МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное

учреждение высшего образования

«Казанский национальный исследовательский технический

университет им. А.Н. Туполева-КАИ»

(КНИТУ-КАИ)

Институт компьютерных технологий и защиты информации

(наименование института (факультета), филиала)

\_\_Кафедра Прикладной Математики и Информатики \_\_

(наименование кафедры)

\_\_\_\_\_\_\_\_09.03.04 «Программная инженерия» \_\_\_

(шифр и наименование направления подготовки (специальности))

пояснительная записка

к Курсовой работе

по дисциплине:

«Технологии Веб-программирования»

на тему: «Клиент-серверное приложение “Учет личных расходов”»

Обучающийся 4310 Нигамадянов Ф.М.

(номер группы) (подпись, дата) (Ф.И.О.)

Руководитель доцент каф. ПМИ Тутубалин П.И.

(должность) (Ф.И.О.)

Курсовая работа зачтена с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

(подпись, дата)

Казань 2023

Оглавление

[1. ПОСТАНОВКА ЗАДАЧИ 4](#_Toc138022704)

[2. МОДЕЛЬ ДАННЫХ СИСТЕМЫ 5](#_Toc138022705)

[3. МАКЕТЫ HTML-СТРАНИЦ СИСТЕМЫ 6](#_Toc138022706)

[4. СТРУКТУРНАЯ СХЕМА МОДУЛЕЙ ПО, СОСТАВЛЯЮЩИХ СИСТЕМУ 12](#_Toc138022707)

[4.1. Общая архитектура приложения 12](#_Toc138022708)

[4.2. Структурная схема сервера 13](#_Toc138022709)

[4.3. Структурная схема клиента 14](#_Toc138022710)

[5. ДИАГРАММА КЛАССОВ РАЗРАБОТАННОЙ СИСТЕМЫ 16](#_Toc138022711)

[6. ОБЗОР ПО, ВЫБРАННОГО ДЛЯ РЕАЛИЗАЦИИ СИСТЕМЫ 17](#_Toc138022712)

[6.1. Среда разработки – Visual Studio Code 17](#_Toc138022713)

[6.2. Веб-браузер Google Chrome и инструменты разработчика 17](#_Toc138022715)

[6.3. Программная платформа Node.js 18](#_Toc138022716)

[6.4. СУБД MongoDB 18](#_Toc138022720)

[6.5. React 19](#_Toc138022721)

[6.5.1 React. Хуки. 21](#_Toc138022722)

[7. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 23](#_Toc138022723)

[7.1. Руководство для пользователя 23](#_Toc138022724)

[7.2. Руководство для администратора 28](#_Toc138022725)

[ИСПОЛЬЗОВАННЫЕ ИСТОЧНИКИ 32](#_Toc138022726)

[Приложение. 33](#_Toc138022727)

# 1. ПОСТАНОВКА ЗАДАЧИ

Целью курсовой работы является разработка и программная реализация клиент-серверного приложения, а также закрепления знаний, полученных при изучении дисциплины, и получение практических навыков программирования информационных систем с использованием современных технологий и инструментальных средств и оформления сопровождающей документации программного обеспечения.

Задача курсовой работы состоит в разработке клиент-серверного приложения «Учет личных расходов» (web-клиента и web-сервера). Для выполнения этой задачи, были определены следующие требования:

* произвольная среда разработки – удобный современный текстовый редактор;
* применение системы контроля версий – git;
* опора на представление «Model-View-Controller»;
* разработка web-клиента с применением html, css, JavaScript, JQuery;
* разработка web-сервера с применением JSON, AJAX, Node.js, Express и NPM;
* использование базы данных (БД) на стороне web-сервера с применением СУБД – MongoDB;
* разворачивание ПО web-сервера либо на отдельном хосте, либо в среде виртуальной машины VirtualBox.

Актуальность работы заключается в том, что на технологии клиент-серверного взаимодействия строится практически вся сеть Интернет. Технология позволяет передавать информацию от серверного устройства к устройствам-клиентам на расстоянии.

Объектом курсовой работы является создание клиент-серверного приложения «Учет личных расходов».

Предметом работы являются языки программирования для разработки приложений web-сервера и web-клиента.

# 2. МОДЕЛЬ ДАННЫХ СИСТЕМЫ

В данной работе используется уже ставшая классической модель Web-приложений на базе AJAX (см. рис. 1). Основная идея AJAX – возможность для приложения получать информацию и отдавать ее другим компьютерам без перезагрузки веб-страницы.

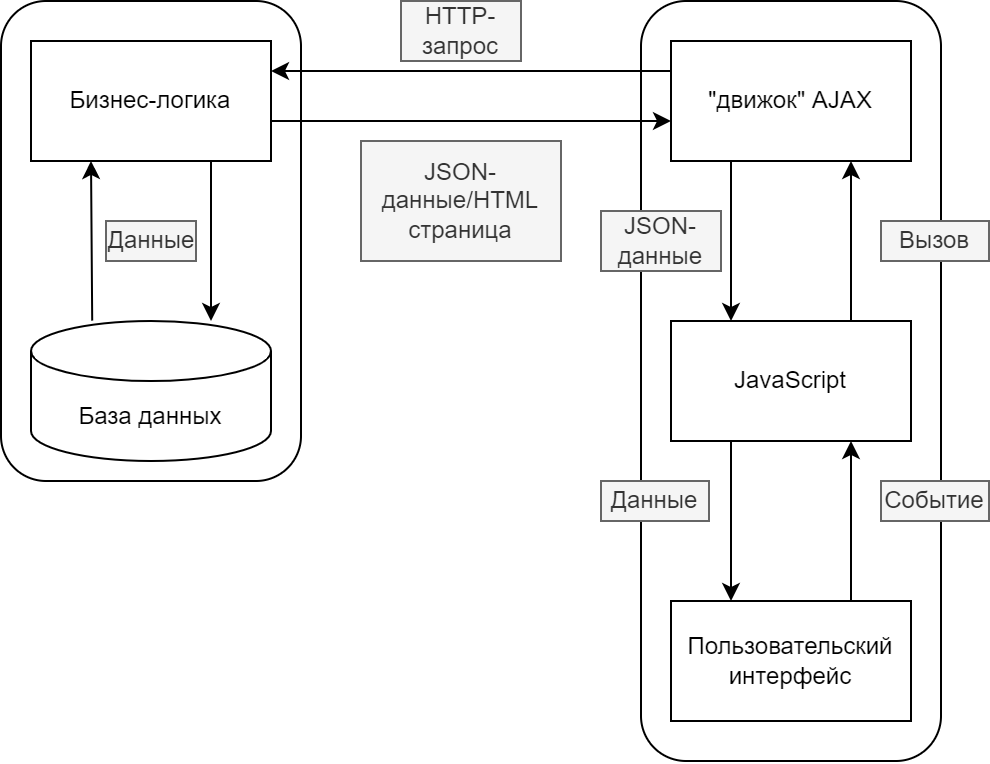


Рис. 2.1. Модель Web-приложений на базе AJAX

JavaScript в браузере с помощью объекта XMLHttpRequest (посредством объекта axios) подает запрос серверу по указанному пути, передавая по необходимости некоторые данные. Маршрутизатор на стороне сервера получает запрос, соответствующий пути, данные обрабатываются согласно бизнес-правилам, формулам и т.д.. После обработки сервер возвращает JavaScript данные или возвращает html страницу.

# 3. МАКЕТЫ HTML-СТРАНИЦ СИСТЕМЫ

3.1. Макет шапки домашнего окна:

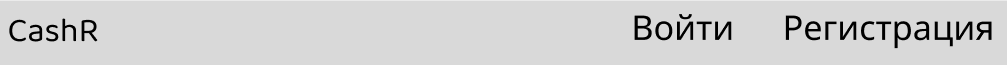


Рис. 3.1. Макет шапки домашнего окна

3.2. Макет домашнего окна

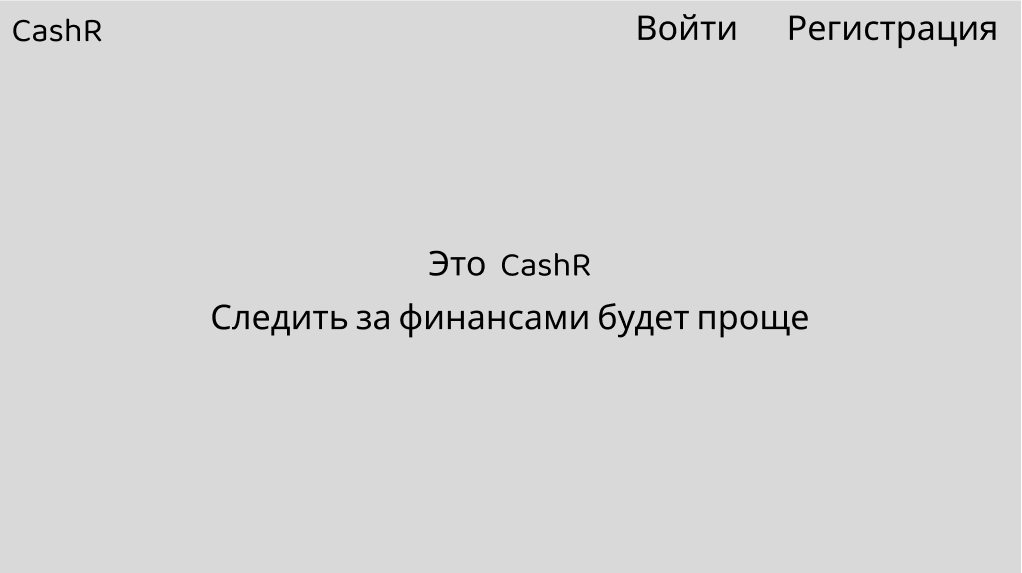


Рис. 3.2. Макет домашнего окна

3.3. Макеты окна авторизации

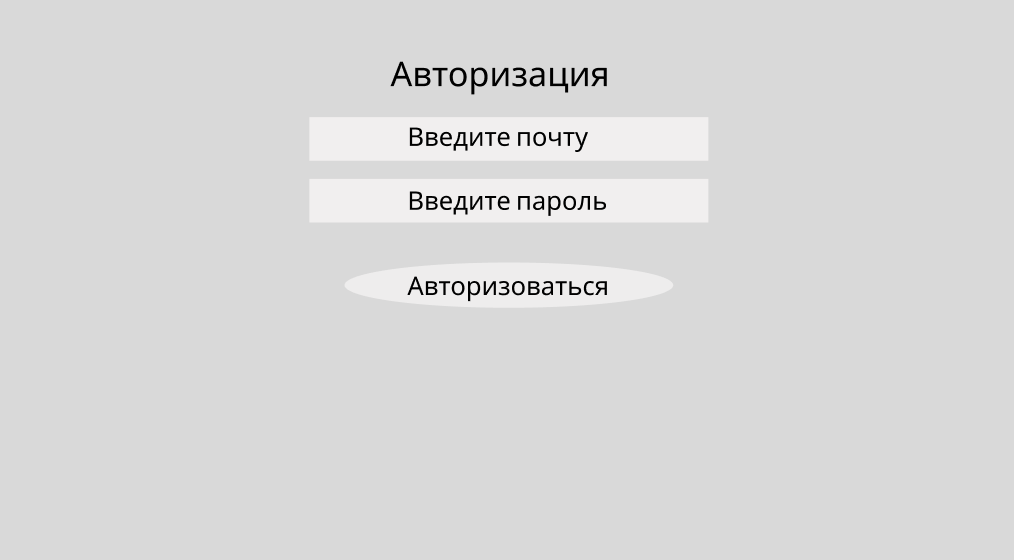


Рис. 3.3. Макет окна авторизации

3.4 Макет окна регистрации

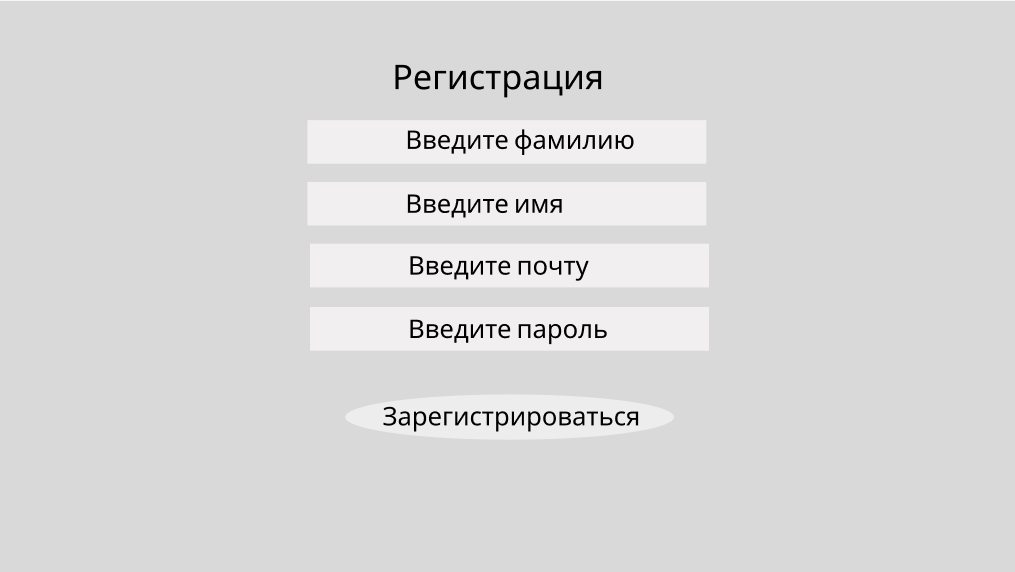


Рис. 3.4. Макет окна регистрации

3.5. Макет шапки администратора



Рис. 3.5. Макет шапки администратора

3.6. Макет основной страницы для администратора

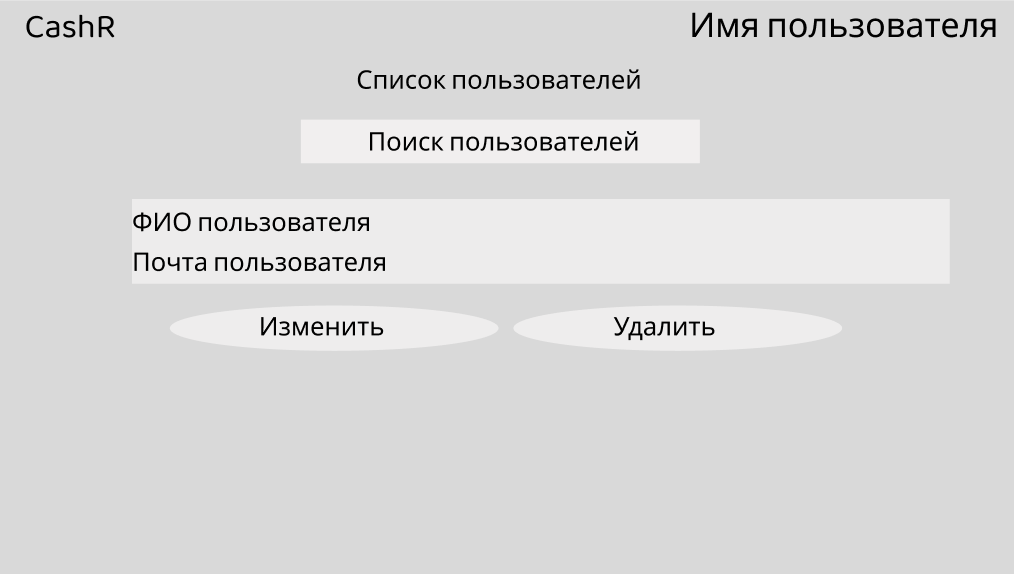


Рис. 3.6. Макет основной страницы для администратора

3.7. Макет боковой панели администратора



Риc. 3.7. Макет боковой панели администратора

3.8. Макет шапки пользователя



Рис. 3.8. Макет шапки пользователя

3.9. Макет окна транзакций пользователя.



Рис. 3.9. Макет окна транзакций пользователя

3.10. Макеты боковых панелей пользователя

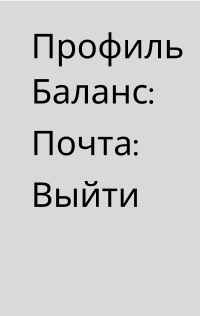
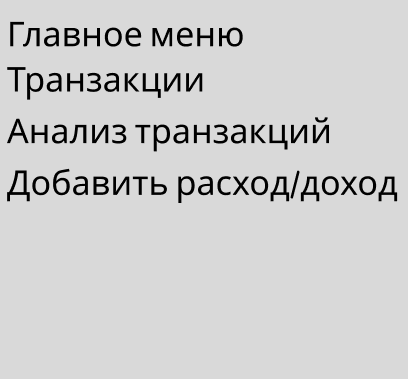
 

Рис. 3.10. Макеты боковых панелей пользователя

3.11. Макет окна фильтрации транзакций

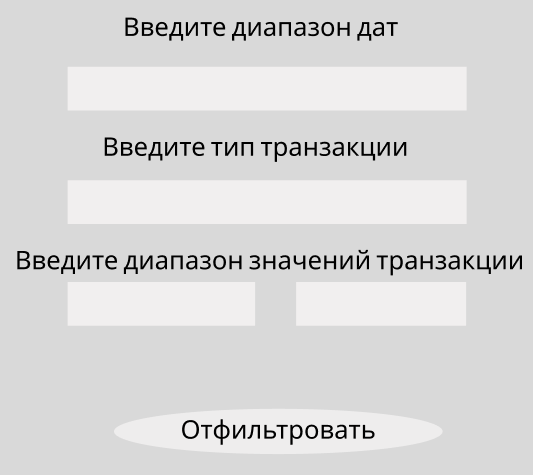


Рис. 3.11. Макет окна фильтрации транзакций

3.12. Макет окна добавления транзакции

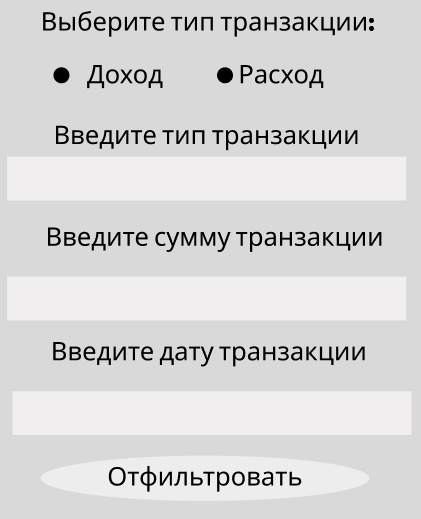


Рис. 3.12. Макет окна добавления транзакции

3.13. Макет окна анализа транзакций

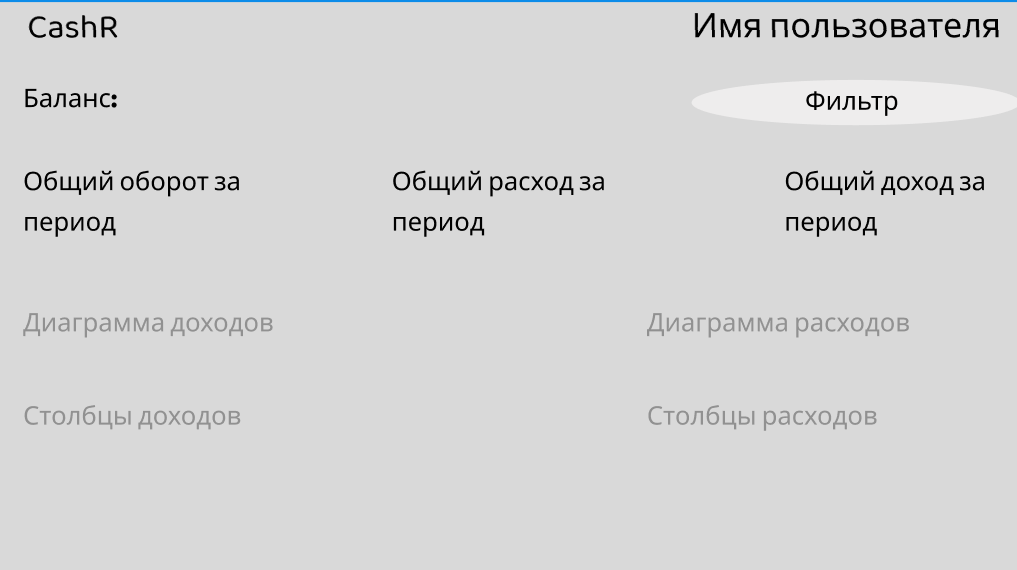


Рис. 3.13. Макет окна анализа транзакций

3.14. Макет окна выбора диапазона дат

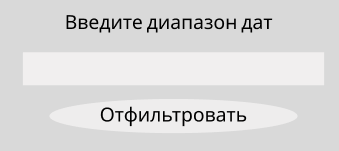


Рис. 3.14. Макет окна выбора диапазона дат

# 4. СТРУКТУРНАЯ СХЕМА МОДУЛЕЙ ПО, СОСТАВЛЯЮЩИХ СИСТЕМУ

## 4.1. Общая архитектура приложения

Система имеет 3-уровневую архитектуру, схема которой показана на рис.4.1.

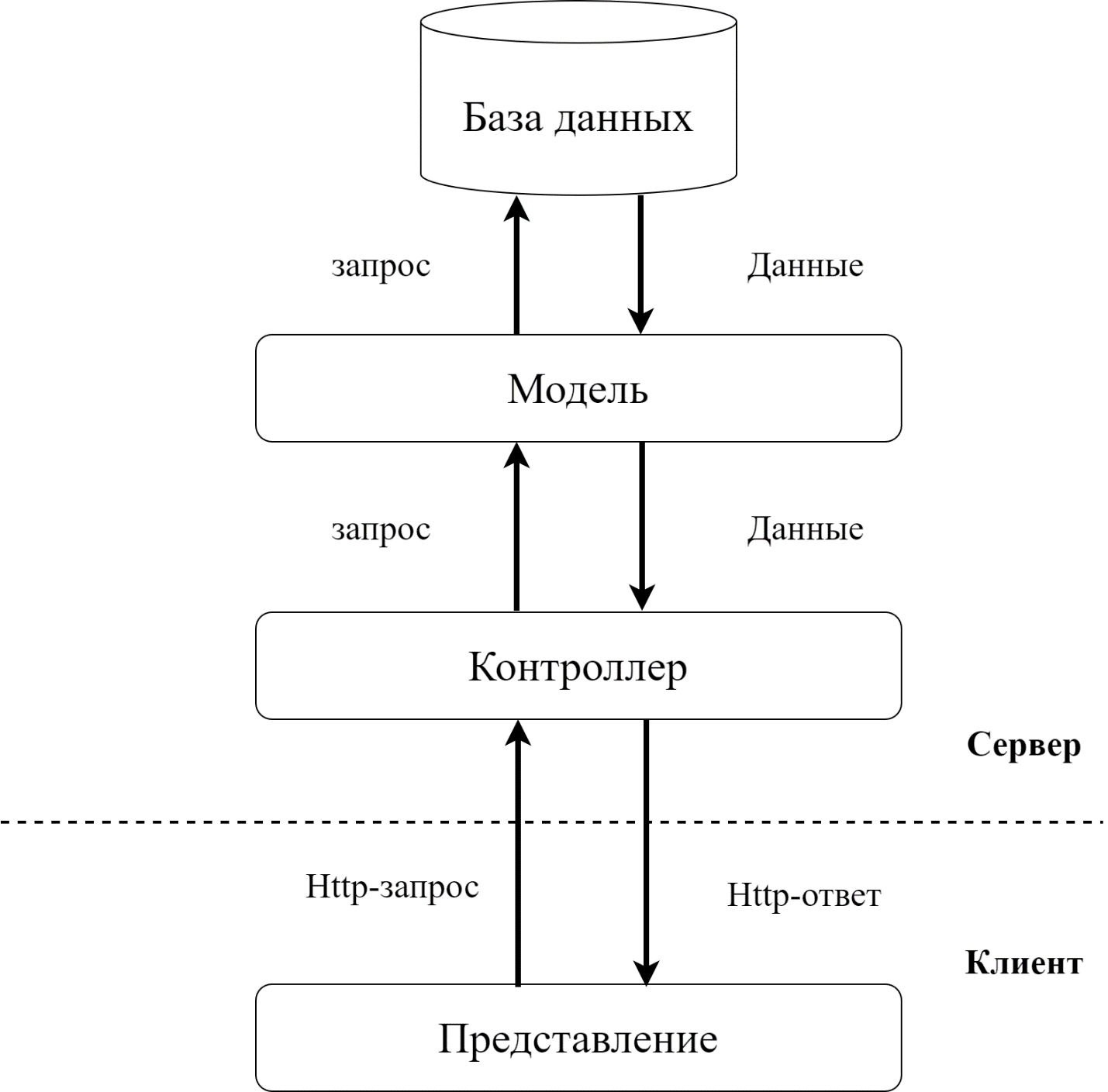


Рис. 4.1. Структурная схема приложения

Дадим определение уровням, изображенным на рисунке.

База данных (в данном случае – MongoDB) – это упорядоченный набор структурированной информации или данных, которые обычно хранятся в электронном виде в компьютерной системе.

Модель – это объектная абстракция элементов в базе данных, с ней взаимодействует контроллер.

Контроллер – преобразует запрос к серверу в действие.

Представление – в нашем случае это React-приложение на клиентской стороне. Отправляет http-запросы на сервер, получает ответ и решает, как его представить.

Как видно из рис. 4.2. Приложение можно разбить на две части: клиент и сервер. Рассмотрим их структуры подробнее

## 4.2. Структурная схема сервера

Структурная схема сервера представлена на рис. 4.2.

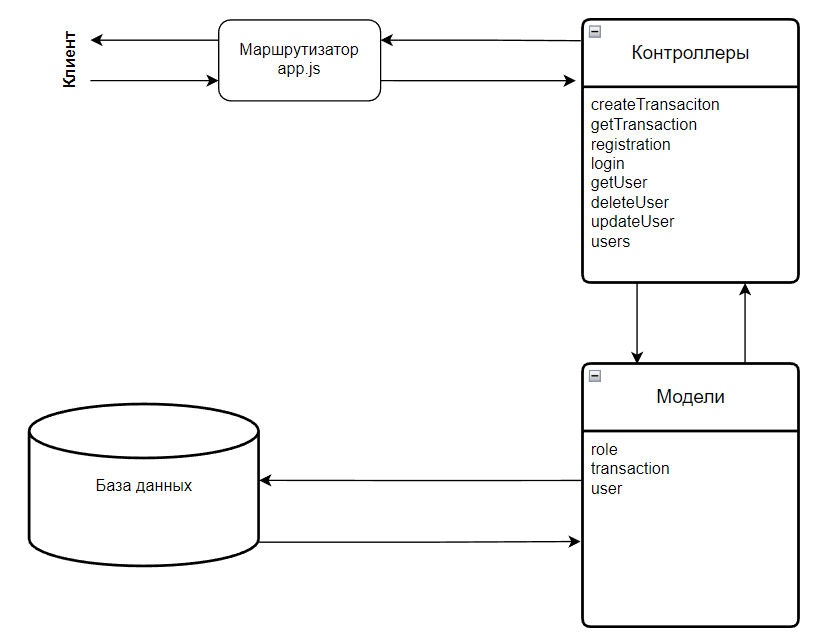


Рис. 4.2. Структурная схема сервера приложения

## 4.3. Структурная схема клиента

Работу клиента можно поделить на 2 части: часть для обыкновенных пользователей и часть для администратора. Структурная схема части для обыкновенных пользователей представлена на рис. 4.3 в виде компонент, отвечающих за страницы сайта.

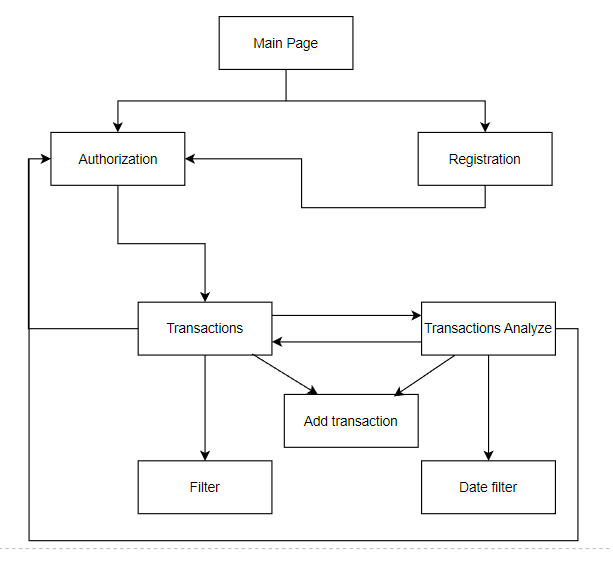


Рис. 4.3. Страницы доступные обыкновенным пользователям

Структурная схема для администратора представлена на рис. 4.4.

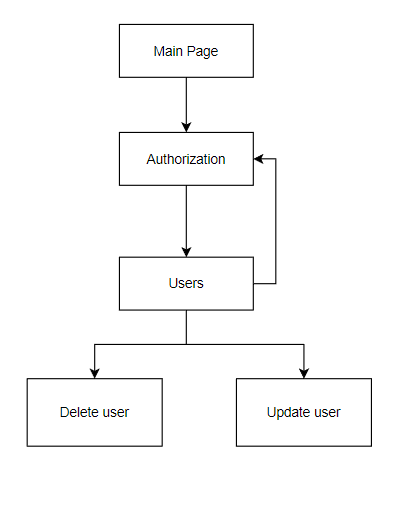


Рисунок 4.4. Страницы доступные администратору.

5. ДИАГРАММА КЛАССОВ РАЗРАБОТАННОЙ СИСТЕМЫ

Диаграмма классов разработанной системы изображена на рис. 5.1:

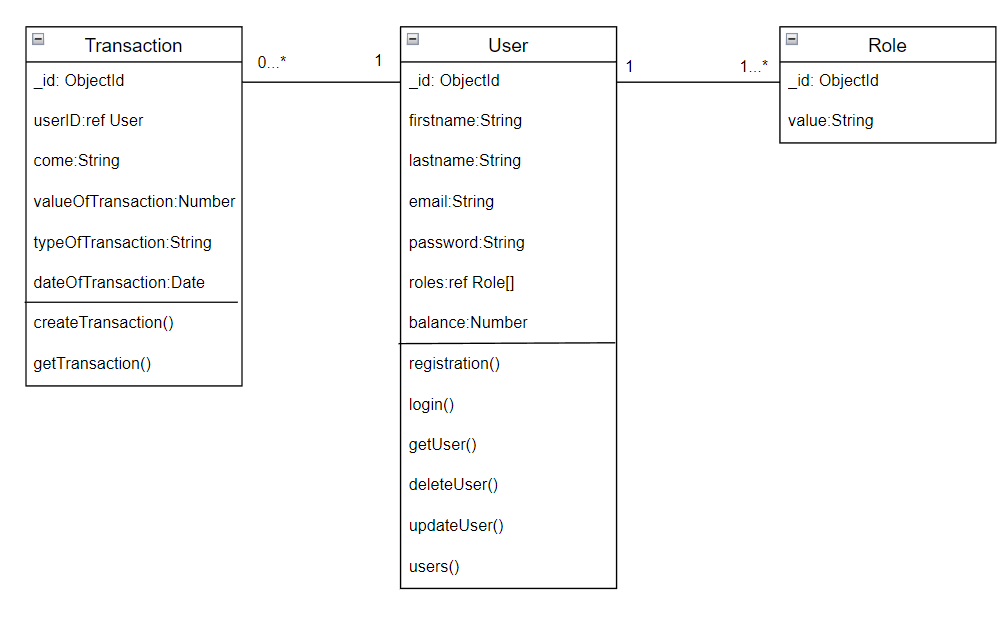


Рис. 5.1. Диаграмма классов разработанной системы

# 6. ОБЗОР ПО, ВЫБРАННОГО ДЛЯ РЕАЛИЗАЦИИ СИСТЕМЫ

## 6.1. Среда разработки – Visual Studio Code

# Visual Studio Code (VS Code) - это бесплатный редактор кода, разработанный компанией Microsoft, который может использоваться для различных задач, связанных с программированием, таких как написание кода, отладка, управление версиями, работа с Git и т.д. Он доступен для Windows, Mac и Linux и поддерживает множество языков программирования, фреймворков и сред разработки. VS Code также имеет множество плагинов и расширений, которые позволяют пользователю настраивать его под свои нужды и предпочтения. В целом, Visual Studio Code является мощным инструментом для разработчиков и программистов всех уровней опыта.

## 6.2. Веб-браузер Google Chrome и инструменты разработчика

Google Chrome - это веб-браузер, разработанный компанией Google, который используется для просмотра веб-страниц и веб-приложений. Он является одним из самых популярных веб-браузеров в мире и предлагает множество функций, таких как быстродействие, безопасность и простота использования.

Один из основных инструментов, присутствующих в Google Chrome, это инструменты разработчика, которые предназначены для содействия и отладки веб-страниц и веб-приложений. Они включают в себя следующие инструменты:

1. Инспектор элементов (Element Inspector) - инструмент, который позволяет узнать HTML-странцы и CSS-свойства на странице.

2. Консоль (Console) - инструмент, который используется для отладки JavaScript-кода и выполнения JavaScript-команд напрямую в браузере.

3. Сетевой анализатор (Network Analyzer) - инструмент, который позволяет анализировать сетевой трафик между браузером и сервером, загрузку файлов и многое другое.

4. Аудит страница (Page Audit) - инструмент, который позволяет анализировать производительность загруженной страницы, идентифицируя узкие места и предоставляя советы по оптимизации.

Инструменты разработчика Google Chrome являются незаменимым помощником в работе веб-разработчиков и дизайнеров.

## 6.3. Программная платформа Node.js

# Node.js - это среда выполнения на стороне сервера, которая позволяет запускать JavaScript-код вне браузера. Она основана на движке V8 JavaScript, который разработан компанией Google для браузера Google Chrome. Node.js позволяет разработчикам создавать серверные приложения на JavaScript, а также обеспечивает многопоточность, асинхронность и высокую производительность.

# Node.js используется для разработки разнообразных приложений, таких как веб-серверы, микросервисы, API, средства управления зависимостями и многое другое. Она также является чрезвычайно популярным инструментом среди разработчиков JavaScript и используется многими крупными компаниями, включая Netflix, Uber, PayPal и др.

# Основные преимущества Node.js включают быстродействие, масштабируемость, высокую производительность и большое количество плагинов и библиотек, доступных для использования. Node.js обеспечивает удобный способ написания серверного кода и управления им, и может быть установлен на различных платформах, включая Windows, Mac OS и Linux.

## 6.4. СУБД MongoDB

MongoDB - это одна из наиболее популярных нереляционных баз данных (NoSQL), которая использует документоориентированную модель данных. MongoDB позволяет хранить и обрабатывать данные в виде документов, которые могут содержать любые типы данных, включая вложенные документы и массивы. Написана на языке C++. Используется в веб-разработке, в частности, в рамках JavaScript-ориентированного стека MERN.

Основные преимущества MongoDB включают масштабируемость, высокую доступность, гибкость и удобство в работе с данными. Она обеспечивает быстрый доступ к данным, что делает ее очень привлекательной для проектов с высокой нагрузкой на чтение. MongoDB также поддерживает горизонтальное масштабирование, что позволяет ей легко масштабироваться по мере увеличения объема данных и количества пользователей.

Благодаря своей гибкой схеме данных, MongoDB идеально подходит для задач, связанных с хранением и обработкой неструктурированных данных, таких как хранение и обработка лог-файлов, систем мониторинга и протоколирования данных. Еще одним преимуществом MongoDB является полная совместимость с языком программирования JavaScript и возможность использования языка запросов на основе JSON, что делает ее легко интегрируемой в различные проекты на основе JavaScript. Для просмотра базы данных используется MongoDBCompass - бесплатный интерактивный инструмент для запросов, оптимизации и анализа данных MongoDВ, который позволяет получать ключевую информацию, перетаскивать элементы для создания пайплайнов и многое другое.

## 6.5. React

React - это JavaScript-библиотека, созданная компанией Facebook в 2011 году, для разработки пользовательских интерфейсов (UI). React позволяет создавать масштабируемые, быстрые и легко переносимые приложения. Его главная идея заключается в создании компонентов (Component-based architecture), которыми можно манипулировать, чтобы отображать данные изменения в различных состояниях.

Главная идея

Основная идея React заключается в разделении пользовательского интерфейса на небольшие и независимые компоненты. Каждый компонент представляет собой отдельную часть интерфейса, которая может иметь свое состояние и свойства (props). React обеспечивает простоту и эффективность в обновлении и перерисовке только тех компонентов, которые изменились, вместо обновления всего интерфейса.

React был разработан, чтобы обеспечить более быстрое и удобное обновление UI без перезагрузки страницы. Он использует Virtual DOM (виртуальный DOM), который представляет собой копию реального DOM-дерева, который вместо обновления элементов по целому дереву каждый раз обновляет только измененные элементы. Это позволяет ускорить процесс обновления UI

Основные преимущества

Основными преимуществами React являются:

- Простота разработки - React отлично подходит для создания масштабируемых приложений благодаря базовому подходу к составлению компонентов и удобной системе управления состоянием;

- Быстрая разработка - React предлагает быструю и простую проверку и управление приложением;

- Высокая производительность - React использует Virtual DOM, что позволяет обновлять только необходимые элементы, увеличивая скорость работы приложения;

- Гибкость - React использует только слой представления, позволяя использовать любые другие инструменты, например, Redux или AngularJS;

- Однонаправленный поток данных: React пропагандирует однонаправленный поток данных (от родительского компонента к дочернему). Это делает код более предсказуемым и упрощает отслеживание изменений.

Использование React

React может использоваться для разработки различных видов приложений, включая веб-приложения, мобильные приложения (с использованием React Native) и настольные приложения (с использованием Electron). Он активно применяется в индустрии и широко используется множеством компаний для создания масштабируемых и эффективных интерфейсов.

DOM и Virtual DOM

DOM (Document Object Model) - это программное представление веб-страницы, которое предоставляет доступ к элементам и их содержимому на странице. DOM представляет HTML-документ в виде дерева объектов, где каждый элемент становится объектом, атрибуты и содержимое элемента - свойствами объекта.

Виртуальный DOM (Virtual DOM) - это концепция, используемая React для оптимизации обновления пользовательского интерфейса. Виртуальный DOM представляет собой внутреннее представление DOM в памяти, которое React использует для эффективного обновления интерфейса. При изменении состояния приложения, React сравнивает предыдущий и новый виртуальный DOM, вычисляет разницу и применяет минимальные изменения в реальном DOM для обновления интерфейса.

### 6.5.1 React. Хуки.

Хуки (Hooks) являются новым механизмом, добавленным в React версии 16.8. Они позволяют использовать состояние и другие функции React без использования классового компонента.

Хуки в React представляют собой функции, которые позволяют компонентам React добавлять состояние и другие функциональные возможности, которые ранее были доступны только в классовых компонентах. С помощью Хуков компоненты становятся более читаемыми и простыми в использовании.

Кроме того, Хуки позволяют избежать проблем, связанных с созданием React компонентов на основе наследования, а также позволяют улучшить производительность приложения за счет предотвращения ненужных перерендеров компонентов.

Наиболее часто используемые Хуки в React включают:

- useState - позволяет добавлять в функциональный компонент состояние.

- useEffect - позволяет выполнить эффекты, связанные с компонентом (например, запрос на сервер), после изменения состояния.

- useContext - позволяет получить доступ к контексту из любого места в React-приложении.

- useMemo - позволяет кэшировать результаты вычислений и избежать ненужных вычислений.

- useCallback – позволяет запомнить коллбэк и избежать ненужного перерендера дочерних компонентов.

Хуки в React имеют свои особенности и требуют правильного использования, чтобы не создавать проблемы в разработке приложения. Однако, с помощью Хуков можно создавать более удобный и масштабируемый код, который легче поддерживать и обновлять.

# 7. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

7.1. Руководство для пользователя

Для перехода к сайт необходимо ввести http://localhost:3000/ в строке ввода пути в браузере. Вы попадете на главную страницу (рис. 7.1)



Рис. 7.1. Главная страница

На этой странице можно авторизоваться либо зарегистрироваться как новый пользователь (рис. 7.2 и рис. 7.3.).

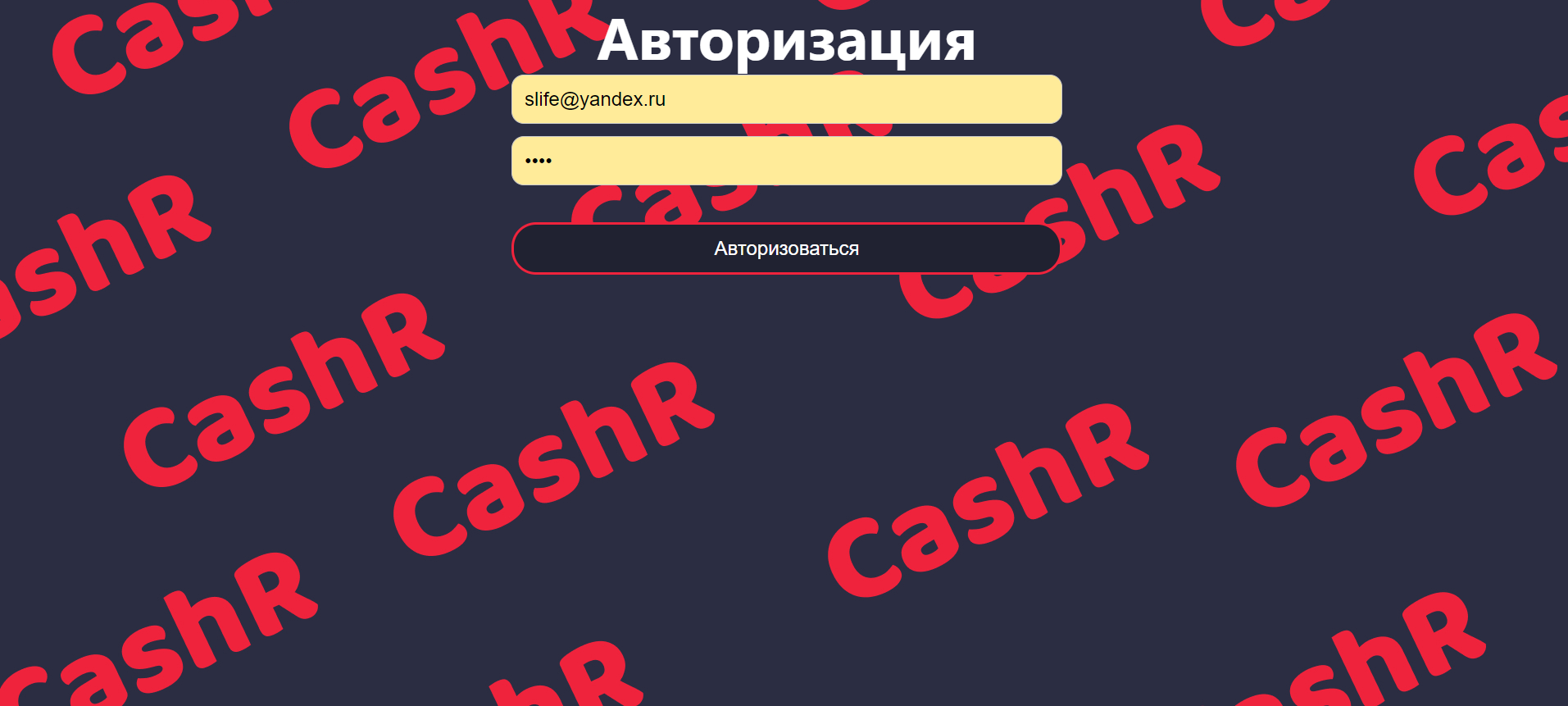


Рис. 7.2. Страница авторизации пользователя

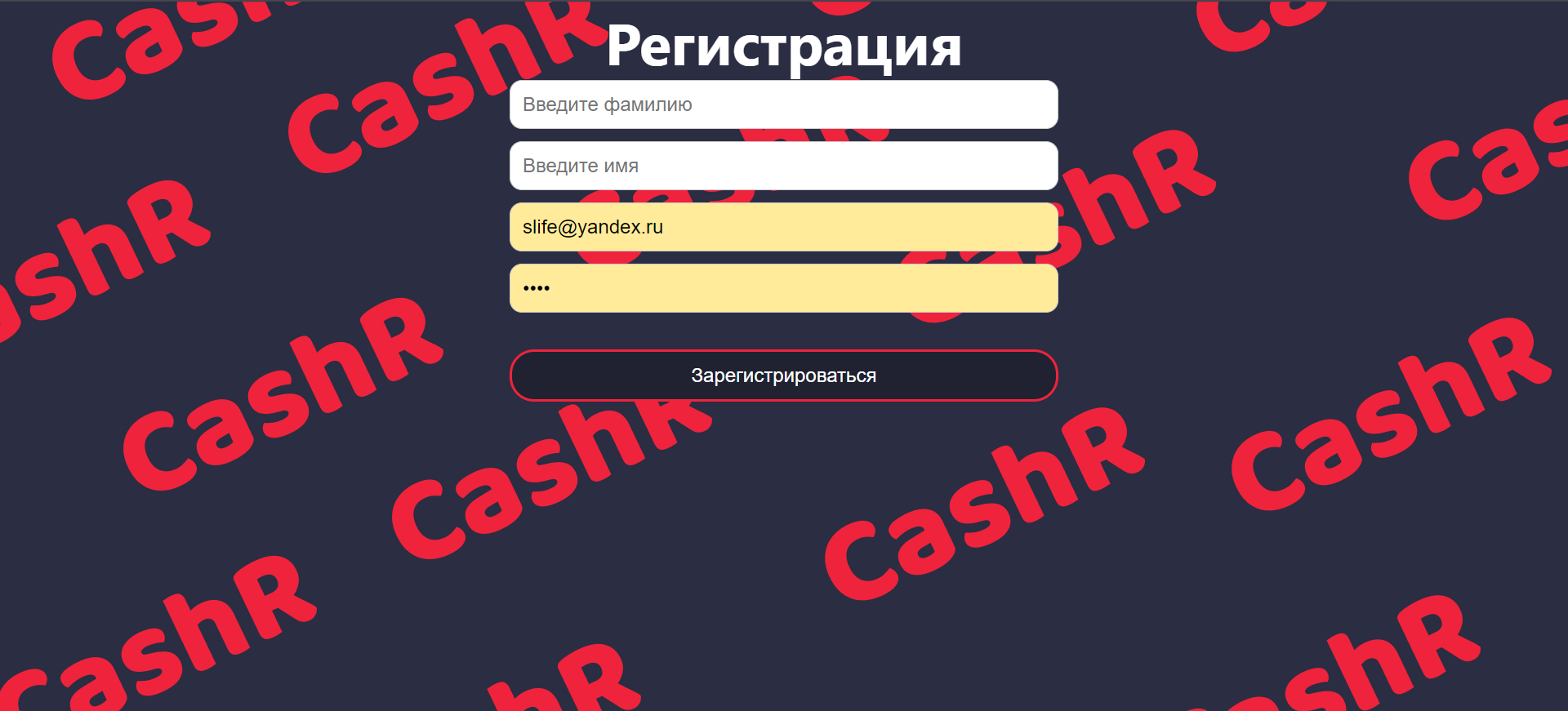


Рис. 7.3. Страница регистрации пользователя

Чтобы завершить регистрацию, требуется заполнить поля данных и нажать на кнопку «Зарегистрироваться». После Вы автоматически перейдете на страницу авторизации, где Вам снова будет необходимо ввести ваши данные. Нажимаем на кнопку «Авторизоваться». Вы перенаправитесь на страницу транзакций.

На странице транзакций пользователь может просмотреть все свои транзакции (рис. 7.7).



Рис. 7.7. Страница транзакций пользователя

На этой странице Вы можете отсортировать все свои транзакции, нажав на кнопку фильтра, который находиться под именем пользователя. Здесь вы можете отсортировать ваши транзакции по датам, по типу транзакции (зарплата, еда и т.д.), по сумме транзакции (рис. 7.8.)

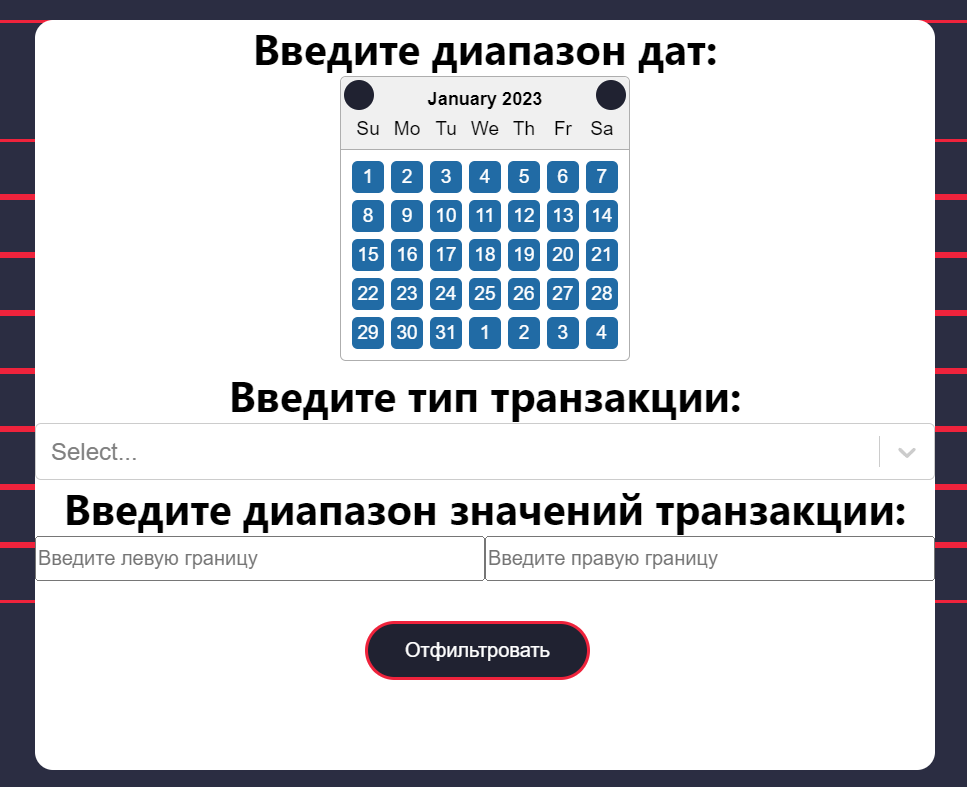


Рис. 7.8. Модальное окно фильтрации

Если нажать на имя пользователя, всплывает боковое окно с основной информацией пользователя. Здесь же можно осуществить выход из аккаунта и перейти на страницу авторизации (рис. 7.9.).

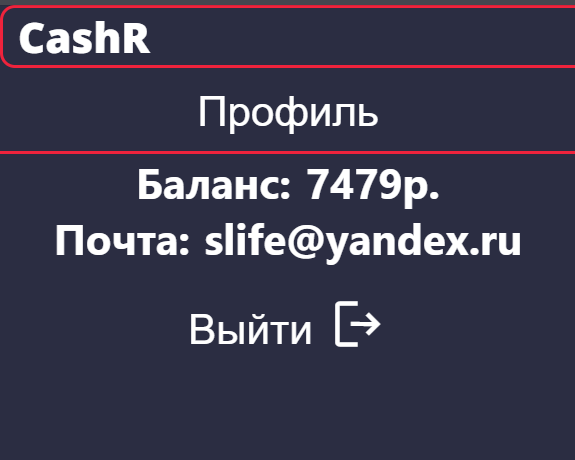


Рис. 7.9. Боковая пользовательская панель

При нажатии на логотип сайта, слева всплывает другая панель, где Вы можете осуществлять переход между страницами «Транзакции» и «Анализ транзакций». Помимо этого здесь же Вы можете открыть модальное окно добавления дохода/расхода (рис. 7.10).

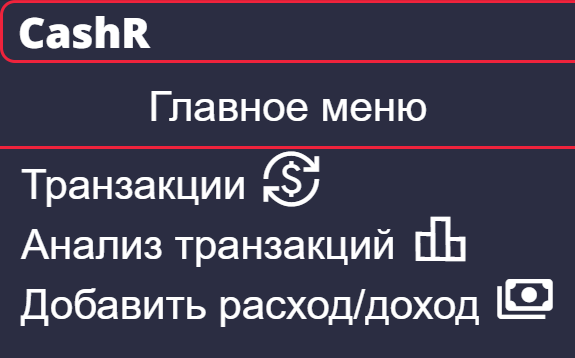


Рис. 7.10. Боковое окно переходов и добавления дохода/расхода

Давайте рассмотрим модальное окно добавления транзакции. Для этого нужно нажать на текст «Добавить расход/доход». Затем вам необходимо заполнить все поля: доход это или расход, тип транзакции, сумма транзакции, дата транзакции (рис.7.11).

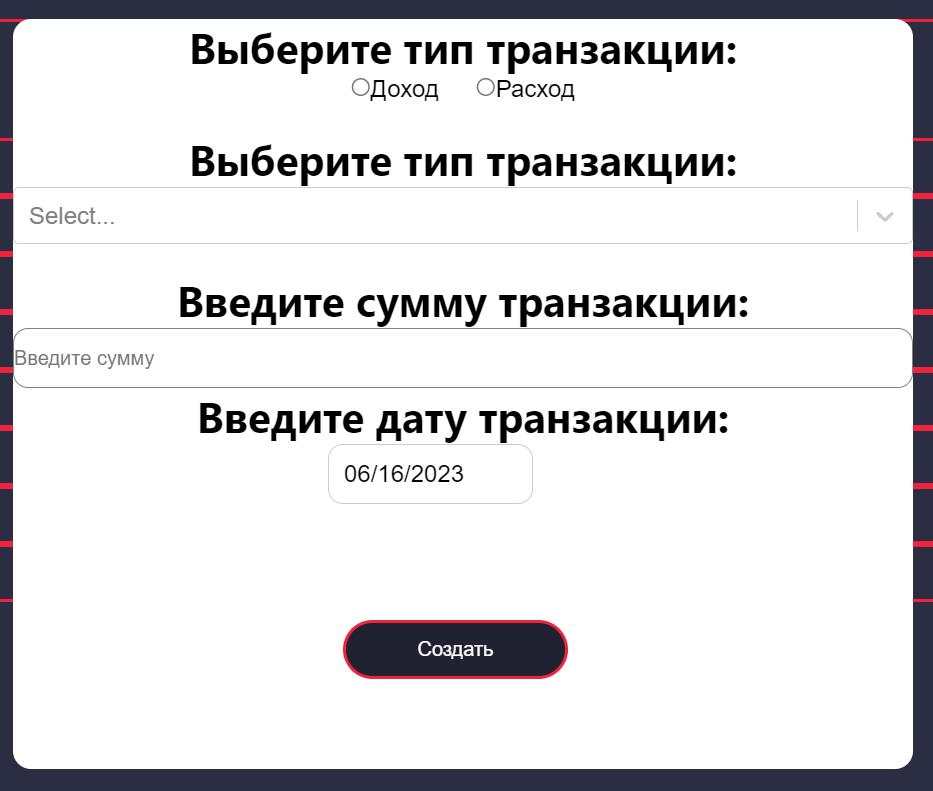


Рис. 7.11. Модальное окно создания транзакции

Перейдем на страницу «Анализ транзакций». Здесь Вы можете наглядно увидеть, как Ваши деньги приходят и уходят. Также в цифрах отображаются: баланс, общий оборот за период, общий расход за период, общий доход за период. При начальном переходе на эту страницу диаграммы будут описывать все транзакции. Однако той же кнопкой фильтра вы можете настроить диапазон интересующих Вас дат (рис. 7.12 и рис. 7.13).

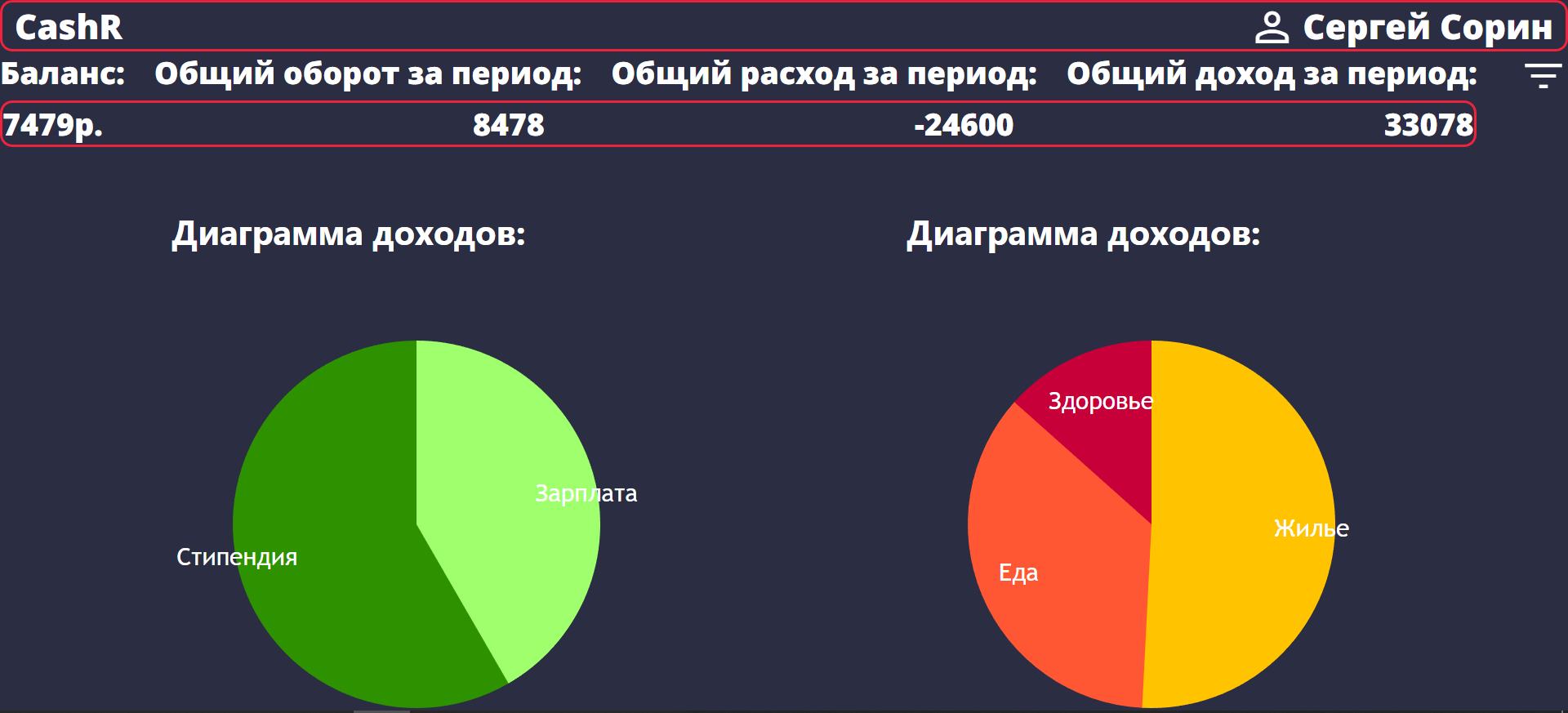


Рис. 7.12

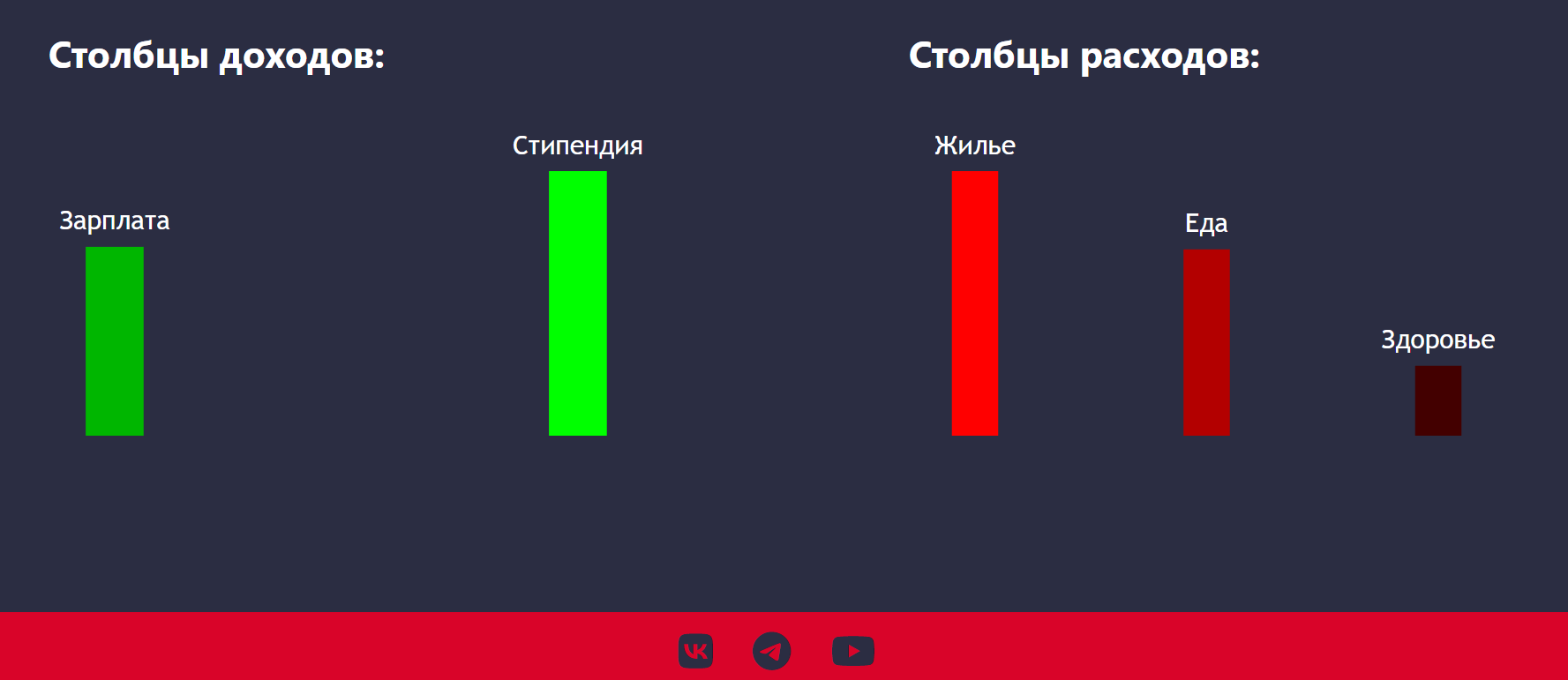


Рис. 7.13

7.2. Руководство для администратора

Чтобы авторизоваться в качестве администратора, требуется также нажать кнопку «Войти». Вы перейдете на страницу пользователей (рис. 7.2).

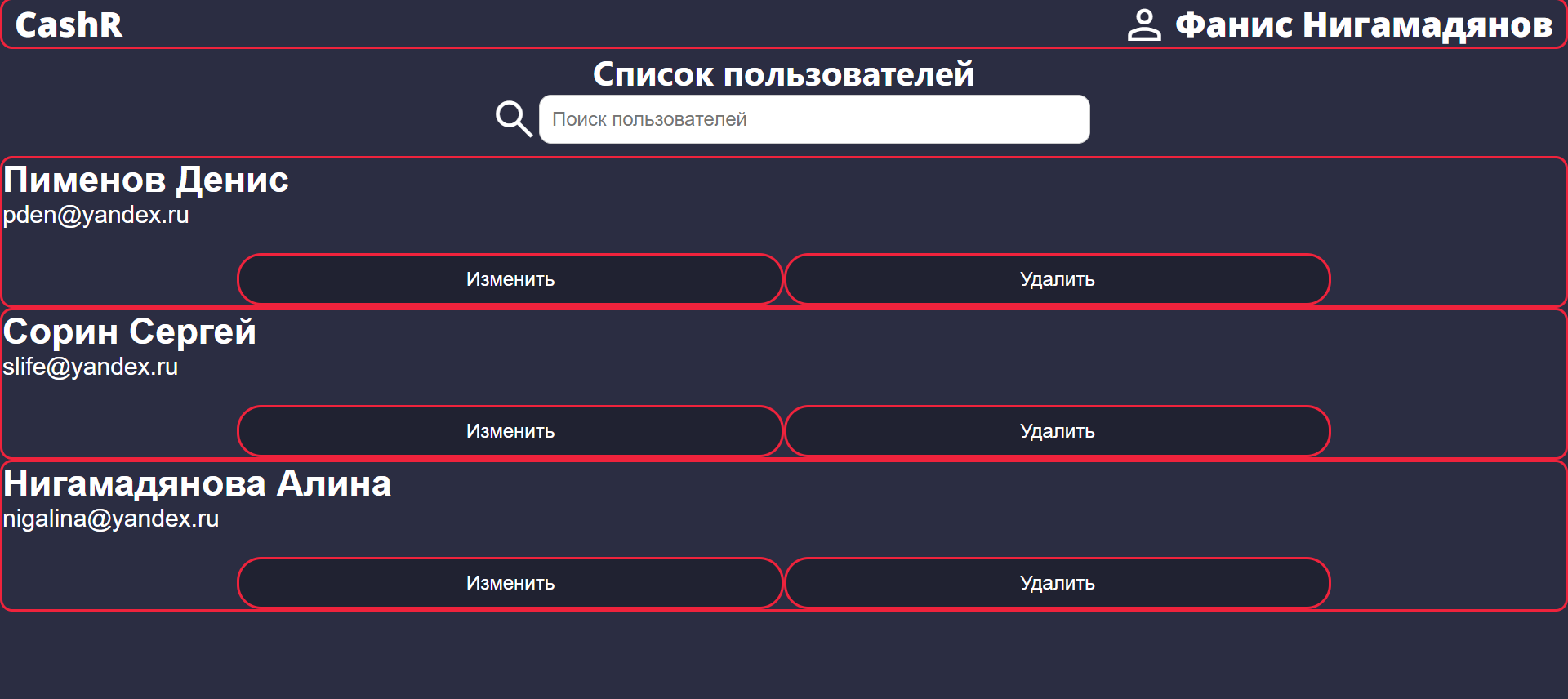


Рис. 7.12. Страница пользователей для администратора

Для поиска определенного пользователя можете начать ввести его фамилию, имя либо почту в поле для ввода текста с надписью «Поиск пользователей», как это сделано на рисунке 7.13.

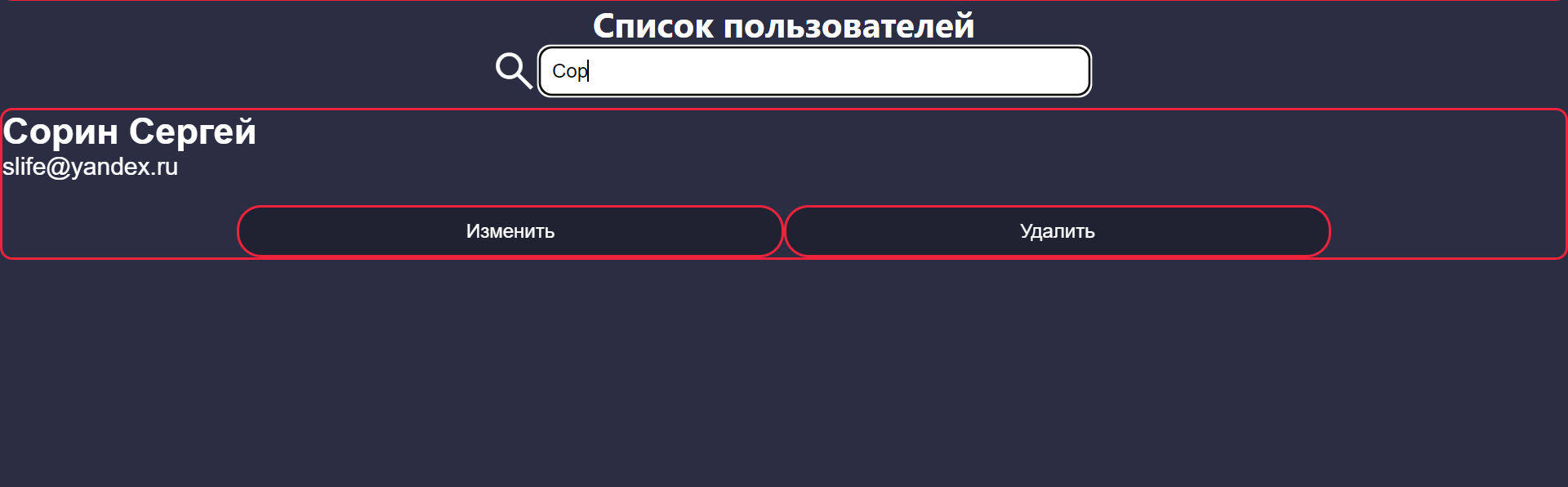


Рис. 7.13

Для удаления пользователя Вам необходимо нажать на кнопку «Удалить». Перед Вами всплывет модально окно с надписью о том, уверены ли Вы в вашем действии (рис.7.14).

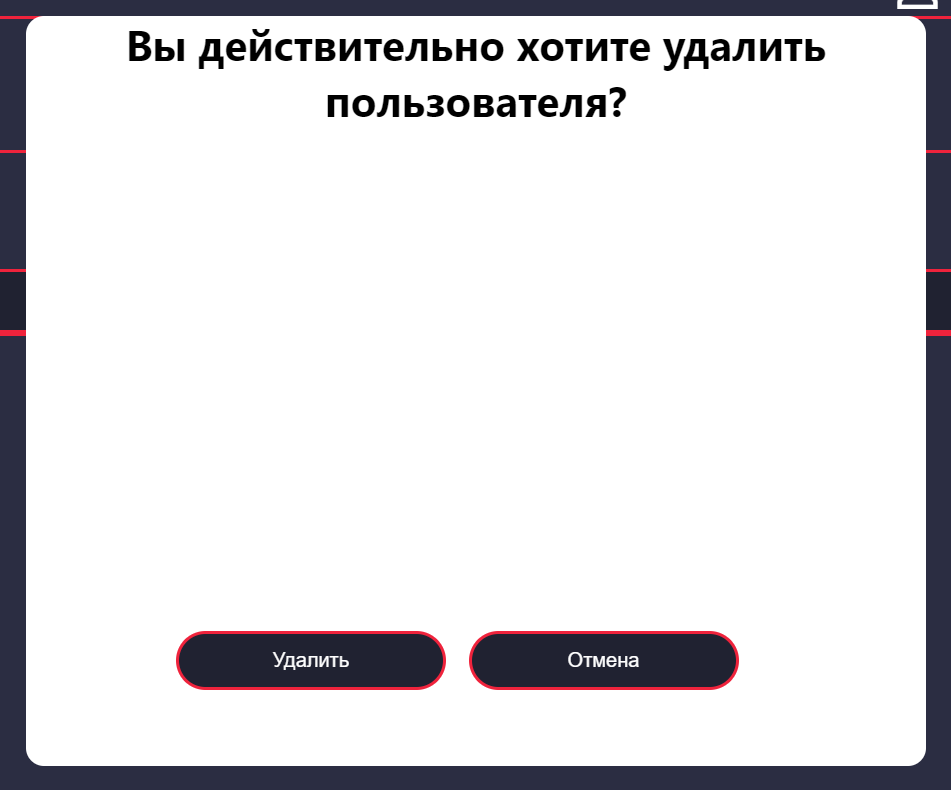


Рис. 7.14.

Для изменения данных о пользователе Вам необходимо нажать на кнопку «Изменить». Перед Вами всплывет модально окно с полями, заполненными данными пользователя. Здесь Вы можете изменить эти данные (Рис. 7.15).

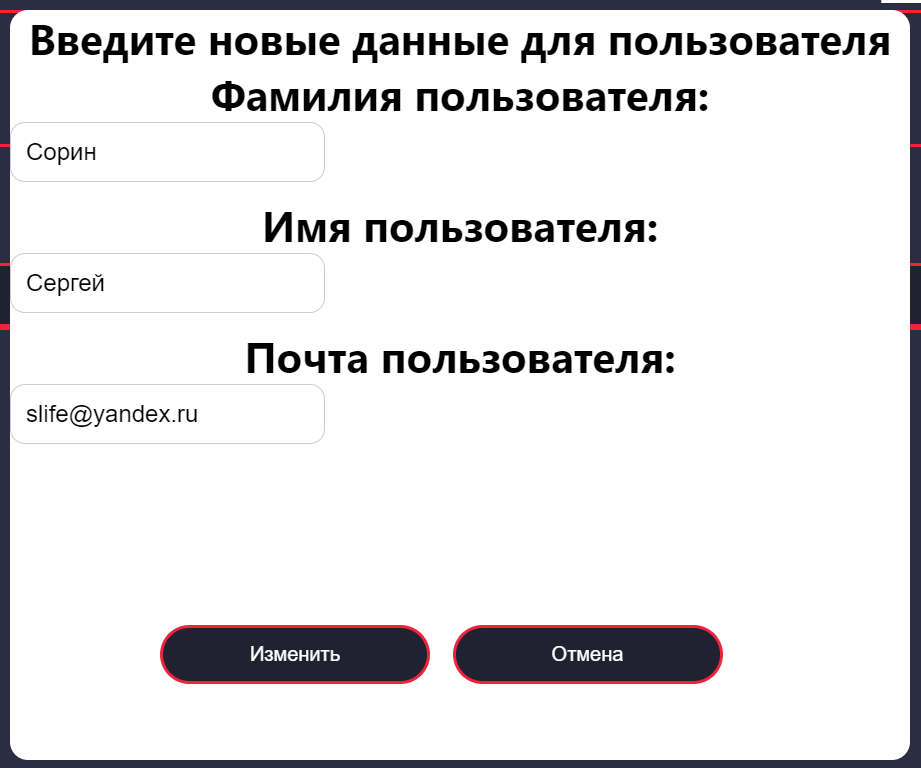


Рис. 7.15.

# ИСПОЛЬЗОВАННЫЕ ИСТОЧНИКИ

1. Документация по React: <https://react.dev> – (Дата обращения: 01.05.2023)

2. Флэнаган Д. JavaScript. Подробное руководство. – Пер. с англ. – СПб:

Символ-Плюс, 2008. – 992 с., ил. – (Дата обращения: 21.04.2023)

3. Онлайн-руководство по MongoDB. // METANIT – сайт о

программировании [Электронный ресурс]. – Режим доступа:

https://metanit.com/nosql/mongodb/. – (Дата обращения: 19.05.2020)

4. MongoDB Documentation [Электронный ресурс]. URL: https://www.mongodb.com/docs/ (дата обращения: 19.05.2022)

5. Documentation | Node.js [Электронный ресурс]. URL: https://nodejs.org/en/docs/ (дата обращения: 19.05.2022)

6. Mongoose Documentation [Электронный ресурс]. URL: <https://mognoosejs.com/docs/documents.html> (дата обращения: 19.05.2022)

7. Документация ReactBootstrap: <https://react-bootstrap.github.io/> – (Дата обращения: 02.05.2023)

# Приложение.

# index.js

const express = require('express'); //import express from package express  
const PORT = process.env.PORT || 5000; //choosing the port (system or default)  
const mongoose = require('mongoose');  
const bodyParser = require('body-parser');  
const authRouter = require('./routs/authRouter');  
const adminRouter = require('./routs/adminRouter');  
const userRouter = require('./routs/userRouter');  
const app = express(); //make app from express  
const cors = require('cors');  
  
app.use(cors());  
app.use(express.json());  
app.use(bodyParser.json());  
app.use("/auth", authRouter);  
app.use("/admin", adminRouter);  
app.use("/user", userRouter);  
  
const start = async () => {  
 try{  
 await mongoose.connect(`mongodb+srv://\*\*\*@cluster0.yv7s4qc.mongodb.net/?retryWrites=true&w=majority`);  
 app.listen(PORT, () => console.log(`server started on port ${PORT}`));  
 }catch(e){  
 console.log(e);  
 }  
}  
start();

role.js

const {Schema, model} = require('mongoose');  
  
const Role = new Schema({  
 value:{type:String, unique: true},  
  
})  
  
module.exports = model('Role', Role)

transactions.js

const {Schema, model, Types} = require('mongoose')  
  
  
const Transaction = new Schema({  
 userID:{type: Types.ObjectId, required: true},  
 come:{type: String, required:true},  
 valueOfTransaction:{type: Number, required: true},  
 typeOfTransaction:{type: String, required:true},  
 dateOfTransaction:{type: Date, required:true}  
})  
  
module.exports = model('Transaction', Transaction)

user.js

const {Schema, model} = require('mongoose')  
  
  
const User = new Schema({  
 firstname:{type: String, unique: false, required: true},  
 lastname:{type: String, unique: false, required: true},  
 email: {type: String, unique: true, required: true},  
 password: {type: String, required: true},  
 roles: [{type: String, ref: 'Role'}],  
 balance: {type: Number, required: true}  
})  
  
module.exports = model('User', User)

adminRouter.js

const Router = require('express');  
const router = new Router();  
const controller = require('../controllers/adminConroller');  
const {check} = require("express-validator");  
  
router.get('/users', controller.users);  
router.post('/deleteUser', controller.deleteUser);  
router.post('/getUser', controller.getUser);  
router.put('/updateUser', controller.updateUser)  
  
module.exports = router;

adminController.js

const bcrypt = require('bcryptjs');  
const jwt = require('jsonwebtoken');  
const {validationResult} = require('express-validator');  
const {secret} = require("../config");  
const express = require("express");  
const path = require("path")  
const bodyParser = require('body-parser');  
  
const User = require('../models/user');  
const Role = require('../models/role');  
  
class adminController{  
 async users(req, res){  
 try {  
 const users = await User.find({roles:"User"});  
 return res.json(users);  
  
 } catch (e) {  
 console.log(e);  
 res.status(400).json({message: 'Getting users error'});  
 }  
 }  
  
 async deleteUser(req, res){  
 try {  
 const userID = req.body.selectedUserID;  
 await User.findOneAndDelete({\_id:userID});  
 const obj = {  
 isDeleted: true,  
 message: "Пользователь удален!"  
 };  
 return res.json(obj);  
  
 } catch (e) {  
 console.log(e);  
 res.status(400).json({isDeleted: false, message: "Ошибка при удалении пользователя!"});  
 }  
 }  
 async getUser(req, res){  
 try {  
 const userID = req.body.selectedUserID;  
 const user = await User.findOne({\_id:userID});  
 const obj = {  
 firstname: user.firstname,  
 lastname: user.lastname,  
 email: user.email  
 };  
 return res.json(obj);  
  
 } catch (e) {  
 console.log(e);  
 res.status(400).json({isDeleted: false, message: "Ошибка!"});  
 }  
 }  
 async updateUser(req, res){  
 try {  
 const userID = req.body.id;  
 const userFirstName = req.body.firstname;  
 const userLastName = req.body.lastname;  
 const userEmail = req.body.email;  
 const result = await User.findOneAndUpdate({\_id:userID}, { $set: {firstname:userFirstName, lastname:userLastName, email: userEmail}});  
 const obj = {  
 message: "Пользователь изменен!"  
 };  
 return res.json(obj);  
  
 }catch (e){  
 console.log(e);  
 res.status(400).json({message:'Project users getting error'});  
 }  
 }  
}  
module.exports = new adminController();

userRouter.js

const Router = require('express');  
const router = new Router();  
const controller = require('../controllers/userController');  
const {check} = require("express-validator");  
  
router.post('/createTransaction', controller.createTransaction);  
router.post('/getTransactions', controller.getTransactions);  
module.exports = router;

userController.js

const bcrypt = require('bcryptjs');  
const jwt = require('jsonwebtoken');  
const {validationResult} = require('express-validator');  
const {secret} = require("../config");  
const express = require("express");  
const path = require("path")  
const bodyParser = require('body-parser');  
  
const User = require('../models/user');  
const Role = require('../models/role');  
const Transaction = require('../models/transaction')  
  
class userController {  
 async createTransaction(req,res){  
 console.log(req.body)  
 const userID = req.body.userID;  
 const come = req.body.come;  
 const valueOfTransaction = req.body.valueOfTransaction;  
 const typeOfTransaction = req.body.typeOfTransaction.value;  
 const dateOfTransaction = req.body.dateOfTransaction;  
  
 const newTransaction = new Transaction({  
 userID: userID,  
 valueOfTransaction: valueOfTransaction,  
 come: come,  
 typeOfTransaction: typeOfTransaction,  
 dateOfTransaction: dateOfTransaction  
 });  
 await newTransaction.save();  
  
 const user = await User.findById(userID);  
 const balance = { amount: user.balance };  
  
 if (come === 'Income') {  
 balance.amount += parseInt(valueOfTransaction);  
 }  
 else if(come ==='Outcome'){  
 balance.amount += valueOfTransaction;  
 }  
  
 const result = await User.findOneAndUpdate({\_id:userID}, { $set: {balance: balance.amount}});  
  
 res.status(200).json({ message: 'Транзакция успешно создана!' });  
 }  
 async getTransactions(req,res){  
 console.log(req.body)  
 const userID = req.body.userID;  
 const transactions = await Transaction.find({ userID: userID });  
 console.log(transactions);  
 res.status(200).json(transactions);  
 }  
}  
  
module.exports = new userController();

authRouter.js

const Router = require('express');  
const router = new Router();  
const controller = require('../controllers/authController');  
const {check} = require("express-validator");  
  
router.post('/registration',[  
 check('firstName', "Фамилия пользователя не может быть пустым!").notEmpty(),  
 check('lastName', "Имя пользователя не может быть пустым!").notEmpty(),  
 check('email', "Почта не может быть пустым!").notEmpty(),  
 check('password', "Пароль должен содержать не менее 4-ех и не более 10-и символов!").isLength({min:4, max:10})  
], controller.registration);  
router.post('/login', controller.login);  
router.get('/user', controller.getUser);  
  
module.exports = router;

authController.js

const bcrypt = require('bcryptjs');  
const jwt = require('jsonwebtoken');  
const {validationResult} = require('express-validator');  
const {secret} = require("../config");  
const config = require('../config.js');  
const express = require("express");  
const path = require("path")  
const bodyParser = require('body-parser');  
  
const User = require('../models/user');  
const Role = require('../models/role');  
  
  
  
const generateAccessToken = (id, roles) =>{  
 const payload = {  
 id,  
 roles  
 }  
 return jwt.sign(payload, secret, {expiresIn:"24h"});  
}  
  
class authController {  
 async getUser(req, res){  
 try{  
 const authHeader = req.headers.authorization;  
 const token = authHeader && authHeader.split(' ')[1];  
  
 if (!token) {  
 return res.sendStatus(401);  
 }  
 jwt.verify(token, config.secret, async (err, decoded) => {  
 if (err) {  
 console.log(err); // Вывод информации об ошибке в консоль  
 return res.status(401).json({error: 'Unauthorized'});  
 }  
 const userId = decoded.id;  
 const user = await User.findOne({\_id: userId});  
  
 if (!user) {  
 return res.status(404).json({error: 'User not found'});  
 }  
 res.json(user);  
 });  
 }  
 catch(e){  
 console.log(e);  
 res.status(400).json({message:'Getting users error'});  
 }  
 }  
 async registration(req, res) {  
 try {  
 const errors = validationResult(req);  
 if (!errors.isEmpty()) {  
 return res.status(400).json({message: "Ошибка при регистрации", errors});  
 }  
 const {firstName, lastName, email, password} = req.body;  
 const candidate = await User.findOne({email});  
 if (candidate) {  
 return res.status(400).json({message: 'Пользователь с таким именем уже существует'});  
 }  
 const hashPassword = bcrypt.hashSync(password, 7);  
 const userRole = await Role.findOne({value:"User"});  
 const user = new User({  
 firstname: firstName,  
 lastname: lastName,  
 email: email,  
 password: hashPassword,  
 roles: userRole.value,  
 balance: 0  
 })  
 await user.save();  
 const obj = {  
 isRegistered: true,  
 message: "Регистрация прошла успешно!"  
 };  
 res.json(obj);  
 } catch (e) {  
 console.log(e);  
 res.status(400).json({message: 'Registration error'});  
 }  
 }  
  
 async login(req, res) {  
 try {  
 const {email, password} = req.body;  
 const user = await User.findOne({email});  
 if (!user) {  
 return res.status(400).json({message: `Пользователь ${email} не найден!`});  
 }  
 const validPassword = bcrypt.compareSync(password, user.password);  
 if (!validPassword) {  
 return res.status(400).json({message: `Введен неверный пароль!`});  
 }  
 const token = generateAccessToken(user.\_id, user.roles)  
 //res.cookie('token', token);  
 const obj = {  
 token: token,  
 role: user.roles,  
 message: "Авторизация успешна!"  
 };  
 return res.json(obj);  
  
 } catch (e) {  
 console.log(e);  
 res.status(400).json({message: 'Login error'});  
 }  
 }  
}  
  
  
  
module.exports = new authController();

index.html

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="utf-8" />  
 <meta name="viewport" content="width=device-width, initial-scale=1" />  
 <meta name="theme-color" content="#000000" />  
 <title>CashR</title>  
 <script src="https://kit.fontawesome.com/0a2fe38d56.js" crossorigin="anonymous"></script>  
 <link rel="stylesheet" href="https://fonts.googleapis.com/css2?family=Material+Symbols+Outlined:opsz,wght,FILL,GRAD@20..48,100..700,0..1,-50..200" />  
 <link rel="stylesheet" href="https://fonts.googleapis.com/css2?family=Material+Symbols+Outlined:opsz,wght,FILL,GRAD@20..48,100..700,0..1,-50..200" />  
 <link rel="stylesheet" href="https://fonts.googleapis.com/css2?family=Material+Symbols+Outlined:opsz,wght,FILL,GRAD@20..48,100..700,0..1,-50..200" />  
 <link rel="stylesheet" href="https://fonts.googleapis.com/css2?family=Material+Symbols+Outlined:opsz,wght,FILL,GRAD@48,400,0,0" />  
 <link rel="stylesheet" href="https://fonts.googleapis.com/css2?family=Material+Symbols+Outlined:opsz,wght,FILL,GRAD@20..48,100..700,0..1,-50..200" />  
 <link rel="preconnect" href="https://fonts.googleapis.com">  
 <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>  
 <link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">  
 <link rel="stylesheet" href="https://fonts.googleapis.com/css2?family=Material+Symbols+Outlined:opsz,wght,FILL,GRAD@48,400,0,0" />  
 <link href="https://fonts.googleapis.com/css2?family=Noto+Sans:wght@900&display=swap" rel="stylesheet">  
 <link rel="shortcut icon" href="./CashR.png" type="image/x-icon"/>  
</head>  
 <body>  
 <noscript>You need to enable JavaScript to run this app.</noscript>  
 <div id="root"></div>  
 </body>  
</html>

DatePicker.jsx

import React, { useState } from "react";  
import DatePicker from "react-datepicker";  
import "react-datepicker/dist/react-datepicker.css";  
  
const MyDatePicker=({startDate, setStartDate}) => {  
  
 let handleColor = (time) => {  
 return time.getHours() > 12 ? "text-success" : "text-error";  
 };  
  
 return (  
 <DatePicker  
 showTimeSelect  
 selected={startDate}  
 onChange={(date) => setStartDate(date)}  
 timeClassName={handleColor}  
 />  
 );  
};  
  
export default MyDatePicker;

DatePickerRange.jsx

import React, { useState } from "react";  
import DatePicker from "react-datepicker";  
import "react-datepicker/dist/react-datepicker.css";  
import MyDatePicker from "./DatePicker";  
  
const MyDatePickerRange=({startDate, setStartDate, endDate, setEndDate}) => {  
  
 const onChange = (dates) => {  
 const [start, end] = dates;  
 setStartDate(start);  
 setEndDate(end);  
 };  
 return (  
 <DatePicker  
 selected={startDate}  
 onChange={onChange}  
 startDate={startDate}  
 endDate={endDate}  
 selectsRange  
 inline  
 />  
 );  
};  
  
export default MyDatePickerRange;

DropDownMulti.jsx

import React, { useState, useEffect } from "react";  
import Select from "react-select";  
  
const MultiSelect = ({selectedTypes, setSelectedTypes}) => {  
  
 const[typesOfComes, setTypesOfComes] = useState(["Еда", "Здоровье", "Спорт", "Жилье", "Зарплата", "Стипендия"]);  
  
 const options = typesOfComes.map((typeOfCome) => ({  
 label: typeOfCome,  
 value: typeOfCome,  
 }));  
  
 const handleSelectedTypes = (selectedOptions) => {  
 setSelectedTypes(selectedOptions);  
 };  
  
 return (  
 <Select  
 value={selectedTypes}  
 onChange={handleSelectedTypes}  
 options={options}  
 isMulti  
 />  
 );  
};  
  
export default MultiSelect;

DropDownSingle.jsx

import React, { useState, useEffect } from "react";  
import Select from "react-select";  
const SingleSelect = ({typesOfComes, selectedType ,setSelectedType}) => {  
  
 const options = typesOfComes.map((typeOfCome) => ({  
 label: typeOfCome,  
 value: typeOfCome,  
 }));  
  
 const handleSelectedType = (selectedOptions) => {  
 setSelectedType(selectedOptions);  
 };  
  
 return (  
 <Select  
 value={selectedType}  
 onChange={handleSelectedType}  
 options={options}  
 />  
 );  
};  
  
export default SingleSelect;

Home.jsx

import React from 'react';  
import './Home.css'  
import {Link} from "react-router-dom";  
const Home = () => {  
 return (  
 <div className="Home">  
 <header>  
 <div className="MainBar">  
 <h1>CashR</h1>  
 <div className="RegAuth">  
 <Link to="/login" className="BarText">Войти</Link>  
 <Link to="/registration" className="BarText">Регистрация</Link>  
 </div>  
 </div>  
 </header>  
 <div className="aligner">  
 <div className="MainText text-1">  
 Это  
 </div>  
 <div className="MainText text-2">  
 CashR  
 </div>  
 </div>  
  
 <div className="MainText text-3">  
 Следить за финансами будет проще  
 </div>  
 </div>  
 );  
};  
  
export default Home;

MainPage.jsx

import React, {useEffect, useState} from 'react';  
import './MainPage.css';  
import axios from 'axios';  
import {useNavigate} from 'react-router-dom';  
import {Backdrop, CircularProgress} from '@mui/material';  
  
import {Link} from "react-router-dom";  
import Menu from "../Menu/Menu";  
import ModalAddTrans from "../Modal/ModalAddTrans";  
import MyDatePickerRange from "../DatePicker/DatePickerRange";  
import {disableBodyScroll, enableBodyScroll} from "body-scroll-lock";  
import ProfileMenu from "../Menu/ProfileMenu";  
import ModalFilter from "../Modal/ModalFilter";  
  
const MainPage = () => {  
 const [isLoading, setIsLoading] = useState();  
 const [user, setUser] = useState({  
 lastname: "",  
 firstname: "",  
 email: "",  
 role: ""  
 });  
 const [users, setUsers] = useState([]);  
 const [menuActive, setMenuActive] = useState(false);  
 const[profileActive, setProfileActive] = useState(false);  
 const[modalFilterIsOpened, setModalFilterIsOpened] = useState(false);  
 const navigate = useNavigate();  
  
 const [come, setCome] = useState();  
 const [addTransactionIsOpened, setAddTransactionIsOpened] = useState(false);  
 const [typesOfOutcomes, setTypesOfOutcomes] = useState(["Еда", "Здоровье", "Спорт", "Жилье"]);  
 const [typesOfIncomes, setTypesOfIncomes] = useState(["Зарплата", "Стипендия"]);  
 const [transactions, setTransactions] = useState([]);  
 const [sortedTransactions, setSortedTransactions] = useState([]);  
  
 const [startDate, setStartDate] = useState(new Date());  
 const [endDate, setEndDate] = useState(null);  
  
 const[leftBorder, setLeftBorder] = useState();  
 const[rightBorder, setRightBorder] = useState();  
  
 const MenuItems = [{value: "Транзакции", action: handleTransClick, icon: "trans"},{  
 value: "Анализ транзакций",  
 action: handleTransAnalyseClick,  
 icon: "analyse"  
 }, {  
 value: "Добавить расход/доход",  
 action: handleAddTransClick,  
 icon: "coin"  
 }];  
  
 const ProfileItems=[{value: "Выйти", action: handleLogoutClick, icon: "logout"}]  
  
 function handleTransAnalyseClick(){  
 navigate('/transAnalyse');  
 }  
 function handleTransClick(){  
 navigate('/transactions');  
 }  
 function handleLogoutClick() {  
 navigate('/login');  
 localStorage.clear();  
 }  
  
 function handleAddTransClick() {  
 setMenuActive(false);  
 setAddTransactionIsOpened(true);  
 }  
  
 useEffect(() => {  
 setIsLoading(true);  
 const fetchData = (async () => {  
 const token = localStorage.getItem('token');  
 if (!token) {  
 console.error('Token not found in localStorage');  
 return;  
 }  
 try {  
 const {data: response} = await axios.get('http://localhost:5000/auth/user', {  
 headers: {  
 Authorization: `Bearer ${token}`,  
 },  
 });  
  
 await setUser({  
 \_id: response.\_id,  
 lastname: response.lastname,  
 firstname: response.firstname,  
 email: response.email,  
 role: response.roles[0],  
 balance: response.balance,  
 });  
  
 const transactionsResponse = await fetch('http://localhost:5000/user/getTransactions', {  
 method: 'POST',  
 headers: {'Content-Type': 'application/json'},  
 body: JSON.stringify({userID: response.\_id}),  
 });  
  
 const transactionsData = await transactionsResponse.json();  
 setTransactions(transactionsData);  
  
  
 } catch (error) {  
 console.error(error.message);  
 }  
  
 setIsLoading(false);  
 });  
 fetchData();  
 }, []);  
  
 function getDateValue(dateString) {  
 return new Date(dateString).getTime();  
 }  
 useEffect(()=>{  
 const srtdTransactions = [...transactions].sort(function(a, b) {  
 return getDateValue(b.dateOfTransaction) - getDateValue(a.dateOfTransaction);  
 });  
 setSortedTransactions(srtdTransactions);  
 },[transactions])  
  
 function toggleMenu() {  
 setMenuActive(!menuActive);  
 }  
 function toggleProfile(){  
 setProfileActive(!profileActive);  
 }  
 function toggleModalFilter(){  
 setModalFilterIsOpened(!modalFilterIsOpened);  
 }  
  
 const[standartSet, setStandartSet] = useState([]);  
 function getDateValue(dateString) {  
 return new Date(dateString).getTime();  
 }  
 useEffect(()=>{  
 const srtdTransactions = [...transactions].sort(function(a, b) {  
 return getDateValue(b.dateOfTransaction) - getDateValue(a.dateOfTransaction);  
 });  
 setStandartSet(srtdTransactions);  
 },[transactions])  
  
  
 return (  
 <div className="Users">  
 <header>  
 <div className="MainBar">  
 <h1 className="logo" onClick={toggleMenu}>CashR</h1>  
 <div className="userName" onClick={toggleProfile}>  
 <div className="projectIcon">  
 <span className="material-symbols-outlined">person</span>  
 </div>  
 <h1>{user.lastname} {user.firstname}</h1>  
 </div>  
 </div>  
 </header>  
 <div className="mainContent">  
 <div className="filterRow">  
 <h1>Транзакции:</h1>  
 <span onClick={toggleModalFilter} className="material-symbols-outlined">filter\_list</span>  
 </div>  
  
  
 <div className="transHeader">  
 <div><h1>Сумма</h1></div>  
 <div><h1>Тип транзакции</h1></div>  
 <div><h1>Дата транзакции</h1></div>  
 </div>  
  
 <div>  
 {sortedTransactions && sortedTransactions.map((transaction) => (  
 <div className="transRow" key={transaction.\_id}>  
 <div className='comeContainer'>  
 <div className='comeOutcome'>  
 <h2>{transaction.come === 'Outcome' && transaction.valueOfTransaction}</h2></div>  
 <div className='comeIncome'>  
 <h2>{transaction.come === 'Income' && '+' + transaction.valueOfTransaction}</h2>  
 </div>  
 </div>  
 <div><h2>{transaction.typeOfTransaction}</h2></div>  
 <div><h2>{new Intl.DateTimeFormat('ru-Ru', {  
 year: 'numeric',  
 month: 'numeric',  
 day: 'numeric',  
 hour: 'numeric',  
 minute: 'numeric',  
 second: 'numeric'  
 }).format(new Date(transaction.dateOfTransaction))}</h2></div>  
 </div>  
 ))}  
 </div>  
 </div>  
 <ModalFilter leftBorder={leftBorder} standartSet={standartSet} setStandartSet={setStandartSet} transactions={transactions} setLeftBorder={setLeftBorder} rightBorder={rightBorder} setRightBorder={setRightBorder} modalFilterIsOpened={modalFilterIsOpened} setModalFilterIsOpened={setModalFilterIsOpened} sortedTransactions={sortedTransactions} setSortedTransactions={setSortedTransactions}/>  
 <ModalAddTrans come={come} setUser={setUser} user={user} setTransactions={setTransactions}  
 setIsLoading={setIsLoading} setCome={setCome} typesOfIncomes={typesOfIncomes}  
 typesOfOutcomes={typesOfOutcomes} setAddTransactionIsOpened={setAddTransactionIsOpened}  
 addTransactionIsOpened={addTransactionIsOpened}/>  
 <Menu active={menuActive} setActive={setMenuActive} action={true} header={"Главное меню"}  
 items={MenuItems}/>  
 <ProfileMenu items={ProfileItems} userBalance={user.balance} userEmail={user.email} active={profileActive} setActive={setProfileActive} action={true} header={"Профиль"}/>  
 <Backdrop open={isLoading}>  
 <CircularProgress/>  
 </Backdrop>  
 <div id="addGap"></div>  
 <footer>  
 <a href="https://vk.com/fanis\_ng" target="\_blank"><i className="fa-brands fa-vk"></i></a>  
 <a href="https://t.me/fanis\_ng" target="\_blank"><i className="fa-brands fa-telegram"></i></a>  
 <a href="https://www.youtube.com/@fanisnigamadyanov8262/featured" target="\_blank"><i className="fa-brands fa-youtube"></i></a>  
 </footer>  
</div>  
);  
};  
  
export default MainPage;

TransAnalyze.jsx

import React, {useEffect, useState} from 'react';  
import './MainPage.css';  
import axios from 'axios';  
import {useNavigate} from 'react-router-dom';  
import {Backdrop, CircularProgress} from '@mui/material';  
  
import {Link} from "react-router-dom";  
import Menu from "../Menu/Menu";  
import ModalAddTrans from "../Modal/ModalAddTrans";  
import MyDatePickerRange from "../DatePicker/DatePickerRange";  
import {disableBodyScroll, enableBodyScroll} from "body-scroll-lock";  
import ProfileMenu from "../Menu/ProfileMenu";  
import ModalFilter from "../Modal/ModalFilter";  
import ModalFilterRange from "../Modal/ModalFilterRange";  
import IncomePieChart from "../PieCharts/IncomePieCharts";  
import OutcomePieChart from "../PieCharts/OutcomePieCharts";  
import IncomePieBar from "../PieCharts/IncomePieBars";  
import OutcomePieBar from "../PieCharts/OutcomePieBars";  
  
const TransAnalyse = () => {  
 const [isLoading, setIsLoading] = useState();  
 const [user, setUser] = useState({  
 lastname: "",  
 firstname: "",  
 email: "",  
 role: ""  
 });  
 const [users, setUsers] = useState([]);  
 const [menuActive, setMenuActive] = useState(false);  
 const[profileActive, setProfileActive] = useState(false);  
 const[modalFilterIsOpened, setModalFilterIsOpened] = useState(false);  
 const navigate = useNavigate();  
  
 const [come, setCome] = useState();  
 const [addTransactionIsOpened, setAddTransactionIsOpened] = useState(false);  
 const [typesOfOutcomes, setTypesOfOutcomes] = useState(["Еда", "Здоровье", "Спорт", "Жилье"]);  
 const [typesOfIncomes, setTypesOfIncomes] = useState(["Зарплата", "Стипендия"]);  
 const [transactions, setTransactions] = useState([]);  
 const [sortedTransactions, setSortedTransactions] = useState([]);  
  
 const[expensesSum, setExpensesSum] = useState();  
 const[incomesSum, setIncomeSum] = useState();  
 const[total, setTotal] = useState();  
  
 const [startDate, setStartDate] = useState(new Date());  
 const [endDate, setEndDate] = useState(null);  
  
 const[leftBorder, setLeftBorder] = useState();  
 const[rightBorder, setRightBorder] = useState();  
  
 const MenuItems = [{value: "Транзакции", action: handleTransClick, icon: "trans"},  
 {  
 value: "Анализ транзакций",  
 action: handleTransAnalyseClick,  
 icon: "analyse"  
 }, {  
 value: "Добавить расход/доход",  
 action: handleAddTransClick,  
 icon: "coin"  
 }];  
  
 const ProfileItems=[{value: "Выйти", action: handleLogoutClick, icon: "logout"}]  
  
 function handleTransAnalyseClick(){  
 navigate('/transAnalyse');  
 }  
 function handleTransClick(){  
 navigate('/transactions');  
 }  
 function handleLogoutClick() {  
 navigate('/login');  
 localStorage.clear();  
 }  
  
 function handleAddTransClick() {  
 setMenuActive(false);  
 setAddTransactionIsOpened(true);  
 }  
  
 useEffect(() => {  
 setIsLoading(true);  
 const fetchData = (async () => {  
 const token = localStorage.getItem('token');  
 if (!token) {  
 console.error('Token not found in localStorage');  
 return;  
 }  
 try {  
 const {data: response} = await axios.get('http://localhost:5000/auth/user', {  
 headers: {  
 Authorization: `Bearer ${token}`,  
 },  
 });  
  
 await setUser({  
 \_id: response.\_id,  
 lastname: response.lastname,  
 firstname: response.firstname,  
 email: response.email,  
 role: response.roles[0],  
 balance: response.balance,  
 });  
  
 const transactionsResponse = await fetch('http://localhost:5000/user/getTransactions', {  
 method: 'POST',  
 headers: {'Content-Type': 'application/json'},  
 body: JSON.stringify({userID: response.\_id}),  
 });  
  
 const transactionsData = await transactionsResponse.json();  
 setTransactions(transactionsData);  
  
  
 } catch (error) {  
 console.error(error.message);  
 }  
  
 setIsLoading(false);  
 });  
 fetchData();  
 }, []);  
  
 function getDateValue(dateString) {  
 return new Date(dateString).getTime();  
 }  
 useEffect(()=>{  
 const srtdTransactions = [...transactions].sort(function(a, b) {  
 return getDateValue(b.dateOfTransaction) - getDateValue(a.dateOfTransaction);  
 });  
 setSortedTransactions(srtdTransactions);  
 },[transactions])  
  
 function toggleMenu() {  
 setMenuActive(!menuActive);  
 }  
 function toggleProfile(){  
 setProfileActive(!profileActive);  
 }  
 function toggleModalFilter(){  
 setModalFilterIsOpened(!modalFilterIsOpened);  
 }  
  
 const[standartSet, setStandartSet] = useState([]);  
 function getDateValue(dateString) {  
 return new Date(dateString).getTime();  
 }  
 useEffect(()=>{  
 const srtdTransactions = [...transactions].sort(function(a, b) {  
 return getDateValue(b.dateOfTransaction) - getDateValue(a.dateOfTransaction);  
 });  
 setStandartSet(srtdTransactions);  
 },[transactions])  
  
  
 return (  
 <div className="Users">  
 <header>  
 <div className="MainBar">  
 <h1 className="logo" onClick={toggleMenu}>CashR</h1>  
 <div className="userName" onClick={toggleProfile}>  
 <div className="projectIcon">  
 <span className="material-symbols-outlined">person</span>  
 </div>  
 <h1>{user.lastname} {user.firstname}</h1>  
 </div>  
 </div>  
 </header>  
  
  
 <div className="mainContent">  
 <div className="filterRow">  
 <h2>Баланс:</h2>  
 <h2>Общий оборот за период:</h2>  
 <h2>Общий расход за период:</h2>  
 <h2>Общий доход за период:</h2>  
 <span onClick={toggleModalFilter} className="material-symbols-outlined">filter\_list</span>  
 </div>  
 <div className="resultRow">  
 <h2>{user.balance}р.</h2>  
 <h2>{total}</h2>  
 <h2>{expensesSum}</h2>  
 <h2>{incomesSum}</h2>  
 </div>  
  
 </div>  
 <div className="PieCharts">  
 <IncomePieChart sortedTransactions={sortedTransactions}/>  
 <OutcomePieChart sortedTransactions={sortedTransactions}/>  
 </div>  
  
 <div className="PieBars">  
 <IncomePieBar sortedTransactions={sortedTransactions}/>  
 <OutcomePieBar sortedTransactions={sortedTransactions}/>  
 </div>  
 <ModalFilterRange expensesSum={expensesSum} setExpensesSum={setExpensesSum} incomesSum={incomesSum} setIncomeSum={setIncomeSum} total={total} setTotal={setTotal} sortedTransactions={sortedTransactions} modalFilterIsOpened={modalFilterIsOpened} setModalFilterIsOpened={setModalFilterIsOpened} setSortedTransactions={setSortedTransactions} transactions={transactions}/>  
 <ModalAddTrans come={come} setUser={setUser} user={user} setTransactions={setTransactions}  
 setIsLoading={setIsLoading} setCome={setCome} typesOfIncomes={typesOfIncomes}  
 typesOfOutcomes={typesOfOutcomes} setAddTransactionIsOpened={setAddTransactionIsOpened}  
 addTransactionIsOpened={addTransactionIsOpened}/>  
 <Menu active={menuActive} setActive={setMenuActive} action={true} header={"Главное меню"}  
 items={MenuItems}/>  
 <ProfileMenu items={ProfileItems} userBalance={user.balance} userEmail={user.email} active={profileActive} setActive={setProfileActive} action={true} header={"Профиль"}/>  
 <Backdrop open={isLoading}>  
 <CircularProgress/>  
 </Backdrop>  
  
 <div id="addGap"></div>  
 <footer>  
 <a href="https://vk.com/fanis\_ng" target="\_blank"><i className="fa-brands fa-vk"></i></a>  
 <a href="https://t.me/fanis\_ng" target="\_blank"><i className="fa-brands fa-telegram"></i></a>  
 <a href="https://www.youtube.com/@fanisnigamadyanov8262/featured" target="\_blank"><i className="fa-brands fa-youtube"></i></a>  
 </footer>  
 </div>  
 );  
};  
  
export default TransAnalyse;

Menu.jsx

import React from 'react';  
import './Menu.css';  
const Menu = ({header, items, active, setActive, action}) => {  
 return (  
 <div className={active?'menu active':'menu'} onClick={()=>setActive(false)}>  
 <div className="blur"/>  
 <div className="menu\_\_content" onClick={e=>e.stopPropagation()}>  
 <div className="menu\_\_header">{header}</div>  
 <ul>  
 {items.map(item=>  
 <li>  
 <p onClick={item.action}>{item.value}</p>  
 {item.icon==="coin"&& <span className="material-symbols-outlined">payments</span>}  
 {item.icon==="logout"&&<span className="material-symbols-outlined">logout</span>}  
 {item.icon==="trans"&&<span className="material-symbols-outlined">currency\_exchange</span>}  
 {item.icon==="analyse"&&<span className="material-symbols-outlined">leaderboard</span>}  
  
 </li>  
 )}  
 </ul>  
 </div>  
 </div>  
 );  
};  
  
export default Menu;

ProfileMenu.jsx

import React from 'react';  
import './Menu.css';  
const ProfileMenu = ({header, items, active, setActive, action, userBalance, userEmail}) => {  
 return (  
 <div className={active?'menu active':'menu'} onClick={()=>setActive(false)}>  
 <div className="blur"/>  
 <div className="menu\_\_content" onClick={e=>e.stopPropagation()}>  
 <div className="menu\_\_header">{header}</div>  
 <div id="balance">  
 <h1>Баланс:</h1>  
 <h1>{userBalance}р.</h1>  
 </div>  
 <div id="email">  
 <h1>Почта:</h1>  
 <h1>{userEmail}</h1>  
 </div>  
 <br/>  
 <ul>  
 {items.map(item=>  
 <li>  
 <p onClick={item.action}>{item.value}</p>  
 {item.icon==="logout"&&<span className="material-symbols-outlined">logout</span>}  
 </li>  
 )}  
 </ul>  
 </div>  
 </div>  
 );  
};  
  
export default ProfileMenu;

ModaAddTrans.jsx

import React, {useEffect, useState} from 'react';  
import './Modal.css'  
import axios from "axios";  
import SingleSelect from "../DropDown/DropDownSingle";  
import MyDatePicker from "../DatePicker/DatePicker";  
  
const ModalAddTrans = ({setAddTransactionIsOpened, setUser, setTransactions, setIsLoading, user, setCome, come, addTransactionIsOpened, typesOfIncomes, typesOfOutcomes}) => {  
  
 const[typesOfComes, setTypesOfComes] = useState([]);  
 const[selectedType, setSelectedType] = useState("");  
 const[sumOfTrans, setSumOfTrans] = useState();  
 const [startDate, setStartDate] = useState(new Date());  
  
  
 useEffect(()=>{  
 if(come==="Income"){  
 setTypesOfComes(typesOfIncomes);  
 }  
 else if(come ==="Outcome"){  
 setTypesOfComes(typesOfOutcomes);  
 }  
 }, [come])  
  
 async function handleCreateTransClick(){  
 if(!come){  
 alert("Выберите, это доход или расход!");  
 return;  
 }  
 else if(come==="Income"&&selectedType===""){  
 alert("Выберите тип дохода!")  
 return;  
 }  
 else if(come==="Outcome"&&selectedType===""){  
 alert("Выберите тип расхода!")  
 return;  
 }  
 else if(!sumOfTrans){  
 alert("Введите сумму транзакции!")  
 return;  
 }  
  
 let transaction;  
 if (come === "Outcome") {  
 transaction = {  
 userID: user.\_id,  
 come: come,  
 valueOfTransaction: -sumOfTrans,  
 typeOfTransaction: selectedType,  
 dateOfTransaction: startDate  
 };  
 } else if (come === "Income") {  
 transaction = {  
 userID: user.\_id,  
 come: come,  
 valueOfTransaction: sumOfTrans,  
 typeOfTransaction: selectedType,  
 dateOfTransaction: startDate  
 };  
 }  
  
 setIsLoading(true);  
 await fetch('http://localhost:5000/user/createTransaction', {  
 method: 'POST',  
 headers: {'Content-Type': 'application/json'},  
 body: JSON.stringify(transaction)  
 })  
 .then(response => response.json())  
 .then(data => {  
 if (data){  
 alert(data.message);  
 }  
 })  
 .catch(error => console.error(error));  
  
 try{  
 const transactionsResponse = await fetch('http://localhost:5000/user/getTransactions', {  
 method: 'POST',  
 headers: { 'Content-Type': 'application/json' },  
 body: JSON.stringify({ userID: user.\_id }),  
 });  
  
 const transactionsData = await transactionsResponse.json();  
 setTransactions(transactionsData);  
  
 }  
 catch(error){  
 console.error(error.message);  
 }  
 const token = localStorage.getItem('token');  
 if (!token) {  
 console.error('Token not found in localStorage');  
 return;  
 }  
  
 const { data: response } = await axios.get('http://localhost:5000/auth/user', {  
 headers: {  
 Authorization: `Bearer ${token}`,  
 },  
 });  
  
 await setUser({  
 ...user,  
 balance: response.balance,  
 });  
  
 setStartDate(new Date())  
 setIsLoading(false);  
 }  
  
 return (  
 <div className={addTransactionIsOpened ? "modal active" : "modal"} onClick={function () { setAddTransactionIsOpened(false)}}>  
 <div className="modal\_\_content" onClick={e => e.stopPropagation()}>  
  
 <h1>Выберите тип транзакции:</h1>  
 <div className="come">  
 <div className="Income">  
 <input  
 type="radio"  
 name="come"  
 id="Income"  
 value="Income"  
 onChange={(e) => setCome(e.target.value)}  
 />  
 <label htmlFor="Income">Доход</label>  
 </div>  
 <div className="Outcome">  
 <input  
 type="radio"  
 name="come"  
 id="Outcome"  
 value="Outcome"  
 onChange={(e) => setCome(e.target.value)}  
 />  
 <label htmlFor="Outcome">Расход</label>  
 </div>  
 </div>  
 <br/>  
 <h1>Выберите тип транзакции:</h1>  
 <SingleSelect typesOfComes={typesOfComes} setSelectedType={setSelectedType} selectedType={selectedType}/>  
 <br/>  
  
 <h1>Введите сумму транзакции:</h1>  
 <div className="amountOfTrans">  
 <input type="number" placeholder={"Введите сумму"} onChange={(e) => setSumOfTrans(e.target.value)} required />  
 </div>  
  
 <h1>Введите дату транзакции:</h1>  
 <div className="dateOfTrans">  
 <MyDatePicker wrapperClassName="datePicker" startDate={startDate} setStartDate={setStartDate}/>  
 </div>  
 <div className="addButton">  
 <button onClick={handleCreateTransClick}>Создать</button>  
 </div>  
 </div>  
 </div>  
 );  
};  
export default ModalAddTrans;

ModalChange.jsx

import React, {useEffect, useState} from 'react';  
import './Modal.css'  
import axios from "axios";  
const ModalChange = ({changeButtonClicked, setUsers, setIsLoading, selectedUserID, changeUser, setSelectedUserEmail, setSelectedUserLastName, setSelectedUserFirstName, setChangeButtonClicked, selectedUserFirstName, selectedUserLastName, selectedUserEmail}) => {  
  
 function handleCancelClick(){  
 setChangeButtonClicked(false);  
 }  
  
 async function handleChangeClick(){  
 if(selectedUserFirstName===""){  
 alert("Введите фамилию пользователя!");  
 return;  
 }  
 else if(selectedUserLastName===""){  
 alert("Введите имя пользователя!");  
 return;  
 }  
 else if(selectedUserEmail===""){  
 alert("Введите почту пользователя!");  
 return;  
 }  
  
 setIsLoading(true);  
 const user={id: selectedUserID, firstname:selectedUserFirstName, lastname: selectedUserLastName, email:selectedUserEmail};  
 await fetch('http://localhost:5000/admin/updateUser', {  
 method: 'PUT',  
 headers: {'Content-Type': 'application/json'},  
 body: JSON.stringify(user)  
 })  
 .then(response => response.json())  
 .then(data => {  
 alert(data.message);  
 })  
 .catch(error => console.error(error))  
  
 try {  
 const {data: response} = await axios.get('http://localhost:5000/admin/users');  
 setUsers(response);  
 } catch (error) {  
 console.error(error.message);  
 }  
  
 setIsLoading(false);  
 }  
  
 return (  
 <div className={changeButtonClicked ? "modal active" : "modal"} onClick={function () { setChangeButtonClicked(false)}}>  
 <div className="modal\_\_content" onClick={e => e.stopPropagation()}>  
 <h1>Введите новые данные для пользователя</h1>  
 <div className="userFirstName">  
 <h1>Фамилия пользователя:</h1>  
 <input type="text" value={selectedUserFirstName} placeholder={"Введите фамилию пользователя"} onChange={(e) => setSelectedUserFirstName(e.target.value)} required />  
 </div>  
 <div className="userLastName">  
 <h1>Имя пользователя:</h1>  
 <input type="text" value={selectedUserLastName} placeholder={"Введите имя пользователя"} onChange={(e) => setSelectedUserLastName(e.target.value)} required />  
 </div>  
 <div className="userEmail">  
 <h1>Почта пользователя:</h1>  
 <input type="text" value={selectedUserEmail} placeholder={"Введите почту пользователя"} onChange={(e) => setSelectedUserEmail(e.target.value)} required />  
 </div>  
 <div className="buttons">  
 <button onClick={handleChangeClick}>Изменить</button>  
 <button onClick={handleCancelClick}>Отмена</button>  
 </div>  
 </div>  
 </div>  
 );  
};  
export default ModalChange;

ModalDelete.jsx

import React, {useEffect, useState} from 'react';  
import './Modal.css'  
import axios from "axios";  
const Modal = ({deleteButtonClicked, setDeleteButtonClicked, deleteUser}) => {  
  
 function handleClick(){  
 setDeleteButtonClicked(false);  
 }  
  
 return (  
 <div className={deleteButtonClicked ? "modal active" : "modal"} onClick={function () { setDeleteButtonClicked(false)}}>  
 <div className="modal\_\_content" onClick={e => e.stopPropagation()}>  
 <h1>Вы действительно хотите удалить пользователя?</h1>  
 <div className="buttons">  
 <button onClick={deleteUser}>Удалить</button>  
 <button onClick={handleClick}>Отмена</button>  
 </div>  
 </div>  
 </div>  
 );  
};  
export default Modal;

ModalFilter.jsx

import React, {useEffect, useState} from 'react';  
import './Modal.css'  
import axios from "axios";  
import MultiSelect from "../DropDown/DropDownMulti";  
import MyDatePickerRange from "../DatePicker/DatePickerRange";  
const ModalFilter = ({setModalFilterIsOpened, modalFilterIsOpened, transactions, standartSet, setStandartSet, sortedTransactions, setSortedTransactions, leftBorder, rightBorder, setLeftBorder, setRightBorder}) => {  
  
 const [selectedTypes, setSelectedTypes] = useState([]);  
 const [startDate, setStartDate] = useState(new Date("2023-01-01"));  
 const [endDate, setEndDate] = useState(new Date("2040-01-01"));  
 const [multiSelectIsOpened, setMultiSelectIsOpened] = useState(false);  
  
 function filterByValue(transactions, leftBorder, rightBorder) {  
 return transactions.filter((transaction) => {  
 return parseInt(transaction.valueOfTransaction) >= leftBorder && parseInt(transaction.valueOfTransaction)<= rightBorder;  
 });  
 }  
 function filterByTypeOfValue(transactions, selectedTypes) {  
 const selectedValues = selectedTypes.map((selectedType) => selectedType.value);  
 return transactions.filter((transaction) => {  
 return selectedValues.includes(transaction.typeOfTransaction);  
 });  
 }  
  
 /\*const srtdTransactions = [...transactions].sort(function(a, b) {  
 return getDateValue(b.dateOfTransaction) - getDateValue(a.dateOfTransaction);  
 });  
 setSortedTransactions(srtdTransactions);  
  
 function getDateValue(dateString) {  
 return new Date(dateString).getTime();  
 }\*/  
 function filterByDate(transactions, startDate, endDate) {  
 const filteredTransactions = transactions.filter((transaction) => {  
 const transactionDate = new Date(transaction.dateOfTransaction).getTime();  
 return transactionDate >= startDate && transactionDate <= endDate;  
 });  
  
 const sortedTransactions = filteredTransactions.sort((a, b) => {  
 const dateA = new Date(a.dateOfTransaction);  
 const dateB = new Date(b.dateOfTransaction);  
 return dateB - dateA;  
 });  
  
 return sortedTransactions;  
 }  
  
  
 async function handleFilterClick() {  
 const filteredByDate = await filterByDate(transactions, startDate, endDate);  
 let sortedTransactions = filteredByDate;  
  
 if (selectedTypes.length !== 0) {  
 sortedTransactions = await filterByTypeOfValue(sortedTransactions, selectedTypes);  
 }  
  
 if ((leftBorder === "" || leftBorder === undefined) && (rightBorder === "" || rightBorder === undefined)) {  
 // do nothing  
 } else if ((leftBorder === "" || leftBorder === undefined) && !(rightBorder === "" || rightBorder === undefined) || (rightBorder === "" || rightBorder === undefined) && !(leftBorder === "" || leftBorder === undefined)) {  
 alert("Заполните оба поля!");  
 } else if (leftBorder > rightBorder) {  
 alert("Левая граница не может быть больше правой!");  
 } else {  
 sortedTransactions = await filterByValue(sortedTransactions, leftBorder, rightBorder);  
 }  
  
 setSortedTransactions(sortedTransactions);  
 }  
  
  
  
  
  
  
 /\*useEffect(()=>{  
 if(selectedTypes.length===0){  
 setSortedTransactions(standartSet);  
 }  
 else{  
 setSortedTransactions(filterByTypeOfValue(sortedTransactions, selectedTypes));  
 }  
 },[selectedTypes])\*/  
  
 /\*useEffect(()=>{  
 setSortedTransactions(filterByDate(sortedTransactions, startDate, endDate));  
 },[startDate, endDate])\*/  
  
 /\*useEffect(() => {  
 if (leftBorder === "" && rightBorder === "") {  
 setSortedTransactions(standartSet);  
 } else {  
 setSortedTransactions(filterByValue(sortedTransactions, leftBorder, rightBorder));  
 }  
 }, [leftBorder, rightBorder]);\*/  
  
  
  
 return (  
 <div className={modalFilterIsOpened ? "modal active" : "modal"} onClick={function () { setModalFilterIsOpened(false)}}>  
 <div className="modal\_\_content" onClick={e => e.stopPropagation()}>  
 <h1>Введите диапазон дат:</h1>  
  
 <div id="datePickerRange">  
 <MyDatePickerRange startDate={startDate} setStartDate={setStartDate} endDate={endDate} setEndDate={setEndDate} className="hidden"/>  
 </div>  
  
 <h1>Введите тип транзакции:</h1>  
 <MultiSelect selectedTypes={selectedTypes} setSelectedTypes={setSelectedTypes} onClick={function (){setMultiSelectIsOpened(!multiSelectIsOpened)}}/>  
  
 <h1>Введите диапазон значений транзакции:</h1>  
 <div className="rangeOfSum">  
 <input type="number" value={leftBorder} placeholder={"Введите левую границу"} onChange={(e) => {setLeftBorder(e.target.value)}} required />  
 <input type="number" value={rightBorder} placeholder={"Введите правую границу"} onChange={(e) => {setRightBorder(e.target.value)}} required />  
 </div>  
  
 <div className="addButton">  
 <button onClick={handleFilterClick}>Отфильтровать</button>  
 </div>  
 </div>  
 </div>  
 );  
};  
export default ModalFilter;

ModalFilterRange.jsx

import React, {useEffect, useState} from 'react';  
import './Modal.css'  
import axios from "axios";  
import MultiSelect from "../DropDown/DropDownMulti";  
import MyDatePickerRange from "../DatePicker/DatePickerRange";  
const ModalFilterRange = ({setModalFilterIsOpened, expensesSum, incomesSum, setIncomeSum, total, setTotal, setExpensesSum, modalFilterIsOpened, transactions, sortedTransactions, setSortedTransactions}) => {  
  
 const [startDate, setStartDate] = useState(new Date("2023-01-01"));  
 const [endDate, setEndDate] = useState(new Date("2040-01-01"));  
  
  
 let values = sortedTransactions.map(transaction => transaction.valueOfTransaction);  
  
 let expenses = values.filter(transaction => transaction < 0);  
 let incomes = values.filter(transaction => transaction > 0);  
  
 setExpensesSum(expenses.reduce((total, amount) => total + amount, 0));  
 setIncomeSum(incomes.reduce((total, amount) => total + amount, 0));  
 setTotal(expensesSum + incomesSum);  
  
 function filterByDate(transactions, startDate, endDate) {  
 return transactions.filter((transaction) => {  
 const transactionDate = new Date(transaction.dateOfTransaction);  
 return transactionDate >= startDate && transactionDate <= endDate;  
 });  
 }  
  
 function handleFilterClick() {  
 setSortedTransactions(filterByDate(transactions, startDate, endDate));  
  
 setExpensesSum(expenses.reduce((total, amount) => total + amount, 0));  
 setIncomeSum(incomes.reduce((total, amount) => total + amount, 0));  
 setTotal(expensesSum + incomesSum);  
  
 values = sortedTransactions.map(transaction => transaction.valueOfTransaction);  
  
 expenses = values.filter(transaction => transaction < 0);  
 incomes = values.filter(transaction => transaction > 0);  
 }  
  
 return (  
 <div className={modalFilterIsOpened ? "modal active" : "modal"} onClick={function () { setModalFilterIsOpened(false)}}>  
 <div className="modal\_\_content" onClick={e => e.stopPropagation()}>  
 <h1>Введите диапазон дат:</h1>  
  
 <div id="datePickerRange">  
 <MyDatePickerRange startDate={startDate} setStartDate={setStartDate} endDate={endDate} setEndDate={setEndDate} className="hidden"/>  
 </div>  
  
  
 <div className="addButton">  
 <button onClick={handleFilterClick}>Отфильтровать</button>  
 </div>  
 </div>  
 </div>  
 );  
};  
export default ModalFilterRange;

IncomePieCharts.jsx

import React, { useEffect, useState } from "react";  
import {VictoryBar, VictoryPie} from "victory";  
import { CSSTransition } from "react-transition-group";  
import "../MainPage/MainPage.css";  
  
const IncomePieBar = ({ sortedTransactions }) => {  
 const types = {};  
  
 if (  
 sortedTransactions &&  
 sortedTransactions.length > 0 &&  
 sortedTransactions !== undefined  
 ) {  
 sortedTransactions.forEach((transaction) => {  
 if (transaction.valueOfTransaction > 0) {  
 if (types[transaction.typeOfTransaction]) {  
 types[transaction.typeOfTransaction] += transaction.valueOfTransaction;  
 } else {  
 types[transaction.typeOfTransaction] = transaction.valueOfTransaction;  
 }  
 }  
 });  
  
 const data = Object.keys(types).map((type, index) => ({  
 x: type,  
 y: types[type],  
 label: `${type}`,  
 }));  
  
 const [showChart, setShowChart] = useState(false);  
  
 useEffect(() => {  
 const timer = setTimeout(() => {  
 setShowChart(true);  
 }, 1000);  
 return () => clearTimeout(timer);  
 }, []);  
  
 return (  
 <div>  
 <h1>Столбцы доходов:</h1>  
 {data.length > 0 ? (  
 <CSSTransition  
 in={showChart}  
 timeout={500}  
 classNames="fade"  
 unmountOnExit  
 >  
 <VictoryBar  
 data={data}  
 labelRadius={100}  
 style={{  
 labels: { fontSize: 20, fill: "white" },  
 data: {  
 fill: ({ datum }) => {  
 const greenValue = Math.floor(  
 (datum.y / Math.max(...data.map((d) => d.y))) \* 255  
 );  
 return `rgb(0, ${greenValue}, 0)`;  
 },  
 },  
 }}  
 animate={{ duration: 1000 }}  
 />  
 </CSSTransition>  
 ) : null}  
 </div>  
 );  
 } else {  
 return null;  
 }  
};  
  
export default IncomePieBar;

OutcomePieCharts.jsx

import React, { useEffect, useState } from "react";  
import { VictoryPie } from "victory";  
import { CSSTransition } from "react-transition-group";  
import "../MainPage/MainPage.css";  
  
const OutcomePieChart = ({ sortedTransactions }) => {  
 const types = {};  
  
 sortedTransactions.forEach((transaction) => {  
 if (transaction.valueOfTransaction < 0) {  
 if (types[transaction.typeOfTransaction]) {  
 types[transaction.typeOfTransaction] += transaction.valueOfTransaction;  
 } else {  
 types[transaction.typeOfTransaction] = transaction.valueOfTransaction;  
 }  
 }  
 });  
  
 const data = Object.keys(types).map((type, index) => ({  
 x: type,  
 y: -types[type],  
 label: `${type}`,  
 }));  
  
 const [showChart, setShowChart] = useState(false);  
  
 useEffect(() => {  
 const timer = setTimeout(() => {  
 setShowChart(true);  
 }, 1000);  
 return () => clearTimeout(timer);  
 }, []);  
  
 return (  
 <div>  
 <h1>Диаграмма доходов:</h1>  
 <CSSTransition  
 in={showChart}  
 timeout={500}  
 classNames="fade"  
 unmountOnExit  
 >  
 <VictoryPie  
 data={data}  
 colorScale={["#FFC300", "#FF5733", "#C70039"]}  
 labelRadius={100}  
 style={{ labels: { fontSize: 20, fill: "white" } }}  
 animate={{ duration: 1000 }}  
 />  
 </CSSTransition>  
 </div>  
 );  
};  
  
export default OutcomePieChart;

PrivateRoute.jsx

import React from 'react';  
import { Navigate, Outlet } from 'react-router-dom';  
  
const PrivateRoute = () => {  
 const token = localStorage.getItem('token');  
 const role = localStorage.getItem('role');  
  
 if (!token) {  
 return <Navigate to="/login" />;  
 }  
 else if(role!=='Admin'){  
 return <Navigate to="/login" />;  
 }  
 else {  
 return <Outlet />;  
 }  
};  
  
export default PrivateRoute;

PrivateRouteUser.jsx

import React from 'react';  
import { Navigate, Outlet } from 'react-router-dom';  
  
const PrivateRouteUser = () => {  
 const token = localStorage.getItem('token');  
 const role = localStorage.getItem('role');  
  
 if (!token) {  
 return <Navigate to="/login" />;  
 }  
 else if(role!=='User'){  
 return <Navigate to="/login" />;  
 }  
 else {  
 return <Outlet />;  
 }  
};  
  
export default PrivateRouteUser;

Login.jsx

import React, { useState } from "react";  
import {useNavigate} from 'react-router-dom';  
import "./AuthReg.css";  
  
const Login = (props) => {  
 const navigate = useNavigate();  
 const [email, setEmail] = useState("");  
 const [password, setPassword] = useState("");  
  
 function handleChangeEmail(event) {  
 setEmail(event.target.value);  
 const { value } = event.target.elements.login;  
 }  
  
 function handleChangePassword(event) {  
 setPassword(event.target.value);  
 }  
  
 async function handleSubmit(event) {  
 event.preventDefault();  
 const user = {email, password};  
 await fetch('http://localhost:5000/auth/login', {  
 method: 'POST',  
 headers: {'Content-Type': 'application/json'},  
 body: JSON.stringify(user),  
 })  
 .then(response => response.json())  
 .then(data => {  
 if (data.token && data.role[0] ==="Admin") {  
 localStorage.setItem('token', data.token);  
 localStorage.setItem('role', data.role[0]);  
  
 alert(data.message);  
 const timer = setTimeout(() => {  
 navigate('/users');  
 }, 1000);  
 }  
 else if(data.token && data.role[0] ==="User"){  
 localStorage.setItem('token', data.token);  
 localStorage.setItem('role', data.role[0]);  
 alert(data.message);  
 navigate("/transactions");  
 }  
 else{  
 alert(data.message);  
 }  
 })  
 .catch(error => console.error(error));  
 }  
  
 return (  
 <div className="auth\_reg">  
 <div className="title\_text">  
 <h1>Авторизация</h1>  
 </div>  
 <form className="auth\_reg\_form" onSubmit={handleSubmit}>  
 <input type="email" value={email} placeholder={"Введите почту"} onChange={(e) => setEmail(e.target.value)} required />  
 <input type="password" value={password} placeholder={"Введите пароль"} onChange={(e) => setPassword(e.target.value)} required />  
 <button type="submit">Авторизоваться</button>  
 </form>  
 </div>  
 );  
}  
export default Login;

Registration.jsx

import React, { useState } from "react";  
import "./AuthReg.css";  
import {useNavigate} from 'react-router-dom';  
  
const Registration = () => {  
 const navigate = useNavigate();  
 const [email, setEmail] = useState("");  
 const [firstName, setFirstName] = useState("");  
 const [lastName, setLastName] = useState("");  
 const [password, setPassword] = useState("");  
 function handleChangeFirstName(event) {  
 setFirstName(event.target.value);  
 }  
  
 function handleChangeLastName(event) {  
 setLastName(event.target.value);  
 }  
 function handleChangeEmail(event) {  
 setEmail(event.target.value);  
 }  
  
 function handleChangePassword(event) {  
 setPassword(event.target.value);  
 }  
  
 async function handleSubmit(event) {  
 event.preventDefault();  
 const user = {firstName, lastName, email, password};  
 await fetch('http://localhost:5000/auth/registration', {  
 method: 'POST',  
 headers: {'Content-Type': 'application/json'},  
 body: JSON.stringify(user),  
 })  
 .then(response => response.json())  
 .then(data => {  
 if (data.isRegistered) {  
 alert(data.message);  
 navigate('/login');  
 }  
 else{  
 alert(data.message);  
 }  
  
 })  
 .catch(error => console.error(error));  
 }  
  
 return (  
 <div className="auth\_reg">  
 <div className="title\_text">  
 <h1>Регистрация</h1>  
 </div>  
 <form className="auth\_reg\_form" onSubmit={handleSubmit}>  
 <input  
 type="text"  
 value={firstName}  
 placeholder={"Введите фамилию"}  
 onChange={(e) => setFirstName(e.target.value)}  
 required  
 />  
 <input  
 type="text"  
 value={lastName}  
 placeholder={"Введите имя"}  
 onChange={(e) => setLastName(e.target.value)}  
 required  
 />  
 <input  
 type="email"  
 value={email}  
 placeholder={"Введите почту"}  
 onChange={(e) => setEmail(e.target.value)}  
 required  
 />  
 <input  
 type="password"  
 value={password}  
 placeholder={"Введите пароль"}  
 onChange={(e) => setPassword(e.target.value)}  
 required  
 />  
 <button type="submit">Зарегистрироваться</button>  
 </form>  
 </div>  
 );  
};  
export default Registration;

Users.jsx

import React, {useEffect, useState} from 'react';  
import './Users.css';  
import axios from 'axios';  
import {useNavigate} from 'react-router-dom';  
import {Backdrop, CircularProgress} from '@mui/material';  
import ModalDelete from '../Modal/ModalDelete'  
import ModalChange from "../Modal/ModalChange";  
  
import {Link} from "react-router-dom";  
import Menu from "../Menu/Menu";  
  
const Users = () => {  
 const [isLoading, setIsLoading] = useState(true);  
 const [user, setUser] = useState({  
 lastname:"",  
 firstname:"",  
 email:"",  
 role:""  
 });  
 const [users, setUsers] = useState([]);  
 const[deleteButtonClicked, setDeleteButtonClicked] = useState(false);  
 const[changeButtonClicked, setChangeButtonClicked] = useState(false);  
 const[selectedUserID, setSelectedUserID] = useState();  
 const [searchQuery, setSearchQuery] = useState("");  
 const [menuActive, setMenuActive] = useState(false);  
 const[selectedUserFirstName, setSelectedUserFirstName] = useState();  
 const[selectedUserLastName, setSelectedUserLastName] = useState();  
 const[selectedUserEmail, setSelectedUserEmail] = useState();  
 const navigate = useNavigate();  
  
 const MenuItems=[{value:"Выйти", action:handleLogoutClick, icon: "logout"}];  
  
  
 const filteredUsers = users.filter(  
 (user) =>  
 (user.firstname + " " + user.lastname).toLowerCase().includes(searchQuery.toLowerCase()) ||  
 user.email.toLowerCase().includes(searchQuery.toLowerCase())  
 );  
  
 useEffect(()=>{  
 console.log(users);  
 },[users])  
  
 function handleLogoutClick(){  
 navigate('/login');  
 localStorage.clear();  
 }  
  
 useEffect(() => {  
 const fetchData = (async () => {  
 const token = localStorage.getItem('token');  
 if (!token) {  
 console.error('Token not found in localStorage');  
 return;  
 }  
 try {  
 const {data: response} = await axios.get('http://localhost:5000/auth/user', {  
 headers: {  
 Authorization: `Bearer ${token}`  
 }  
 });  
 setUser({  
 lastname: response.lastname,  
 firstname: response.firstname,  
 email: response.email,  
 role: response.roles[0]  
 });  
 } catch (error) {  
 console.error(error.message);  
 }  
  
 try {  
 const {data: response} = await axios.get('http://localhost:5000/admin/users');  
 setUsers(response);  
 } catch (error) {  
 console.error(error.message);  
 }  
  
 setIsLoading(false);  
 });  
 fetchData();  
 }, []);  
  
 useEffect(()=>{  
 try {  
 console.log(selectedUserID);  
 fetch('http://localhost:5000/admin/getUser', {  
 method: 'POST',  
 headers: {'Content-Type': 'application/json',  
 'Accept': 'application/json',  
 },  
 body: JSON.stringify({selectedUserID}),  
 })  
 .then(response => response.json())  
 .then(async data => {  
 if(data){  
 setSelectedUserFirstName(data.firstname);  
 setSelectedUserLastName(data.lastname);  
 setSelectedUserEmail(data.email);  
 }  
 })  
 .catch(error => console.error(error));  
 }catch (e){  
  
 }  
 }, [selectedUserID])  
  
 function handleDeleteClick(id) {  
 return function() {  
 handleDelete(id);  
 }  
 }  
 function handleChangeClick(id){  
 return function() {  
 handleChange(id);  
 }  
 }  
 async function handleChange(key){  
 setSelectedUserID(key);  
 setChangeButtonClicked(true);  
 }  
 function handleDelete(key) {  
 setDeleteButtonClicked(true);  
 setSelectedUserID(key);  
 }  
 async function changeUser(){  
  
 }  
 async function deleteUser(){  
 try {  
 setIsLoading(true);  
 await fetch('http://localhost:5000/admin/deleteUser', {  
 method: 'POST',  
 headers: {'Content-Type': 'application/json',  
 'Accept': 'application/json',  
 },  
 body: JSON.stringify({selectedUserID}),  
 })  
 .then(response => response.json())  
 .then(async data => {  
 if(data.isDeleted){  
 try {  
 const {data: response} = await axios.get('http://localhost:5000/admin/users');  
 setUsers(response);  
 } catch (error) {  
 console.error(error.message);  
 }  
 alert(data.message);  
 }  
 else{  
 alert(data.message);  
 }  
 })  
 .catch(error => console.error(error));  
 }catch (e){  
  
 }  
 setIsLoading(false);  
 }  
  
 return (  
 <div className="Users">  
 <header>  
 <div className="MainBar">  
 <h1 className="logo" onClick={()=>setMenuActive(!menuActive)}>CashR</h1>  
 <div className="userName">  
 <div className="projectIcon">  
 <span className="material-symbols-outlined">person</span>  
 </div>  
 <h1>{user.lastname} {user.firstname}</h1>  
 </div>  
 </div>  
 </header>  
 <Menu active={menuActive} setActive={setMenuActive} action={true} header={"Главное меню"} items = {MenuItems}/>  
 <ModalChange setUsers={setUsers} selectedUserID={selectedUserID} setIsLoading={setIsLoading} changeButtonClicked={changeButtonClicked} setChangeButtonClicked={setChangeButtonClicked} changeUser={changeUser} setSelectedUserFirstName={setSelectedUserFirstName} setSelectedUserLastName={setSelectedUserLastName} setSelectedUserEmail={setSelectedUserEmail} selectedUserFirstName={selectedUserFirstName} selectedUserLastName={selectedUserLastName} selectedUserEmail={selectedUserEmail}/>  
 <ModalDelete deleteButtonClicked={deleteButtonClicked} setDeleteButtonClicked={setDeleteButtonClicked} deleteUser={deleteUser}/>  
 <Backdrop open={isLoading}>  
 <CircularProgress/>  
 </Backdrop>  
  
 <div className="verticalAligner">  
 <h1>Список пользователей</h1>  
 <div className="searchAligner">  
 <span className="material-symbols-outlined">search</span>  
 <input  
 style={{width: '450px'}}  
 type="text"  
 placeholder="Поиск пользователей"  
 value={searchQuery}  
 onChange={(e) => setSearchQuery(e.target.value)}  
 />  
 </div>  
 </div>  
  
 <div className="userAligner">  
 <div>  
 {filteredUsers.map((user) => (  
 <div className="appUser" key={user.\_id}>  
 <h2>{user.firstname} {user.lastname}</h2>  
 <p>{user.email}</p>  
 <div id="userButtons">  
 <button onClick={handleChangeClick(user.\_id)} type="submit">Изменить</button>  
 <button onClick={handleDeleteClick(user.\_id)} type="submit">Удалить</button>  
 </div>  
 </div>  
 ))}  
 </div>  
 </div>  
 </div>  
 );  
};  
  
export default Users;