МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«Южно-Уральский государственный университет (национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

РАБОТА ПРОВЕРЕНА

Рецензент
к.п.н., доцент кафедры ЭВМ
ЮУрГУ
_____ Ю.Г. Плаксина
«___»_____ 2020 г.

ДОПУСТИТЬ К ЗАЩИТЕ

Заведующий кафедрой, д.ф.-м.н., профессор
_____ Л.Б. Соколинский
«___»_____ 2020 г.

Разработка веб-приложения «Awesome Chat»

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 02.03.02.2020.308-546.ВКР

Научный руководитель,
ст. преподаватель кафедры СП
_____ Н.С. Силкина

Автор работы,
студент группы КЭ-402
_____ Х.М. Фам

Ученый секретарь
(нормоконтролер)
_____ И.Д. Володченко
«___»_____ 2020 г.

Челябинск-2020

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«Южно-Уральский государственный университет (национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

УТВЕРЖДАЮ
Зав. кафедрой СП
_____ Л.Б. Соколинский
09.02.2020

ЗАДАНИЕ

на выполнение выпускной квалификационной работы бакалавра

студенту группы КЭ-402
Фам Хонг Ман
обучающемуся по направлению
02.03.02 «Фундаментальная информатика и информационные технологии»

1. **Тема работы** (утверждена приказом ректора от 24.04.2020 № 627)
Разработка веб-приложения «Awesome Chat».
2. **Срок сдачи студентом законченной работы:** 02.06.2020.
3. **Исходные данные к работе**
3.1. Leonard R. & Mike A. RESTful Web APIs - O'REILLY, 2013.
3.2. Marijin H. Eloquent Javascript - No Starch Press, 2018.
3.3. Tim M. Design and Implementation of a NoSQL-concept for an international and multicentral clinical database, 2016.
4. **Перечень подлежащих разработке вопросов**
4.1. Провести анализ предметной области.
4.2. Проектирование базы данных.
4.3. Проектирование приложения.
4.4. Реализация и тестирование приложения.
5. **Дата выдачи задания:** 08.02.2020.

Научный руководитель,
Ст. преподаватель кафедры СП Н.С. Силкина

Задание принял к исполнению Х.М. Фам

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность темы

Веб-чаты, голосовые и видео веб-чаты – это веб-коммуникации для сайтов. Веб-коммуникации обеспечивают общение пользователей через Интернет с использованием веб-браузера. В основном веб-коммуникации реализованы на сетевой архитектуре клиент-сервер. *Клиент* – это прикладная программа, которая посылает запросы серверу и получает от сервера требуемую информацию. *Сервер* – это программа, работающая на отдельном компьютере (хосте), которая предназначена для хранения, обработки и выдачи клиентам информации по запросам. Взаимодействие клиента и сервера осуществляется на основе определённого протокола [11].

Взаимодействие веб-браузера с веб-сервером осуществляется по прикладному протоколу – HTTP. Протокол HTTP – это сеансовый протокол, работающий по принципу «запрос-ответ». Клиентское веб-приложение посылает запрос на сервер, используя протокол версии HTTP 1.1, и получает от сервера обновленную веб-страницу. Чтобы получить от сервера веб-страницу целиком веб-браузер формирует серию запросов к веб-серверу: к DNS-серверу для преобразования символьного имени хоста в IP-адрес, для закачки HTML структуры страницы, CSS, JS, изображений и других объектов.

Для общения в чате можно использовать специфическое программное обеспечение, устанавливаемое на компьютер пользователя, так и веб-браузер. *Веб-чат* – это система диалогового текстового общения в реальном времени, построенная на базе веб-сервера [11]. В веб-чатах роль клиентской части веб-приложений выполняет браузер. Это объясняется тем, что в браузеры встроен интерпретатор языка программирования JavaScript, на базе которого создаются клиентские приложения на Ajax. Серверная часть веб чата размещается на веб-сервере. На практике применяются веб чаты двух типов: для персонального

общения (консультаций) и группового общения (обсуждения какой-либо темы). Под словом чат обычно понимается групповое общение, хотя к ним можно отнести и обмен текстом «один на один» посредством программ мгновенного обмена сообщениями, например, XMPP, ICQ или даже SMS.

Современный ритм жизни – это вечная спешка, из-за которой не остается времени на общение. А ведь общение – это выражение своих эмоций, чувств, отношения друг к другу, общие интересы и увлечения. С помощью веб-чата вы можете общаться с друзьями, родственниками или вашими любимыми с любой точки планеты, а также со случайными незнакомцами по всему миру. С веб-чатом можно весело провести свободное время, завести новые знакомства, получить грандиозные впечатления от общения с удивительными, необычными и, несомненно, харизматичными людьми и узнать много интересного.

Цель и задачи работы

Целью данной работы является разработка веб-чата.

Для достижения поставленной цели необходимо решить следующие задачи:

1) выполнить анализ предметной области и разработать требования;
2) выполнить анализ существующих решений и выбрать технологии необходимые для реализации приложения;
3) спроектировать структуру и логику приложения;
4) спроектировать базу данных;
5) реализовать базу данных;
6) реализовать приложение;
7) произвести тестирование приложения;

Структура и объем работы

Выпускная квалификационная работа состоит из введения, четырех разделов, заключения, библиографии, приложения. Объем работы составляет 38 страницы, объем библиографии – 21 наименований.

В первом разделе «Анализ предметной области» приведен обзор существующих аналогов и используемых технологий.

Второй раздел «Проектирование» описывает процесс проектирования приложения. Раздел содержит диаграмму вариантов использования, проектирование базы данных и макеты страниц сайта.

Третий раздел «Реализация» содержит описание реализации веб-чат приложения.

Четвертый раздел «Тестирование» включает описание процесса тестирования приложения.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. Анализ аналогичных проектов

В данном разделе описан анализ аналогичных проектов, в ходе которого необходимо выявить базовые требования к приложению, определить плюсы и минусы аналогичных проектов.

Chatrandom

Chatrandom является одним из ведущих сайтов в списке «10 топовых чат сайтов» по мнению организации TopChatSites [13], благодаря большому количеству пользователей и исключительными особенностям. Chatrandom имеет очень много дополнительных возможностей и услуг, что выделяет его среди всех остальных сайтов подобного типа. Высокоскоростное соединение, возможность увеличения видео экрана и простота в использовании – это всего несколько причин, почему на данном сайте более 20 000 онлайн пользователей в любое время суток. Главная страница представлена на рисунке 1.

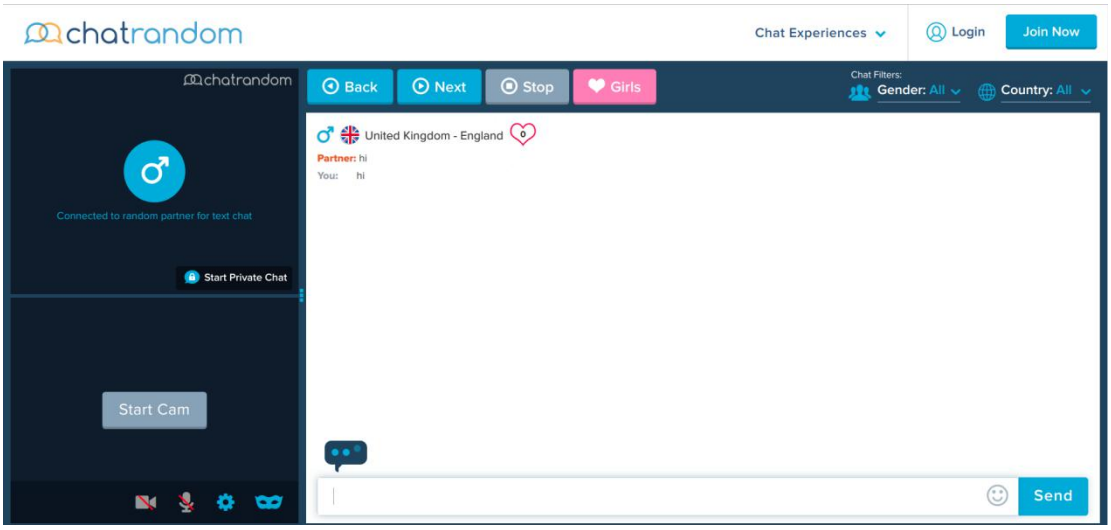

Рис. 1. Главная страница «Chatrandom»

Достоинства:

- Красивый интерфейс, простой в использовании.
- Случайные собеседники со всего мира, возможность встретить новых друзей.

Недостатки:

- Некоторые нужные функции платные.

Chatspin

Chatspin являлся одним из первых альтернативных чатов рулеток. На сайте множество пользователей, благодаря всем особенностям, которые делают сайт интересным и легким в использовании. Сайт хорошо продуман и имеет простой в использовании интерфейс. Он быстро загружается и имеет тысячи онлайн пользователей, поэтому найти с кем пообщаться на сайте будет несложно. Сайт переведен на английский, испанский, немецкий, французский, итальянский и португальский языки, но большинство людей, которые пользуются Chatspin говорят на английском [14], представлен на рисунке 2.

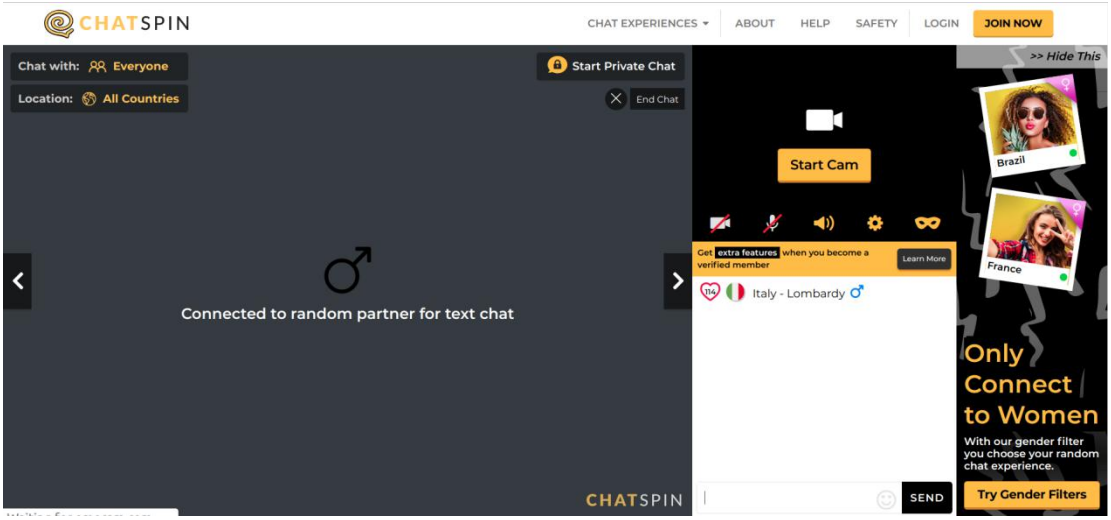

Рис. 2. Главная страница «Chatspin»

Достоинства:

- Chatspin обладает большим количеством онлайн пользователей.
- Chatspin предлагает намного больше дополнительных возможностей в своем видео приложении.
- Можно пользоваться без регистрации.

Недостатки:

- Некоторые нужные функции платные.
- Наличие рекламы.

E-Chat

E-Chat [16] – это популярный бесплатный чат, который позволяет пользователям со всего мира анонимно общаться со случайными людьми. На веб-сайте доступны различные готовые чаты, и пользователи могут присоединиться к ним, создав учетную запись или в качестве гостя.

Вы можете создать учетную запись, если хотите, и войти, если у вас уже есть учетная запись. Если вы не хотите создавать учетную запись, вы можете сразу же присоединиться к чату в качестве гостя и начать общаться. Войти в систему как гость очень легко, и вы можете войти, просто выбрав имя пользователя и выполнив проверку с помощью сервиса cupcha.

После входа в систему вы можете выбрать чат-комнату, к которой хотите присоединиться, из разных комнат. Вы можете иметь личный чат с любым пользователем, которого вы выберите, а также добавить их в друзья при желании.

E-Chat также позволяет пользователям создавать свои собственные частные чаты и приглашать других людей в чат-комнату, используя общую ссылку. Как только люди начинают присоединяться к чату, вы становитесь модератором чата и можете управлять различными аспектами чата, представлен на рисунке 3.

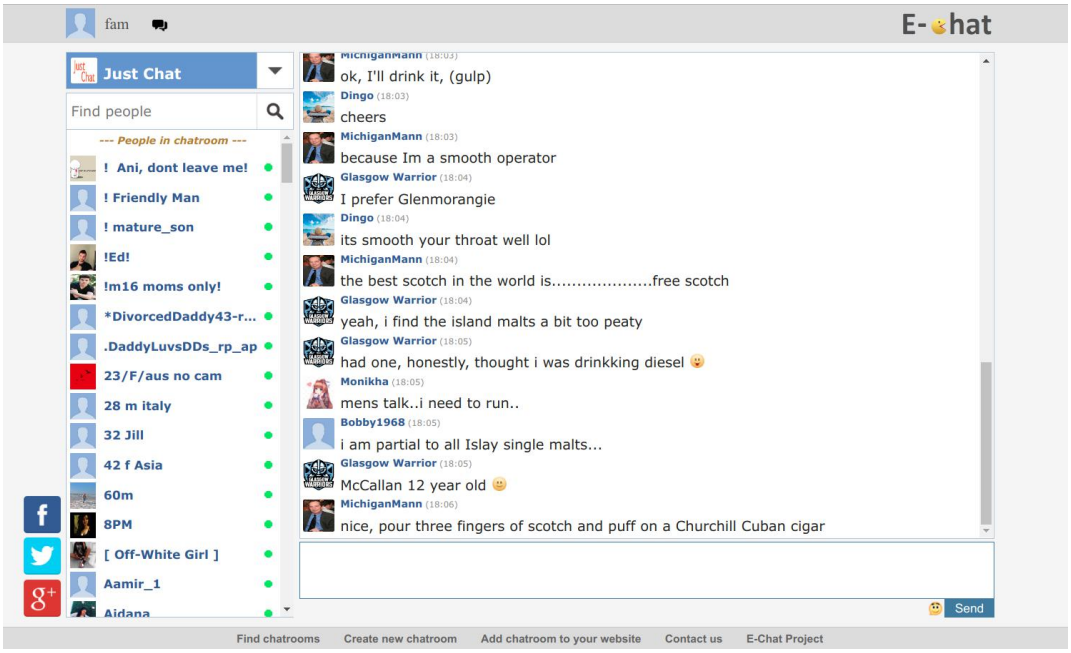

Рис. 3. Главная страница «E-Chat»

Достоинства.

- Возможность знакомства с новыми людьми, которые разделяют ваши интересы.
- Возможность создать свой собственный чат с полным контролем над ним.
- Возможность добавить окно чата на другой веб-сайт.
- Все чаты работают очень быстро и не требуют плагинов.
- Это было, есть и всегда будет бесплатным.

Недостатки.

- Отсутствие интерфейса для мобильного телефона.

Исходя из анализа, можно выделить основные функции поиск друзей, добавь друзей, отправлять текстовые сообщения со смайликами и с личными изображениями в режиме реального времени, приложение должно иметь удобный интерфейс, простой в ознакомлении и использовании. Подробные требования к приложению определены в следующей главе.

1.2. Технологии и инструменты

В данном разделе приведен анализ технологий, которые могут быть использованы для реализации веб-чата. Технологии выбирались на основе их функциональных возможностей и актуальности их использования в настоящее время.

Для реализации были выбраны следующие технологии:

Node.js – это среда выполнения JavaScript, основанная на движке Chrome V8 JavaScript Node.js использует управляемую событиями асинхронную неблокирующую модель ввода-вывода Node.js работает в цикле событий одного потока [2, 6].

Express.js – это фреймворк веб-приложений для Node.js. Он спроектирован для создания веб-приложений и API. Де-факто является стандартным каркасом для Node.js [8].

Socket.IO – JavaScript-библиотека для веб-приложений и обмена данными в реальном времени. Состоит из двух частей: клиентской, которая запускается в браузере и серверной для node.js. Оба компонента имеют похожее API. Подобно Node.js, Socket.IO событийно-ориентированная [9].

MongoDB – это система управления базами данных, которая значительно отличается от MySQL. Основная разница заключается в том, что в MySQL запросы пишутся на языке SQL, а в MongoDB на BSON (бинарный JSON). Это значит, что работа с этой системой может осуществляться в основном через JavaScript выражения [1, 10].

ReactJS – это JavaScript-библиотека для создания пользовательских интерфейсов. Библиотека «на основе компонентов» означает, что вы можете создавать строительные блоки и составлять интерфейс из этих повторно используемых компонентов. Create-React-App, с другой стороны, является средой React с нулевой конфигурацией. Он работает из коробки с одной командой оболочки и поддерживает вашу среду в актуальном состоянии [7].

AntdDesign – это комплексная система проектирования, включающая полный набор компонентов React. Поскольку React основан на компонентах, довольно просто использовать компоненты React от Ant Design в качестве строительных блоков, чтобы быстро собрать прототип [17].

Таким образом, была осуществлена постановка задачи, произведен анализ аналогичных программных продуктов, а также произведен обзор средств реализации веб-чат.

2. ПРОЕКТИРОВАНИЕ

2.1. Анализ требований

В ходе проектирования веб-приложения были определены функциональные возможности, предоставляемые пользователю.

- Веб-приложение должно предоставлять возможность идентификации пользователей (регистрации и авторизации пользователей).
- Веб-приложение должно предоставлять возможность подтверждения адреса электронной почты для пользователей.
- Веб-приложение должно предоставлять возможность обновить информацию о пользователе (аватар, ФИО, email, и т.д.).
- Веб-приложение должно предоставлять возможность поиска и добавления друзей.
- Веб-приложение должно предоставлять возможность управления списком друзей, добавления, удаления, подсчета количества друзей, отправки запросов на добавление в друзья, отмены запросов, подтверждения друзей и т.д.
- Веб-приложение должно автоматически отправлять уведомления в режиме реального времени, когда кто-то отправляет приглашение другу и когда кто-то отправляет сообщение.
- Веб-приложение должно предоставлять возможность отправлять текстовые сообщения со смайликами, с личными изображениями.
- Веб-приложение должно предоставлять возможность пользователям создавать голосовой чат, видео чат.
- Веб-приложение должно предоставлять возможность отображает статус пользователя: онлайн, офлайн.

Помимо функциональных требований можно выделить следующие нефункциональные требования.

- Веб-приложение должно иметь интуитивно понятный интерфейс.
- Веб-приложение должно хранить данные пользователя на удаленном сервере.

• Веб-приложение должно корректно работать во всех популярных браузерах.

2.2. Варианты использования

Исходя из поставленной задачи, можно построить диаграмму вариантов использования и определить требования для разрабатываемого приложения.

Диаграмма вариантов использования – диаграмма UML, отражающая отношения между актёрами и прецедентами и являющаяся составной частью модели прецедентов, позволяющей описать систему на концептуальном уровне.

Вариант использования – возможность моделируемой системы (часть её функциональности), благодаря которой пользователь может получить конкретный, измеримый и нужный ему результат. Вариант использования соответствует отдельному сервису системы, определяет один из вариантов её использования и описывает типичный способ взаимодействия пользователя с системой. Варианты использования обычно применяются для спецификации внешних требований к системе [5].

Определим основных актеров, взаимодействующих с системой.

• Гость – неавторизированный посетитель сайта.

• Пользователь – пользователь, прошедший авторизацию.

Пользователю должен быть доступен весь функционал сайта, гость может пройти регистрацию и выполнить вход.

На рисунке 4 представлена диаграмма вариантов использования разрабатываемого приложения. Рассмотрим возможные варианты использования для обозначенных актеров.

Гость может.

• Выполнить вход – авторизоваться в системе используя свою электронную почту и пароль.

• Зарегистрироваться – создать аккаунт в системе.

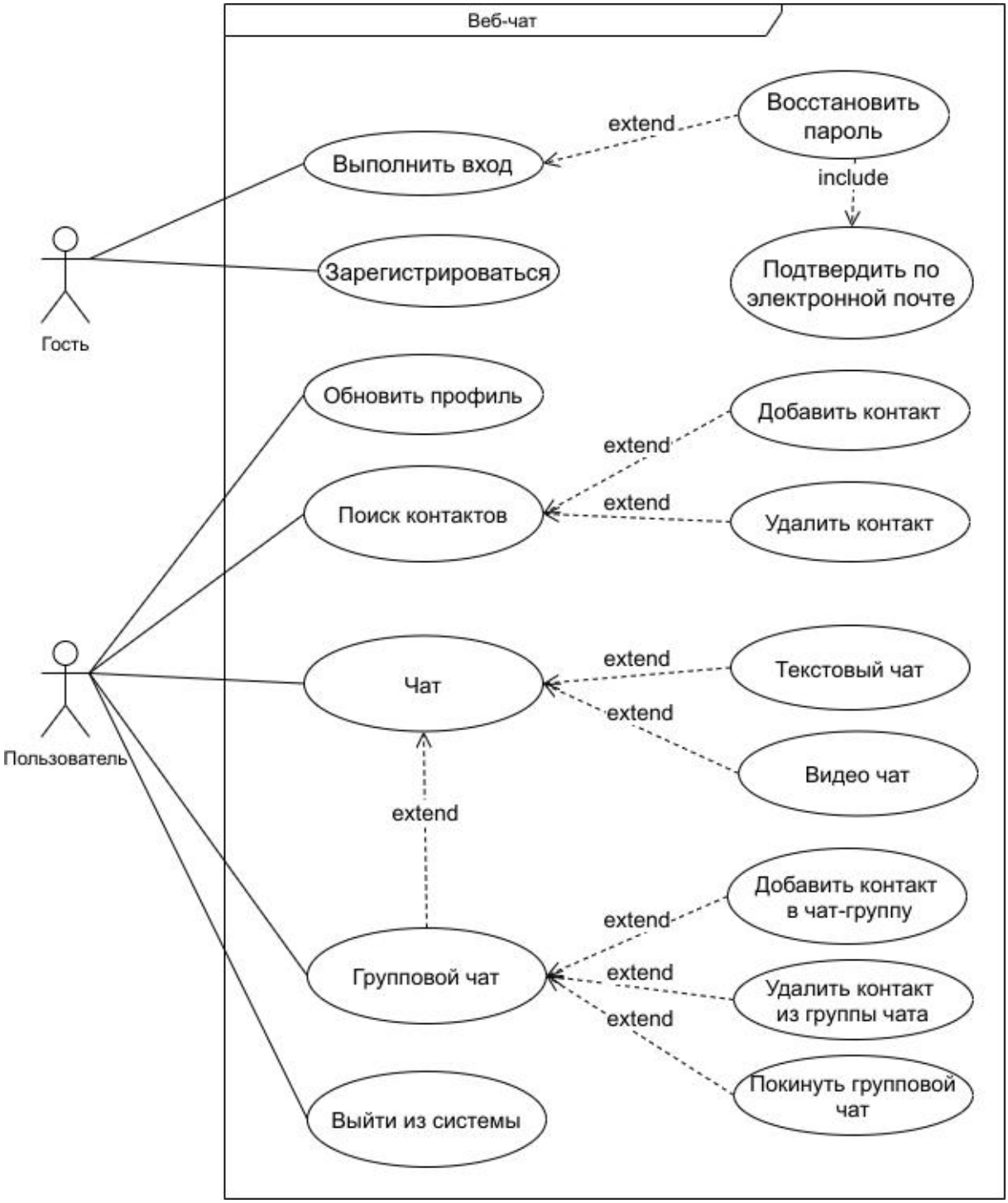

Рис. 4. Диаграмма вариантов использования

Пользователь может.

• Обновить профиль – пользователь может вносить изменения в свой профиль.

• Поиск контактов – пользователь может искать других пользователей.

• Чат – пользователь может общаться в чате с другими пользователями.

• Групповой чат – пользователь может сделать групповой чат с другими пользователями.

• Выйти из системы – пользователь может выйти из системы.

2.3. Проектирование базы данных

База данных является важнейшим компонентом системы, в ней хранится вся информация, которая необходима пользователю, на рисунке 5 представлена схема базы данных.

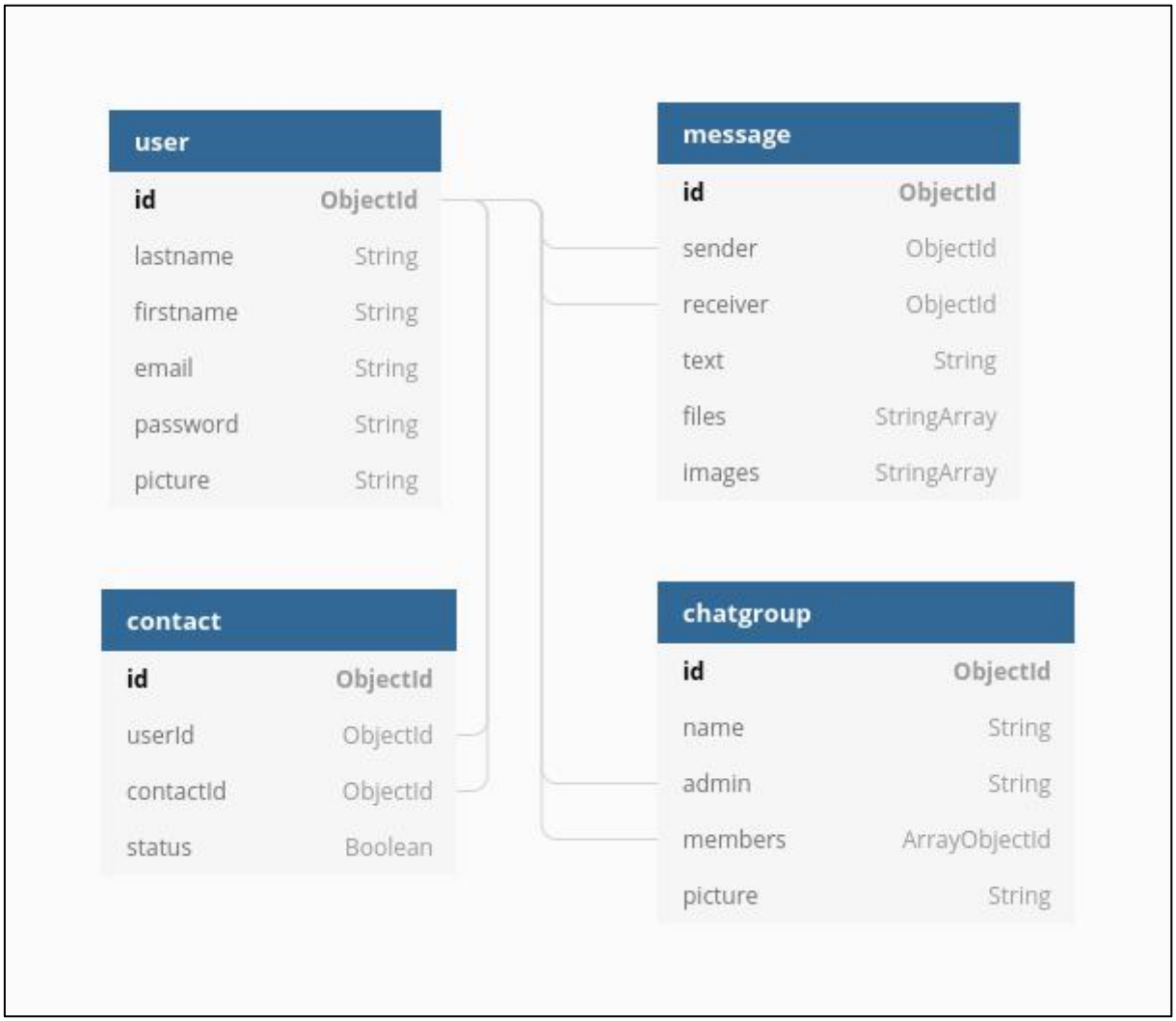

Рис. 5. Схема базы данных

В коллекции «user» хранится информация о пользователях. Коллекция состоит из следующих полей:

- id – тип ObjectId, идентификатор пользователя;
- lastname – тип String, имя пользователя;
- firstname – тип String, фамилия пользователя;
- email – тип String, электронная почта пользователя;
- password – тип String, зашифрованный пароль;
- picture – тип String, путь к файлу.

В коллекции «contact» хранится информация о контакте. Коллекция состоит из следующих полей:

- id – тип ObjectId, идентификатор контакта;
- userId – тип ObjectId, идентификатор пользователя, который отправил запрос о добавлении в контакты;
- contactId – тип ObjectId, идентификатор пользователя, который получил запрос о добавлении в контакты;
- status – тип Boolean, принят или не принят запрос.

В коллекции «message» хранится информация о сообщениях. Коллекция состоит из следующих полей:

- id – тип ObjectId, идентификатор контакта;
- senderId – тип ObjectId, идентификатор отправителя сообщения;
- receiverId – тип ObjectId, идентификатор получателя сообщения;
- text – тип String, текст сообщения;
- files – тип StringArray, массив содержит путь к файлам;
- images – тип StringArray, массив содержит путь к фотографиям.

В коллекции «chatgroup» хранится информация о чат-группе. Коллекция состоит из следующих полей:

- id – тип ObjectId, идентификатор контакта;
- name – тип String, имя группы;
- admin – тип ObjectId, идентификатор администратор;
- members – тип Array, массив идентификаторов членов группы;

- picture – тип String, путь к файлу.

2.4. Проектирование интерфейса

Интерфейс – это граница между взаимодействующими независящими друг от друга сущностями, которая определяется стандартами, устанавливает характеристики, операции и методы взаимодействия объектов.

Интерфейс пользователя – элементы и компоненты программы, умеющие проявлять воздействие на работу пользователя с программным обеспечением. А также следующие.

- Ресурсы отражения информации, выводящиеся сведения.
- Командные режимы, язык пользователь-интерфейс.
- Инструменты и способы занесения информации.
- Диалог, связь и транзакции пользователя и компьютера.
- Обратная взаимосвязь с пользователем.

На рисунке 6 представлен макет главной страницы сайта.

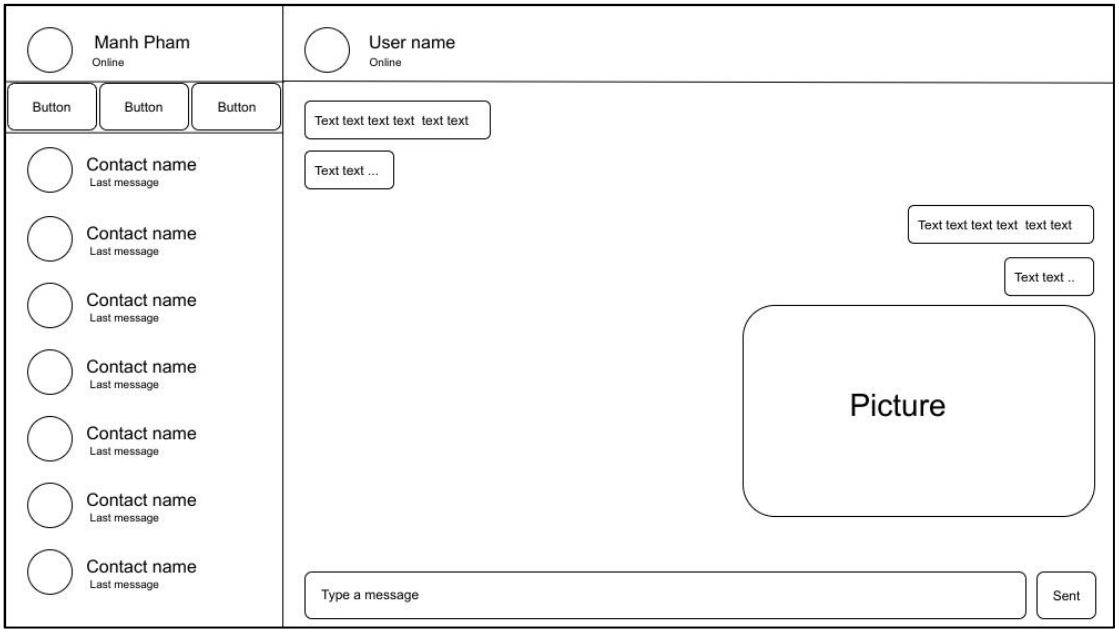

Рис. 6. Макет главной страницы

Пользователь сайта должен зарегистрироваться и авторизоваться на сайте, что представлено на рисунке 7.

Рис. 7. Страница регистрация

Таким образом, были определены требования к веб-приложению, на их основе была построена диаграмма вариантов использования. Кроме этого, были спроектированы база данных и интерфейсы приложения.

2.5. JSON Web Token

JSON Web Token (JWT) – это JSON объект, который определен в открытом стандарте RFC 7519. Он считается одним из безопасных способов передачи информации между двумя участниками. Формат данных возможен в двух формах сериализованный и десериализованный.

Сериализованный формат используется для передачи токена по сети, пример сериализованный токен показан на листинге 1.

Листинг 1. Сериализованный токен

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJleHAiOiIzNjAwMDAiLCJzdWIiOiI1ZTNiZD
I2OWUyMTI1MDQ0NWI2Y2QxMWUifQ.yinULzWUocxBysJimAXAeL6_WT53LMnFDUhKFKwnveg
```

Структура состоит из трех частей:

- Header – заголовок (тип токена, алгоритм подписи);
- Payload – полезная нагрузка (какие либо данные, например id, email, имя…);

• Signature – подпись (строка, используется для проверки того, что сообщение не было изменено).

Пример десериализованный токен показан на листинге 2.

Листинг 2. Десериализованный токен

```
{ exp: 1589630342, userId: '5e3bd269e21250445b6cd11e' }
```

В приложении Awesome-chat токен является идентификатором сессии пользователя. При его помощи мы можем верифицировать пользователя и гарантировать, что мы отдадим пользователю именно его данные. При этом токен имеет зашифрованную подпись, что дает некоторую защиту от его подмены на стороне клиента.

На рисунке 8. представлен процесс общения клиента и сервера.

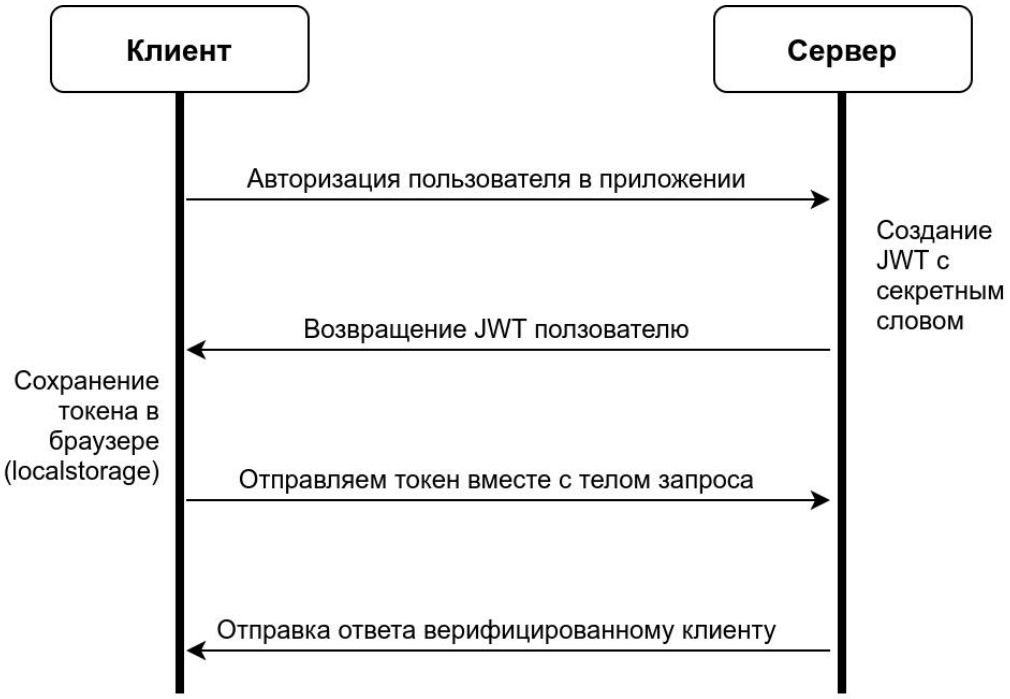

Рис. 8. Процесс общения клиента и сервера

Приложение использует JWT для проверки аутентификации пользователя следующим образом:

1) сперва пользователь заходит на сервер аутентификации с помощью аутентификационного ключа (с использованием логин и пароль);

2) затем сервер аутентификации создает jwt и отправляет его пользователю;

3) когда пользователь делает запрос к api приложения, он добавляет к нему полученный ранее jwt;

4) когда пользователь делает api запрос, приложение может проверить по переданному с запросом jwt является ли пользователь тем, за кого себя выдает.

Для реализации стандарта аутентификации используется расширение jsonwebtoken [19], которое реализует весь необходимый функционал. Информация об аутентификации пользователя хранится в локальном хранилище (localStorage).

2.6. Архитектура приложения

Архитектура пользовательского интерфейса представлена на рисунке 9.

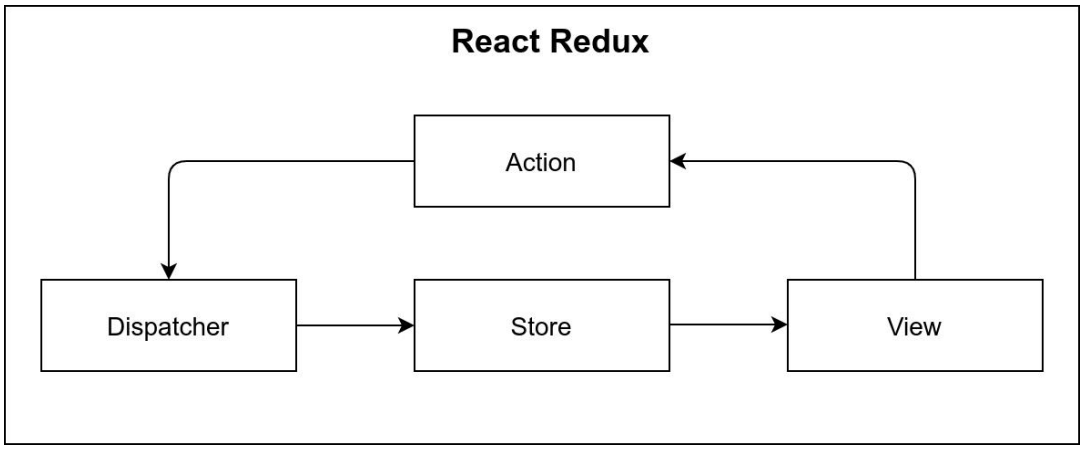

Рис. 9. Архитектура пользовательского интерфейса

View – Это компоненты, которые непосредственно создаются в React и которые в перспективе будут отражать на экране монитора. В браузере эти компоненты получают данные из Store.

Store хранит состояние приложения. Store является единственным и общим хранилищем для всех наших компонентов.

Dispatcher это центральный узел, который обрабатывает все наши потоки данных. То есть – всё проходит через него. А конкретно, через него проходит те данные, которые определяют как раз-таки action.

Архитектура сервера представлена на рисунке 10.

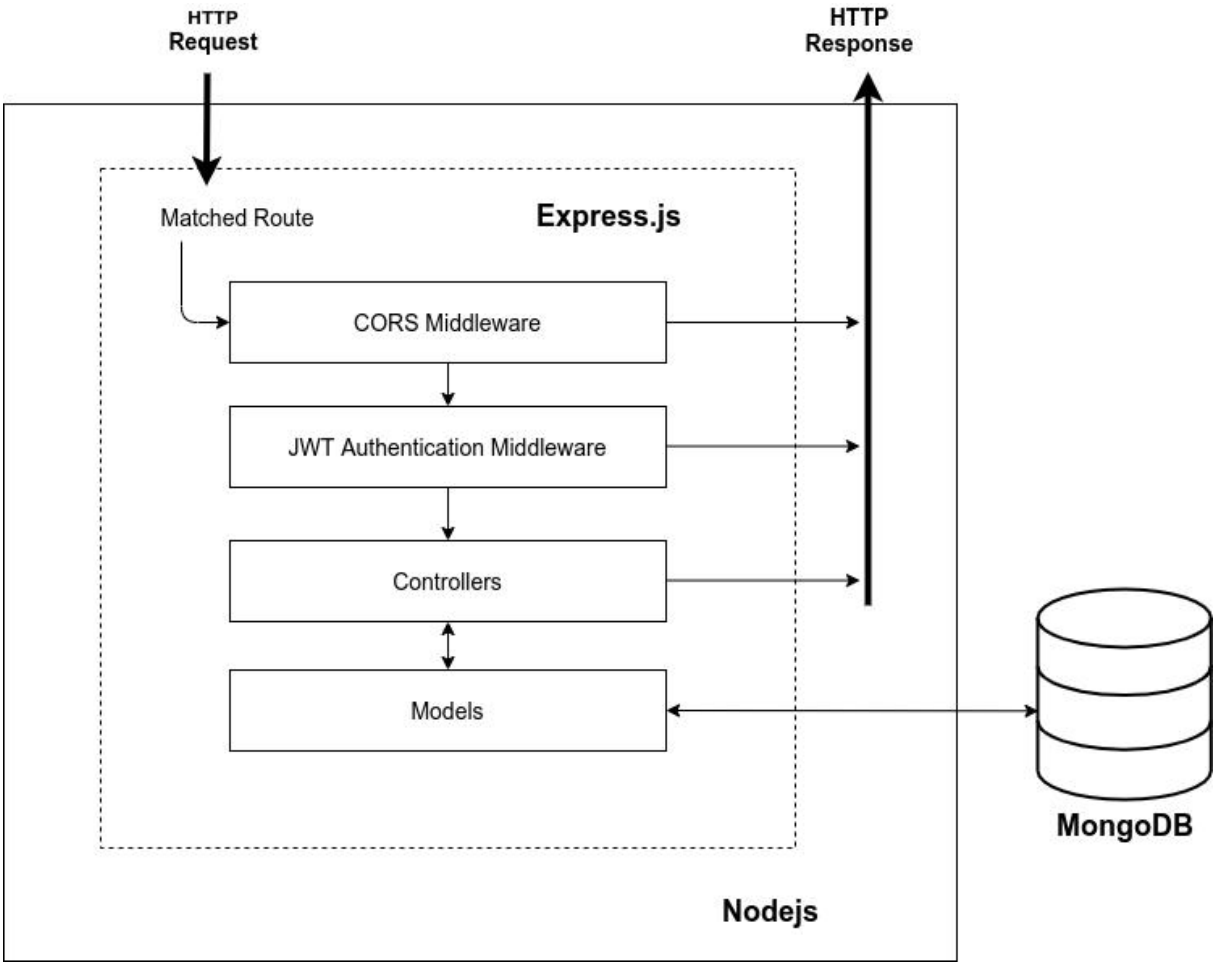

Рис. 10. Архитектура сервера

Через экспресс-маршруты HTTP-запрос, соответствующий маршруту, будет проверен CORS Middleware перед переходом на аутентификации JWT. Cross-Origin Resource Sharing (CORS) – механизм, использующий дополнительные HTTP-заголовки, чтобы дать возможность агенту пользователя получать разрешения на доступ к выбранным ресурсам с сервера на источнике (домене), отличном от того, что сайт использует в данный момент. Говорят, что агент пользователя делает запрос с другого источника (cross-origin HTTP request), если источник текущего документа отличается от запрашиваемого ресурса доменом, протоколом или портом.

Аутентификации JWT верифицировать пользователя. Если эти промежуточные программы выдают какую-либо ошибку, сообщение будет отправлено как HTTP-ответ.

Контроллеры – блок, который получает данные от пользователя, обрабатывает, нормализует их, также выполняет проверку правильности ввода и передает эти обработанные данные в нужную модель. Также он принимает данные от модели и отправляет HTTP-ответ.

Модели – блок, который получает данные от контроллера, далее на основе этих данных производит необходимые преобразования, а затем передает результат своей работы назад контроллеру.

3. РЕАЛИЗАЦИЯ

3.1. Реализация REST API

Для разработки REST API используется язык программирования Nodejs версии 12.13.1 и фреймворк для создания веб-приложений Expressjs. Express предоставляет гибкую настройку веб-приложения с минимальным количеством зависимостей.

Пакетный менеджер npm [5] предоставляет множество расширений для языка Nodejs, благодаря чему реализация взаимодействия между компонентами системы сводится к написанию бизнес логики.

REST (Representational State Transfer) – архитектурный стиль взаимодействия компонентов распределённого приложения в сети. REST представляет собой согласованный набор ограничений, учитываемых при проектировании распределённой гипермедиа-системы [18].

Основная идеи REST API заключается в разделении разных операций (чаще всего CRUD) при обращении к одному и тому же URL с помощью HTTP методов, основные из которых:

- GET – используется для получения данных;
- POST – используется для создания новой записи(ей);
- PUT – используется для обновления уже существующей записи(ей);
- PATCH – используется для обновления, но только тогда, когда изменяется идентификатор записи(ей);
- DELETE - используется для удаления записи(ей).

Все ответы используют формат данных JSON и содержат код состояния в соответствии со стандартом RFC 7231 [21].

Описание API

API для работы со средствами аутентификации представлен в таблице 1.

Табл. 1. API для работы со средствами аутентификации

Метод	Роутер	Описание
POST	/auth/register	Регистрация пользователей
POST	/auth/login	Авторизоваться
POST	/auth/send-password-reset	Запросить сброс пароля
POST	/auth/reset-password	Сброс пароля

API для работы с пользователями представлен в таблице 2.

Табл. 2. API для работы с пользователями

Метод	Роутер	Описание
GET	/user	Получить информацию о пользователе
POST	/user/avatar	Загрузить аватар пользователя
PATCH	/user	Обновить пользователя
PATCH	/user/change-password	Обновить пароль пользователя

API для работы с чат-группами представлен в таблице 3.

Табл. 3. API для работы с чат-группами

Метод	Роутер	Описание
GET	/chat-group/<id>	Получить информацию о чат-группе по идентификатору
POST	/chat-group	Создать чат-группу
PATCH	/chat-group	Обновить чат-группу
DELETE	/chat-group/<id>	Удалить чат-группу по идентификатору

API для работы с контактами представлен в таблице 4.

Табл. 4. API для работы с контактами

Метод	Роутер	Описание
GET	/contact/<id>	Получить контакт по идентификатору
POST	/contact	Создать контакт
PATCH	/contact	Обновить контакт
DELETE	/contact/<id>	Удалить контакт по идентификатору

API для работы с сообщением представлен в таблице 5.

Табл. 5. API для работы с сообщением

Метод	Роутер	Описание
GET	/message/<id>	Получить сообщение по идентификатору контакта
POST	/message/images	Загрузить изображения
POST	/message/files	Загрузить файлы

3.2. Подключиться к mongodb

Для работы с базой данных используется рекомендованный способ для работы с MongoDB из среды Nodejs пакет Mongoose. При помощи mongoose.connection можно получить стандартный объект Connection. После подключения в экземпляре Connection возникает событие open, представлен на листинге 3.

Доступ к конфигурации приложения происходит посредством обращения к переменным среду nodejs. Благодаря тому, что данная база является документно-ориентированной с представлением объектов в типе BSON отсутствует миграция базы данных.

Листинг 3. Программный код модуля mongoose.js

```
const mongoose = require('mongoose');
const logger = require('./../config/logger');
const { mongo, env } = require('./vars');
mongoose.Promise = Promise;
mongoose.connection.on('error', (err) => {
  logger.error(`MongoDB connection error: ${err}`);
  process.exit(-1);
});

exports.connect = () => {
  mongoose
    .connect(mongo.uri, {
      useCreateIndex: true,
      keepAlive: 1,
      useNewUrlParser: true,
      useUnifiedTopology: true,
      useFindAndModify: false,
    })
    .then(() => console.log('mongoDB connected...'));
  return mongoose.connection;
};
```

3.3. Реализация пользовательского интерфейса

Для реализации веб-клиента используется веб-фреймворк React. React предоставляет разработчику два основных механизма для передачи данных компонентам. Это – свойства (props) и состояние (state). Свойства предназначены только для чтения и позволяют родительским компонентам передавать дочерним компонентам атрибуты. Состояние – это локальная сущность, инкапсулированная внутри компонента, которая может в любое время, в жизненном цикле компонента, измениться. Так как состояние – это крайне полезный механизм, используемый для создания мощных динамических React-приложений, возникает необходимость в правильном управлении им. Redux – это библиотека, предназначенная для создания контейнеров, используемых для хранения состояния приложения. Она предлагает разработчику понятные инструменты для управления

состоянием, которые ведут себя предсказуемо. Redux решает проблему управления состоянием в приложении, предлагая хранить данные в глобальном State, и централизованно изменяя его.

На рисунке 11 представлена страница авторизации.

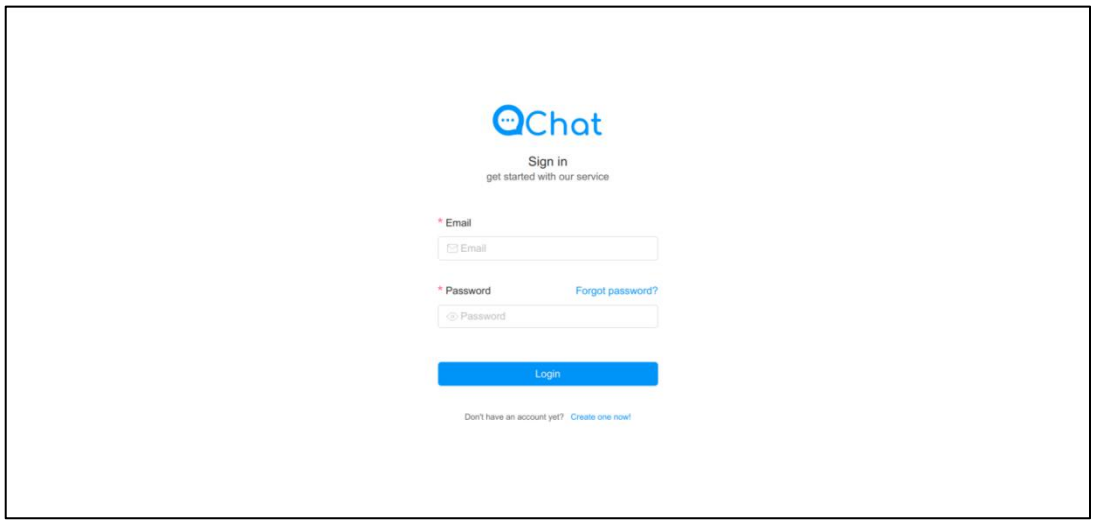

Рис. 11. Страница авторизации

На рисунке 12 представлена страница регистрации.

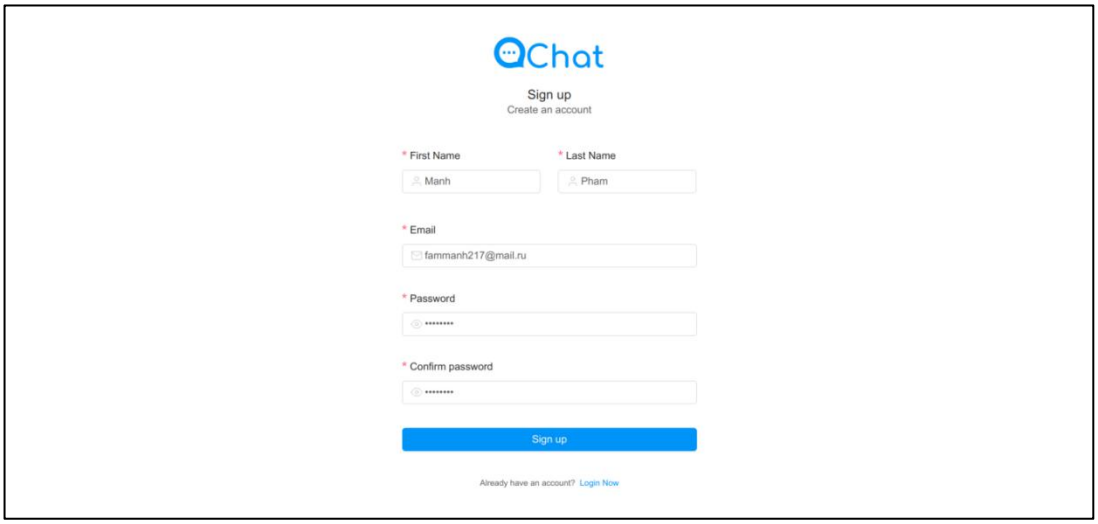

Рис. 12. Страница регистрации

На рисунке 13 представлена главная страница.

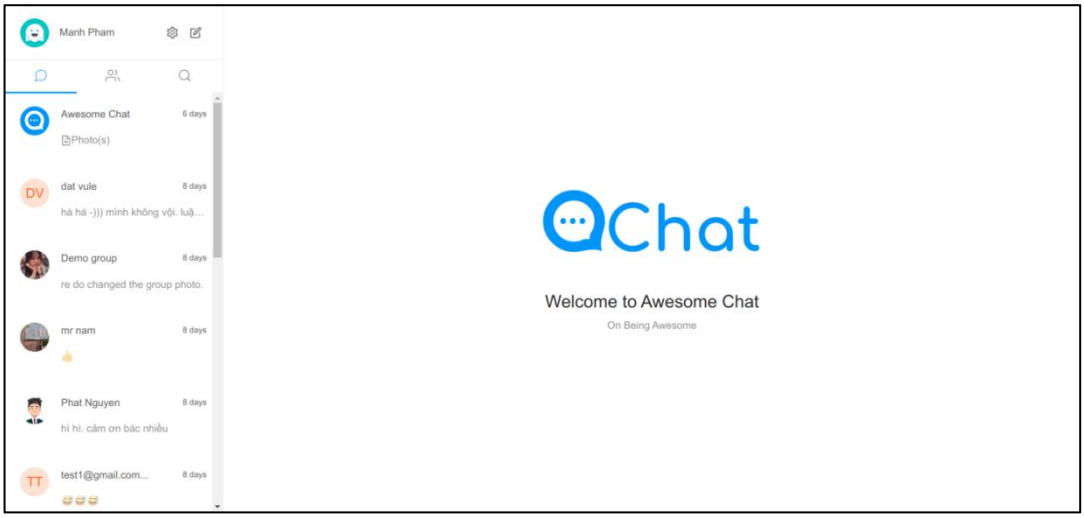

Рис. 13. Главная страница

На рисунке 14 представлена чат страница.

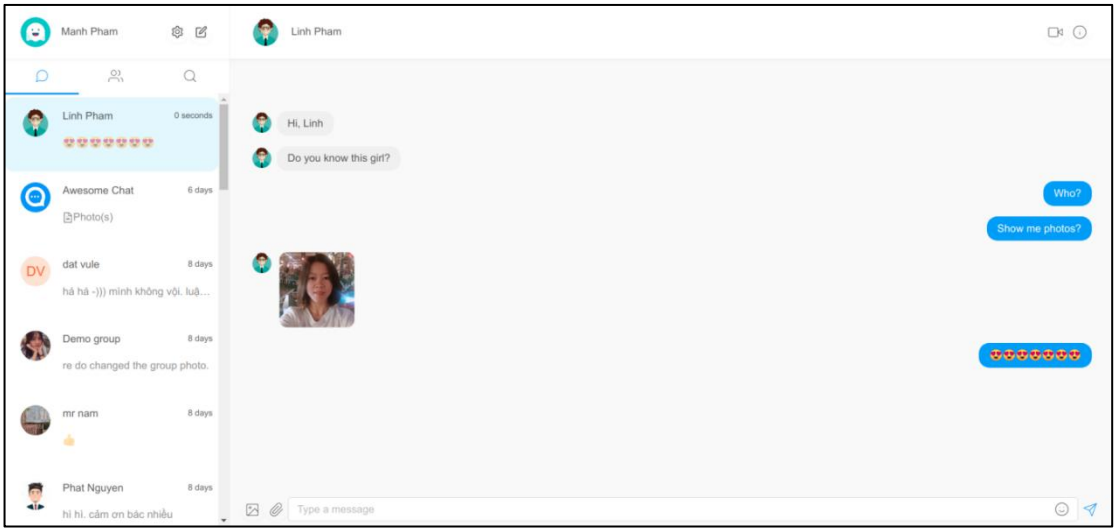

Рис. 14. Чат страница

На рисунке 15 и на рисунке 16 представлен интерфейс веб-приложения, который отображается на мобильных устройствах.

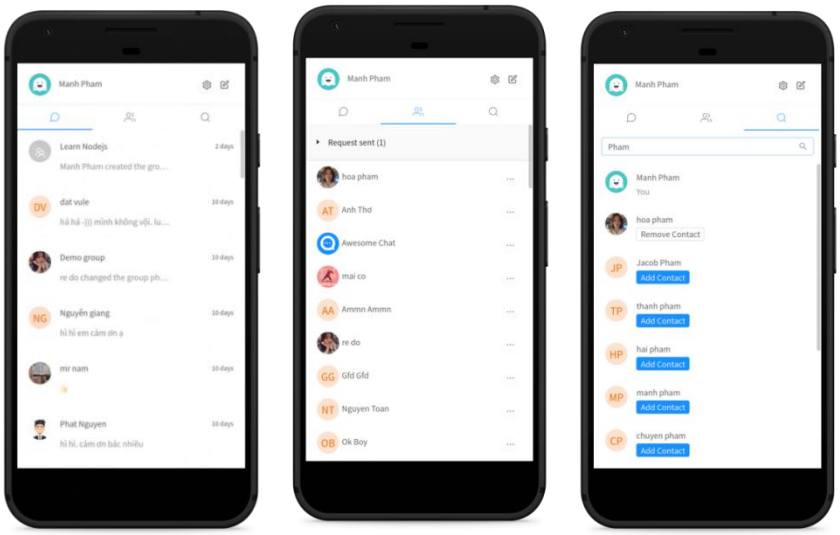

Рис. 15. Веб-приложение на мобильном устройстве

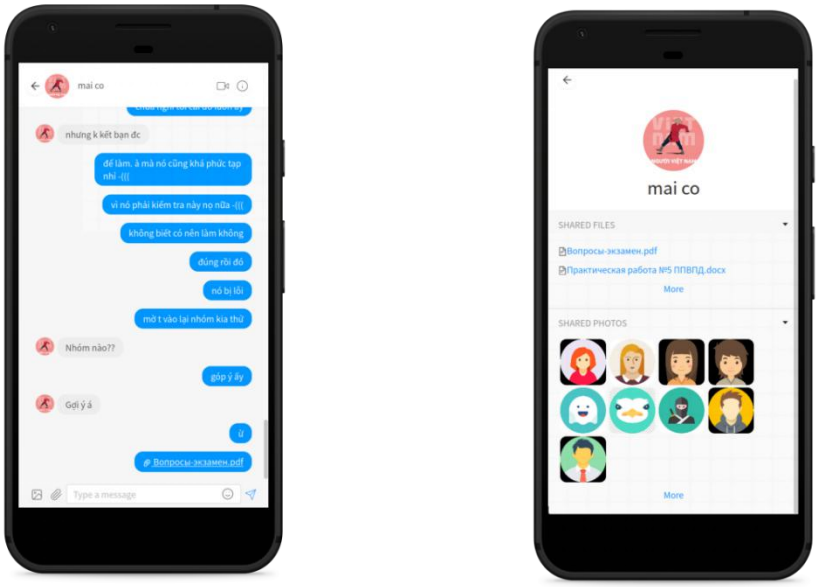

Рис. 16. Веб-приложение на мобильном устройстве

4. ТЕСТИРОВАНИЕ

4.1. Функциональное тестирование

Функциональное тестирование – это тестирование программного обеспечения в целях проверки реализуемости функциональных требований, то есть способности программного обеспечения в определенных условиях решать задачи. Функциональные требования определяют, что именно делает программное обеспечение, какие задачи оно решает. Также с помощью функционального тестирования проверим работоспособность ссылок.

Тест № 1. Регистрация пользователя.

Входные данные: вводятся данные нового пользователя.

Ожидаемый результат: если пользователь с заданной почтой уже существует выводится сообщение об необходимости использовать другую почту.

Полученный результат: на вход подана почта уже зарегистрированного пользователя, отобразилось соответствующее сообщение. На вход поданные подходящие данные регистрация, регистрация прошла успешно, пользователь перенаправлен на главную страницу.

Тест успешно пройден.

Тест № 2. Вход в систему.

Входные данные: на форму входа подаются данные для входа (почта и пароль).

Ожидаемый результат: если данный пользователь существует, происходит вход, пользователь перенаправляется на главную страницу, иначе выдается сообщение о неправильно введённых данных.

Полученный результат: на вход поданы данные несуществующего пользователя, выдалось соответствующее уведомление. На вход подаются данные существующего пользователя, вход произошел успешно, пользователь перенаправлен на главную страницу.

Тест успешно пройден.

Тест № 3. Добавить пользователя в контакт.

Входные данные: выбранный пользователя и нажимает кнопку «Добавить в контакт».

Ожидаемый результат: уведомление будет отправлено получателю в режиме реального времени, если получатель находится в сети.

Полученный результат: полученный результат совпал с ожидаемым.

Тест успешно пройден.

Тест № 4. Текстовый чат с друзьями**.**

Входные данные: пользователь введет текстовое сообщение и нажимает кнопку «отправить сообщение».

Ожидаемый результат: сообщение будет отправлено получателю в режиме реального времени, если получатель находится в сети.

Полученный результат: полученный результат совпал с ожидаемым.

Тест успешно пройден.

Тест № 5. Создать чат-группу.

Входные данные: Заполнить форму и нажать на кнопку создать.

Ожидаемый результат: пользователь перенаправлен на страницу чат и уведомление будет отправлено всем членам чат-группы в режиме реального времени.

Полученный результат: полученный результат совпал с ожидаемым.

Тест успешно пройден.

Тест № 6. Покинуть чат-группу.

Входные данные: Выбранная чат-группа.

Ожидаемый результат: пользователь будет удален из чат-группы и перенаправлен на главную страницу.

Полученный результат: полученный результат совпал с ожидаемым.

Тест успешно пройден.

Тест № 7. Удалить участника из группы.

Входные данные: выбранная чат-группа и выбранный участник.

Ожидаемый результат: данный пользователь будет удален из чат-группы.

Полученный результат: полученный результат совпал с ожидаемым.

Тест успешно пройден.

Тест № 8. Выход из учетной записи.

Входные данные: выход из учётной записи. Осуществляется выход из учетной записи.

Ожидаемый результат: после нажатия кнопки выход, произведен выход пользователь перенаправлен на страницу входа.

Полученный результат: полученный результат совпал с ожидаемым.

Тест успешно пройден.

Тест № 9. Сбросить пароль, когда забыть пароль.

Входные данные: введет адрес электронной почты и нажимает кнопку «сброса пароля».

Ожидаемый результат: ссылка для сброса пароля будет отправлена на пользователя почтовый ящик.

Полученный результат: полученный результат совпал с ожидаемым.

Тест успешно пройден.

В ходе проведения функционального тестирования, были проверены все функции приложения, ошибок выявлено не было, тестирование пройдено успешно.

4.2. Тестирование API ресурсов веб-сервера.

В процессе разработки производилось тестирование ресурсов посредством интегрированной среды тестирования API Postman [2020], пример запроса с результатом показан на рисунке 17.

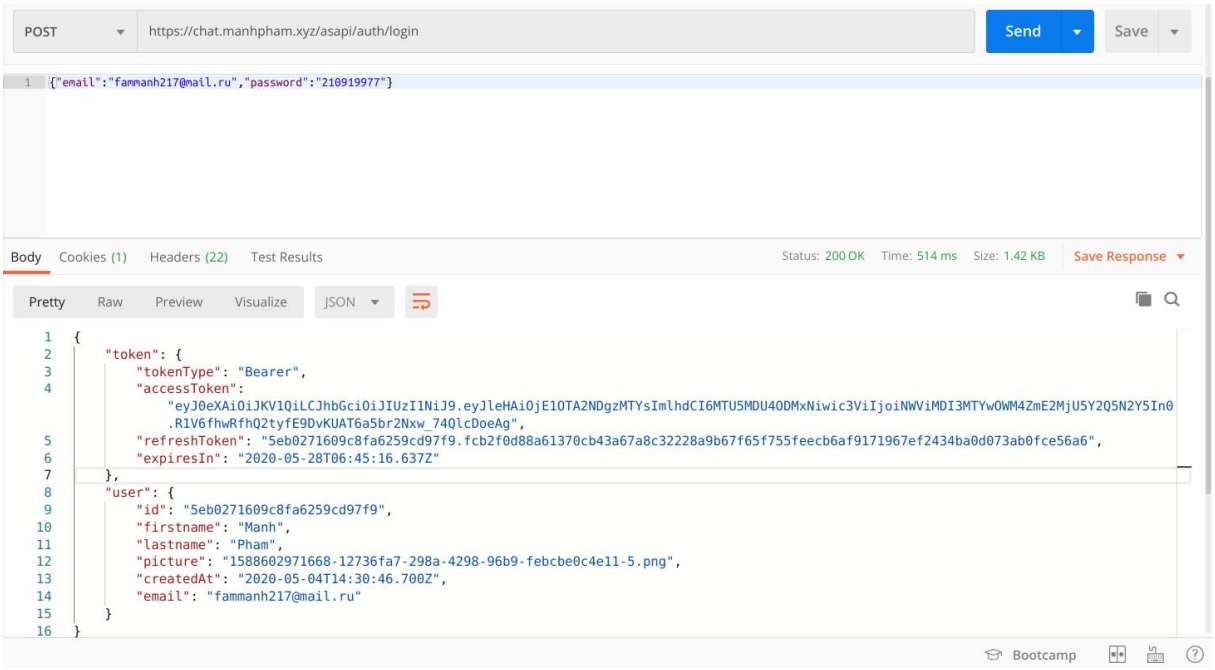

Рис. 17. Обработка запроса в Postman

Данным образом были протестированы все ресурсы, предоставляемые веб-сервером.

4.3. Тестирование интерфейса

Важно, чтобы сайт правильно отображался в разных браузерах, и на экранах разных размеров, тестирование было проведено в браузерах Google chrome версии 81.0.4044, Mozilla Firefox 76.0.1.

Во всех браузерах все страницы сайта отображались правильно, никаких нарушений не выявлено, тест пройден успешно.

Было проведено тестирование на экранах разных размеров. Мобильные устройства, ширина экрана меньше 580px, нарушений не выявлено, на всех страницах ширина контента уменьшается пропорционально ширине экрана.

4.4. Развертывание приложения на vps сервере

Приложение было развернуто на VPS от Google с 06.05.2020 по адресу https://chat.manhpham.xyz. По состоянию на 31.05.2020: 567 пользователей зарегистрировались для пробного использования,

отправлено 1 763 сообщения, создано 670 контактов и 8 чат-групп, что представлено на рисунке 18. На сервер поступило 97 014 запросов и 2 906 посетителей, что представлено на рисунке 19.

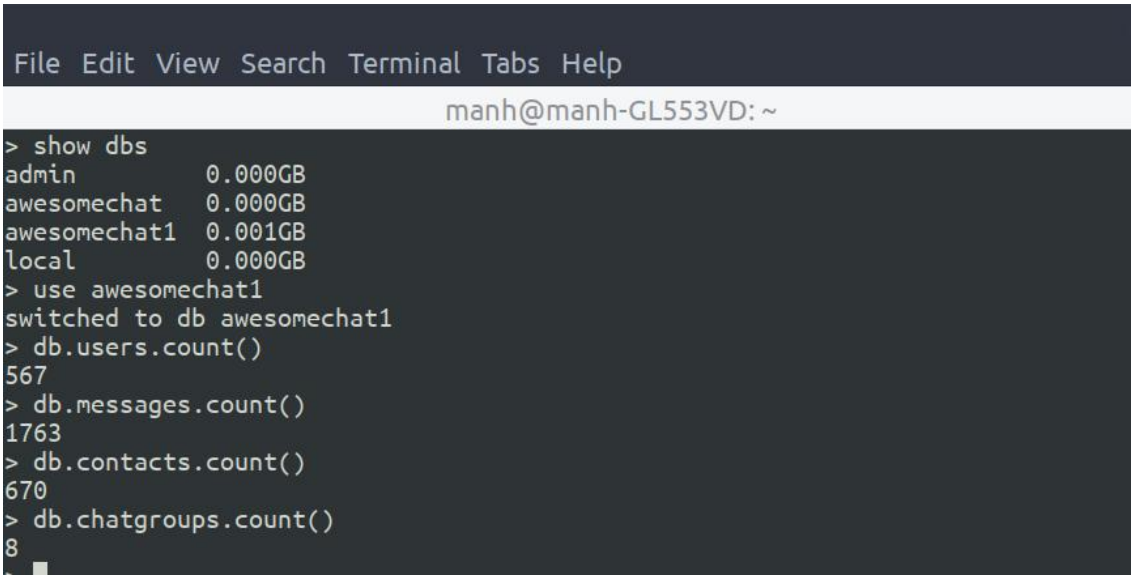

Рис. 18. 567 пользователей зарегистрировались

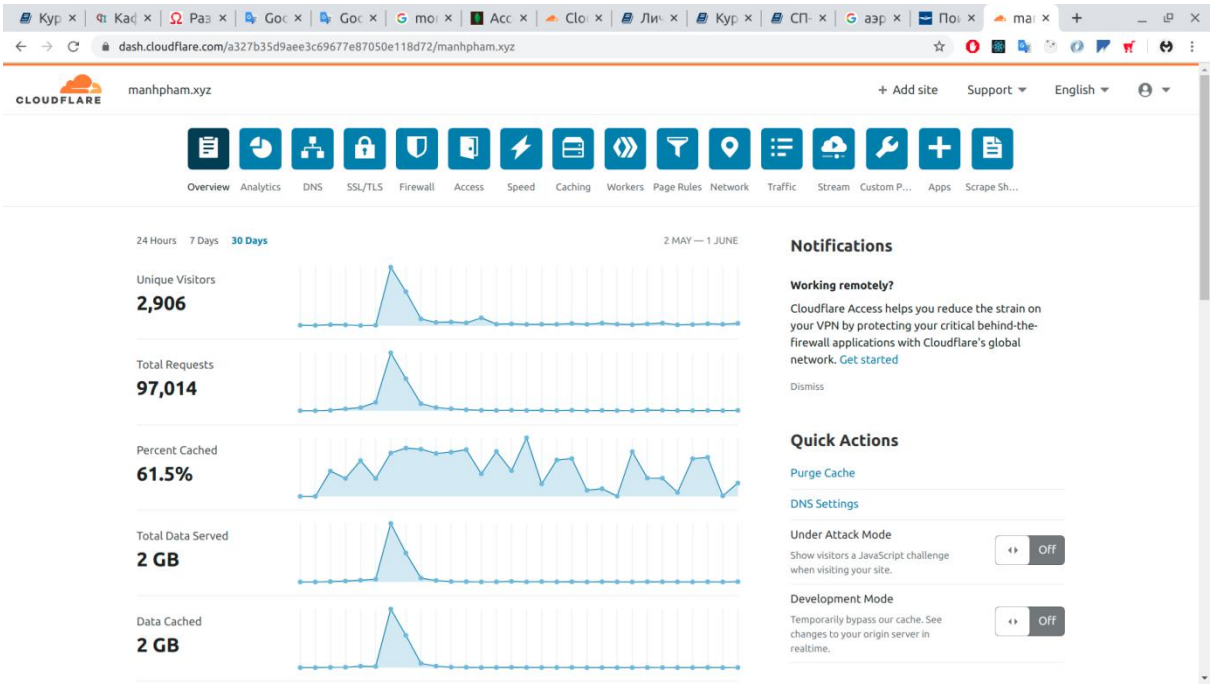

Рис. 19. Cloudflare аналитика

Был реализован на VPS со следующими основными характеристиками:

- оперативная память (ОП) – 3.75 GB;

- процессор – Intel(R) Xeon(R) CPU @ 2.00GHz;
- тип операционной системы – Linux;
- версия ОС – Centos 7.

Все функции работают правильно, ошибок выявлено не было.

Вывод

В ходе тестирования была протестирована работоспособность RESTful API и веб-интерфейсов системы. По результатам тестирования можно сделать выводы о том, что разработанная система работает корректно и функциональность системы соответствует заявленной.

ЗАКЛЮЧЕНИЕ

В ходе выполнения выпускной квалификационной работы было разработано веб-приложения «Awesome Chat», данное веб-приложения позволяет общаться с друзьями, родственниками и с любимыми, были выполнены следующие задачи:

Для достижения данной цели были решены следующие задачи:

1) выполнен анализ предметной области;
2) выполнен анализ существующих решений;
3) спроектирована структуру и логику приложения;
4) спроектирована базу данных;
5) реализована базу данных;
6) реализована веб-приложени;
7) произведено тестирование веб-приложение.

Приложение имеет адаптивный интерфейс, что позволяет пользоваться им на устройствах с экранами разных размеров. При реализации использовался шаблон проектирования MVC благодаря чему система является масштабируемой и позволяет в будущем дополнять функционал или создавать версии под различные операционные системы.

В результате выполнения выпускной квалификационной работы были решены все поставленные задачи, таким образом, цель данной работы достигнута.

При дальнейшем развитии проекта планируется реализовать следующий функционал:

- голосовой чат;
- голосовой чат для группы;
- видео чат для группы;
- онлайн/офлайн функция.

ЛИТЕРАТУРА

1. Tim M. Design and Implementation of a NoSQL-concept for an international and multicentral clinical database, 2016. [Электронный ресурс] URL: http://dbis.eprints.uni-ulm.de/1448/1/MA_MOH_2016.pdf (дата обращения 02.02.2020).

2. Marijin H. Eloquent Javascript – No Starch Press, 2018. – 436 p. https://eloquentjavascript.net/Eloquent_JavaScript.pdf.

3. Leonard R., Mike A. RESTful Web APIs. O'REILLY, 2013. 404 c.

4. Компьютер пресс. URL: https://compress.ru/article.aspx?id=10364 (дата обращения 10.03.2020).

5. Официальный сайт менеджера пакетов npm. URL: https://docs.npmjs.com/about-npm/ (дата обращения 10.02.2020).

6. Официальный сайт Node.js. URL: https://nodejs.org/ru/ (дата обращения 18.02.2020).

7. Документация сайт Reactjs. URL: https://reactjs.org/docs (дата обращения 18.02.2020).

8. Официальный сайт фреймворка Expressjs. URL: https://expressjs.com/ (дата обращения 18.02.2020).

9. Документация Socket.IO. URL: https://socket.io/docs (дата обращения 18.02.2020).

10. Документация сайт база данных MongoDB. URL: https://docs.mongodb.com/manual (дата обращения 18.02.2020).

11. Веб чаты для сайтов. URL: www.lessons-tva.info/ (дата обращения 24.03.2020).

12. Буч Г. Введение в UML от создателей языка. М.: ДМК Пресс, 2015. 496 с.

13. Сайт TopChatSites. URL: https://ru.topchatsites.com (дата обращения 05.03.2020).

14. Сайт chatspin. URL: https://chatspin.com (дата обращения 06.03.2020).

15. Сайт chatrandom. URL: https://chatrandom.com (дата обращения 06.03.2020).

16. Сайт e-chat. URL: http://www.e-chat.co (дата обращения 06.03.2020).

17. Документация сайт Antdesign. URL: https://ant.design (дата обращения 06.03.2020).

18. Mark Masse O'. REST API Design Rulebook – Reilly Media, 2011. – 116 с.

19. Документация сайт jwt. URL: https://npmjs.com/package/jsonwebtoken (дата обращения 20.04.2020).

20. Официальный сайт Postman среда разработки API. URL: https://www.postman.com (дата обращения 20.04.2020).

21. Hypertext Transfer Protocol (HTTP) Status Code Registry. URL: https://www.iana.org/assignments/http-status-codes/http-status-codes.xhtml (дата обращения 20.04.2020).