**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №5**

**по дисциплине «Web-технологии»**

Тема: **МОДУЛЬ АДМИНИСТРИРОВАНИЯ**

**ПРИЛОЖЕНИЯ «БИРЖА АКЦИЙ»**

|  |  |  |
| --- | --- | --- |
| Студент гр. 2381 |  | Богатов И.А. |
| Преподаватель |  | Беляев С.А. |

Санкт-Петербург

2025

­

**1. Общая информация**

Проект представляет собой **полноценное веб-приложение** с использованием **React** на клиентской стороне и **NestJS** на сервере. В приложении реализованы:

1. Работа с брокерами и акциями.
2. Обновление и управление данными через HTTP-запросы.
3. Реализация маршрутизации на клиенте с React Router.
4. Хранилище Redux для управления состоянием приложения.
5. WebSocket для передачи данных в реальном времени.

**2. Структура клиентской части (React)**

Файлы и их назначения:

1. **CSS-файлы**:
   * app.brokers.css, app.exchange.css, app.header.css, app.startPage.css, app.stocks.css  
     Отвечают за стилизацию соответствующих компонентов.
2. **JS-файлы**:
   * **app.brokers.js** — компонент для работы с брокерами.
   * **app.exchange.js** — отображает обмен или взаимодействие между брокерами.
   * **app.header.js** — главный заголовок (шапка) приложения.
   * **app.startPage.js** — стартовая страница приложения.
   * **app.stocks.js** — компонент для отображения и управления акциями.
   * **app.socket.js** — отвечает за реализацию WebSocket-­подключения.
   * **brokersSlice.js** — Redux slice для управления состоянием брокеров.
   * **store.js** — конфигурация хранилища Redux.
   * **index.js** — точка входа в клиентское приложение.
3. **index.js**:
   * Подключает **Redux** и **React Router**.
   * Настраивает маршруты:
     + **/** — стартовая страница.
     + **/brokers** — управление брокерами.
     + **/stocks** — просмотр акций.
     + **/exchange** — обмен акциями.
4. **Хранилище Redux**:
   * Настраивается через configureStore из @reduxjs/toolkit.
   * Редьюсер brokersSlice регистрируется под ключом **brokers**.

**3. Структура серверной части (NestJS)**

Файлы и их назначения:

1. **Контроллер (AppController)**:
   * Обрабатывает HTTP-запросы, связанные с брокерами и акциями.
   * Использует **файловую систему** (fs) для чтения и записи данных в JSON-файлы.
2. **Основные эндпоинты**:
   * **Брокеры**:
     + GET /brokers — возвращает список брокеров.
     + POST /connectNewBroker — добавляет нового брокера.
     + POST /deleteBroker — удаляет брокера.
     + POST /updateBrokersList — обновляет список брокеров.
   * **Акции**:
     + GET /stocks — возвращает список акций.
     + POST /showStocks — возвращает данные об акциях на основе символа.
     + POST /startSale — начинает торговлю с таймером и передаёт данные через **WebSocket**.
     + POST /stopSale — останавливает торговлю.
     + POST /buyStocks — покупка акций брокером.
     + POST /saleStocks — продажа акций брокером.
3. **WebSocket**:
   * Используется для передачи данных о ходе торгов в реальном времени.
   * Метод PostStartSale отправляет данные с заданным интервалом через socketService.

**4. Логика и взаимодействие**

1. **Хранение данных**:
   * Данные брокеров и акций хранятся в **JSON-файлах**:
     + brokers-list.json — список брокеров.
     + stocks-list.json — список доступных акций.
     + companies/{symbol}.json — данные о конкретной акции (цена и даты).
2. **Покупка и продажа акций**:
   * Покупка/продажа акций обновляет информацию о деньгах и портфеле брокера в файле brokers-list.json.
   * Реализована проверка наличия акций перед покупкой или продажей.
3. **Торги**:
   * При старте торгов (startSale), сервер отправляет данные о ценах акций через WebSocket с заданной скоростью (changeSpeed).
   * Метод stopSale останавливает интервал передачи данных.

**5. Интеграция Redux и React**

1. **Redux**:
   * Используется для управления состоянием брокеров.
   * Slice brokersSlice обрабатывает состояния: загрузка, обновление и управление брокерами.
2. **React Router**:
   * Обеспечивает маршрутизацию между страницами:
     + **/** — стартовая страница.
     + **/brokers** — управление брокерами.
     + **/stocks** — просмотр акций.
     + **/exchange** — обмен акциями.
3. **Компоненты**:
   * **Header**: Шапка приложения.
   * **StartPage**: Начальная страница.
   * **Brokers**: Работа с брокерами.
   * **Stocks**: Просмотр и взаимодействие с акциями.
   * **Exchange**: Обмен данными об акциях.

**6. Заключение**

Проект объединяет клиентскую и серверную часть для управления брокерами и торговлей акциями:

* **React** используется для интерфейса и маршрутизации.
* **Redux** управляет состоянием приложения.
* **NestJS** реализует серверную логику с поддержкой WebSocket.
* Данные хранятся в **JSON-файлах**, что упрощает работу без полноценной БД.

Архитектура чёткая и модульная, с возможностью масштабирования.

Приложение:







