МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ  
Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ   
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационных систем и технологий

Специальность 1-98 01 03 «Программное обеспечение информационной безопасности мобильных систем»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА КУРСОВОГО ПРОЕКТА**

по дисциплине «Технологии frontend-разработки»

Тема «Web-сайт «система обмена сообщений»

**Исполнитель**

студент 2 курса 7 группы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Д. В. Тимошенко

подпись, дата

**Руководитель**

доцент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Н. А. Жиляк

должность, учен. степень, ученое звание подпись, дата

Допущен к защите \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

дата, подпись

Курсовой проект защищен с оценкой

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_ Н. А. Жиляк

подпись дата инициалы и фамилия

Минск 2022

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования   
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий   
Кафедра информационных систем и технологий

Утверждаю

Заведующий кафедрой

\_\_\_\_\_\_\_\_\_\_\_ В.В. Смелов

подпись инициалы и фамилия

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2022 г.

**ЗАДАНИЕ**

**на курсовое проектирование**

по дисциплине«Технологии frontend-программирования»

Курс 2 Группа:7 Специальность: 1-98 01 03 «Программное обеспечение информационной безопасности мобильных систем»

Студент: Тимошенко Дмитрий Валерьевич

**1. Тема:** Web-сайт «система обмена сообщений»

**2. Срок выполнения курсового проекта**: с 01 сентября 2022 г. по 16 декабря 2022 г.

**3. Технические требования:**

3.1 Работа с create-react-app, Virtual DOM, JSX:

* работа c React Hooks (применение правил использования и пишем свои);
* тестирование компонент с помощью Jest и ReactTestingLibrary;
* подключение стилей с css modules;
* переиспользование кода с помощью наследования, декораторов и кастомных хуков;
* связь с DOM: keys & refs;
* подключение сторонних компонентов.

3.2 Использование Redux.js: функциональный подход, Redux dev tools:

* Store как иммутабельный объект: Redux + Immutable.js;
* настраивание синхронного потока с Redux, переиспользование созданных ранее компонент;
* использование React-redux для связи компонентов с логикой;
* использование мемоизированных селекторов reselect;
* применение Side-effects в Redux: создание и использование Middlewares.
* получение данных от сервера.
* использование асинхронных экшенов с redux-thunk;
* применение Redux ToolKit;

3.3 Веб-сайт должен содержать несколько веб-страниц;

3.4 Верстка сайта должна быть адаптивной и кроссбраузерной;

3.5 Для тестирования использовать Linter;

3.6 Проект и пояснения к проекту должны быть размещены на GitHub.

**4. Содержание пояснительной записки**

1. Титульный лист;

2. Задание на курсовое проектирование;

3. Введение;

4. Постановка задачи;

5. Проектирование веб-сайта;

6. Реализация структуры веб-сайта;

7. Тестирование веб-сайта;

8. Заключение;

9. Список использованных литературных источников;

10. Приложения (полный исходный текст программы разработанного приложения с подробными комментариями).

**5. Форма представления на GitHub выполненного курсового проекта:**

– теоретическая часть пояснительной записки курсового проекта должна быть представлена в формате MS Word;

– оформление записки должно быть согласно правилам;

– листинги представляются в приложении.

#### Календарный план

|  |  |  |
| --- | --- | --- |
| № п/п | Наименование этапов курсового проекта | Срок выполнения этапов проекта |
| 1 | Задание на курсовое проектирование. Титульный лист | 14.09.2022 |
| 2 | Введение. Постановка задачи | 22.09.2022 |
| 3 | Проектирование веб-сайта | 28.09.2022 |
| 4 | Реализация структуры веб-сайта на HTML5 и внешнего оформления на Sass/СSS3 | 09.10.2022 |
| 5 | Тестирование веб-сайта | 30.10.2022 |
| 6 | Заключение | 02.11.2022 |
| 7 | Список использованных литературных источников. Приложения | 02.11.2022 |
| 6 | Подготовка и оформление пояснительной записки курсового проекта | 03.12.2022 |
| 7 | Сдача и защита курсового проекта | 10.12.2022 |

**5. Дата выдачи задания** «01» сентября 2022 г.

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Н.А. Жиляк

(подпись)

Задание принял к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата и подпись студента)

**Содержание**

[Введение 5](#_Toc121854143)

[1. Обзор технических методов и программных средств разработки веб-сайта 6](#_Toc121854144)

[1.1 Обследование объекта и обоснование необходимости создания программного продукта 6](#_Toc121854145)

[1.2. Формирование требований к ПП 7](#_Toc121854146)

[1.3. Выбор средств реализации программного продукта 8](#_Toc121854147)

[1.4. Вывод 9](#_Toc121854148)

[2. Последовательность разработки содержания и структуры веб-сайта c использованием компонентов 10](#_Toc121854149)

[2.1. Структура ПП 10](#_Toc121854150)

[2.2. Выбор стилевого оформления 10](#_Toc121854151)

[2.3. Выбор шрифтового оформления 11](#_Toc121854152)

[2.4. Разработка логотипа 11](#_Toc121854153)

[2.5. Разработка пользовательских элементов 12](#_Toc121854154)

[2.6. Вывод 13](#_Toc121854155)

[3. Последовательность разработки содержания и структуры веб-сайта c использованием компонентов 14](#_Toc121854156)

[3.1. Внедрение в проект облачной платформы Firebase 14](#_Toc121854157)

[3.2. Использование Redux и Redux Toolkit 15](#_Toc121854158)

[3.3. Использование Material-UI 16](#_Toc121854159)

[3.4. Вывод 17](#_Toc121854160)

[4. Описание хода работы программного средства 18](#_Toc121854161)

[4.1. Руководство пользователя 18](#_Toc121854162)

[4.2. Тестирование ПП 20](#_Toc121854163)

[4.3. Вывод 21](#_Toc121854164)

[Заключение 22](#_Toc121854165)

[Список использованных литературных источников 23](#_Toc121854166)

[7. Приложение 24](#_Toc121854167)

[Приложение листинги компонентов 24](#_Toc121854168)

**Введение**

В нашем современном мире имеется огромный спрос пользователей на веб-мессенджеры, которые могли бы работать и позволять комфортно обмениваться сообщениями как на персональном компьютере и ноутбуке, так и на мобильном телефоне и планшете, но также необходимо учитывать необходимость поддержания кроссбраузерности веб-мессенжера, для возможности правильного отображения сообщений в браузере, вне зависимости от его производителя.

В связи с актуальностью данной работы целью курсового проекта является разработка и реализация структуры кроссбраузерного веб-приложения системы обмена сообщений с использованием JavaScript-библиотек ­ React и Redux, а также вспомогательных библиотек.

К задачам курсового проекта относится: обзор аналогичных решений, характеристика решаемой задачи, выбор средств реализации; проектирование страниц веб-приложения, включающее разбиение веб-приложения на компоненты, стилевого оформления, пользовательских элементов, логотипа; реализация структуры веб-приложения, включающая листинги структуры JSX-документов, таблиц стилей, использования Redux Toolkit; руководство пользователя; формирование вывода по каждому разделу; заключение, включающее вывод по проделанной работе.

Спектр использования веб-приложений очень широк. Веб-приложение представляет собой веб-сайт, на котором размещены страницы с частично либо полностью несформированным содержимым. Окончательное содержимое формируется только после того, как посетитель сайта запросит страницу с веб-сервера. В связи с тем, что окончательное содержимое страницы зависит от запроса, созданного на основе действий посетителя, такая страница называется динамической.

**1. Обзор технических методов и программных средств разработки веб-сайта**

**1.1 Обследование объекта и обоснование необходимости создания программного продукта**

В сети интернет существует множество систем обмена сообщениями (мессенджеров). Анализируя многие примеры мессенджеров, можно сделать вывод, что весь контент размещен по центру, для удобного просмотра. Примером такого дизайна может стать раздел сообщений веб-сайта социальной сети «Вконтакте», представленный на рисунке 1.1.

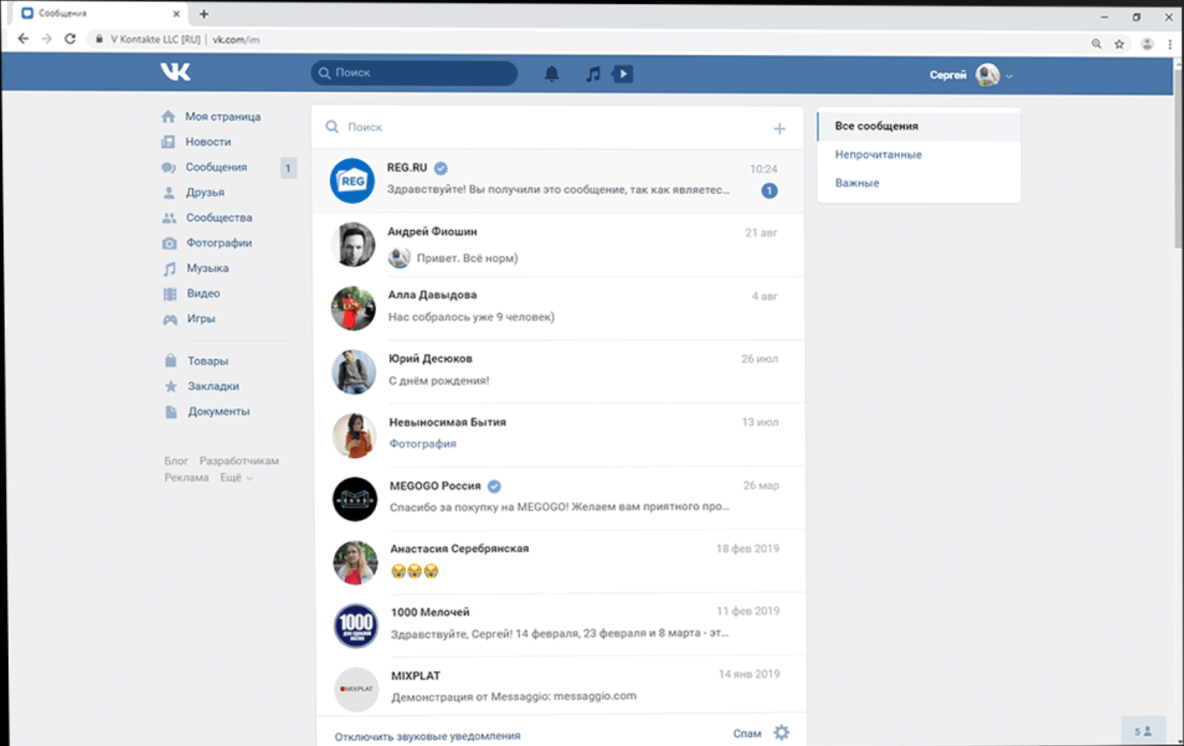


Рисунок 1.1 – раздел сообщений «Вконтакте»

Данная социальная сеть создана с использованием библиотеки React.js, что демонстрирует, что с помощью данной библиотеки можно реализовать удобный мессенджер. Пример показывает, что очень удобно, когда можно выбрать с кем именно вести диалог(переписываться).

Также примером дизайна может стать веб-версия мессенджера «Telegram», представленная на рисунке 1.2.

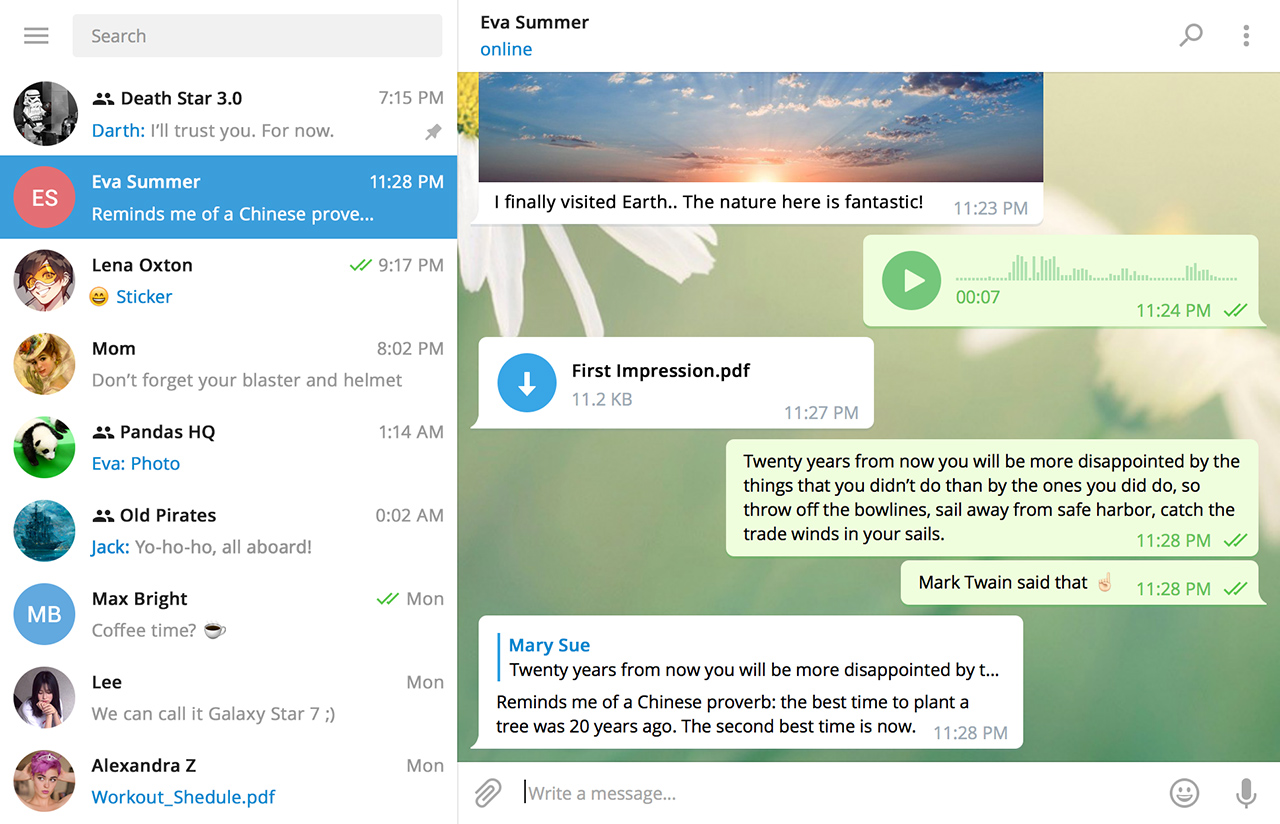


Рисунок 1.2 – мессенджер «Telegram»

В данном примере представлен другое оформление страницы выбора чатов, в сочетании с новой идеей – темой самого окна чата.

**1.2. Формирование требований к ПП**

Исходя из изученных примеров можно сделать вывод: в данном проекте мессенджер должен содержать страницы логина, выбора комнаты и саму страницу переписки. Страница выбора комнаты чата должна быть оснащена формой создания комнаты для общения, с функцией защиты от несанкционнированного входа пользователей, а также карточками уже существующих комнат для последующего присоединения к ним.

На странице переписки должно быть расположено окно чата и форма ввода, обработки и отправки сообщения.

Также должна быть реализована страница входа, содержащая кнопки входа с использованием информации о пользователе, получаемой из социальных сетей.

Основное требование – реализации возможности вести переписку в рамках одного чата.

Основная функция – дать возможность пользователям общаться с использованием разработанной системы обмена сообщениями.

Основными задачами курсового проекта являются: создания дизайна, разбиение приложения на компоненты и их реализация, разработка функциональной состовляющей.

**1.3. Выбор средств реализации программного продукта**

Для реализации программного продукта были выбраны следующие инструменты: язык разметки HTML5; язык стилей CSS3; язык программирования JavaScript; язык программирования NodeJS, библиотеки React, Redux, Material UI, и другие вспомогательные библиотеки; пакетный менеджер npm; облачная платформа Firebase; инструмент Create React App; текстовый редактор Microsoft Visual Code.

**HTML** (Hypertext Markup Language) – это язык гипертекстовой разметки, который используется для структурирования и отображения веб-страницы и её контента. Например, контент может быть структурирован внутри множества параграфов, маркированных списков или с использованием изображений и таблиц данных.

**CSS (Cascading Style Sheets, каскадные таблицы стилей)** – язык описания внешнего вида HTML-документа. Это одна из базовых технологий в современном интернете. Практически ни один сайт не обходится без CSS, поэтому HTML и CSS действуют в единой связке.

**JavaScript** – это язык программирования, который используют для написания frontend - и backend-частей сайтов, а также мобильных приложений. Часто в текстах и обучающих материалах название языка сокращают до JS. Это язык программирования высокого уровня, то есть код на нем понятный и хорошо читается.

**Node.js** – это кроссплатформенная среда выполнения для JavaScript, которая работает на серверах. Платформа Node.js построена на базе JavaScript движка V8 от Google, который используется в браузере Google Chrome. Данная платформа, в основном, используется для создания веб-серверов, однако сфера её применения этим не ограничивается.

**React.js** ‑– это инструмент для создания пользовательских интерфейсов. Его главная задача — обеспечение вывода на экран того, что можно видеть на веб-страницах. React значительно облегчает создание интерфейсов благодаря разбиению каждой страницы на небольшие фрагменты. Эти фрагменты называются компонентами.

**Redux** – это инструмент для управления состоянием данных и пользовательским интерфейсом в приложениях JavaScript с большим количеством сущностей. Представляет собой библиотеку JavaScript.

**Firebase** – это API для синхронизации данных приложений, работающих на устройствах Android, Web и iOS. Разработчики приложений могут использовать эту платформу для создания совместимых с различными платформами приложений в реальном времени.

**Material-UI –** это собрание простых и настраиваемых компонентов, благодаря которому React-приложения становятся быстрее, красивее и доступнее.

**Create React App** – это инструмент, который позволяет нам настроить среду для создания одностраничных веб-приложений на основе React, посредством выполнения всего одной команды.

**npm** – менеджер пакетов, входящий в состав Node.js, благодаря которому производится управление пакетами.

Курсовой проект выполняется в текстовом редакторе Microsoft Visual Code.

**1.4. Вывод**

В данном разделе были рассмотрены аналогичные решения, были проанализированы их недостатки и преимущества, для создания в последующем качественного мессенджера. Были поставлены конкретные задачи для создания интерактивного веб-приложения. Все вышеперечисленные пункты понадобятся для достижения поставленной задачи и для создания в дальнейшем качественного продукта.

**2. Последовательность разработки содержания и структуры веб-сайта c использованием компонентов**

## **2.1. Структура ПП**

Данный проект является веб-приложением, состоящим из множества компонентов. Каждая страница включает в себя шапку. Шапка состоит из надписи «Чатик» и кнопки «Login», если пользователь не авторизован, в другом случае кнопка заменена аватаркой пользователя, при нажатии на которое открывает меню с двумя кнопками: «Settings» и «Logout». Структура приложения представлена на рисунке 2.1.

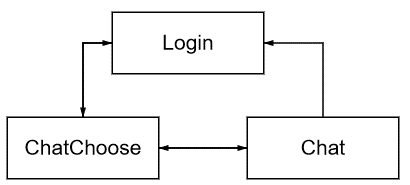


Рисунок 2.1 – Структура сайта

Мессенджер должен корректно отображаться в основных браузерах, для удобного взаимодействия с пользователями.

На странице «Login» располагается 3 кнопки для входа, используя аккаунты из веб-платформ, таких как Facebook, Google, а также GitHub.

На странице «ChatChooser» располагается форма создания комнаты чата, включающая название и пароль комнаты, и карточки уже созданных комнат, при нажатии на которые, если они защищены паролем появляется модальное окно для ввода пароля, иначе происходит перенаправление на страницу выбранного чата.

На странице «Chat» располагается окошко отображения сообщений и форма для ввода сообщения с возможностью приложения, к отправляемому сообщению, изображений.

Также должна быть страница «404 Not found», которая будет указывать пользователям, что страницы, на которой они находятся, не существует.

Сайт поддерживает две темы: светлая и темная. Переключение между ними выполняется в выподающем окошке.

**2.2. Выбор стилевого оформления**

При создании стиля и дизайна приложения были выбраны броские цвета, гармонично сочетающиеся между собой, а также были использованы шрифты, для увеличения привлекательности и комфортности в использовании.

Веб-сайт не загромождается лишними деталями (боковыми панелями и т.п.), большим количеством спецэффектов. Акцент сделан на максимально эффективное и комфортное пользование мессенджером.

Используемые цвета: #1976D2 – основной цвет светлой темы, #FFFFFF – цвет текста, #272727 – основной цвет темной темы, #90CAF9 –цвет для кнопок темной темы, #342B45 – цвет фона темной темы (рисунок 2.2).

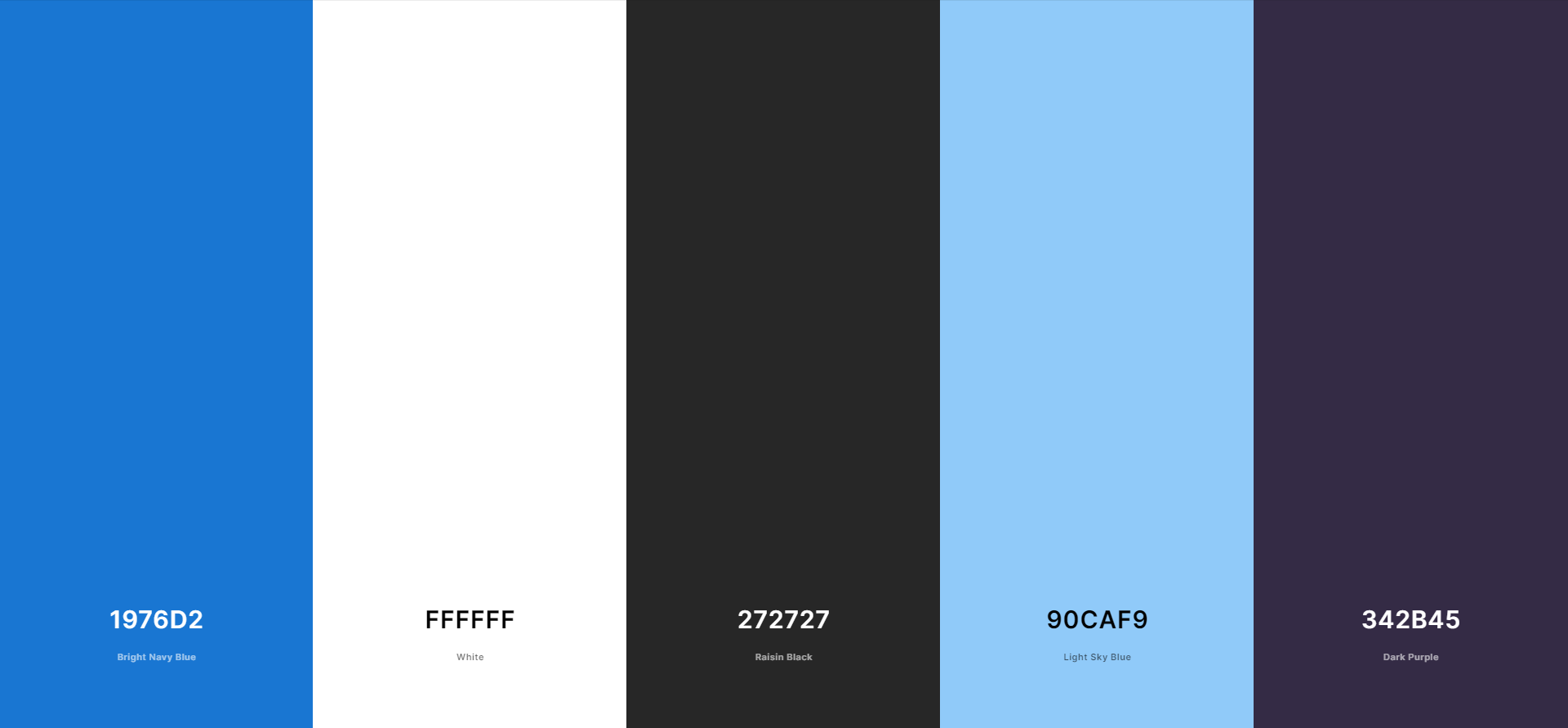


Рисунок 2.2 –Цветовая палитра сайта

**2.3. Выбор шрифтового оформления**

В данном проекте будут использованы шрифты: Roboto − основной шрифт для текста, Helvetica – вторичный шрифт для текста, Arial – запасной шрифт, отображаемый, если другие шрифты не поддерживаются. Шрифты поддерживают кириллическое написание и не конфликтуют друг с другом по стилевым характеристикам.

**2.4. Разработка логотипа**

Логотип создавался с помощью бесплатного редактора изображений - GIMP. Дизайн прост, рисованная синяя призма, подходящая под цветовую гамму приложения. Рисунок 2.3.

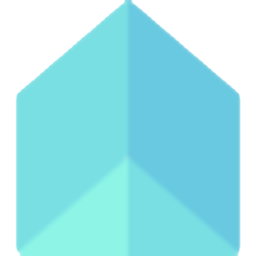


Рисунок 2.3 – Логотип веб-сайта

**2.5. Разработка пользовательских элементов**

В проекте будут представлены такие элементы пользовательского интерфейса, как навигационная панель веб-сайта (рисунок 2.4), карточки комнат и табличка авторизации.



Рисунок 2.4 – Навигационная панель веб-сайта

У пользователя должна быть возможность выбрать комнату чата, к которому он присоединится. Для этого необходимо создать простые карточки содержашие сведения о чате (рисунок 2.5).

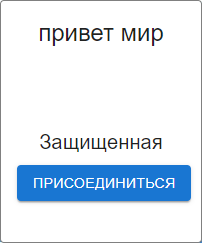


Рисунок 2.5 – Карточка комнаты чата

Лицо любого сайта — это первая страница, на которую попадает любой пользователь при посещении сайта, и если данная страница будет плохо выполнена, то пользователь может покинуть ресурс для того, чтобы это избежать была разработана красивая табличка авторизации в совокупности с страницей авторизации (рисунок 2.6).

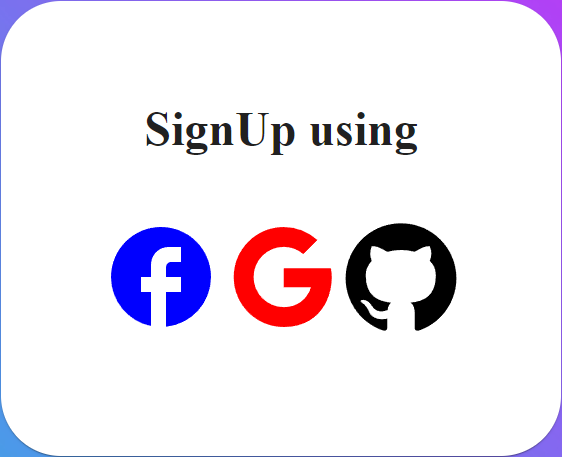


Рисунок 2.6 – Табличка авторизации

**2.6. Вывод**

На данном этапе было упорядочено расположение всех элементов на страницах (навигационного меню, логотипа, основного контента каждой страницы, расположение картинок). Было определено стилевое оформление веб-сайта, то есть основная цветовая гамма, шрифтовое оформление, элементы пользовательского интерфейса.

**3. Последовательность разработки содержания и структуры веб-сайта c использованием компонентов**

## **3.1. Внедрение в проект облачной платформы Firebase**

Сайт разрабатывался с помощью текстового редактора – MS Visual Code.

Так как в данном проекте используется библиотека React, то всё приложение было разбито на логические компоненты. Все, получившиеся, компоненты были реализованы как функциональные.

В данном проекте используется облачный сервис Firebase, который позволяет использовать, в создаваемом приложении, находящиеся в облаке: базу данных, хранилище файлов, аутентификацию, хостинг и другие функции, поставляемые на полубесплатной основе.

Из перечисленных функций в программном продукте используется: аутентификация (Authentication), реализованная для трех различных сервисов: Google, Facebook, GitHub; хранилище (Storage), необходимое для хранения и получения доступа к загружаемым на сервер файлам; база данных (Firestore database), для хранения комнат и привязанных к ним сообщений; хостинг (Hosting), для доступа к приложению из любой точки мира.

В приведенном ниже скриншоте продемонстрирован код подключения приложения к Firebase (листинг 3.1).

|  |
| --- |
| const app = initializeApp({    apiKey: "AIhzSyAlw5PZH6Uzn7y7ks4Eo\_eHowyOiuJHejM",    authDomain: "reactcourseproject-a53e5.firebaseapp.com",    projectId: "reactcourseproject-a53e5",    storageBucket: "reactcourseproject-a53e5.appspot.com",    messagingSenderId: "630145701678",    appId: "1:630145701678:web:e796aa64g7ee4dbbdee7bf"  }); |

Листинг 3.1 – Подключение Firebase к приложению

Для создания аутентификации были использованы поставляемые промисы: OathProvider (Google), GithubAuthProvider (GitHub), FacebookAuthProvider (Facebook), а также были написаны функции для проведения авторизации (листинг 3.2).

|  |
| --- |
| //Login with github account      const loginGitHub = async () => {          const provider = new GithubAuthProvider();          await auth.signInWithPopup(auth.getAuth(), provider).then((result) => {              dispatch(LoginAction(result.user.uid));              user = result.user;          })          .catch((error) => {              //if not login (email exists in db with other method of login)              if (error.code === 'auth/account-exists-with-different-credential') {                  linkOtherAccounts(error);      }});      } |

Листинг 3.2 – Функция аутентификации

Пример использования базы данных Firebase, представлен в листинге 3.3.

|  |
| --- |
| const AddRoom = async () => {          const docRef = firestore.doc(firestore.collection(db, 'rooms'));          if (name.length < 3) return;          await firestore.setDoc(docRef, {              id: docRef.id,              uid: user.uid,              displayName: name,              password: password,              createdAt: firestore.serverTimestamp()          });          setName('');          setPassword('');      } |

Листинг 3.3 – Добавление строки в базу данных

Пример демонстрирует функцию отправки данных комнаты в облачную базу Firestore database.

## **3.2. Использование Redux и Redux Toolkit**

Для реализации темной и светлой темы, а также для проверки авторизации был использован Redux в связке с Redux ToolKit. Он может хранить и предоставлять всем компонентам информацию, хранящуюся у него, что очень облегчает разработку приложения.

Redux Toolkit же облегчает работу с Redux, путем упрощения написания множества функций и добавлением нового контента.

Пример, демонстрирующий использование Redux и Redux ToolKit, представлен в листинге 3.4.

|  |
| --- |
| const rootReducer = createReducer([], (builder) => {      builder          .addCase(LOGIN, (state = initialState, action) => {              return {...state, 'LOGIN': action.payload}          })          .addCase(LOGOUT, (state = initialState, action) => {              return {...state, 'LOGIN': action.payload}          })          .addCase(DIALOG\_OPEN, (state = initialState, action) => {              return {...state, 'DIALOG\_OPEN': action.payload}          })          .addCase(THEME, (state = initialState, action) => {              return {...state, 'THEME': action.payload}          })  }) |

Листинг 3.4 – использование Redux и Redux ToolKit

Также листинг, демонстрирующий использование Redux для смены темы (листинг 3.5).

|  |
| --- |
| const handleThemeChange =(e) => {      store.getState().THEME.payload === 'light' ? dispatch(ThemeChangeAction({payload: 'dark'}))      : dispatch(ThemeChangeAction({payload: 'light'}));      colorMode.toggleColorMode();      setAnchorEl(null);    } |

Листинг 3.5 – код функции изменения темы

## **3.3. Использование Material-UI**

Для реализации большей части внешнего вида ПП была использована бибилиотека Material-UI, предоставляющая готовые стилевые решения. Она включает в себя множество элементов, комбинирование которых позволяет создать уникальный и в тоже время красивый дизайн. Пример использования комбинации элементов данной библиотеки продемонстрирован на листинге 3.6.

|  |
| --- |
| <div className="login">              <Paper sx={{textAlign: "center",  width: 450,  minHeight: 300,  borderRadius: 15,  padding: "55px",  paddingTop: "100px"}}>                  <Typography variant="h3" sx={{fontFamily: "Poppins-Bold",  fontWeight: 600,  paddingBottom: "60px"  }}>SignUp using</Typography>                  <div style={{                      display: "flex",                      flexWrap: "wrap",                      justifyContent: "center",                      alignItems: "center"                  }}>                  <FacebookRoundedIcon className="AnimateIcons" sx={{width: 120, height: 120, color: "blue"}} onClick={loginFacebook}/>                  <GoogleIcon className="AnimateIcons" sx={{width: 120, height: 120, color: "red"}} onClick={loginGoogle}/>                  <GitHubIcon className="AnimateIcons" sx={{width: 120, height: 120, color: "black"}} onClick={loginGitHub}/>                  </div>              </Paper>          </div> |

Листинг 3.6 – использование Material-UI для создания страницы авторизации

Также посредством использования Material-UI была реализована смены темы приложения между темной и светлой для более комфортного времяпрепровождения в мессенджере. Код функций для смены темы приложения представлен на листинге 3.7.

|  |
| --- |
| const colorMode = React.useMemo(      () => ({        toggleColorMode: () => {          setCookie('theme', mode === 'light' ? 'dark' : 'light');          setMode((prevMode) => (prevMode === 'light' ? 'dark' : 'light'));        },      }),      [mode],    );    const theme = React.useMemo(      () =>        createTheme({          palette: {            mode,          },        }),      [mode],    ); |

Листинг 3.7 – использование Material-UI для создания страницы авторизации

**3.4. Вывод**

В данном разделе была продемонстрирована часть стека технологий, использованного при разработке программного продукта. Были продемонстрированы листинги кода приложения.

**4. Описание хода работы программного средства**

## **4.1. Руководство пользователя**

При попадании на веб-сайт пользователь видит перед собой страницу авторизации, которая содержит 3 кнопки, для авторизации из разных сервисов (рисунок 4.1).

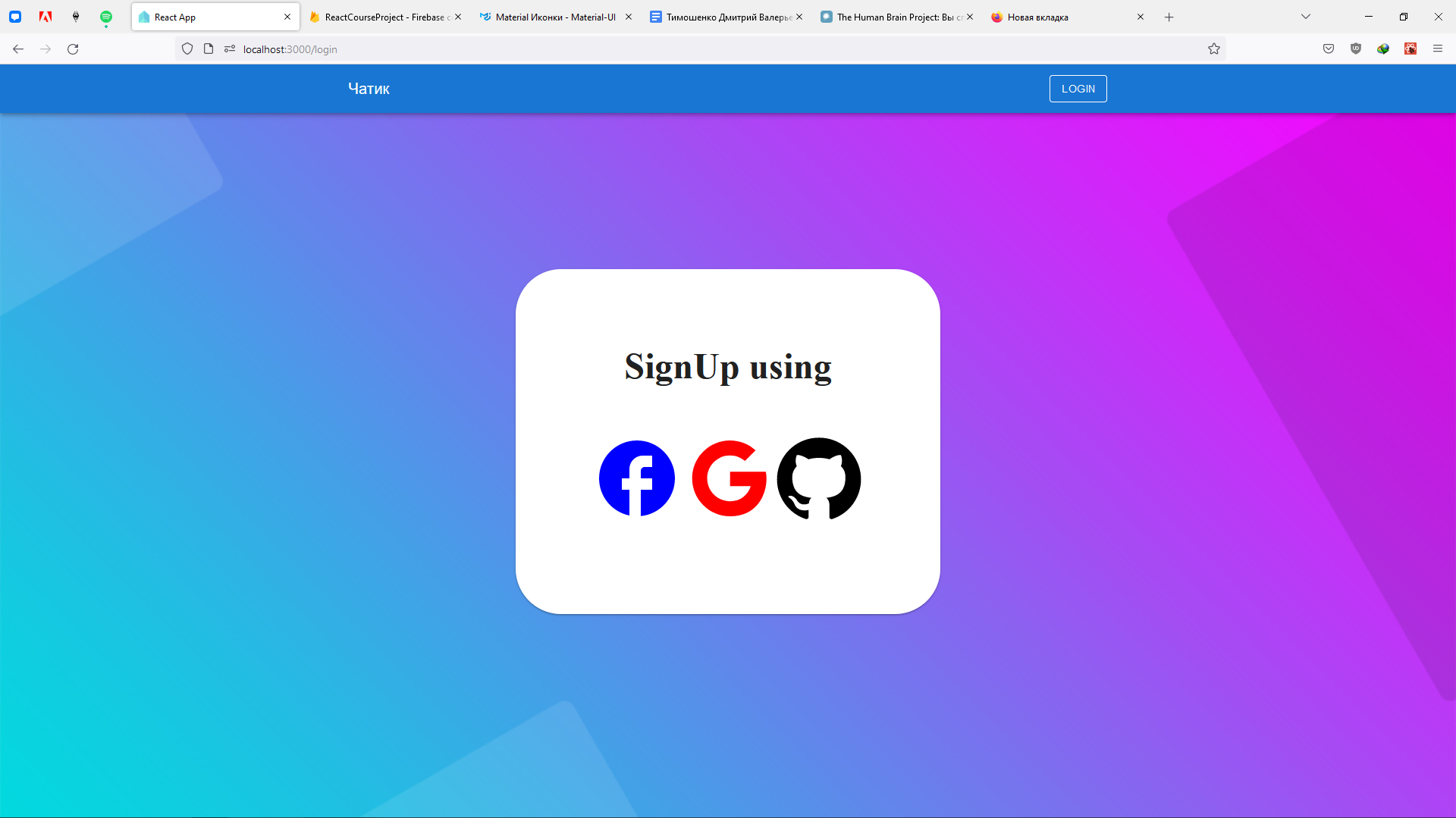


Рисунок 4.1 – Страница авторизации

Для авторизации пользователь может воспользоваться любым из предложенных сервисов.

После входа, посетителя перебросит на страницу выбора комнаты чата, содержащую карточки этих комнат. Также, благодаря имеющейся на этой странице форме, пользователь сможет создать свою комнату переписки (рисунок 4.2), созданные пользователем конаты могут быть удалены, путем нажатия на иконку корзины на карточке комнаты.

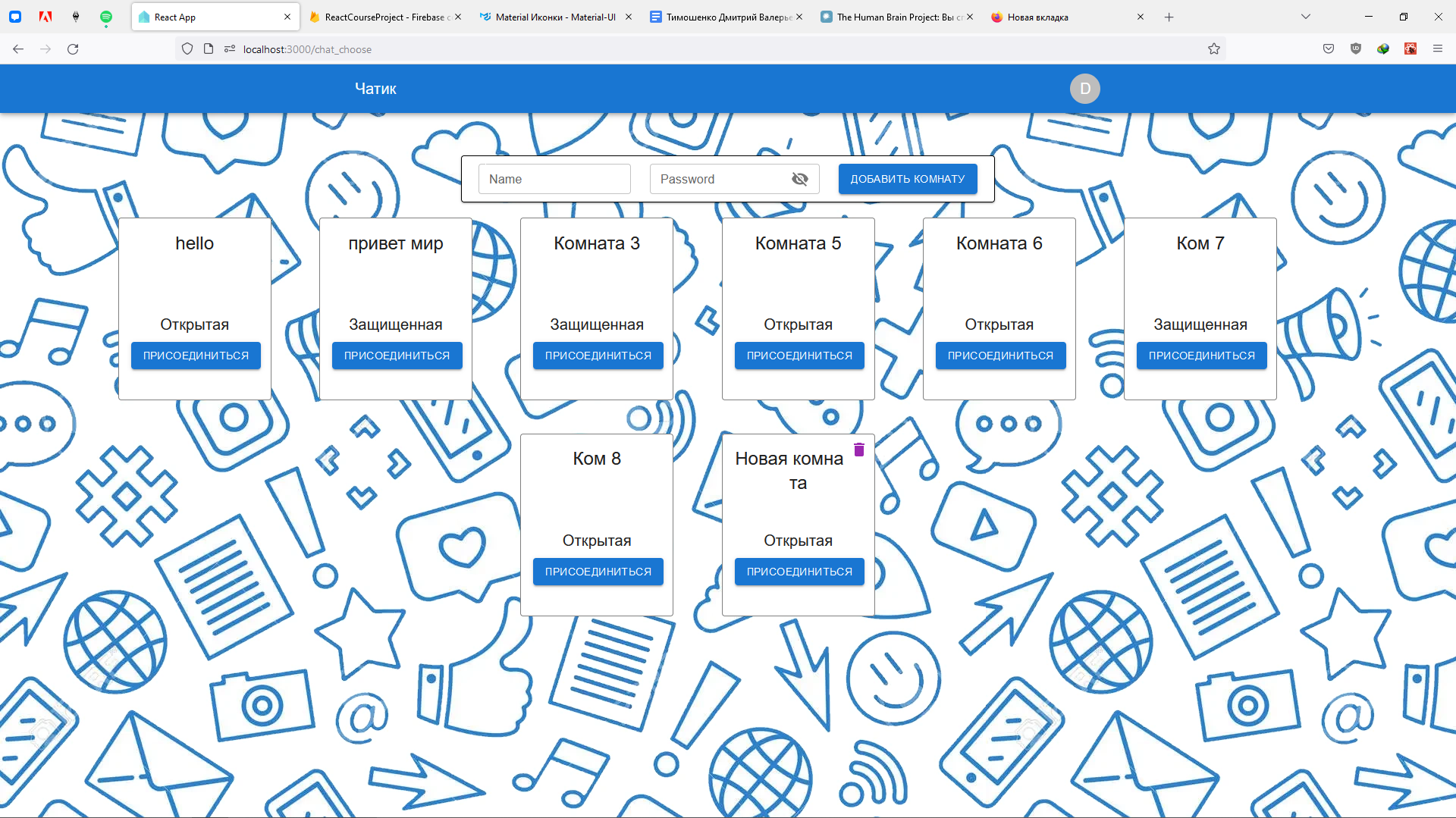


Рисунок 4.2 – Страница выбора чата

При выборе для входа защищенной комнаты всплывает окошко, просящее ввести пароль от данной комнаты (рисунок 4.3).

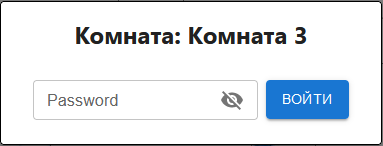


Рисунок 4.3 – Модальное окно для ввода пароля

После правильно введенного пароля, ползователь переходит на страницу самого чата, состоящую из окошка чата и формы ввода сообщений с возможностью прикрепления к сообщению файлов изображений (рисунок 4.4).



Рисунок 4.4 – Страница чата

У пользователя есть возможность поменять тему сайта между двумя видами: темной и светлой. Для этого необходимо кликнуть по своей аватарке, расположенной в навигационной панели сайта.

Откроется меню с двумя кнопками: «Settings» и «Logout». Кнопка «Logout» предназначена для выхода из системы. А кнопка «Settings» открывает меню настроек. В этом меню содержится настройка темы и информация о сайте (рисунок 4.5).

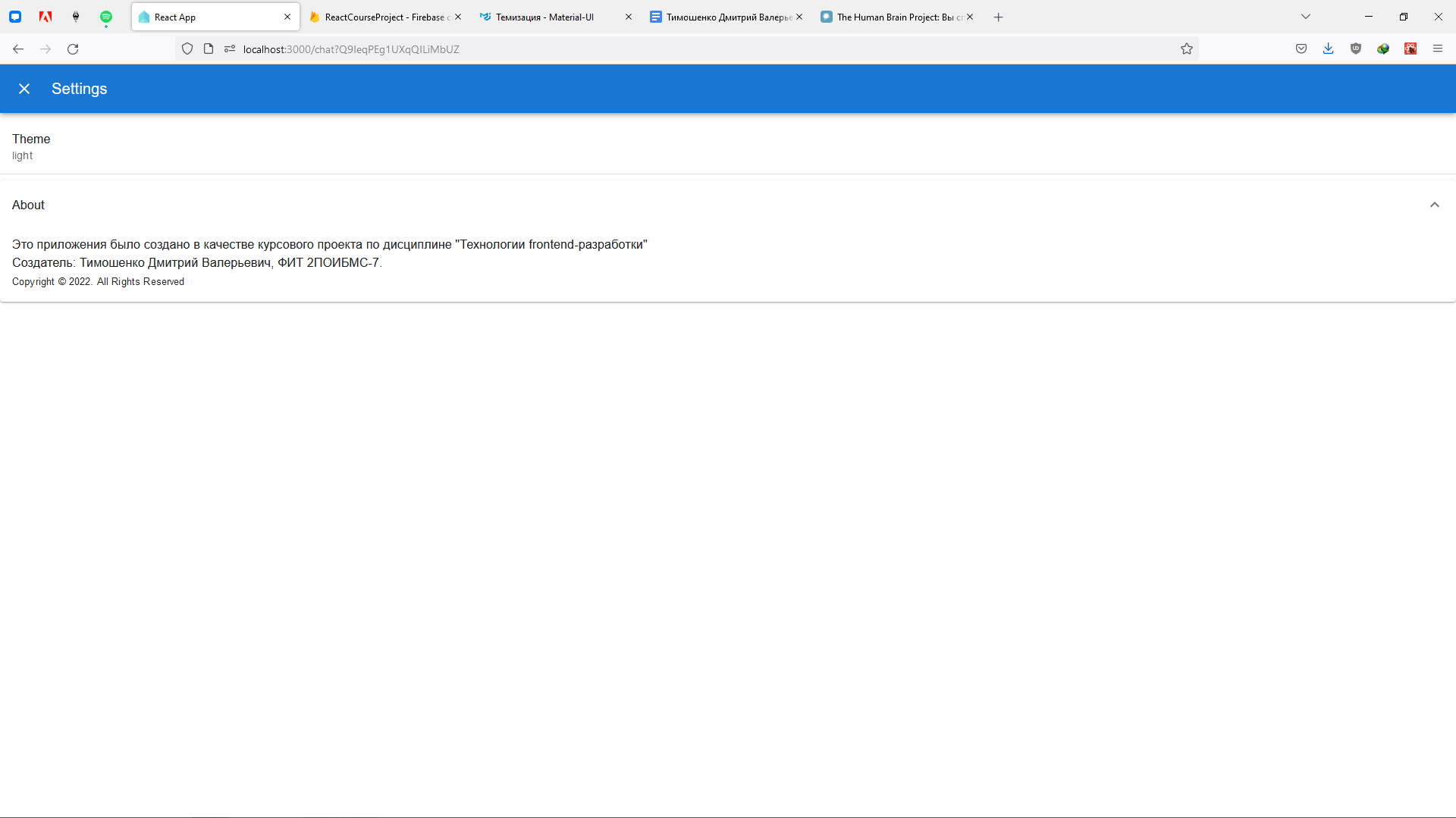


Рисунок 4.5 – Страница настроек

**4.2. Тестирование ПП**

Кроссбраузерность – это способность веб-ресурса отображаться одинаково и работать во всех популярных браузерах, без перебоев в функционировании и ошибок в верстке, а также с одинаково корректной читабельностью контента.

Для проверки, созданной кроссбраузерности веб-сайта, будем использовать такие современные браузеры, как Google Chrome, Microsoft Edge, Brave, Mozilla Firefox. Пример корректной работы веб-сайта представлен на рисунке 4.6.

|  |  |
| --- | --- |
| Google Chrome | Mozilla Firefox |
| Brave | Microsoft Edge |

Рисунок 4.6 – Веб-сайт в разных браузерах

**4.3. Вывод**

В данном разделе было описано подробное руководство пользования веб-приложением, которое позволит использовать програмный продукт в полной мере его возможностей, а также проведены кроссбраузерные тесты для проверки работоспособности приложения в разных браузерах, которые показали, что разработанный веб-сайт отлично совместим с современными браузерами.

# **Заключение**

Задача данной курсовой работы состояла в том, чтобы создать одностраничное веб-приложение по теме «системы обмена сообщений».

В ходе выполнения данной задачи были рассмотрены аналогичные решения. В процессе осмотра выяснилось, каких тенденции в разработке дизайна веб-сайта стоит придерживаться.

При формировании технического задания были продумано содержание основных страниц приложения.

Были выбраны инструменты для разработки веб-сайта.

При проектировании была разработана основная структура приложения, а далее продемонстрирован код некоторых функций веб-сайта.

Были разработаны пользовательские элементы и логотип, для идентификации веб-сайта, среди миллионов других интернет-ресурсов.

Для качественного отображения картинок, на любом устройстве, было решено применять SVG-изображения, также была создана база данных для хранения информации с веб-сайта.

Было создано подробное руководство пользователя для использования веб-сайта.

В ходе тестирования была продемонстрирована кроссбраузерность приложения.

Для разработки были использованы технологии, не изучаемые на данном курсе, такие, как Material-UI, включая библиотеку Redux ToolKit.

Как итог, было создано и размещено на хостинге кроссбраузерное веб-приложение, состоящее из множества уникальных компонентов, предназначенных для надежной работы веб-сайта, и использующий огромную часть изученного материала.

**Список использованных литературных источников**

1. Документация по Material-UI [Электронный ресурс] / Справочник Режим доступа:

https://v4.mui.com/ru/ – Дата доступа 20.10.2022.

1. Документация по HTML [Электронный ресурс] / Справочник Режим доступа:

https://developer.mozilla.org/ru/docs/Web/HTML/ – Дата доступа 05.10.2022.

1. Документация по CSS [Электронный ресурс] / Справочник Режим доступа:

https://developer.mozilla.org/ru/docs/Web/CSS/ – Дата доступа 08.10.2022.

1. Документация по React-Redux [Электронный ресурс] / Справочник Режим доступа:

https://react-redux.js.org/ – Дата доступа 28.10.2022.

1. Учебник по JavaScript [Электронный ресурс] / Справочник Режим доступа:

https://learn.javascript.ru/ – Дата доступа 24.10.2022.

1. Документация по JavaScript [Электронный ресурс] / Справочник Режим доступа:

https://developer.mozilla.org/ru/docs/Web/JavaScript – Дата доступа 26.10.2022.

1. Документация по React [Электронный ресурс] / Справочник Режим доступа:

https://ru.reactjs.org/ – Дата доступа 01.11.2022.

1. Документация по Redux ToolKit [Электронный ресурс] / Справочник Режим доступа:

https://redux-toolkit.js.org/ – Дата доступа 06.11.2022.

1. Статья по React-Redux [Электронный ресурс] / Справочник Режим доступа:

https://habr.com/ru/post/498860/ – Дата доступа 12.11.2022.

1. Документация по Firebase [Электронный ресурс] / Справочник Режим доступа:

https://firebase.google.com/docs – Дата доступа 18.11.2022.

1. Форум StackOverflow [Электронный ресурс] / Справочник Режим доступа:

https://ru.stackoverflow.com/ – Дата доступа 25.11.2022.

1. Документация по React Router DOM [Электронный ресурс] / Справочник Режим доступа:

https://v5.reactrouter.com/web/guides/ – Дата доступа 28.11.2022.

# **7. Приложение**

## **Приложение листинги компонентов**

|  |
| --- |
| import React, { createContext } from 'react';  import ReactDOM from 'react-dom/client';  import './index.css';  import App from './App';  import { initializeApp } from 'firebase/app';  import {    getAuth,    onAuthStateChanged,    GoogleAuthProvider,    signInWithPopup,    signOut,    fetchSignInMethodsForEmail  } from 'firebase/auth';  import {    getFirestore,    collection,    addDoc,    deleteDoc,    query,    orderBy,    limit,    onSnapshot,    setDoc,    updateDoc,    doc,    serverTimestamp,  } from 'firebase/firestore';  import {    getStorage,    ref,    uploadBytesResumable,    getDownloadURL,  } from 'firebase/storage';  import {Provider} from 'react-redux';  import {store} from './../src/store/store';  const app = initializeApp({    apiKey: "AIhzSyAlw5PZH6Uzn7y7ks4Eo\_eHowyOiuJHejM",    authDomain: "reactcourseproject-a53e5.firebaseapp.com",    projectId: "reactcourseproject-a53e5",    storageBucket: "reactcourseproject-a53e5.appspot.com",    messagingSenderId: "630145701678",    appId: "1:630145701678:web:e796aa64g7ee4dbbdee7bf"  });  const auth = {getAuth,    onAuthStateChanged,    GoogleAuthProvider,    signInWithPopup,    signOut,    fetchSignInMethodsForEmail};  const firestore = {    getFirestore,    collection,    addDoc,    deleteDoc,    query,    orderBy,    limit,    onSnapshot,    setDoc,    updateDoc,    doc,    serverTimestamp,  };  const fStorage = {    getStorage,    ref,    uploadBytesResumable,    getDownloadURL,  }  const db = getFirestore(app);  export const Context = createContext(null);  const root = ReactDOM.createRoot(document.getElementById('root'));  root.render(    <React.StrictMode>    <Context.Provider value={{      firestore,      db,      auth,      fStorage,    }}>    <Provider store={store}>        <App/>        </Provider>    </Context.Provider>    </React.StrictMode>  ); |

Листинг index.js

|  |
| --- |
| import './App.css';  import React from 'react';  import {BrowserRouter} from 'react-router-dom';  import Navbar from './components/Navbar';  import AppRouters from './components/AppRouters';  import Loader from './components/Loader';  import { useAuthState } from 'react-firebase-hooks/auth';  import { useContext } from 'react';  import { Context } from './index';  import { ThemeProvider } from '@mui/material/styles';  import { createTheme } from '@mui/material/styles';  import { store } from './store/store';  import {getCookie, setCookie } from './components/FuncForCookie';  export const ColorModeContext = React.createContext({ toggleColorMode: () => {} });  function App() {    const {auth} = useContext(Context);    const [user, loading] = useAuthState(auth.getAuth());    if (!getCookie('theme')) setCookie('theme', 'light');    const [mode, setMode] = React.useState(getCookie('theme') === 'dark' ? 'dark' : 'light');    const colorMode = React.useMemo(      () => ({        toggleColorMode: () => {          setCookie('theme', mode === 'light' ? 'dark' : 'light');          setMode((prevMode) => (prevMode === 'light' ? 'dark' : 'light'));        },      }),      [mode],    );    const theme = React.useMemo(      () =>        createTheme({          palette: {            mode,          },        }),      [mode],    );    if (loading) {      return <Loader/>  }    if (store.getState().THEME.payload === 'dark')      document.body.style.backgroundImage = "url('images/dark.webp')";    else document.body.style.backgroundImage = "url('images/light.webp')";    return (      <ColorModeContext.Provider value={colorMode}>      <ThemeProvider theme={theme}>        <BrowserRouter>          <Navbar/>          <AppRouters/>        </BrowserRouter>      </ThemeProvider>      </ColorModeContext.Provider>    );  }  export default App; |

Листинг App.js

|  |
| --- |
| import {configureStore} from '@reduxjs/toolkit';  import { LOGIN } from './actions/LoginActions';  import { initialState } from './initialState';  import rootReducer from './reducers/rootReducer';  export const store = configureStore({      reducer: rootReducer,      preloadedState: initialState,      middleware: (getDefaultMiddleware) =>      getDefaultMiddleware({        serializableCheck: {          // Ignore Login action types          ignoredActions: [LOGIN]        }})      }); |

Листинг store.js

|  |
| --- |
| import { createReducer } from '@reduxjs/toolkit';  import { DIALOG\_OPEN, THEME } from '../actions/SettingsAction';  import { initialState } from '../initialState';  import {LOGIN, LOGOUT} from './../actions/LoginActions';  const rootReducer = createReducer([], (builder) => {      builder          .addCase(LOGIN, (state = initialState, action) => {              return {...state, 'LOGIN': action.payload}          })          .addCase(LOGOUT, (state = initialState, action) => {              return {...state, 'LOGIN': action.payload}          })          .addCase(DIALOG\_OPEN, (state = initialState, action) => {              return {...state, 'DIALOG\_OPEN': action.payload}          })          .addCase(THEME, (state = initialState, action) => {              return {...state, 'THEME': action.payload}          })  })  export default rootReducer; |

Листинг rootReducer.js

|  |
| --- |
| import { CHAT\_ROUTE, CHATCHOOSE\_ROUTE, LOGIN\_ROUTE } from "./constants";  import Login from "./Login";  import Chat from "./Chat";  import ChatChooser from "./ChatChooser";  import NotFound from "./NotFound";  export const publicRoutes = [      {          path: LOGIN\_ROUTE,          component: <Login/>      },      {          path: "\*",          component: <NotFound/>      }  ];  export const privateRoutes = [      {          path: CHAT\_ROUTE,          component: <Chat/>      },      {          path: CHATCHOOSE\_ROUTE,          component: <ChatChooser/>      },      {          path: "\*",          component: <NotFound/>      }  ] |

Листинг routers.js

|  |
| --- |
| import React from 'react';  import AppBar from '@mui/material/AppBar';  import Box from '@mui/material/Box';  import Toolbar from '@mui/material/Toolbar';  import Typography from '@mui/material/Typography';  import Button from '@mui/material/Button';  import AvatarUser from './AvatarUser';  import { NavLink } from 'react-router-dom';  import { CHATCHOOSE\_ROUTE, LOGIN\_ROUTE } from './constants';  import { useContext } from 'react';  import { Context } from '../index';  import { useHistory } from 'react-router-dom';  import { useAuthState } from 'react-firebase-hooks/auth';  const Navbar = () => {    const {auth} = useContext(Context);    const history = useHistory();    const [user] = useAuthState(auth.getAuth());      return (      <Box sx={{ flexGrow: 1, justifyContent: 'space-around', position: 'fixed', width: '100%', zIndex: 999}}>        <AppBar position="static">          <Toolbar sx={{             justifyContent: 'space-around',             paddingLeft: 5,             }}>            <Typography variant="h6" component="div" sx={{              flexGrow: 0, cursor: 'pointer'              }} onClick={() => history.push(CHATCHOOSE\_ROUTE)}>              Чатик            </Typography>            {user ? <AvatarUser/> : <NavLink to={LOGIN\_ROUTE} style={{textDecoration: "none"}}><Button variant='outlined' color='inherit' sx={{ color: "white"}}>Login</Button></NavLink>}          </Toolbar>        </AppBar>      </Box>      );  }  export default Navbar; |

Листинг Navbar.jsx

|  |
| --- |
| import React from 'react';  import { useContext } from 'react';  import { Context } from '../index';  import { useAuthState } from 'react-firebase-hooks/auth';  import { Paper, Typography } from '@mui/material';  import FacebookRoundedIcon from '@mui/icons-material/FacebookRounded';  import GoogleIcon from '@mui/icons-material/Google';  import GitHubIcon from '@mui/icons-material/GitHub';  import './Login.css';  import { FacebookAuthProvider, GithubAuthProvider, linkWithCredential, OAuthProvider } from 'firebase/auth';  import { LoginAction } from '../store/actionCreators/LoginActionCreator';  import {useDispatch} from 'react-redux';  const Login = () => {      const dispatch = useDispatch();      const {auth} = useContext(Context);      let [user] = useAuthState(auth.getAuth());      //link credential from others auths      const linkCredentialFunc = (result, email, pendingCred) => {          //check identity of emails          if (result.user.email === email)          //link credential to exist login          return linkWithCredential(result.user, pendingCred).then().catch(e => console.log(e));          else {              alert("Неправильный адрес почты");              return;          }      }      //linking other accounts to auth      const linkOtherAccounts = (error) => {          //get credential and mail          const pendingCred = OAuthProvider.credentialFromError(error);          var email = error.customData.email;          //get methods of signIn          auth.fetchSignInMethodsForEmail(auth.getAuth(), email).then((methods) => {              alert("К этой почте уже привязана учетная запись. Требуется авторизация с основной учетной записи.")              // searching methods of auth              if (methods[0] === "github.com") {                  //signIn with google                  auth.signInWithPopup(auth.getAuth(), new auth.GithubAuthProvider()).then((result) => {                      // linking credential                     linkCredentialFunc(result, email, pendingCred);                  })              }              else if (methods[0] === "facebook.com") {                   //signIn with facebook                   auth.signInWithPopup(auth.getAuth(), new auth.FacebookAuthProvider()).then((result) => {                      //linking credential                      linkCredentialFunc(result, email, pendingCred);                  })              }              else if (methods[0] === "google.com") {                  //signIn with google                  auth.signInWithPopup(auth.getAuth(), new auth.GoogleAuthProvider()).then((result) => {                      // linking credential                     linkCredentialFunc(result, email, pendingCred);                  })              }              return;          })          .catch((error) => {              console.log(error);          })      }      //Login with google account      const loginGoogle = async () => {          const provider = new auth.GoogleAuthProvider();          await auth.signInWithPopup(auth.getAuth(), provider).then((result) => {              dispatch(LoginAction(result.user.uid));              user = result.user;            })          .catch((error) => {              //if not login (email exists in db with other method of login)              if (error.code === 'auth/account-exists-with-different-credential') {                  linkOtherAccounts(error);              }});      }      //Login with facebook account      const loginFacebook = async () => {          const provider = new FacebookAuthProvider();          await auth.signInWithPopup(auth.getAuth(), provider).then((result) => {              dispatch(LoginAction(result.user.uid));              user = result.user;          })          .catch((error) => {              //if not login (email exists in db with other method of login)              if (error.code === 'auth/account-exists-with-different-credential') {                 linkOtherAccounts(error);      }});      }      //Login with github account      const loginGitHub = async () => {          const provider = new GithubAuthProvider();          await auth.signInWithPopup(auth.getAuth(), provider).then((result) => {              dispatch(LoginAction(result.user.uid));              user = result.user;          })          .catch((error) => {              //if not login (email exists in db with other method of login)              if (error.code === 'auth/account-exists-with-different-credential') {                  linkOtherAccounts(error);      }});      }      return (          <div className="login">              <Paper sx={{textAlign: "center", width: 450, minHeight: 300, borderRadius: 15, padding: "55px", paddingTop: "100px"}}>                  <Typography variant="h3" sx={{fontFamily: "Poppins-Bold", fontWeight: 600, paddingBottom: "60px"}}>SignUp using</Typography>                  <div style={{                      display: "flex",                      flexWrap: "wrap",                      justifyContent: "center",                      alignItems: "center"                  }}>                  <FacebookRoundedIcon className="AnimateIcons" sx={{width: 120, height: 120, color: "blue"}} onClick={loginFacebook}/>                  <GoogleIcon className="AnimateIcons" sx={{width: 120, height: 120, color: "red"}} onClick={loginGoogle}/>                  <GitHubIcon className="AnimateIcons" sx={{width: 120, height: 120, color: "black"}} onClick={loginGitHub}/>                  </div>              </Paper>          </div>      );  }  export default Login; |

Листинг Login.jsx

|  |
| --- |
| import React from 'react';  import './Loader.css';  import {Paper} from '@mui/material';  import { Triangle } from 'react-loader-spinner';  const Loader = () => {      return (          <div className='login loader'>              <Paper sx={{textAlign: "center", width: 450, minHeight: 300, borderRadius: 15, padding: "55px", paddingTop: "100px"}}>                  <div style={{                      display: "flex",                      flexWrap: "wrap",                      justifyContent: "center",                      alignItems: "center"                  }}>                  <Triangle                      height="250"                      width="250"                      color="#4fa94d"                      ariaLabel="triangle-loading"                      wrapperStyle={{}}                      wrapperClassName=""                      visible={true}                      />                  </div>              </Paper>          </div>      );  }  export default Loader; |

Листинг Loader.jsx

|  |
| --- |
| export const getCookie = (name) => {      let matches = document.cookie.match(new RegExp(          //eslint-disable-next-line        "(?:^|; )" + name.replace(/([\.$?\*|{}\(\)\[\]\\\/\+^])/g, '\\$1') + "=([^;]\*)"      ));      return matches ? decodeURIComponent(matches[1]) : undefined;    }  export const setCookie = (name, value, options = {}) => {      options = {        path: '/',        SameSite: 'None',        ...options      };      if (options.expires instanceof Date) {        options.expires = options.expires.toUTCString();      }      let updatedCookie = encodeURIComponent(name) + "=" + encodeURIComponent(value);      for (let optionKey in options) {        updatedCookie += "; " + optionKey;        let optionValue = options[optionKey];        if (optionValue !== true) {          updatedCookie += "=" + optionValue;        }      }      document.cookie = updatedCookie;    }      export const deleteCookie = (name) => {      setCookie(name, "", {        'max-age': -1      })    } |

Листинг FuncForCookie.js

|  |
| --- |
| import \* as React from 'react';  import Dialog from '@mui/material/Dialog';  import ListItemText from '@mui/material/ListItemText';  import ListItem from '@mui/material/ListItem';  import List from '@mui/material/List';  import Divider from '@mui/material/Divider';  import AppBar from '@mui/material/AppBar';  import Toolbar from '@mui/material/Toolbar';  import IconButton from '@mui/material/IconButton';  import Typography from '@mui/material/Typography';  import CloseIcon from '@mui/icons-material/Close';  import Slide from '@mui/material/Slide';  import { DialogOpenAction, ThemeChangeAction } from '../store/actionCreators/SettingsActionCreator';  import { store } from '../store/store';  import { useDispatch } from 'react-redux';  import { ColorModeContext } from '../App';  import { Accordion, AccordionDetails, AccordionSummary, Icon } from '@mui/material';  import { ExpandMore } from '@mui/icons-material';  const Transition = React.forwardRef(function Transition(props, ref) {    return <Slide direction="up" ref={ref} {...props} />;  });  const DialogSettings = (props) => {    const [open, setOpen] = React.useState(store.getState().DIALOG\_OPEN);    const dispatch = useDispatch();    const colorMode = React.useContext(ColorModeContext);    const handleClose = () => {      props.dispatch(DialogOpenAction({payload: false}));      setOpen(false);    };    const handleThemeChange =(e) => {      store.getState().THEME.payload === 'light' ? dispatch(ThemeChangeAction({payload: 'dark'}))      : dispatch(ThemeChangeAction({payload: 'light'}));      colorMode.toggleColorMode();    }    return (      <div>        <Dialog          fullScreen          open={open}          onClose={handleClose}          TransitionComponent={Transition}        >          <AppBar sx={{ position: 'relative' }}>            <Toolbar>              <IconButton                edge="start"                color="inherit"                onClick={handleClose}                aria-label="close"              >                <CloseIcon />              </IconButton>              <Typography sx={{ ml: 2, flex: 1 }} variant="h6" component="div">                Settings              </Typography>            </Toolbar>          </AppBar>          <List>            <ListItem button            aria-controls={open ? 'positioned-menu' : undefined}          aria-haspopup="true"          aria-expanded={open ? 'true' : undefined}>              <ListItemText primary="Theme" secondary={store.getState().THEME.payload} onClick={handleThemeChange}/>            </ListItem>            <Divider />          </List>          <Accordion>            <AccordionSummary             expandIcon={<ExpandMore />}            aria-controls="about-content"            id="about-header">              <Typography>                About              </Typography>            </AccordionSummary>            <AccordionDetails>              <Typography>                Это приложения было создано в качестве курсового проекта по дисциплине "Технологии frontend-разработки"<br/>                Создатель: Тимошенко Дмитрий Валерьевич, ФИТ 2ПОИБМС-7.<br/>                <small>Copyright &copy; {new Date().getFullYear()}. All Rights Reserved</small>              </Typography>            </AccordionDetails>          </Accordion>        </Dialog>      </div>    );  }  export default DialogSettings; |

Листинг DialogSettings.jsx

|  |
| --- |
| import React from 'react';  import { useContext, useState } from 'react';  import { Context } from '../index';  import { useAuthState} from 'react-firebase-hooks/auth';  import { useCollectionData } from 'react-firebase-hooks/firestore';  import { Backdrop, Button, Grid, TextField, Card, CardContent, Typography, Modal, Fade, InputAdornment, IconButton, Paper } from '@mui/material';  import { Container } from '@mui/system';  import { useHistory } from 'react-router-dom';  import { CHAT\_ROUTE } from './constants';  import { Delete, Visibility, VisibilityOff } from '@mui/icons-material';  import './ChatChooser.css';  const ChatChooser = () => {      const {auth, db, firestore} = useContext(Context);      const history = useHistory();      const [user] = useAuthState(auth.getAuth());      const [password, setPassword] = useState('');      const [passwordE, setPasswordE] = useState('');      const [errorPassE, seterrorPassE] = useState(false);      const [name, setName] = useState('');      const [showPassword, setShowPass] = useState(false);      const [open, setOpen] = useState([false, null]);      const [rooms, loading] = useCollectionData(          firestore.query(firestore.collection(db, 'rooms'), firestore.orderBy('createdAt'))      );      if (loading) {          return null;      }      const AddRoom = async () => {          const docRef = firestore.doc(firestore.collection(db, 'rooms'));          if (name.length < 3) return;          await firestore.setDoc(docRef, {              id: docRef.id,              uid: user.uid,              displayName: name,              password: password,              createdAt: firestore.serverTimestamp()          });          setName('');          setPassword('');      }      const handlerDeleteRoom = async (room) => {          await firestore.deleteDoc(firestore.doc(db, 'rooms', room.id));      }      const GoingToChat = (room) => {          if (room.password === '' || room.password === passwordE) history.push(CHAT\_ROUTE+'?'+room.id);          else if (open[0] === false) {              setOpen([true, room]);          }          else {              seterrorPassE(true);          }      }      return (          <Container maxWidth={"xl"}>          <Grid container          justifyContent={"center"}          sx={{ height: window.innerHeight - 50, marginTop: 0, alignContent: "flex-start"}}          spacing={10}          direction={"row"}          alignItems={"center"}>              <Grid item xs={6}              sx={{marginTop: 5}}              justifyContent="center"              alignItems="center">              <Paper style={{border: '1px solid black', padding: 10, display: 'flex', justifyContent: 'space-around'}} className="addRoom">                  <TextField label="Name"                  name="name"                  id="name-bas"                  type="text"                  size={"small"}                  inputProps={{maxLength: 16}}                  value={name}                  onChange={e => setName(e.target.value)}                  />                  <TextField label="Password"                  name="password"                  id="password-bas"                  type={!showPassword ? "password" : "text"}                  size={"small"}                  inputProps={{maxLength: 16}}                  value={password}                  onChange={e => setPassword(e.target.value)}                  InputProps={{                endAdornment: <InputAdornment position="end" style={{padding: 0, margin: 0}}>                  <IconButton style={{padding: 0, margin: 0}}                    aria-label="toggle password visibility"                    onClick={() => setShowPass(!showPassword)}                    onMouseDown={e => e.preventDefault()}                  >                    {showPassword ? <Visibility /> : <VisibilityOff />}                  </IconButton>                </InputAdornment>                }}                  />                  <Button variant="contained" color="primary" name='Name' onClick={AddRoom}>Добавить комнату</Button>              </Paper>              </Grid>              <br/>              <Grid container spacing={3}              justifyContent="center"              alignItems="center" >                  {rooms.map((room, index) =>                      <Grid item xs={2} key={index}                      className="Cardclass"                      justifyContent="center" sx={{marginTop: '20px'}}>                          <Card variant="outlined" style={{width:'200px', minHeight: 'calc(20vh + 40px)', height:'180px', border: '1px solid gray', margin: 'auto auto'}}>                          {user.uid === room.uid ? <IconButton arial-label="delete" color="secondary" sx={{float: 'right'}} onClick={() => handlerDeleteRoom(room)}><Delete/></IconButton> : null}                          <CardContent sx={{height: 'calc(16vh + 40px)'}}>                              <Typography gutterBottom sx={{wordBreak: 'break-all', minHeight: '50%'}} align={'center'} variant="h5" component="h2">                                  {room.displayName}                              </Typography>                              <Typography gutterBottom align={'center'} variant="h6" component="h5">                                  {room.password === '' ? 'Открытая' : 'Защищенная'}                              </Typography>                              <Button variant="contained" color="primary" onClick={() => GoingToChat(room)}>                                  Присоединиться                              </Button>                          </CardContent>                          </Card>                      </Grid>)}                </Grid>          </Grid>          <Modal          sx={{display: 'flex',          alignItems: 'center',          justifyContent: 'center',          zIndex: 10}}          aria-labelledby="transition-modal-title"          aria-describedby="transition-modal-description"          open={open[0]}          onClose={() => setOpen([false, null])}          closeAfterTransition          BackdropComponent={Backdrop}          BackdropProps={{            timeout: 500,          }}        >          <Fade in={open[0]}>            <Paper style={{padding: '16px 32px 24px', border: '1px solid black', zIndex:20}}>              <h2 id="transition-modal-title" style={{textAlign: 'center'}}>Комната: {open[1] ? open[1].displayName : null}</h2>              <div id="transition-modal-description" style={{display: 'flex', justifyContent: 'space-between', paddingTop: 30}}>              <form onSubmit={(e) => {e.preventDefault();GoingToChat(open[1])}}>              <TextField label="Password" error={errorPassE}                  placeholder='Password'                  name="Epassword"                  id="password-exist"                  type={!showPassword ? "password" : "text"}                  size={"small"}                  inputProps={{maxLength: 16}}                  value={passwordE}                  onChange={e => setPasswordE(e.target.value)}                  InputProps={{                endAdornment: <InputAdornment position="end" style={{padding: 0, margin: 0}}>                  <IconButton style={{padding: 0, margin: 0}}                    aria-label="toggle password visibility"                    onClick={() => setShowPass(!showPassword)}                    onMouseDown={e => e.preventDefault()}                  >                    {showPassword ? <Visibility /> : <VisibilityOff />}                  </IconButton>                </InputAdornment>              }}                  />                  <Button variant="contained" sx={{marginLeft:1, padding: "7px 16px"}} color="primary" onClick={() => GoingToChat(open[1])}>Войти</Button>                  </form>                  </div>            </Paper>          </Fade>        </Modal>          </Container>      );  }  export default ChatChooser; |

Листинг ChatChooser.jsx

|  |
| --- |
| import { Container, Grid, TextField, Button, Avatar, Paper } from '@mui/material';  import { useAuthState } from 'react-firebase-hooks/auth';  import { useCollectionData } from 'react-firebase-hooks/firestore';  import { useContext, useState } from 'react';  import { Context } from '../index';  import SendIcon from '@mui/icons-material/Send';  import React from 'react';  import { PhotoCamera } from '@mui/icons-material';  import IconButton from '@mui/material/IconButton';  import ModalImage from "react-modal-image";  import './Chat.css';  import NotFound from './NotFound';  //import { LOADING\_MESSAGE\_URL } from './constants';  const Chat = () => {      const [file, setFile] = useState([]);      const [messageRef, setMessageRef] = useState({});      const [bUpdMes, setbUpdMes] = useState(false);      const {auth, db, firestore, fStorage} = useContext(Context);      const [user] = useAuthState(auth.getAuth());      const [value, setValue] = useState('');      const url = document.location.search.slice(1);      try {      var [messages, loading] = useCollectionData(          firestore.query(firestore.collection(db, '/chats/', url, 'messages'), firestore.orderBy('createdAt'))      );}      catch {          var messages = 0;          var loading = 0;          return <NotFound/>      }      const EnterPress = (e) => {          if (e.keyCode === 13 && !e.shiftKey) {              setValue(value.trim());              sendMessage(file);          }      }      const UpdateMessage = (message) => {          if (bUpdMes === false) {               if (message.uid === user.uid) {                  setMessageRef(firestore.doc(db, '/chats/', url, 'messages', message.id));                  setValue(message.text);                  setbUpdMes(true);          }          }else {              if (value === "") {                  firestore.deleteDoc(messageRef);              }              else {              firestore.updateDoc(messageRef, {                  text: value,              })              }              setbUpdMes(false);              setValue('');          }      }      const SaveImages = async (docRef) => {          try {              let fileobj = file;              setFile([]);              let publicImageURL = "";          for (let i = 0; i < fileobj.length; i++) {              if (fileobj[i].size >= 10\*1024\*1024 || !fileobj[i].name.match(/(?:jpg|jpeg|png|webp)$/g))                  continue;              let filePath = `files/${user.uid}/${docRef.id}/${fileobj[i].name}`;              let newImageRef = fStorage.ref(fStorage.getStorage(), filePath);              let fileSnapShot;              await fStorage.uploadBytesResumable(newImageRef, fileobj[i]).then((snapshot) => {                  fileSnapShot = snapshot;              });              publicImageURL += await fStorage.getDownloadURL(newImageRef) + '\n';              await firestore.updateDoc(docRef, {                  imageURL: publicImageURL,                  storageUri: fileSnapShot.metadata.fullPath              })          }      }          catch (error) {              console.error('There was an error uploading a file to Cloud Storage:', error);          }      }      if (loading) {          return null;          //return <Loader/>      }      const sendMessage = async () => {          if (bUpdMes === true) {              UpdateMessage(value);              return;          }          if (value === '' || value === '\n') {              return;          }            const docRef = firestore.doc(firestore.collection(db, '/chats/', url, 'messages'));          await firestore.setDoc(docRef, {              id: docRef.id,              uid: user.uid,              displayName: user.displayName,              photoURL: user.photoURL,              imageURL: null,              text: value.trim(),              createdAt: firestore.serverTimestamp()          });          if (file.length !== 0)              SaveImages(docRef);          setValue('');      }  const Scrolling = (e) => {      e.currentTarget.scrollIntoView(false);  }      return (          <Container>              <Grid                  container                  justifyContent={"center"}                  sx={{height: window.innerHeight - 50, marginTop: 0, alignContent: "flex-start"}}>                  <Paper style={{width:'80%', height: (file.length > 0) ? '60vh': '70vh', border: '1px solid gray', overflowY: 'auto', marginTop: '120px'}} className="Chating">                  {messages.map((message, index) =>                  <div key={index} onLoad={e => Scrolling(e)} style={{                      margin: 10,                      border: user.uid === message.uid ? '2px solid green' : '2px solid red',                      marginLeft: user.uid === message.uid ? 'auto' : '10px',                      width: 'fit-content',                      maxWidth: '55%',                      wordBreak: 'break-word',                      padding: 5,                      borderRadius: "3px"                  }}>                      <Grid container sx={{                          alignItems: 'center'                      }}>                          <Avatar sx={{                              height: '100%',                              width: '40px',                              maxHeight: '40px',                              marginRight: '5px'                              }} src={message.photoURL.toString()}/>                          <div>{message.displayName}</div>                      </Grid>                      <div onClick={e => UpdateMessage(message)} style={{                          marginLeft: '45px',                      }}>{message.text}</div>                      {/\*  Printing files on page with check for empty \*/}                      <div>{message.imageURL ? (message.imageURL.split('\n').map((el, i) =>                         el !== "" ? <ModalImage key={i} small={el.toString()} medium={el.toString()} showRotate/> : null)) : null}</div>                      </div>                  )}                  </Paper>                  <Paper                      className='formInputPaper'                      sx={{width: '80%', margin: 3}}                      direction={'row'}                      alignitems={'flex-start'}                      >                      <IconButton color="primary" aria-label="upload picture" component="label" sx={{width: '5%'}} className='IconButtonform'>                          <input hidden accept="image/\*" type="file" onChange={e => (file.length < 9) ? setFile([...file, e.currentTarget.files[0]]) : null}/>                          <PhotoCamera/>                      </IconButton>                          <TextField multiline                          className='TextFieldform'                          maxRows={'3'}                          onKeyDown={(e) => EnterPress(e)}                          sx={{width: '88%'}}                          aligncontent={'flex-start'}                          variant={'outlined'}                          fullWidth                          value={value}                          onChange={(e) => setValue(e.target.value)}                          />                          <Button                          className='ButtonformInput'                          id='sendMess'                          onClick={() => sendMessage()}                          sx={{width: '2%', height: '100%', minWidth: '7%'}}                          variant={"contained"}                          ><SendIcon/></Button>                      </Paper>                      <Grid                      container                      alignItems={'start'}                      sx={{width: '80%'}}>                      {file.map((fil, index) => {                              return <Grid item key={index}                      sx={{width: '25%'}}                      justifyContent={'space-between'}                      >                      <ModalImage small={URL.createObjectURL(fil)} medium={URL.createObjectURL(fil)}/>                      </Grid>})}</Grid>              </Grid>          </Container>      );  }  export default Chat; |

Листинг Chat.jsx

|  |
| --- |
| import Avatar from '@mui/material/Avatar';  import Box from '@mui/material/Box';  import IconButton from '@mui/material/IconButton';  import Menu from '@mui/material/Menu';  import MenuItem from '@mui/material/MenuItem';  import Tooltip from '@mui/material/Tooltip';  import Typography from '@mui/material/Typography';  import React, { useContext } from 'react';  import { useAuthState } from 'react-firebase-hooks/auth';  import { useDispatch } from 'react-redux';  import { Context } from '../index';  import { DialogOpenAction } from '../store/actionCreators/SettingsActionCreator';  import DialogSettings from './DialogSettings';  import { store } from '../store/store';  const AvatarUser = () => {      const dispatch = useDispatch();      const [anchorElUser, setAnchorElUser] = React.useState(null);      const [opened, setOpened] = React.useState(store.getState().DIALOG\_OPEN.payload);      const subscribe = store.subscribe(() => {        setOpened(!opened);      });      const {auth} = useContext(Context);      const [user] = useAuthState(auth.getAuth());      const settings = ['Settings', 'Logout'];          const handleOpenUserMenu = (event) => {              setAnchorElUser(event.currentTarget);            };              const handleCloseUserMenu = (setting) => {              if (setting === "Settings") {                dispatch(DialogOpenAction({payload: true}));                }              if (setting === "Logout") {                subscribe();                auth.signOut(auth.getAuth()).then(                  ).catch((error) =>                console.log(error)                );              }              setAnchorElUser(null);            };      return (          <Box>          {opened ? <DialogSettings dispatch={dispatch}/> : null}              <Tooltip title="Open settings">                <IconButton onClick={handleOpenUserMenu} sx={{ p: 0 }}>                  <Avatar alt={user ? user.displayName : null} src={user ? user.photoURL : null } />                </IconButton>              </Tooltip>              <Menu                sx={{ mt: '45px' }}                id="menu-appbar"                anchorEl={anchorElUser}                anchorOrigin={{                  vertical: 'top',                  horizontal: 'right',                }}                keepMounted                transformOrigin={{                  vertical: 'top',                  horizontal: 'right',                }}                open={Boolean(anchorElUser)}                onClose={handleCloseUserMenu}              >                {settings.map((setting) => (                  <MenuItem key={setting} onClick={() => handleCloseUserMenu(setting)}>                    <Typography textAlign="center">                        {setting}                    </Typography>                  </MenuItem>                ))}              </Menu>              </Box>      );  }  export default AvatarUser; |

Листинг AvatarUser.jsx

|  |
| --- |
| import React from 'react';  import { useContext } from 'react';  import {Route, Switch, Redirect, useLocation} from "react-router-dom";  import { CHATCHOOSE\_ROUTE, LOGIN\_ROUTE } from './constants';  import { privateRoutes, publicRoutes } from './routes';  import { useAuthState } from 'react-firebase-hooks/auth';  import { Context } from '../index';  const AppRouters = () => {      const {auth} = useContext(Context);      const [user] = useAuthState(auth.getAuth());      const location = useLocation();      return user ?      (          <Switch>              {privateRoutes.map(({path, component}, i) =>                  <Route key={i} path={path} exact={false}>                      {location.pathname === "/login" ?                      (<Redirect to={CHATCHOOSE\_ROUTE}/>) :                      component}                  </Route>              )}            </Switch>      )      :      (          <Switch>              {publicRoutes.map(({path, component}, i) =>                  <Route key={i} path={path} exact={true}>                  {location.pathname.startsWith("/chat") || location.pathname === "/chat\_choose" ?                      (<Redirect to={LOGIN\_ROUTE}/>) :                      component}                  </Route>              )}          </Switch>      );  }  export default AppRouters; |

Листинг AppRouters.jsx

|  |
| --- |
| import React from 'react';  import './NotFound.css';  const NotFound = () => {      return (          <section className="not-found">          <div className="wrap">              <hgroup>                  <h1 className="not-found\_\_title">Упс, что-то пошло не так</h1>                  <h2 className="not-found\_\_subtitle">Давайте начнём сначала!</h2>              </hgroup>              <div className="not-found\_\_picture">                  404                  <img className="not-found\_\_image" src="https://roistat.com/rublog/wp-content/themes/roistat2021v2/assets2019/img/404.png" alt="404"/>              </div>            </div>      </section>      );  }  export default NotFound; |

Листинг NotFound.jsx