**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №6**

**по дисциплине «Web-технологии»**

Тема:Модуль приложения «Покупка и продажа акций»

|  |  |  |
| --- | --- | --- |
| Студентка гр. 1381 |  | Васильева О. М. |
| Преподаватель |  | Беляев С.А. |

Санкт-Петербург

2023

## Цель и задачи работы.

Целью работы является изучение возможностей применения фреймворка Vue (https://v3.ru.vuejs.org/ru/) для разработки интерфейсов пользователя web-приложений и организации E2E тестирования клиентской части приложения. Для достижения поставленной цели требуется решить следующие задачи:

– разработка интерфейса web-приложения;

– создание web-сервера на основе NestJS. Подготовка web-сокетов для обновления информации о стоимости у всех клиентов;

– создание каркаса web-приложения с использованием Vue;

– разработка перечня компонентов;

– создание статической версии интерфейса;

– программирование потока изменения состояний web-приложения;

– разработка скрипта автоматического тестирования web-приложения.

**Основные теоретические сведения.**

Vue – фреймворк для создания пользовательских интерфейсов. В отличие от фреймворков-монолитов, Vue создавался пригодным для постепенного внедрения. Его ядро решает задачи уровня представления, упрощая интеграцию с другими библиотеками и существующими проектами. С другой стороны, Vue подходит и для разработки сложных одностраничных приложений.

Selenium – фреймворк тестирования web-приложений, позволяет программировать автоматизированные тесты клиентской части web-приложений, а также записывать и воспроизводить действия пользователей.

Headless-браузер – браузер без интерфейса пользователя, предназначенный для тестирования web-приложений.

## Общая формулировка задание.

Необходимо создать web-приложение, обеспечивающее использование пользователем социальной сети. Пользователь может зарегистрироваться в социальной сети. Может добавить или удалить свою фотографию, может управлять своими друзьями в социальной сети, может добавить сообщение (новость) на свою страницу, может просматривать список новостей своих друзей.

Основные требования:

Необходимо создать web-приложение, обеспечивающее работу брокера, у него есть запас денежных средств, он имеет возможность купить или продать акции (любое доступное количество), а также контролировать изменение котировок акций. В приложении должен отображаться баланс (запас денежных средств плюс стоимость акций), а также прибыль или убыток, которые он получил по каждой акции. Основные требования следующие:

1. Приложение получает исходные данные из модуля администрирования приложения «Биржа акций» в виде настроек в формате JSON-файла и в виде данных от web-сокета по изменению стоимости акций во времени.

2. В качестве сервера используется NestJS.

3. Участники торгов подключаются к приложению «Покупка и продажа акций».

4. Предусмотрена HTML-страница администратора, на которой отображается перечень участников. Для каждого участника отображается его баланс, количество акций каждого типа у каждого участника и его прибыль или убыток по каждой акции в текущий момент времени.

5. Предусмотрена HTML-страница входа в приложение, где каждый участник указывает (или выбирает из допустимых) свое имя.

6. Предусмотрена HTML-страница, на которой участнику отображаются:

– текущая имитируемая дата;

– текущая стоимость каждой из акций, выставленных на торги;

– общее количество доступных средств;

– количество, стоимость и прибыль/убыток по каждой купленной акции.

На этой же странице у брокера есть возможность:

– открыть диалоговое окно просмотра графика изменения цены каждой акции (с момента начала торгов до текущего момента) с учётом сообщений об изменении стоимости акций;

– купить/продать интересующее его количество акций.

Комментарии:

– брокер не может купить акции, если денег не хватает;

– купля/продажа происходит «мгновенно».

7. Разработаны автоматизированные тесты для проверки корректности работы клиентской части web-приложения с использованием headlessбраузера или фреймворка Selenium. Как минимум необходимо проверить, что при покупке/продаже N акций в определённую дату соответствующим образом изменяется баланс средств брокера и через некоторое время получается правильная прибыль/убыток по данной акции.

Преимуществом будет использовать Material Design Framework (https://vuetifyjs.com/en).

## Выполнение работы.

1. В качестве сервера использовался Nest. Часть функций для покупки и продажи акций, входа в приложение было прописана в файле edit.ts лабораторной работы 5.
2. В папке Puppeteer прописан main.test.ts для тестирования клиентской части приложения с использованием Jest. В данном файле прописан тест при покупке акции (см. рис. 1-3).

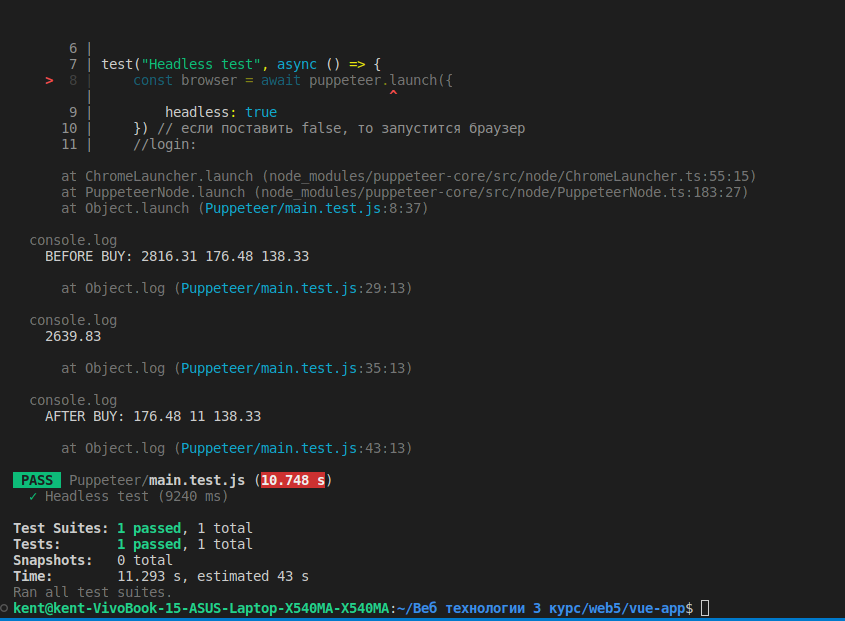


Рис. 1 — результат работы теста

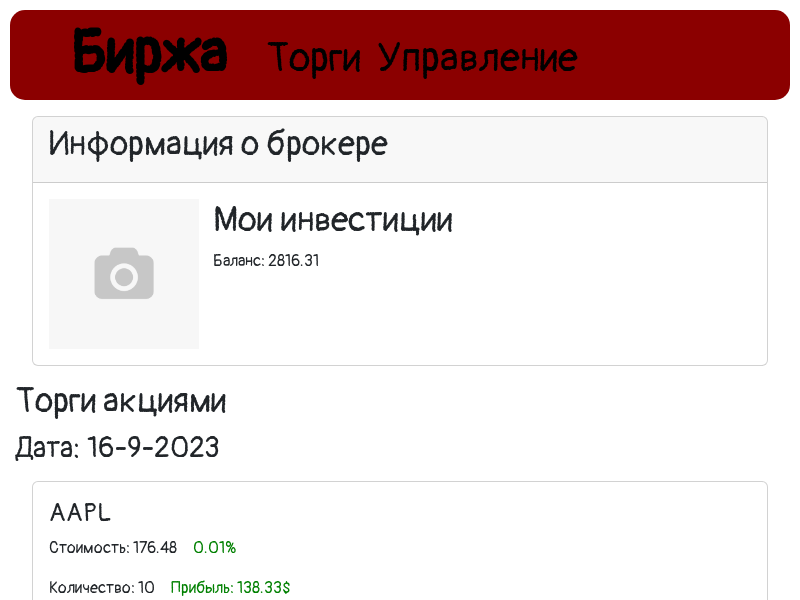


Рис. 2- Скриншот до покупки акции

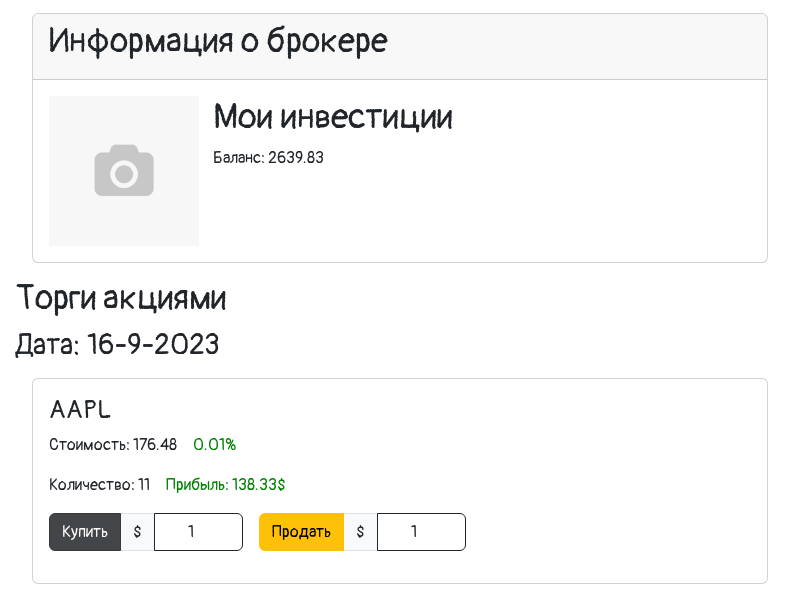


Рис. 3 — Скриншот после покупки акции

3. Маршрутизация (routes.js).

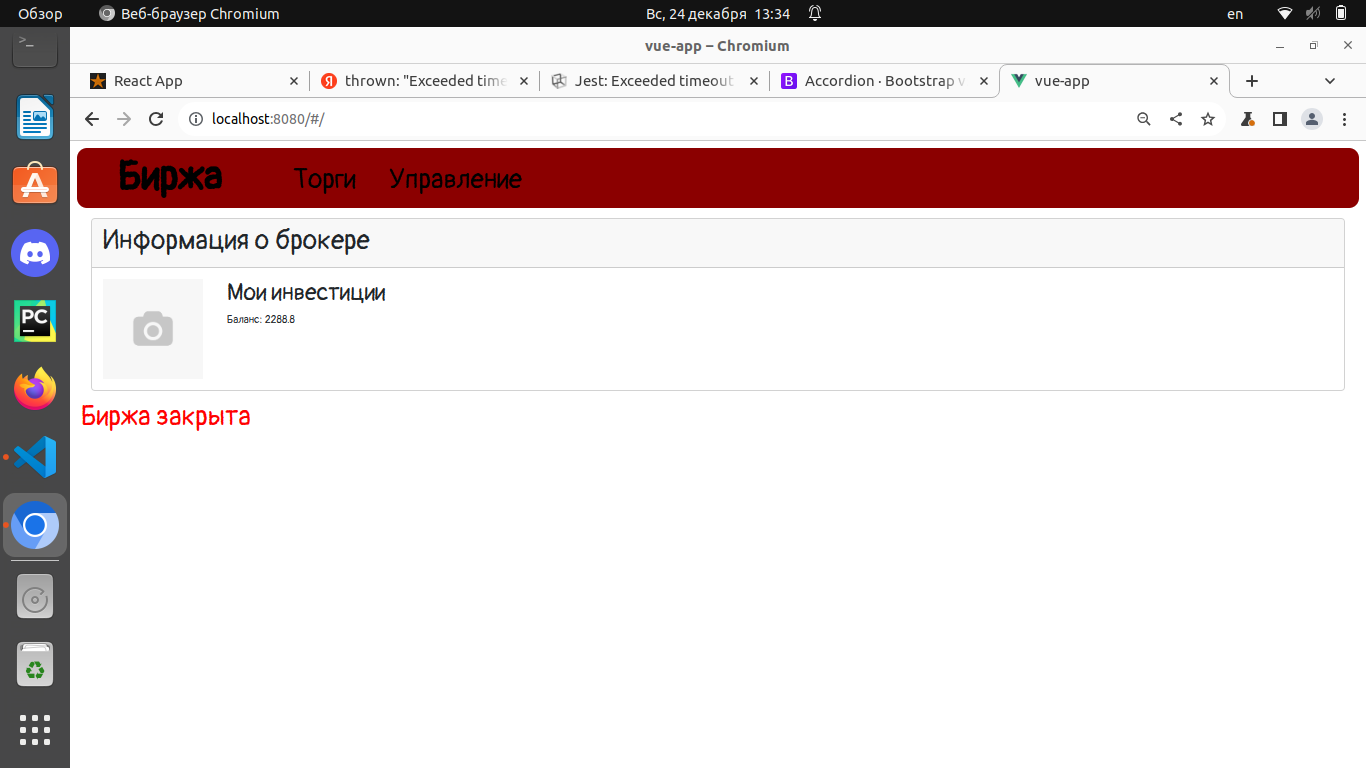
В данном файле прописаны основные пути и компоненты обрабатывающие для них.

4. Файл App.vue.

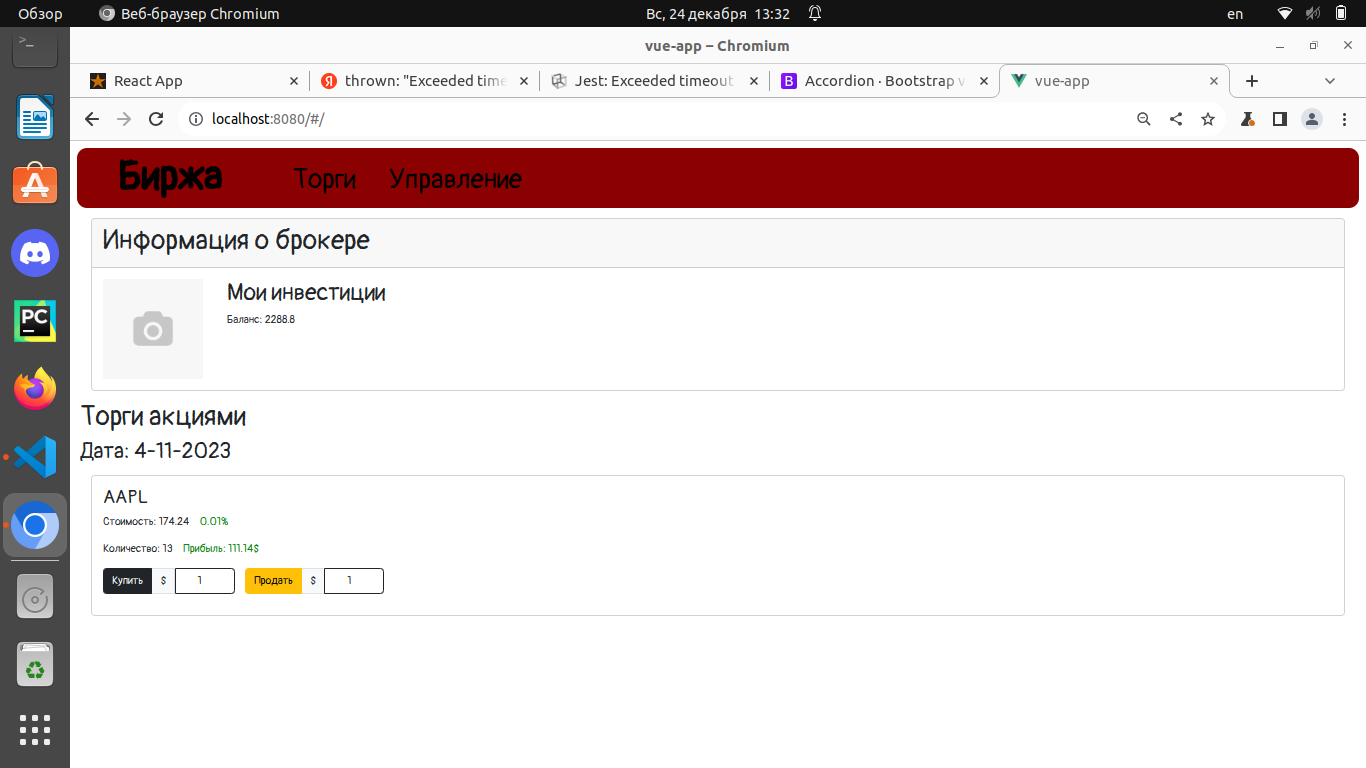
Данный файл отображает маршруты и ссылки переходов между компонентами. Если пользователь является администратором, то он видит кнопку «Управление». Также в данном файле подключается основной стиль для шрифта сайта.

5. Файл Home.vue.

В данном файле описан блок кода для главной страницы (см. рис. 4).

Рис. 4 — Главная страница (биржа закрыта).

На страница отображается карточка пользователя (аватар, имя и баланс), а также проводятся основные торги. Здесь же содержится компонент Component для отображения акций, участвующих в торгах (см. рис. 5).

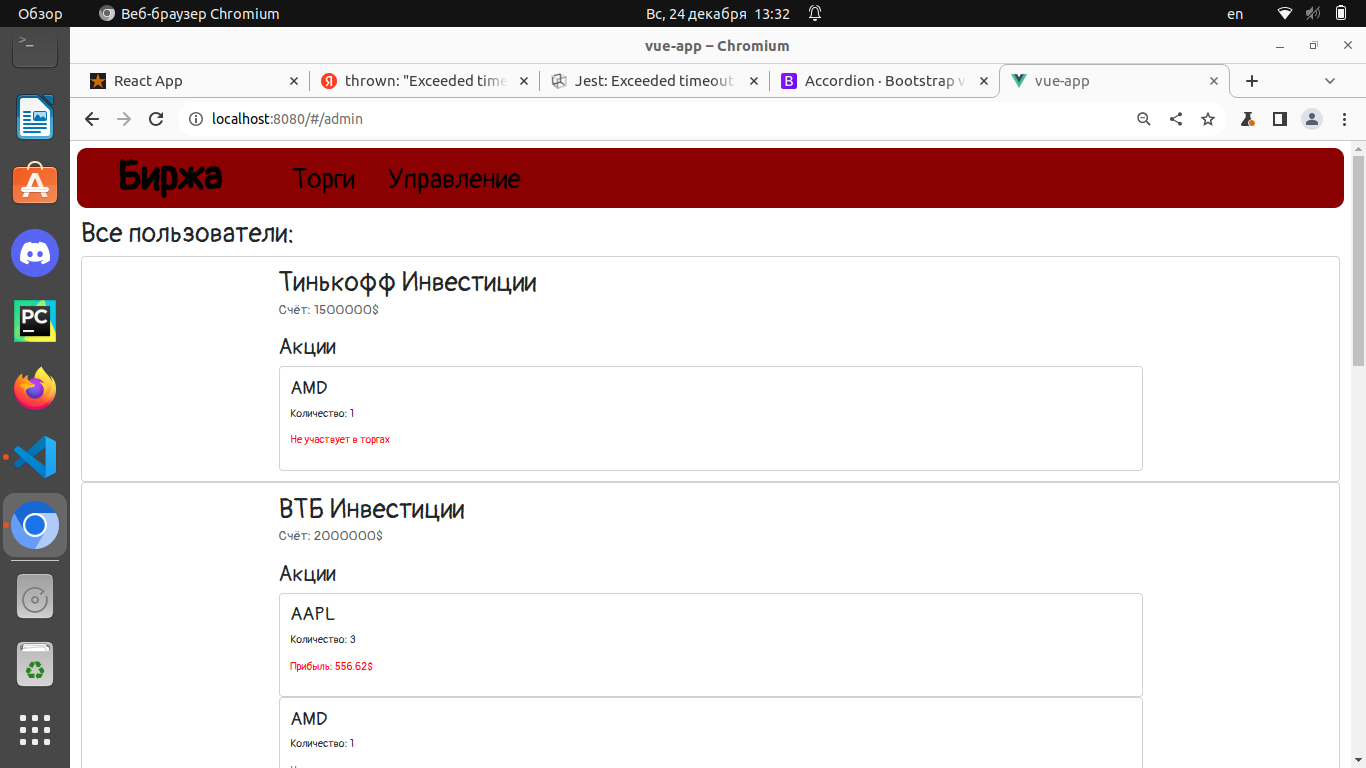
Рис. 5 — Главная страница (торги открыты).

6. Файл Component.vue.

В данном файле расположен блок кода для отображения карточки акции, участвующей в торгах. Сама карточка имеет: название акции, ее стоимость, разницу в % от предыдущей стоимости, количество данной акции у брокера, кнопки купить/продать, прибыль. Причем, если у пользователя данной акции нет, то кнопка продать не высвечивается, также не высвечивается блок количества и прибыли.

7. Файл Admin.vue.

В данном файле имеется блок для управления, который видит только администратор. В нем отображается информация о всех пользователях (см. рис. 6).

Рис. 6 — Страница управления

Для каждого пользователя отображается имя, счет и акции. Для каждой акции выводится количество и прибыль/убыль для акции.

Компонент для самих брокеров прописан в файле Block.vue ( в нем отображается имя брокера и его баланс). Для отображения акции прописан компонент в файле Sblock.vue ( в нем отображается количество данной акции и прибыль/убыль или не участвует в торгах).

8. Файлы store.js и storestocks.js.

В первом файле создается Vuex хранилище с именем "User". В этом хранилище определены состояние (state), геттеры (getters) и мутации (mutations).

* Состояние (state) хранит следующие значения:
  + nameU: значение, полученное из localStorage с ключом "user".
  + money: значение 0.
  + stocks: пустой массив.
  + admin: значение false.
  + photo: значение 5.
* Геттеры (getters) определяют, как получить данные из состояния.
  + get\_stocks(state): возвращает значение state.stocks.
  + get\_stock(state, name): итерируется по массиву state.stocks и возвращает объект со значениями set, count и cost, соответствующими переданному параметру name. Если совпадений нет, возвращает объект с set=false, count=0 и cost=0.
* Мутация (mutation) set\_user определяет, как изменить состояние.
  + set\_user(state): обновляет значение state.nameU на значение, полученное из localStorage с ключом "user". Затем делает запрос на URL "<http://localhost:4000/getbroker>" с добавленными параметрами state.nameU. Полученные данные обновляют значения state.money, state.stocks, state.admin и state.photo.

Во втором файле создается Vuex хранилище с именем "stocksS". В этом хранилище определены состояние (state), геттеры (getters) и мутации (mutations). Также в данном файле имеется сокет, который во время торгов принимает событие старта торгов (сообщение), и обновляет данные в хранилище.

## Вывод.

В ходе лабораторной работы были изучены возможности применения фреймворка Vue (https://v3.ru.vuejs.org/ru/) для разработки интерфейсов пользователя web-приложений и организации E2E тестирования клиентской части приложения.