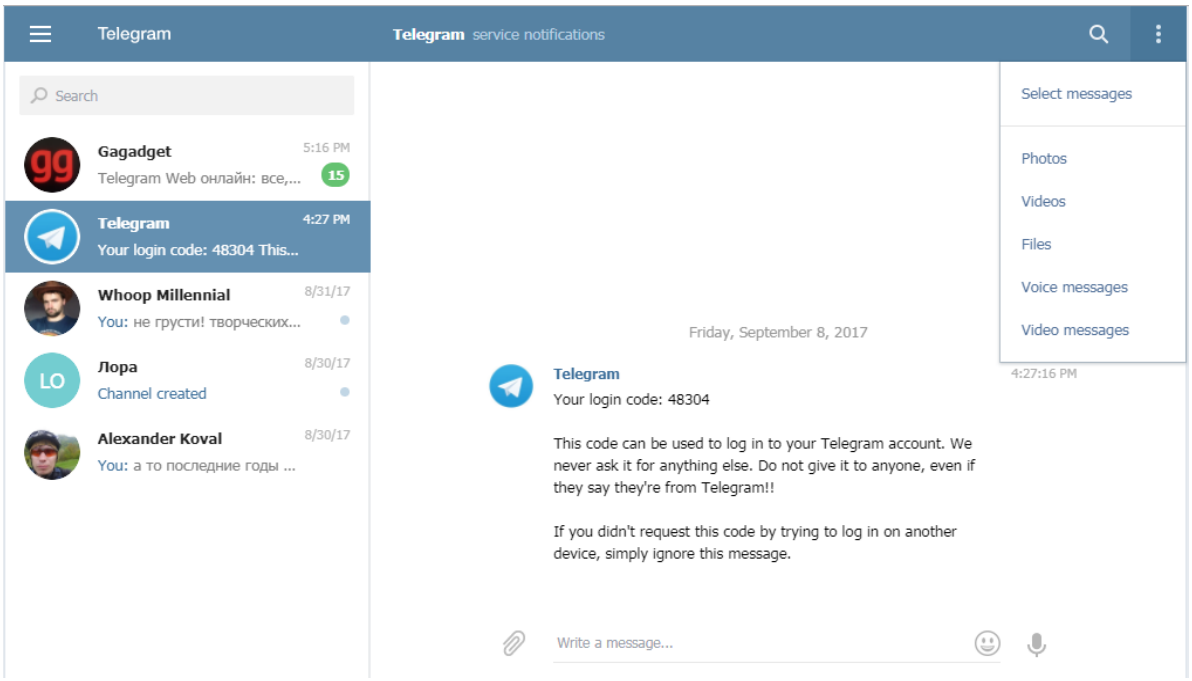

Рисунок 1 - Внешний вид Telegram.

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.

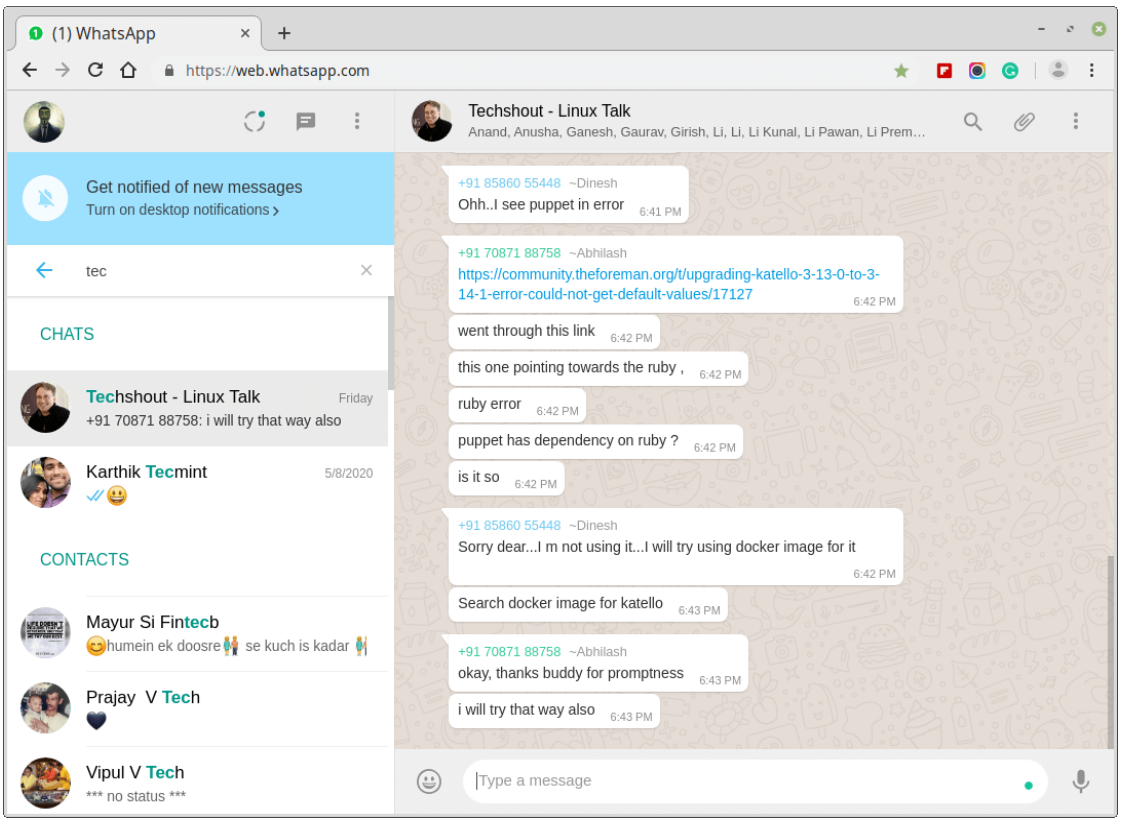

Рисунок 2 - Внешний вид WhatsApp.

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.

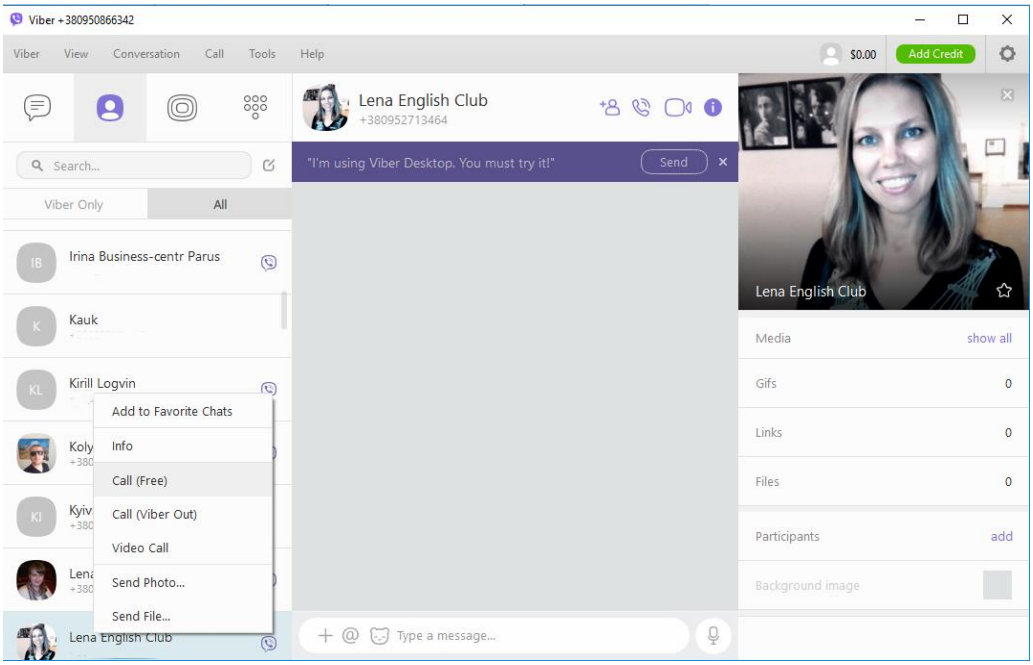

Рисунок 3 - Внешний вид Viber.

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.

2.3 Диаграммы, иллюстрирующие работу системы

2.3.1 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 помимо функций, доступных неавторизированному пользователю, может:

— Редактировать свой профиль.

— Писать сообщения.

— Просматривать чат.

— Выходить из профиля.

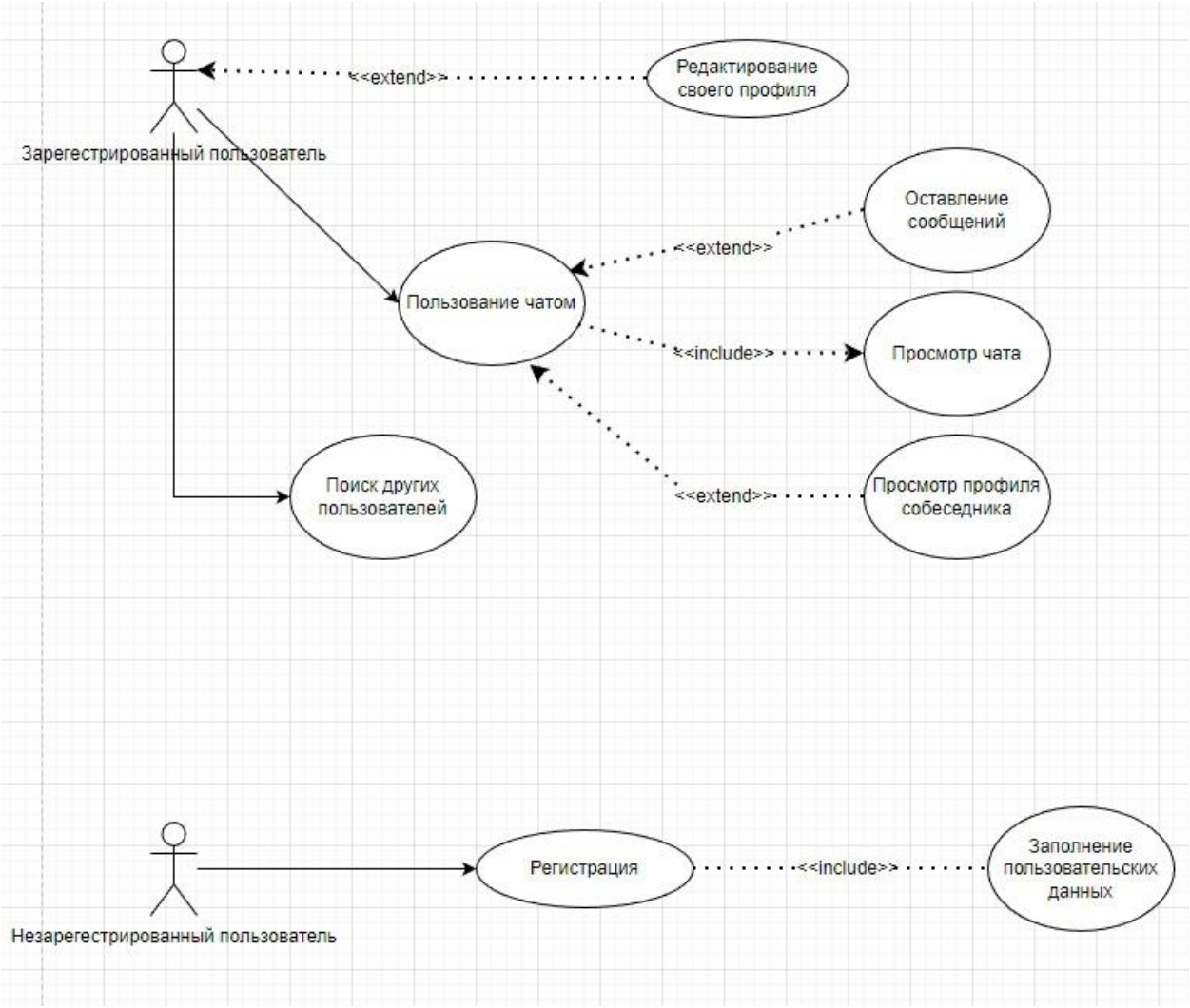

Рисунок 4 - Диаграмма прецедентов (Use case) для авторизованного пользователя.

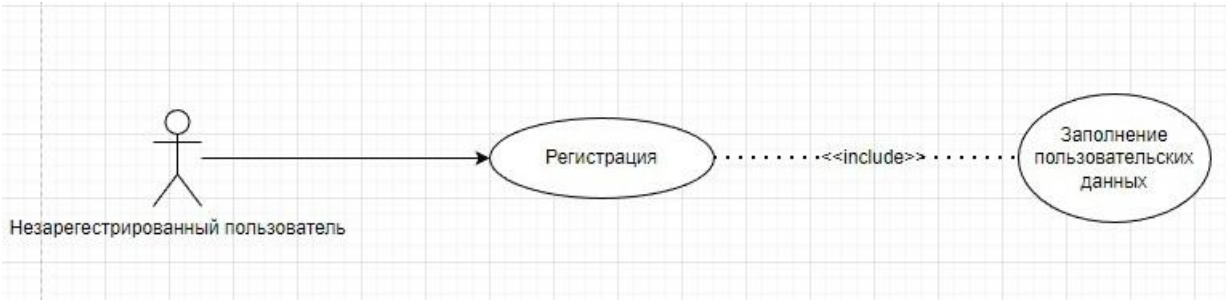

Рисунок 5 - Диаграмма прецедентов (Use case) для незарегистрированного пользователя.

2.3.2 Диаграмма состояний (Statechart diagram)

На рисунке 6 изображена диаграмма состояний. На ней рассмотрены состояния пользования чатом.

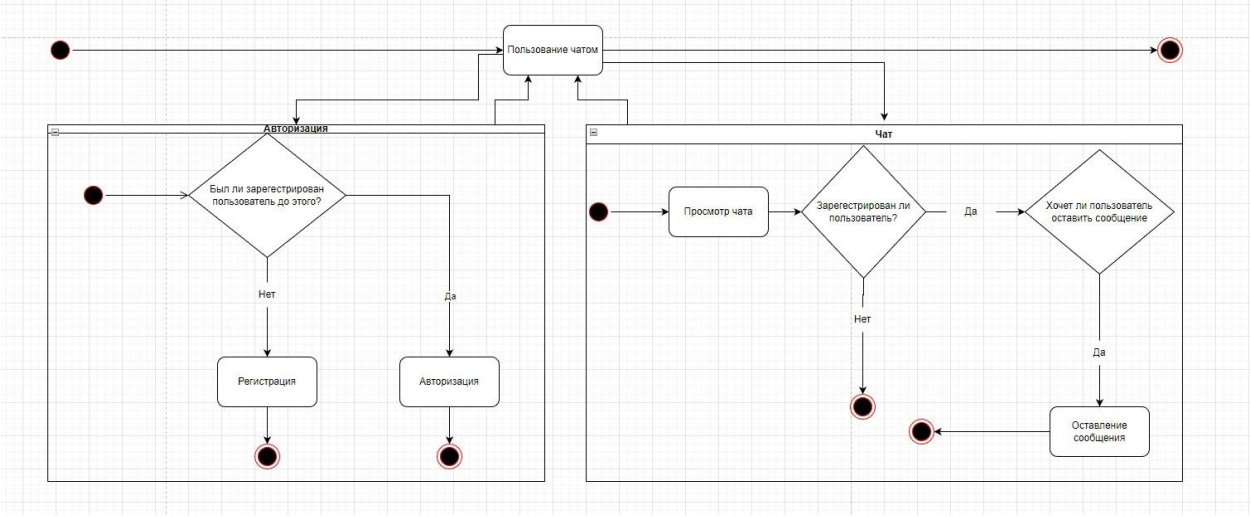

Рисунок 6 - Диаграмма состояний

2.3.3 Диаграмма активностей (Activity diagram)

Диаграмма активности (Рисунок 7) описывает последовательность действий системы или людей, включая состояния, описанные на диаграмме состояний. Она показывает действия пользователя и их последовательный поток.

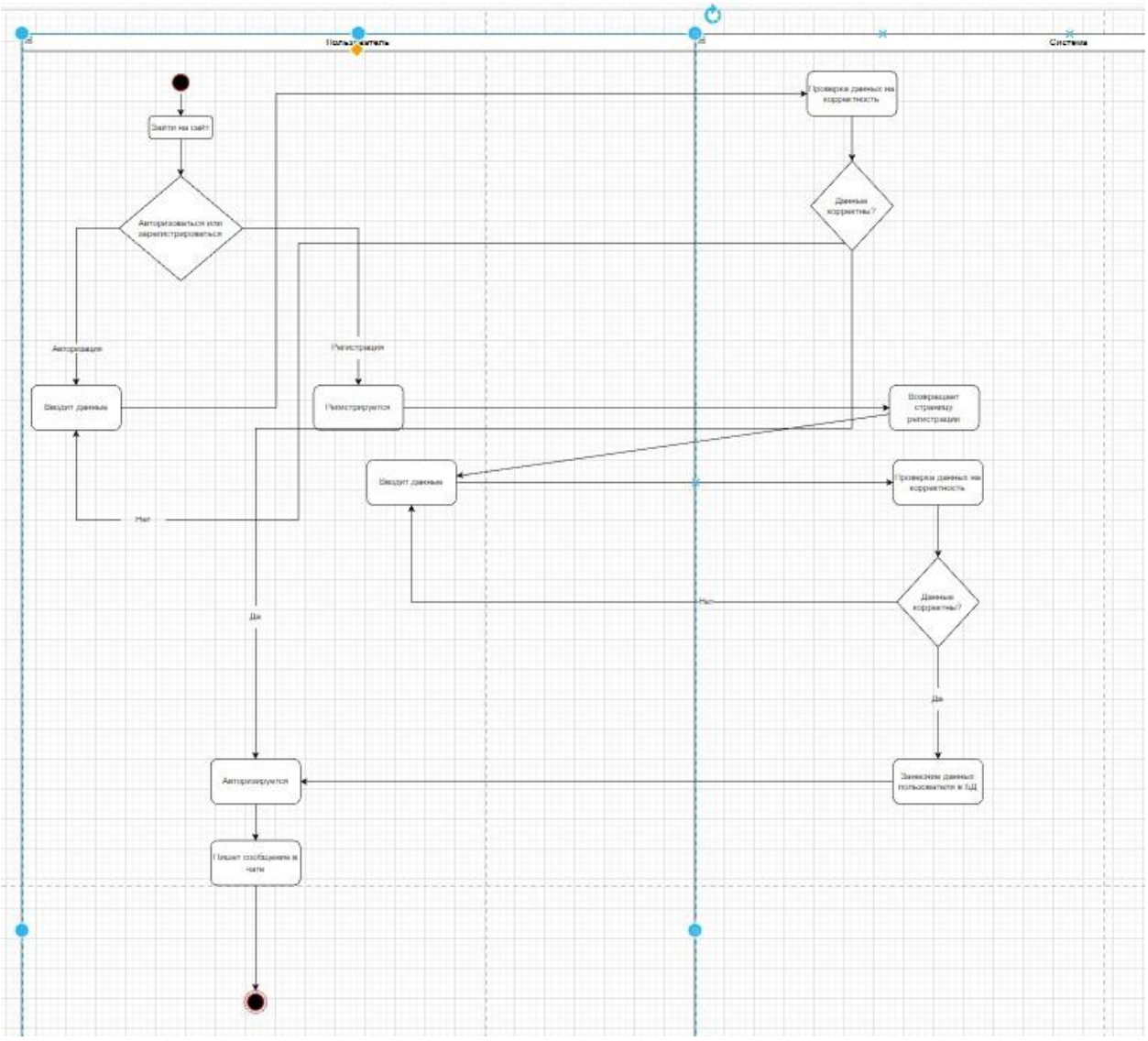

Рисунок 7 - Диаграмма активности (деятельности)

2.3.4 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».

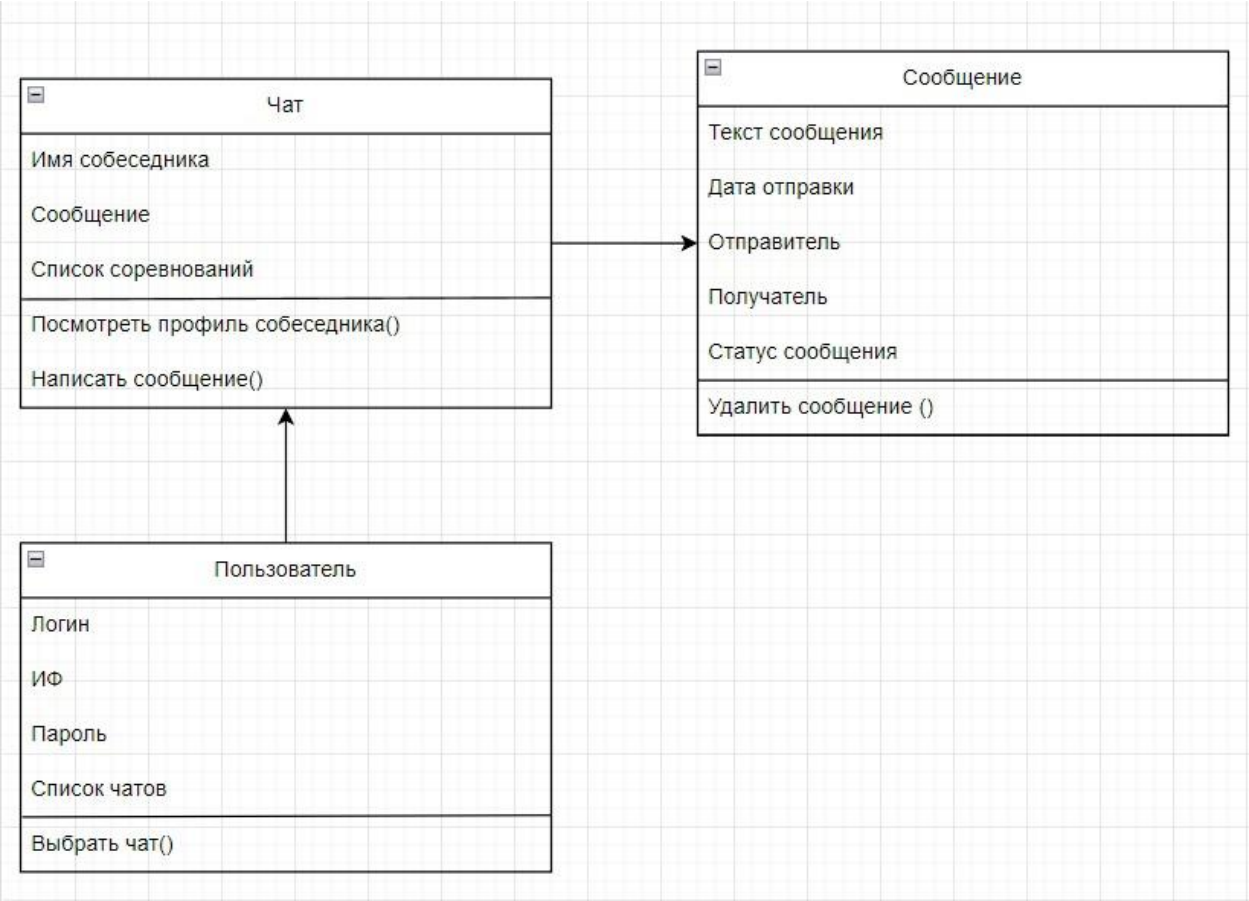

Рисунок 8 - Диаграмма классов

2.3.5 Диаграмма объектов (Object diagram)

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

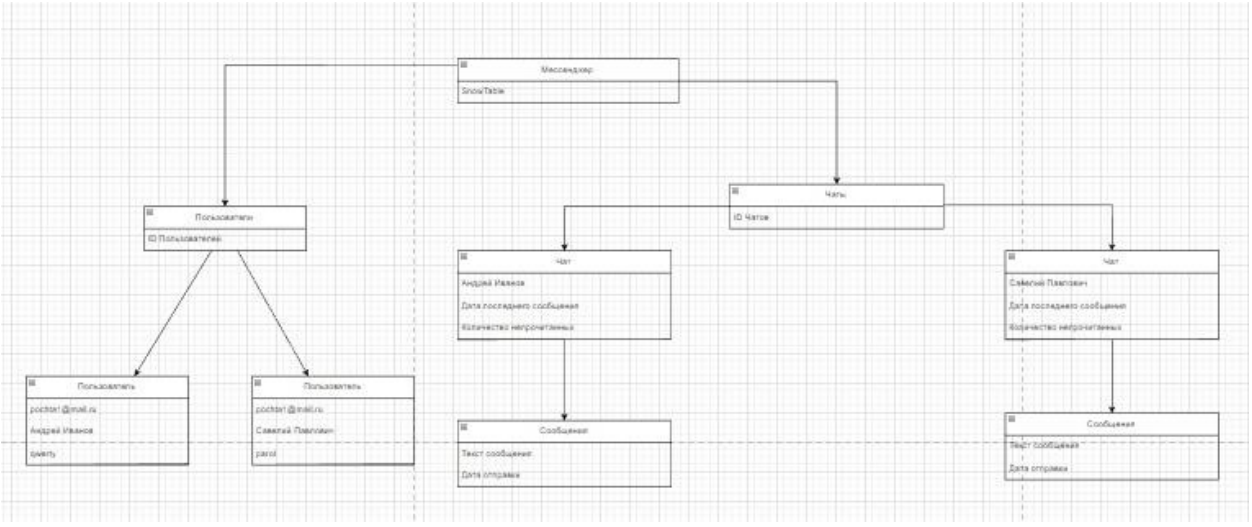

Рисунок 9 - Диаграмма объектов

2.3.6 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.

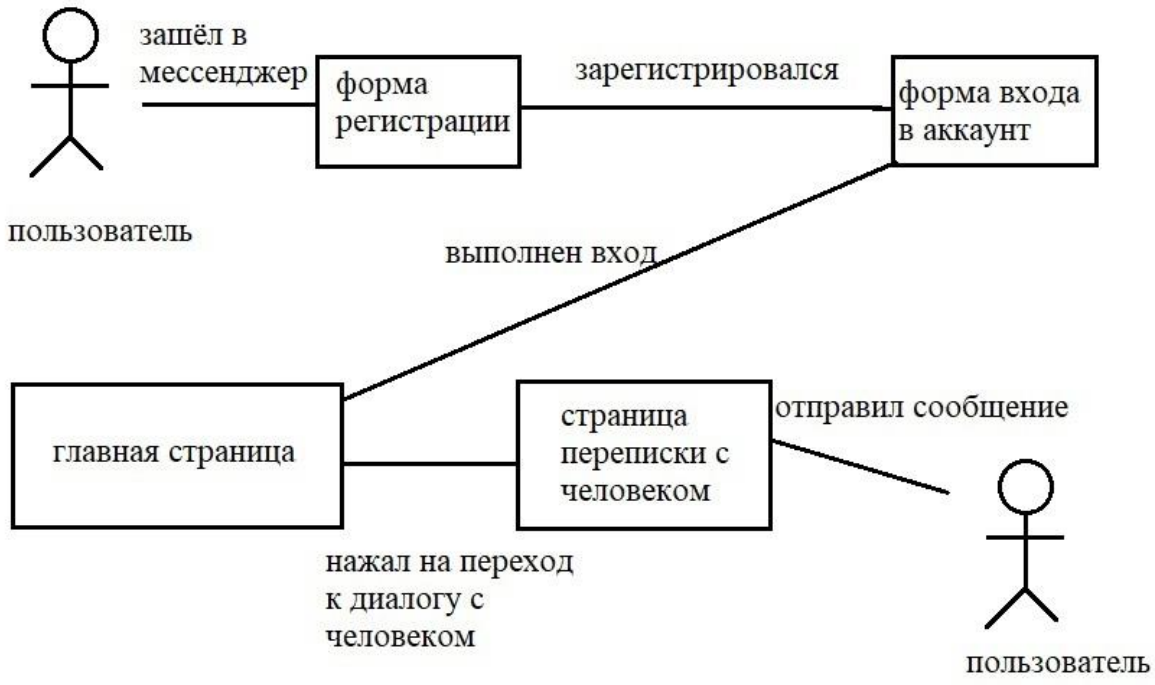

Рисунок 10 - Диаграмма сотрудничества.

2.3.7 Диаграмма развертывания (Deployment diagram)

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

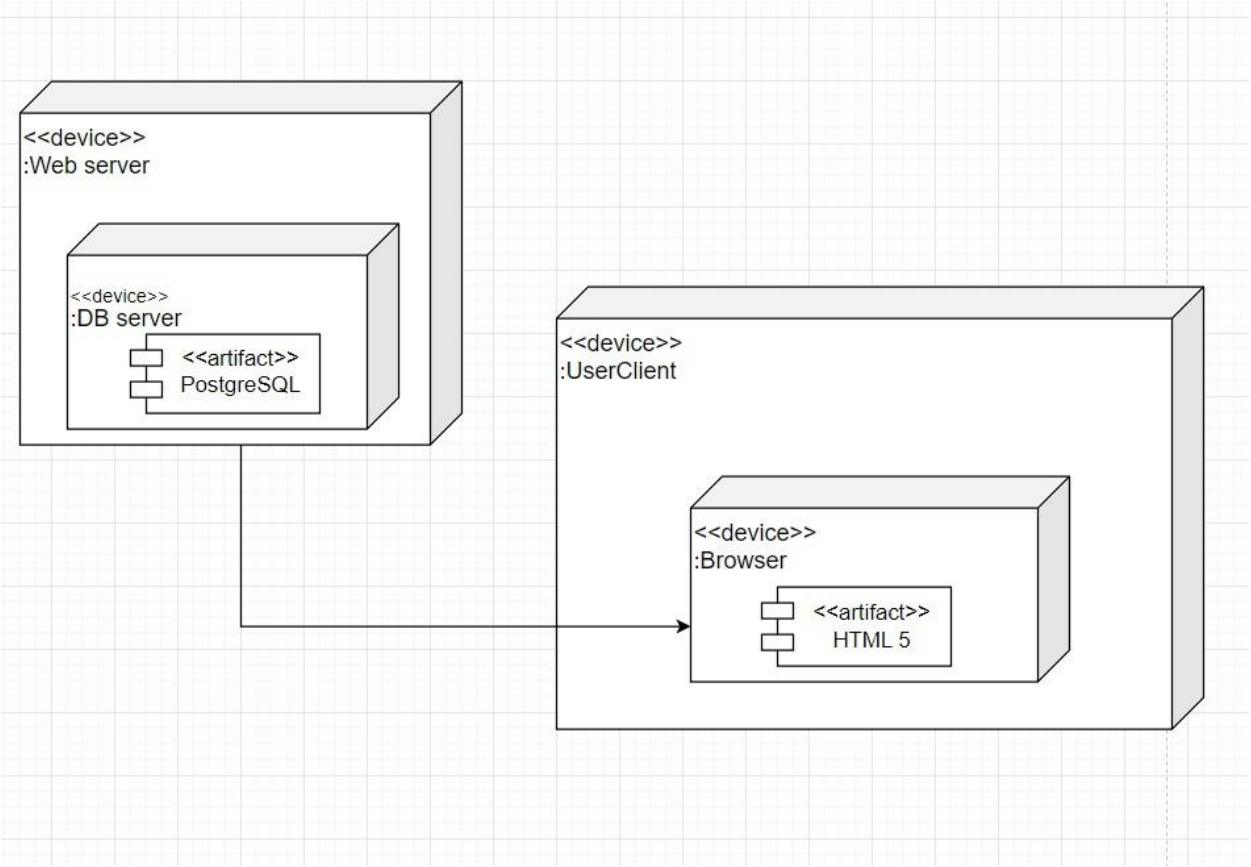

Рисунок 11 - Диаграмма развертывания.

Заключение

Список использованной литературы