ЧАСТНОЕ ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
«МОСКОВСКИЙ ГОРОДСКОЙ ОТКРЫТЫЙ КОЛЛЕДЖ»

ДИПЛОМНАЯ РАБОТА

Тема Разработка логической части веб-приложения социальной сети на фрейворке

(наименование темы)

Специальность 09.02.07 «Информационные системы и программирование»

Группа 15.ИСиП.20.ОФ.С.1

Обучающийся _____ И.В.Штанова
(подпись) (ФИО)

Руководитель _____Д.М.Макеев
(подпись) (ФИО)

Допустить к защите Заместитель директора _____Е.Ф.Гурова
(подпись) (ФИО)

Москва
2023

ЧАСТНОЕ ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
«МОСКОВСКИЙ ГОРОДСКОЙ ОТКРЫТЫЙ КОЛЛЕДЖ»

МОСКОВСКИЙ ГОРОДСКОЙ
ОТКРЫТЫЙ КОЛЛЕДЖ

Рассмотрено на заседании
Предметно-цикловой комиссии
профессионального цикла специальностей
информационных технологий
протокол № 7
от 15.03.2023г.
ПЦК_____ Д.М.Макеев
Подпись ФИО

Утверждаю
Заместитель директора
_____Е.Ф.Гурова
(подпись) (ФИО)
«16» марта 2023г.

ЗАДАНИЕ
на дипломную работу

Обучающемуся Штановой Ирине Вадимовне
(Ф.И.О. полностью)
Курс 3 группа 15.ИСиП.20.ОФ.С.1

Тема дипломной работы: Разработка логической части веб-приложения социальной сети на фреймворке

Перечень технических решений, подлежащих разработке:
1. Продумать функционал и дизайн социальной сети
2. Осуществить разработку веб-приложения

Перечень вопросов, подлежащих рассмотрению в дипломной работе:
1. Введение
2. Глава 1. Теоретическая часть
3. Обоснование выбора дизайнерского решения
4. Глава 2. Практическая часть
5. Проектирование и разработка веб-приложения социальной сети
6. Заключение
7. Список использованных источников

Наименование предприятия, на котором выпускник проходит производственную практику (преддипломную) ФГБОУ ВО «Российский государственный социальный университет»

Задание получил _____ 16 марта 2023г.
Срок сдачи обучающимся законченной работы: 6 июня 2023г.
Руководитель дипломной работы _____ Д.М.Макеев
(подпись) (ФИО)

«СОГЛАСОВАНО»
Представитель работодателя _____
16 марта 2023г.

ЧАСТНОЕ ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
«МОСКОВСКИЙ ГОРОДСКОЙ ОТКРЫТЫЙ КОЛЛЕДЖ»

МОСКОВСКИЙ ГОРОДСКОЙ
ОТКРЫТЫЙ КОЛЛЕДЖ

КАЛЕНДАРНЫЙ ГРАФИК

Выполнения дипломной работы

Обучающийся Штанова Ирина Вадимовна
Специальность, код 09.02.07 «Информационные системы и программирование»
Курс 3 , группа 15.ИСиП.20.ОФ.С.1
Тема дипломной работы: Разработка логической части веб-приложения социальной сети на фреймворке

№п/п	***Наименование этапа работы***	***Срок выполнения***
1	Выбор темы дипломной работы, получение задания	30.03.2023 – 05.04.2023
2	Подбор литературы, ее изучение. Составление списка с основными источниками. Обсуждение выполнения дипломной работы	06.04.2023– 12.04.2023
3	Составление плана работы	13.04.2023 - 19.04.2023
4	Написание теоретической части дипломной работы	20.04.2023 – 26.04.2023
5	Проведение обследования (объекта дипломной работы) или изучение проблемы; обобщение и анализ полученных в ходе обследования или изучения материалов. Накопление, систематизация и анализ практических материалов	20.04.2023 - 17.05.2023
6	Анализ и разработка практической части дипломной работы	27.04.2023 – 17.05.2023
7	Согласование выводов и предложений, написание заключительной части	20.05.2023– 24.05.2023
8	Представление черновика дипломной работы руководителю	25.05.2023
9	Переработка (доработка) дипломной работы в соответствии с замечаниями	25.05.2023 – 27.05.2023
10	Представление дипломной работы научному руководителю на отзыв, для рецензии. Сбор оценочных материалов (отзыв, рецензия)	30.05.2023
11	Подготовка тезисов доклада для защиты, подготовка презентации	31.05.2023
12	Проведение предзащиты	01.06.2023- 07.06.2023
13	Передача переплетенной дипломной работы с отзывом руководителя и рецензией в Государственную экзаменационную комиссию	08.06.2023- 14.06.2023
14	Защита дипломной работы на заседании ГЭК.	15.06.2023

Руководитель дипломной работы _____ Д.М. Макеев
(подпись) (ФИО)
Обучающийся _____ Штанова Ирина Вадимовна

ЧАСТНОЕ ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
«МОСКОВСКИЙ ГОРОДСКОЙ ОТКРЫТЫЙ КОЛЛЕДЖ»

МОСКОВСКИЙ ГОРОДСКОЙ
ОТКРЫТЫЙ КОЛЛЕДЖ

ОТЗЫВ РУКОВОДИТЕЛЯ ДИПЛОМНОЙ РАБОТЫ

Тема Разработка логической части веб-приложения социальной сети на фрейворке

Обучающегося Штановой Ирины Вадимовны
(Ф. И. О.)

Группа15.ИСиП.20.ОФ.С.1 Курс 3

Специальность 09.02.07 «Информационные системы и программирование»

Объем дипломной работы _____ страниц, в т. ч. количество страниц приложений _____

1. Актуальность и практическая значимость темы _____
_____
2. Логическая последовательность _____
_____
3. Аргументированность и конкретность выводов и предложений _____
_____
4. Правильное использование научных/профессиональных терминов и понятий в контексте проблемы _____
_____
5. Уровень использования различных видов литературных источников
_____
6. Качество оформления дипломной работы качество таблиц, иллюстраций и пр.
_____
_____
7. Уровень самостоятельности при работе над темой дипломной работы
_____
8. Недостатки работы_____
_____
9. Дипломная работа соответствует/не соответствует требованиям, предъявляемым к дипломной работе, и может/не может быть рекомендована к защите на заседании Государственной экзаменационной комиссии
нужное подчеркнуть

оценка_____

Руководитель дипломной работы_____ Д.М.Макеев
(подпись) Ф.И.О.

6 июня 2023г.
(дата выдачи)

ОТЗЫВ РЕЦЕНЗЕНТА
О ДИПЛОМНОЙ РАБОТЕ

Дипломная работа выполнена обучающимся Штановой Ирины Вадимовны

Курс 3 Группа 15.ИСиП.20.ОФ.С.1

Специальность 09.02.07 «Информационные системы и программирование»

Руководитель Макеев Д.М.

Тема дипломной работы: Разработка логической части веб-приложения социальной сети на фреймворке

Рецензент Чижиков А.С.

Отмеченные достоинства _____
_____
_____
_____
_____

Практическая значимость работы _____
_____
_____
_____
_____

Отмеченные недостатки _____
_____
_____
_____
_____
_____

Заключение _____
_____

Рецензент _____ / А.С.Чижиков/ ИП РОМАНОВ Д.П 7 июня 2023г.
(подпись) Ф.И.О должность

МП

С рецензией ознакомлен _____ Штанова Ирина Вадимовна
(подпись) (ФИО)

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Создание нового мессенджера в настоящее время является актуальным и важным заданием, поскольку общение через мессенджеры стало неотъемлемой частью нашей жизни. Многие пользователи используют мессенджеры для общения с друзьями, коллегами и близкими, а также для работы и бизнеса. Кроме того, ситуация с пандемией COVID-19 привела к увеличению спроса на мессенджеры и другие средства коммуникации в онлайн-формате.

Существующие мессенджеры имеют свои недостатки, такие как ограниченные функциональные возможности, низкая скорость работы или недостаточная защита данных пользователей. Создание нового мессенджера с учетом этих недостатков может привести к созданию более удобного и безопасного продукта, который будет пользоваться популярностью среди пользователей.

Кроме того, создание нового мессенджера может привести к появлению новых возможностей для коммуникации и социальной интеракции, которые ранее не были доступны в других мессенджерах. Например, это может быть возможность использования искусственного интеллекта для автоматического перевода сообщений на разные языки или создание встроенных функций для организации онлайн-мероприятий и встреч.

Таким образом, создание нового мессенджера является актуальным и перспективным заданием, которое может привести к появлению нового качественного продукта на рынке коммуникационных сервисов.

Целью данной дипломной работы является разработка клиентской части мессенджера «ISKS».

1. ТЕОРИТИЧЕСКАЯ ЧАСТЬ

1.1 Обоснование выбора цветовой палитры для мессенджера

Выбор цветовой гаммы для сайта является важным аспектом дизайна, который может оказать большое влияние на восприятие пользователей. Различные цвета могут вызывать различные эмоции и ассоциации, поэтому важно выбирать цвета, которые соответствуют теме и целям сайта.

Например, использование ярких и насыщенных цветов может создавать эффект привлекательности и энергичности, в то время как более темные и приглушенные цвета могут создавать эффект спокойствия и серьезности.

Кроме того, выбор цветовой гаммы может влиять на удобство использования сайта. Например, яркий текст на темном фоне может быть трудночитаемым для пользователей с ограниченными возможностями зрения.

Тенденция выбора цветовой гаммы для разных платформ может отличаться в зависимости от целевой аудитории и тематики сайта. Например, интернет-магазины часто используют яркие и насыщенные цвета для привлечения внимания к продуктам, в то время как социальные сети могут использовать более приглушенные и спокойные цвета для создания эффекта удобства и комфорта.

В целом, выбор цветовой гаммы для сайта должен быть основан на балансе между эстетическими и функциональными аспектами, чтобы обеспечить удобство использования и привлекательный внешний вид.

Дизайн мессенджера является ключевым фактором, который влияет на его успех и популярность среди пользователей. Правильный выбор цветовой гаммы может создать приятную атмосферу для общения и привлечь новых пользователей. В данной дипломной работе мы будем проектировать мессенджер «ISKS» и выбирать цветовую гамму, которая будет соответствовать его концепции и целям. Мы рассмотрим различные варианты цветовых решений и выберем наиболее подходящий для нашего продукта.

Лавандовый цвет был выбран для дизайна мессенджера «ISKS» по нескольким причинам. Во-первых, он ассоциируется с романтикой, спокойствием и расслабленностью, что создаст приятную атмосферу для пользователей. Во-вторых,

он является необычным и малоиспользуемым цветом в мире мессенджеров, что поможет выделиться на фоне конкурентов.

Сравнение с другими цветами показывает, что красный цвет ассоциируется с энергией, страстью и динамикой, что может быть не подходящим для мессенджера, который должен быть спокойным и уютным. Синий цвет ассоциируется с профессионализмом и надежностью, но также может быть слишком холодным и формальным для мессенджера. Зеленый цвет ассоциируется с природой и экологией, но может быть слишком простым и скучным для дизайна мессенджера. Желтый цвет ассоциируется с оптимизмом и энергией, но может быть слишком ярким и раздражающим для пользователей.

Лавандовый цвет (или #9D81BA в шестнадцатеричной кодировке), выбранный для нашего проекта, является светло-фиолетовым оттенком, который может хорошо сочетаться с различными цветами. Ниже приведены некоторые из лучших и худших сочетаний лавандового цвета:

Лучшие сочетания:

- Лавандовый + белый (#FFFFFF): создает нежный и романтический эффект, идеально подходит для свадебных сайтов (рис. 1).

Рисунок 1 - лавандовый + белый

- Лавандовый + серый (#808080): создает элегантный и стильный дизайн, хорошо подходит для сайтов, посвященных моде и красоте (рис 2).

Рисунок 2 - лавандовый + серый

- Лавандовый + розовый (#FFC0CB): создает женственный и мягкий дизайн, подходит для сайтов, посвященных моде и красоте (рис. 3).

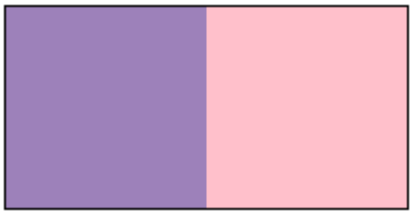

Рисунок 3 - лавандовый + розовый

Худшие сочетания:

- Лавандовый + оранжевый (#FFA500): создает слишком яркий и контрастный эффект, который может быть неприятным для глаз (рис. 4).

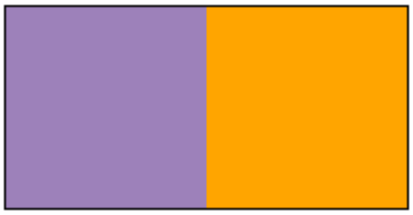

Рисунок 4 - лавандовый + оранжевый

- Лавандовый + желтый (#FFFF00): создает слишком яркий и неестественный эффект, который может отвлекать внимание от основного контента (рис. 5).

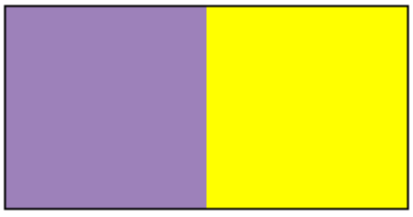

Рисунок 5 - лавандовый + желтый

- Лавандовый + красный (#FF0000): это сочетание может создать слишком яркий и неестественный вид, который может быть неприятным для глаз (рис. 6).

Рисунок 6 - лавандовый + красный

Некоторые примеры использования лавандового цвета с другими цветами:

- Лавандовый фон с белым текстом: #9D81BA на #FFFFFF (рис. 7)

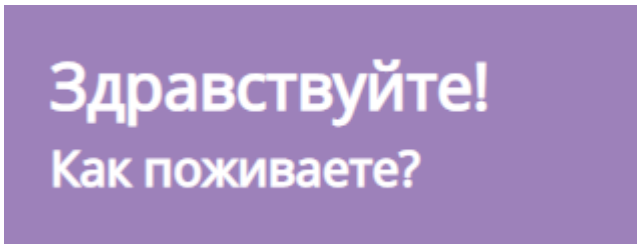

Рисунок 7 - лавандовый фон + белый текст

- Лавандовый фон с серым текстом: #9D81BA на #808080 (рис. 8)

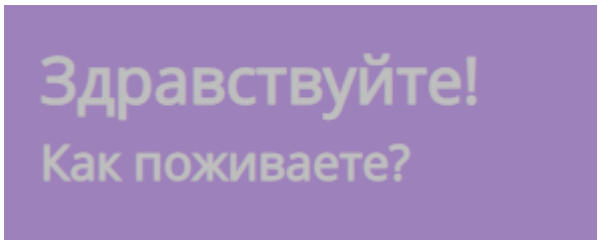

Рисунок 8 лавандовый фон + серый текст

- Лавандовый фон с розовым текстом: #FFC0CB на #00FF00 (рис. 9)

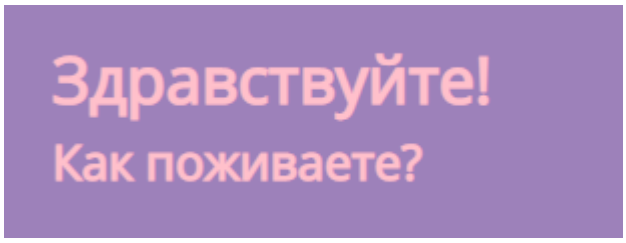

Рисунок 9 - лавандовый фон + розовый текст

- Лавандовый фон с оранжевым текстом: #9D81BA на #FFA500 (рис. 10)

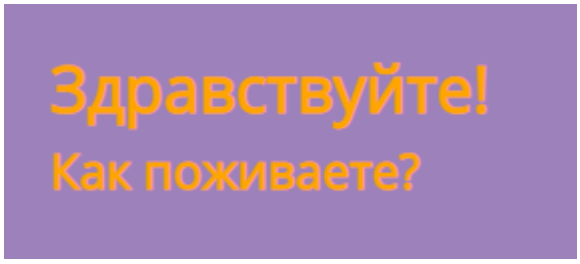

Рисунок 10 - лавандовый фон + оранжевый текст

- Лавандовый фон с желтым текстом: #9D81BA на #FFFF00 (рис. 11)

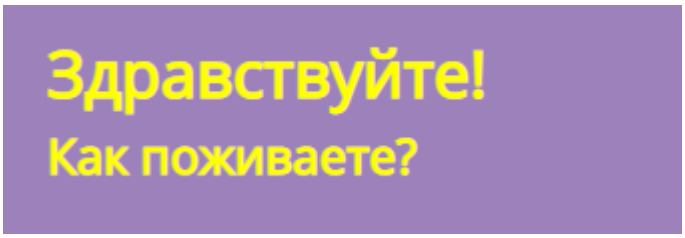

Рисунок 11 - лавандовый фон + желтый текст

- Лавандовый фон с красным текстом: #9D81BA на #FF0000 (рис. 12)

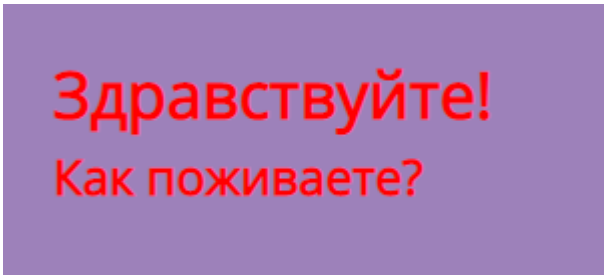

Рисунок 12 - лавандовый фон + красный текст

На основе этих сочетаний было решено выбрать лавандовый цвет (#9D81BA) – основным, белый (#FFFFFF) – цветом текста. Для кнопок и других элементов будут использоваться более тёмные оттенки основного цвета для создания контрастности.

1.2 Обоснование использования React в разработке

Сначала стоит разобраться, что же такое React. React — это библиотека JavaScript (мультипарадигменный язык программирования, поддерживающий объектно-ориентированный, императивный и функциональный стили), которая используется для создания пользовательских интерфейсов. Она позволяет разрабатывать компоненты, которые могут быть использоваться и в других частях кода. React также упрощает работу с динамическими данными и обеспечивает быстрое обновление интерфейса без перезагрузки страницы.

У этого фреймворка множество достоинств, среди которых:

- высокая производительность: React использует виртуальный DOM (Document Object Model - объектная модель документа, которую браузер создает в памяти компьютера на основании HTML-кода (HyperText Markup Language – “язык гипертекстовой разметки”, стандартизированный язык гипертекстовой разметки документов для просмотра веб-страниц в браузере), полученного им

от сервера), что позволяет минимизировать количество операций обновления страницы и улучшает производительность приложения;

- модульность: React позволяет создавать компоненты, которые могут быть использоваться в разных частях приложения, что упрощает разработку и поддержку кода;
- удобство работы с динамическими данными: React позволяет легко обрабатывать изменения данных и обновлять интерфейс без перезагрузки страницы;
- широкое сообщество разработчиков: React является одной из самых популярных библиотек JavaScript, поэтому имеет большое сообщество разработчиков, готовых помочь и поделиться опытом;
- простота в использовании: React имеет простой и понятный синтаксис, что упрощает его использование для начинающих разработчиков.

Использования React так же является логичным выбором, поскольку такой веб-ресурс, как социальная сеть, имеет в себе множество компонентов, повторяющихся на многих страницах (например, навигация, “подвал” и “шапка” сайта). Создав единожды эти элементы, можно с лёгкостью обращаться к ним с других страниц, не нагромождая код однообразными HTML-тэгами.

1.2 Обоснование выбора среды Intellij IDEA для разработки приложения

Intellij IDEA - это интегрированная среда разработки (IDE), которая предназначена для разработки программного обеспечения на языках программирования Java, Kotlin, Groovy, Scala и других. Эта IDE имеет множество функций, которые облегчают процесс разработки, включая автодополнение кода, подсветку синтаксиса, отладку, рефакторинг кода и многое другое.

Intellij IDEA имеет множество функций и возможностей, которые делают ее одной из лучших IDE для разработки на языке Java. Но также эта среда может быть использована для фронт-разработки. Некоторые из ее функций, которые могут быть полезны при фронт-разработке, включают:

- поддержка HTML, CSS и JavaScript: Intellij IDEA имеет встроенную поддержку HTML, CSS и JavaScript, что позволяет разрабатывать фронтенд-часть проекта;
- интеграция с Angular и React: IDE имеет встроенную поддержку фреймворков Angular и React, что упрощает процесс разработки фронтенд-части проекта;
- поддержка Sass и Less: Intellij IDEA имеет встроенную поддержку препроцессоров Sass и Less, что упрощает процесс написания CSS-кода;
- отладка JavaScript: IDE предоставляет возможность отладки JavaScript, что позволяет быстро определить и исправить ошибки в JavaScript-коде;
- рефакторинг CSS и JavaScript: Intellij IDEA предоставляет множество функций рефакторинга CSS и JavaScript-кода, что позволяет быстро оптимизировать код и улучшить его читаемость;
- интеграция с Git: IDE имеет встроенную поддержку Git, что позволяет быстро и легко работать с репозиториями Git при фронт-разработке;
- поддержка множества языков программирования: Intellij IDEA поддерживает множество языков программирования, включая HTML, CSS, JavaScript и другие, что делает ее универсальной и гибкой.

1.3 Обоснование выбора Visual Studio Code для разработки приложения

Visual Studio Code (VS Code) — это бесплатный и открытый исходный код редактор кода, разработанный Microsoft. Он может быть использован для разработки на многих языках программирования, включая JavaScript, HTML и CSS, что делает его идеальным выбором для фронт-разработки.

Некоторые из функций и преимуществ VS Code для фронт-разработки включают:

- интеграция с Git: VS Code имеет встроенную поддержку Git, что позволяет быстро и легко работать с репозиториями Git при фронт-разработке;

- поддержка множества расширений: VS Code имеет множество расширений, которые могут быть установлены для улучшения опыта разработки фронтенд-части проекта;
- отладка JavaScript: IDE предоставляет возможность отладки JavaScript, что позволяет быстро определить и исправить ошибки в JavaScript-коде;
- интеграция с фреймворками: VS Code имеет встроенную поддержку фреймворков Angular и React, что упрощает процесс разработки фронтенд-части проекта;
- редактирование CSS: VS Code имеет функции редактирования CSS, которые позволяют быстро оптимизировать код и улучшить его читаемость;
- поддержка Emmet: VS Code имеет встроенную поддержку Emmet, что упрощает процесс написания HTML и CSS-кода;
- кроссплатформенность: VS Code может быть использован на Windows, macOS и Linux, что делает его универсальным и гибким выбором для разработки фронтенд-части проекта.

1.4 Github как платформа хранения контента

GitHub - это онлайн-платформа для хранения, управления и совместной разработки проектов с использованием системы контроля версий Git. Он является одним из наиболее популярных ресурсов для разработки программного обеспечения и веб-приложений, и используется многими крупными и малыми компаниями, а также открытыми сообществами разработчиков по всему миру.

GitHub предоставляет возможность создания и хранения репозиториев, которые могут быть как публичными, так и приватными. Публичные репозитории могут быть доступны для просмотра и скачивания другими разработчиками, что позволяет делиться кодом и находить партнеров для совместной работы. Приватные репозитории доступны только для разработчиков, которые имеют соответствующие права доступа.

GitHub позволяет пользователям создавать репозитории, которые могут содержать код, документацию, тесты, изображения, видео и другие типы файлов, а

также использовать систему отслеживания ошибок и запросов на изменения, называемую "Issues" и "Pull Requests" соответственно.

Кроме того, GitHub позволяет пользователям работать вместе над проектом, путем "форкинга" (создания копии) существующего репозитория, внесения изменений и предложения их включения в оригинальный проект через запрос на изменение (Pull Request).

GitHub также предоставляет множество инструментов для упрощения совместной работы над проектами, включая системы автоматической сборки и развертывания (Continuous Integration and Deployment), интеграцию с другими сервисами, такими как Travis CI, Heroku, Slack и другими, а также многие другие возможности, которые делают GitHub идеальным местом для хранения и совместной разработки кода.

Одной из главных особенностей GitHub является система контроля версий Git. Git позволяет отслеживать изменения в коде и возвращаться к предыдущим версиям при необходимости. Это обеспечивает лучшую безопасность и управление проектом.

Существует множество причин, по которым стоит использовать GitHub для хранения и совместной разработки проектов. Ниже приведены некоторые из наиболее важных преимуществ GitHub:

- легкость использования: GitHub предоставляет простой и интуитивно понятный интерфейс для создания, хранения и управления репозиториями. Даже новички в области разработки могут быстро научиться использовать GitHub;
- контроль версий: GitHub использует систему контроля версий Git, которая позволяет отслеживать изменения в коде и возвращаться к предыдущим версиям при необходимости. Это обеспечивает лучшую безопасность и управление проектом;
- совместная работа: GitHub позволяет нескольким пользователям работать над одним проектом и объединять свои изменения. Это упрощает совместную работу и сотрудничество между разработчиками;

- открытый исходный код: GitHub позволяет публиковать проекты с открытым исходным кодом, что позволяет разработчикам совместно работать над ними и делать их доступными для общественности;
- интеграция с другими инструментами: GitHub предоставляет множество инструментов для автоматизации процесса разработки, таких как системы непрерывной интеграции, интеграцию с другими сервисами, такими как Travis CI, Heroku, Slack и другими;
- широкое сообщество: GitHub имеет огромное сообщество разработчиков, которые активно обсуждают и совместно работают над проектами. Это обеспечивает доступ к большому количеству образцов кода, библиотек и других полезных ресурсов.

Он позволяет сократить время и усилия, которые требуются для разработки и сотрудничества, и может быть важным фактором успеха вашего проекта. Поэтому использование Github, по моему мнению, является важной частью любого проекта.

2 ПРАКТИЧЕСКАЯ ЧАСТЬ

2.1 Описание предметной области. Обоснование необходимости программного приложения

Предметной областью данной дипломной работы является веб-приложение мессенджер «ISKS».

Создание нового месседжера обоснуется потребностью общества всегда быть на связи и иметь возможность быстро отправлять сообщения. Конечно, альтернативной могут послужить телефонные или электронные письма, но в нынешнем времени это уже не считается столь удобным, и я объясню почему.

Телефонные письма, они же SMS (Short Message Service**)** — «служба коротких сообщений»), передаются посредством сотовой связи и зависят напрямую от её качества. Но зачастую “поймать сигнал” получается плохо, а вот бесплатным Wi-Fi оснащены большая часть общественных мест (кафе, торговые центры, метро и т.д). И пусть мобильные операторы предоставляют определенный пакет услуг, в которых входят бесплатные минуты для звонков, гигабайты интернета и некое количество SMS, намного проще оставить быстрое сообщение в социальной сети или мессенджере.

Что касается электронной почты, то в современном мире она используется исключительно для деловых переписок или для регистрации на различных ресурсах.

Сейчас почти невозможно найти подростка, который бы вел переписку с другом через mail.ru или другой подобный ресурс.

Так почему же создание мессенджера является обоснованным решением? Есть несколько причин:

- существует растущий спрос на коммуникационные инструменты, которые бы позволяли людям общаться в режиме реального времени и удобно обмениваться информацией. Мессенджеры являются одним из наиболее популярных и удобных инструментов для общения, позволяющих быстро и просто передавать текстовые сообщения;
- развитие веб-технологий и увеличение доступности высокоскоростного интернета позволило создать более мощные и функциональные веб-приложения, в том числе и мессенджеры. Веб-приложение мессенджера может быть запущено на любом устройстве, имеющем доступ к интернету, и не требует установки дополнительного программного обеспечения;
- создание веб-приложения мессенджера позволит создателям расширять свой бизнес и привлекать новых пользователей. Существующие мессенджеры, такие как WhatsApp, Telegram, Viber и другие, имеют огромное количество пользователей, что свидетельствует о популярности этого типа приложений. Создание своего мессенджера может быть выгодным шагом для развития бизнеса и привлечения новых пользователей;
- создание веб-приложения мессенджера позволит пользователям сохранять свою приватность и контролировать свои данные. В отличие от существующих мессенджеров, которые могут собирать и анализировать личную информацию пользователей, создание собственного мессенджера позволит управлять этими данными и обеспечить большую защиту приватности;
- мессенджер может быть полезным для корпоративных пользователей, которые могут использовать его для обмена сообщениями и информацией внутри компании. Веб-приложение мессенджера может предоставлять

дополнительные функции, такие как шифрование сообщений, возможность обмена документами и другие.

2.2 Цели и задачи разработки проекта

Целью данной работы (дипломного проекта) является разработка интегрируемого веб-приложения мессенджера, необходимого для обмена сообщениями между пользователями.

Постановка задачи: разработать клиентскую часть веб-приложения, беря во внимание потребности пользователей в подобного вида ресурсах и учитывая тенденции и моду современного мира.

Разработка веб-приложения состоит из нескольких пунктов:

1. Планирование.
2. Проектирование.
3. Тестирование.
4. Запуск продукта.

2.3 Планирование веб-разработки

2.3.1 Продумывание логики веб-приложения

Прежде чем приступать к моделированию нашего веб-приложения, нужно провести тщательный анализ и задать себе несколько вопросов: «Какими функциями должен обладать создаваемый мной мессенджер?», «Какой внешний вид привлечет новых пользователей и будет комфортен для них?», «Что наше веб-приложение может предложить пользователям, из-за чего они бы выбрали наш продукт?».

Раз нашей задачей стала разработка мессенджера, было решено первым делом создать диаграмму перемещений (рис. 13), на которой будут показаны все варианты развития событий.

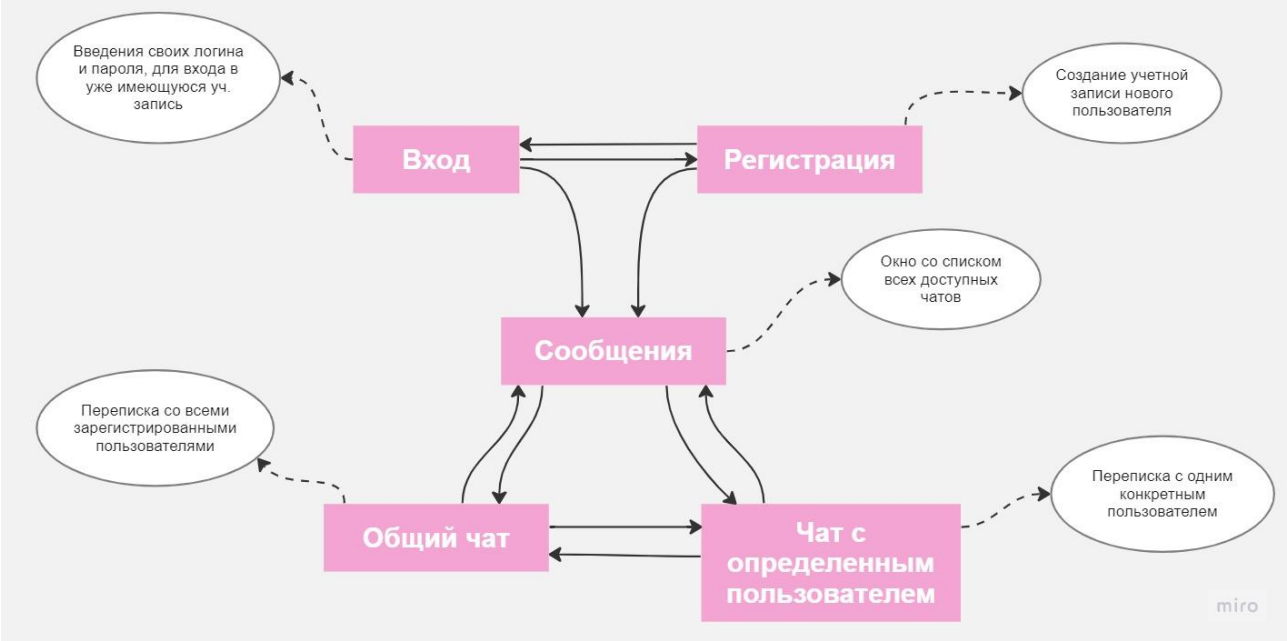

Рисунок 13 - диаграмма перемещений

Из нее видно, что пользователь будет иметь возможность создать новую учетную запись в окне “Регистрация” или зайти в уже существующую в окне “Вход”. Далее он окажется в окне “Сообщения”, откуда уже сможет начать вести диалог с выбранным собеседником или же со всеми сразу.

2.3.2 Разработка макета веб-приложения

Приняв во внимание все возможные нам варианты развития события, мы можем сформировать список страниц, необходимых нам, а именно:

- регистрация;
- вход;
- сообщения.

Стоит пояснить, почему окно “Сообщения” не поделено на чаты с группой и личные сообщения. Страница “Сообщения” будет поделена на два блока: список возможных собеседников и окно сообщений, где и будет отображаться вся переписка пользователя. Такого деления придерживаются такие мессенджеры, как Telegram и WhatsApp, поэтому мы не будем изобретать что-то новое, а будем придерживаться привычной классике.

И так, когда мы знаем необходимое количество страниц, можем приступать к созданию макета. Для этого мы воспользовались сервисом Figma.

Figma – это удобный онлайн-сервис для разработки интерфейсов и прототипирования с возможностью организации совместной работы в режиме реального времени. Сервис доступен по подписке, но также предусмотрен бесплатный тарифный план для одного пользователя. Имеются офлайн-версии для Windows, macOS.

Создав пустой проект, мы видим такое окно, в котором нам сразу предлагают выбрать размер экрана, для которого будет разрабатываться макет (рис. 14, красный маркер). Мы выберем стандартный размера экрана для ноутбука (зеленый маркер) и далее сделаем дизайн более адаптивным при необходимости.

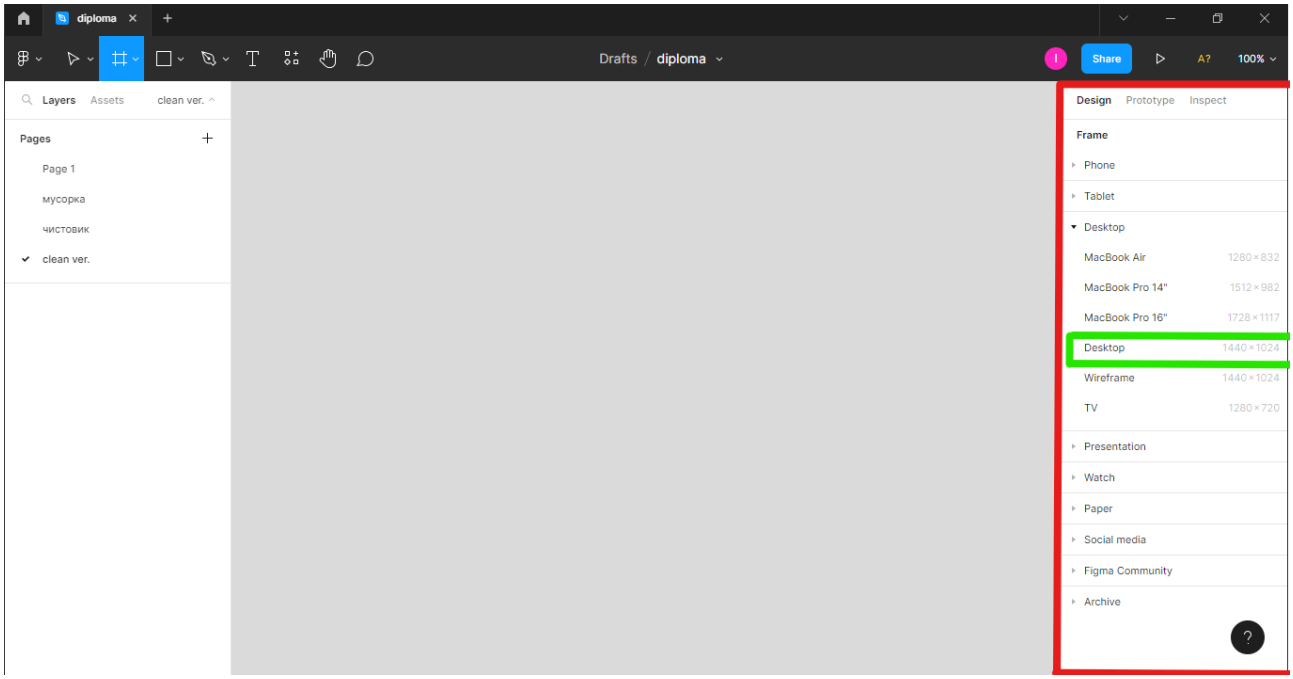

Рисунок 14 - Новый проект в Figma

Используя различные инструменты и формы, мы создали нужные страницы и готовы переходить к их переносу в код.

2.4 Перенос макета в верстку

Закончив с макетом, мы переходим в VS Code и готовимся непосредственно к программированию. Для фронт-энд разработки мы используем HTML (он же “язык разметки”), которым добавляем необходимые элементы и блоки на веб-страницу, и CSS (Cascade Style Sheet – каскадная таблица стилей), отвечающий за внешний вид нашего приложения.

2.4.1 Регистрация

Начнём со страницы “Регистрация”. Сначала подключаем все нужные нам библиотеки (рис. 15). Первой строкой мы импортируем модуль useState из библиотеки React для управления состоянием компонента. Библиотека Axios нам нужна для отправки POST-запроса на сервер, а модули ToastContainer и toast из react-toastify – для отображения уведомлений.

```
1   import { useState } from "react";
2   import axios from "axios";
3   import { ToastContainer, toast } from 'react-toastify';
4
```

Рисунок 15 - Подключение библиотек

Далее мы открываем основную функцию Register, которая является неким контейнером для всего кода одной страницы (рис. 16). За ним следует объявление переменных, через которые мы будем проверять верность веденных пользователем данных. Всего у нас шесть переменных:

- username – хранит в себе Имя пользователя;
- email – записывает электронную почту пользователя, через которую он регистрирует свою учетную запись;
- password – переменная для хранения пароля;
- emailError – сюда записывается текст об ошибки в электронной почте;
- passwordError – хранит ошибку при вводе пароля;
- usernameError – выводит ошибку в имени пользователя.

```
function Register() {
  const [username, setUsername] = useState("");
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  const [emailError, setEmailError] = useState("");
  const [passwordError, setPasswordError] = useState("");
  const [usernameError, setUsernameError] = useState("");
```

Рисунок 16 - Объявление переменных

Следующим шагом мы делаем так называемую “защиту от дурака” – защиту предметов пользования, программного обеспечения и т. п. от очевидно неверных

действий человека, то есть заранее обезопасим себя от того, что какой-то пользователь захочет отправить в базу пустые поля валидации (рис. 17).

```
const validateForm = () => {
  let isValid = true;

  if (!username) {
    setUsernameError("Username is required");
    isValid = false;
  }

  if (!email) {
    setEmailError("Email is required");
    isValid = false;
  }

  if (!password) {
    setPasswordError("Password is required");
    isValid = false;
  }

  return isValid;
};
```

Рисунок 17 - "Защита от дурака"

Для этого мы создаем переменную isValid и по умолчанию ставим ей значение true. Далее прописываем условие if (если) и проверяем переменные, назначенные на поля ввода имени пользователя, электронной почты и пароля – username, email и password, соответственно. Если эта переменная пуста, то выводится сообщение о том, что поле не заполнено, а переменная isValid принимает значение false. В конце функция возвращает значение isValid.

Полученное значение дальше обрабатывается в функции onSubmit и, если isValid равна false, POST-запрос не отправляется(рис.18). В противном случае подключается библиотека axios и, получив данные с полей ввода и отправив их в базу, выводит всплывающее окно об успешной регистрации или об ошибке.

```
const onSubmit = (e) => {
  e.preventDefault();

  if (!validateForm()) {
    return;
  }

  axios
    .post("http://localhost:8080/user/save", {
      username,
      email,
      password,
    })
    .then((response) => response.data)
    .then((data) => {
      if (data.fieldErrors) {
        data.fieldErrors.forEach((fieldError) => {
          if (fieldError.field === "email") {
            setEmailError(fieldError.message);
            toast.error(fieldError.message);
          }

          if (fieldError.field === "password") {
            setPasswordError(fieldError.message);
            toast.error(fieldError.message);
          }
          if (fieldError.field === "username") {
            setUsernameError(fieldError.message);
            toast.error(fieldError.message);
          }
        });
      } else {
        alert("Success!!");
      }
    })
    .catch((err) => console.error(err));
};
```

Рисунок 18 - Обработка полученных данных

Если при проверке данных были обнаружены какие-то ошибки, то они отобразятся под полем ввода. И когда пользователь снова нажмет на форму, то уведомление об ошибке исчезнет, чтобы не отвлекать его (рис. 19).

```
const onEmailFocus = () => {
  setEmailError("");
};

const onUsernameFocus = () => {
  setUsernameError("");
};

const onPasswordFocus = () => {
  setPasswordError("");
};
```

Рисунок 19 - Убираем уведомление об ошибке

И далее, закончив с JavaScript, начинаем формировать нашу страницу (рис. 20). Здесь у нас будет два главных объекта – header (он же “чердак”) и main, в котором будет находиться основная часть нашего сайта.

Рисунок 20 - Разметка страницы "Регистрация"

В header создаем блочный элемент с двумя классами – container и header. Первый из них будет универсальным, его мы будем использовать и в основном блоке, а второй – его вариация непосредственно для “чердака”. Внутри прописываем заголовок второго уровня (тэг h2) и записываем между тэгами название нашего мессенджера – «ISKS».

В следующем объекте main так же создаём блочный элемент с двумя классами – container и form, который является вариацией универсального класса. Внутри снова заготовок второго уровня, говорящий о том, что это форма для регистрации нового пользователя (рис. 21).

Рисунок 21 - Создание формы регистрации

Тэг form, в котором хранятся поля ввода, связываем с прописанной ранее функцией, проверяющей правильность заполнения формы. Дальше блочный элемент с классом form-group, содержащий в себе label – описание поля ввода, сам input, откуда считывается информация, и span, выводящий сообщение о некорректно заполненной форме. Таких form-group будет три штуки – для ввода имени пользователя, электронной почты и пароля.

В самом низу, перед закрытием тэга form, вставляем кнопку, сохраняющую наши результаты и отправляющую их в базу, и ссылку на страницу Входа, если вдруг у пользователя уже имеется учетная запись (рис. 22). И мы так же не забываем про все закрывающие тэги и скобки во избежание лишних ошибок.

```
            <button type="submit">Save</button>
            <a href="http://localhost:3000/">Already have an account? Log in!</a>
          </form>
        </div>
      </main>

      <ToastContainer />
    </div>
  );
}
```

Рисунок 22 - Кнопка "Сохранить" и ссылка на страницу "Вход"

На этом верстка страницы “Регистрации” закончена, переходим к странице “Вход”.

2.4.2 Вход

Далее за “Регистрацией” идет страница “Входа”, с которой могут авторизоваться пользователи, уже имеющие учетные записи.

Её структура ничем не отличается от предыдущей страницы, за исключением одной детали. Вместо проверки на корректность ввода данных мы теперь проверяем наличие этих данных, то есть регистрировал ли пользователь свою почту на нашем веб-ресурсе ранее (стр. 23).

```
try {
  const response = await axios.post("http://localhost:8080/user/login", {
    email: email,
    password: password,
  });

  const data = response.data;

  if (data.message === "Email not exists") {
    alert("Email does not exist");
  } else if (data.message === "Password Not Match") {
    alert("Password does not match");
  } else if (data.message === "Login Success") {
    navigate("/ChatRoom");
  }
} catch (err) {
  console.error(err);
  alert("An error occurred while logging in");
}
}
```

Рисунок 23 - Проверка наличия электронной почты в базе

Начинаем верстать саму форму “Входа”. От страницы “Регистрация” она отличается только заголовком второго уровня, гласящего “Welcome”, и количеством полей ввода – здесь их будет только два: для электронной почты и пароля. Снизу кнопочка “Save” и ссылка на форму с регистрацией для тех, кто учетной записи ещё не имеет.

2.3.3 Сообщения

Пройдя валидацию, пользователь попадает на страницу “Сообщения”. Вот она уже точно отличается от двух предыдущих и имеет более сложный функционал.

Первым делом так же подключаем библиотеки, но они уже отличаются от прошлых страниц (рис. 24).

```
import React, { useEffect, useState } from 'react'
import {over} from 'stompjs';
import SockJS from 'sockjs-client';
```

Рисунок 24 - Подключение библиотек на странице "Сообщения"

В данном случае мы используем библиотеки stompjs и sockjs-client для обмена сообщениями между клиентом и сервером в режиме реального времени. Это позволит нам не перезагружать страницу каждый раз, когда мы ждем ответа от собеседника, оно будет отображаться автоматически.

Для подключения к серверу и отправки сообщений создаем переменную stompClient и инициализируем её как null (значение отсутствует) (рис. 25).

Переменные privateChats и publicChats хранят список приватных и публичных сообщений, а переменная userData хранит информацию о пользователе, такую как:

- имя пользователя (username);
- имя получателя(receivername);
- статус подключения (connected, равное false);
- сообщение(message).

```
var stompClient =null;
const ChatRoom = () => {
    const [privateChats, setPrivateChats] = useState(new Map());
    const [publicChats, setPublicChats] = useState([]);
    const [tab,setTab] =useState("CHATROOM");
    const [userData, setUserData] = useState({
        username: '',
        receivername: '',
        connected: false,
        message: ''
      });
    useEffect(() => {
      console.log(userData);
    }, [userData]);

    const connect =()=>{
        let Sock = new SockJS('http://localhost:8080/ws');
        stompClient = over(Sock);
        stompClient.connect({},onConnected, onError);
    }
```

Рисунок 25 - Реализация функции отправки сообщений

UseEffect вызывается каждый раз, когда изменяется состояние переменной userData, и выводит ее значение в консоль.

Далее идет множество функций, создающий функционал мессенджера:

- функция connect используется для подключения к серверу по адресу http://localhost:8080/ws. После установления соединения функция onConnected вызывается для подписки на каналы сообщений и отправки сообщения о присоединении пользователя (рис. 26);
- функция userJoin отправляет сообщение о присоединении пользователя на сервер;
- функция onMessageReceived вызывается при получении сообщения на публичный канал. В зависимости от статуса сообщения (JOIN или MESSAGE) оно добавляется в список приватных или публичных сообщений (рис. 27);
- функция onPrivateMessage вызывается при получении приватного сообщения от другого пользователя. Она проверяет наличие списка сообщений для данного пользователя и добавляет новое сообщение в список.

Если список еще не создан, он создается и новое сообщение добавляется в него;

- функция onError вызывается при возникновении ошибки подключения к серверу и выводит сообщение об ошибке в консоль (рис. 28);
- функции handleMessage и handleUsername вызываются при изменении значений полей ввода сообщения и имени пользователя соответственно. Они изменяют состояние переменной userData, сохраняя новые значения;
- функции sendValue и sendPrivateValue вызываются при отправке сообщения на публичный или приватный канал соответственно. Они создают объект chatMessage с информацией о отправителе, получателе и сообщении, отправляют его на сервер и очищают поле ввода сообщения;
- функция registerUser вызывается при монтировании компонента и устанавливает соединение с сервером.

```
const onConnected = () => {
    setUserData({...userData,"connected": true});
    stompClient.subscribe('/chatroom/public', onMessageReceived);
    stompClient.subscribe('/user/'+userData.username+'/private', onPrivateMessage);
    userJoin();
}

const userJoin=()=>{
      var chatMessage = {
        senderName: userData.username,
        status:"JOIN"
      };
      stompClient.send("/app/message", {}, JSON.stringify(chatMessage));
}
```

Рисунок 26 - Реализация функции отправки сообщений

```
const onMessageReceived = (payload)=>{
    var payloadData = JSON.parse(payload.body);
    switch(payloadData.status){
        case "JOIN":
            if(!privateChats.get(payloadData.senderName)){
                privateChats.set(payloadData.senderName,[]);
                setPrivateChats(new Map(privateChats));
            }
            break;
        case "MESSAGE":
            publicChats.push(payloadData);
            setPublicChats([...publicChats]);
            break;
    }
}

const onPrivateMessage = (payload)=>{
    console.log(payload);
    var payloadData = JSON.parse(payload.body);
    if(privateChats.get(payloadData.senderName)){
        privateChats.get(payloadData.senderName).push(payloadData);
        setPrivateChats(new Map(privateChats));
    }else{
        let list =[];
        list.push(payloadData);
        privateChats.set(payloadData.senderName,list);
        setPrivateChats(new Map(privateChats));
    }
}
```

Рисунок 27 - Реализация функции отправки сообщений

```
const onError = (err) => {
    console.log(err);

}

const handleMessage =(event)=>{
    const {value}=event.target;
    setUserData({...userData,"message": value});
}
const sendValue=()=>{
        if (stompClient) {
          var chatMessage = {
            senderName: userData.username,
            message: userData.message,
            status:"MESSAGE"
          };
          console.log(chatMessage);
          stompClient.send("/app/message", {}, JSON.stringify(chatMessage));
          setUserData({...userData,"message": ""});
        }
}
```

Рисунок 28 - Реализация функции отправки сообщений

```
const sendPrivateValue=()=>{
    if (stompClient) {
      var chatMessage = {
        senderName: userData.username,
        receiverName:tab,
        message: userData.message,
        status:"MESSAGE"
      };

      if(userData.username !== tab){
        privateChats.get(tab).push(chatMessage);
        setPrivateChats(new Map(privateChats));
      }
      stompClient.send("/app/private-message", {}, JSON.stringify(chatMessage));
      setUserData({...userData,"message": ""});
    }
}

const handleUsername=(event)=>{
    const {value}=event.target;
    setUserData({...userData,"username": value});
}

const registerUser=()=>{
    connect();
}
```

Рисунок 29 - Реализация функции отправки сообщений

Далее займемся верстkой страницы “Сообщения”. Первым делом создадим большой блочный элемент, который будет содержать в себе весь наш чат. Назначим ему уже знакомый нам класс container и дополнительный класс big, который увеличит ширину нашего начального контейнера (рис 30).

```
return (
    <div className="container big">···
    </div>
)
```

Рисунок 30 - Создание списка чатов

Следом формируем боковое меню, в котором отображается список чатов (рис. 31).

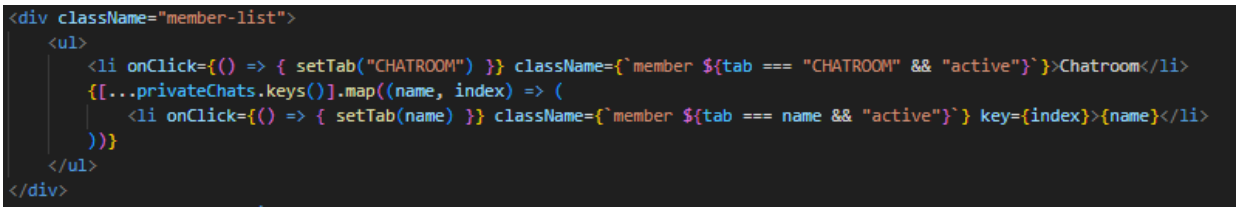

Рисунок 31 - Формирование списка чатов

Мы формирует список через маркированный список, где каждый пункт – это и есть чат с доступным собеседником. Первый пункт по умолчанию всегда будет общим чатом, куда добавляются все зарегистрированные пользователи, следом уже каждый собеседник добавляется в отдельный пункт списка.

В зависимости от текущей вкладки – общий (publicChat) или личный (privateChat) чат – класс chat-content определяет, где отображать сообщения (рис. 32).

Рисунок 32 - Отображение сообщений

Внутри блока "chat-content" отображается список сообщений, которые хранятся в массиве publicChats. Каждое сообщение представлено в виде элемента списка li с классом "message". Если отправитель сообщения совпадает с текущим пользователем (userData.username), то сообщение отображается справа, иначе слева. Также в случае, если отправитель не является текущим пользователем, отображается его имя в виде элемента "avatar".

Поле ввода сообщения представлено в виде input с классом "input-message", значение которого хранится в состоянии userData.message. При изменении значения поля вызывается функция handleMessage, которая обновляет состояние userData.message.

```
.container{
  display: flex;
  flex-direction: row;
  align-items: center;
  gap: 24px;
  width: 65%;
}
```

Кнопка отправки сообщения представлена в виде элемента button с классом "send-button". При клике на кнопку вызывается функция sendValue, которая добавляет новое сообщение в массив publicChats.

Такой же функционал и у личных чатов, но стоит обратить внимание, что в этом случае используется имя пользователя (tab), которому отправляется сообщение, а не его id.

2.3.4 Стили

Несмотря на то, что наш проект может показаться довольно объемным с большим количеством классов, для всех трех страниц мы использовали один файл .css. Это так же обусловлено тем, что мы использовали одни и те же классы на нескольких страницах и тем, что нам не пришлось создавать большое количество однотипных компонентов для выравнивания и позиционирования. Среди этих стилей хотелось бы отметить некоторые из них.

Класс “container”, который мы использовали на каждой нашей странице. Как и говорилось ранее, он выравнивал основные информационные блоки по центру с нужной для нас шириной:

Стиль тэга <input> - нашего поля ввода. Дабы оно не сильно выделялось на фоне лавандового цвета, мы сделали его приглушённого цвета. Мы отошли от варианта с белым цветом, поскольку он бы стал бы слишком ярким пятном на фоне. Так же мы изменили стиль placeholder’a, чтобы он не выбивался из общей композиции:

```
input {
  height: 36px;
  border-radius: 5px;
  border: .5px solid rgb(152, 72, 189);
  background-color: #e0c4ff;
  padding: 10px;
}
::placeholder {
  color: #9f7ec4;
  font-size: 16px;
  font-weight: 400;
}
```

На все кнопки мы наложили особые стили, которые срабатывают, когда на объект наводят мышкой. Это добавило изюминки нашей социальной сети и сделало её более интерактивной. Суть проста: движение мышки является неким триггером, который запускает другой сценарий стилей, в нашем случае – фон и текст кнопки становится более насыщенный, что интуитивно подсказывает пользователю, что её можно нажать:

```
button{
  border: none;
  padding: 10px;
  background: #9f7ec4;
  color:#d3c5c5;
  font-size: 1.2em;
  font-weight: bold;
  transition: .3s;
}
button:hover {
  background: #9364c9;
  color: #fff;
}
```

Особо хотелось бы выделить тэг <a> на формах регистрации и авторизации. Для него был прописан особенно сложный код, в котором прописан такой псевдо-класс, как before. Благодаря ему мы сменили привычное подчёркивание ссылки отдельным псевдо-элементом, которому в последствии прописали отдельный сценарий стилей, подключающийся при наведении, как и к основному элементу <a>:

```
a {
  text-align: center;
```

```
  font-size: 16px;
  font-weight: 600;
  text-decoration: none;
  position: relative;
  display: inline-block;
  color: #fff;
  transition: color .3s ease-out;
}
a::before {
  content: "";
  position: absolute;
  z-index: -1;
  bottom: 0;
  left: 0;
  height: 1px;
  width: 100%;
  background: #9364c9;
  transition: height .3s ease-out;
}
a:hover {
  color: #fff;
}
a:hover::before {
  height: 100%;
}
```

ЗАКЛЮЧЕНИЕ

В заключении моего дипломного проекта, посвященного разработке клиентской части веб-приложения социальной сети, я хотела бы подчеркнуть следующие ключевые моменты:

1. Успешное выполнение данного проекта подтверждает мои знания и навыки в области разработки веб-приложений. Я продемонстрировала способность анализировать требования пользователей, разработку собственного технического задания и реализацию задуманного.

2. Разработанный веб-приложение обладает современным дизайном и удобным интерфейсом, что позволяет пользователям обмениваться сообщениями и информацией эффективно и безопасно. Мною были реализованы такие ключевые функции, как отправка текстовых сообщений и создание групповых чатов.

3. В процессе разработки я активно использовал современные технологии и инструменты, такие как HTML, CSS, JavaScript и фреймворк React. Это позволило создать гибкую и масштабируемую архитектуру приложения, обеспечивая быстродействие и отзывчивость интерфейса.

4. В процессе тестирования приложения были выявлены и успешно исправлены возможные ошибки и недочеты. Пользовательский фидбек и активное участие тестировщиков помогли совершенствовать функциональность и улучшать пользовательский опыт.

В целом, разработка веб-приложения мессенджера была успешной, и полученные результаты полностью удовлетворяют поставленным целям проекта. Созданное мной приложение предоставляет надежный и удобный инструмент для общения и обмена информацией, соответствуя современным требованиям и ожиданиям пользователей.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. ГОСТ 34.003-90. Информационная технология. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Термины и определения.
2. ГОСТ 34.601-90. Информационная технология. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Стадии создания.
3. ГОСТ 34.602-89. Информационная технология. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы.
4. Документация htmlbook (http://htmlbook.ru/).
5. Документация W3C HTML (https://www.w3schools.com/html/).
6. Документация W3C CSS (https://www.w3schools.com/css/).
7. "Изучаем React: Функциональная веб-разработка с использованием React и Redux" - Алекс Бэнкс, Ева Портилло (2018).
8. "Паттерны и лучшие практики React" - Карлос Сантана Родригес (2018).