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»

Студент гр. 1304 _____ Шаврин А.П.

Преподаватель _____ Беляев С.А.

Санкт-Петербург
2023

Цель работы.

Изучение возможностей применения фреймворка Vue (https://v3.ru.vuejs.org/ru/) для разработки интерфейсов пользователя web-приложений и организация E2E-тестирования клиентской части приложения.

Задание.

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 и 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 учетом сообщений об изменении стоимости акций;

– купить/продать интересующее его количество акций.

Комментарии:

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

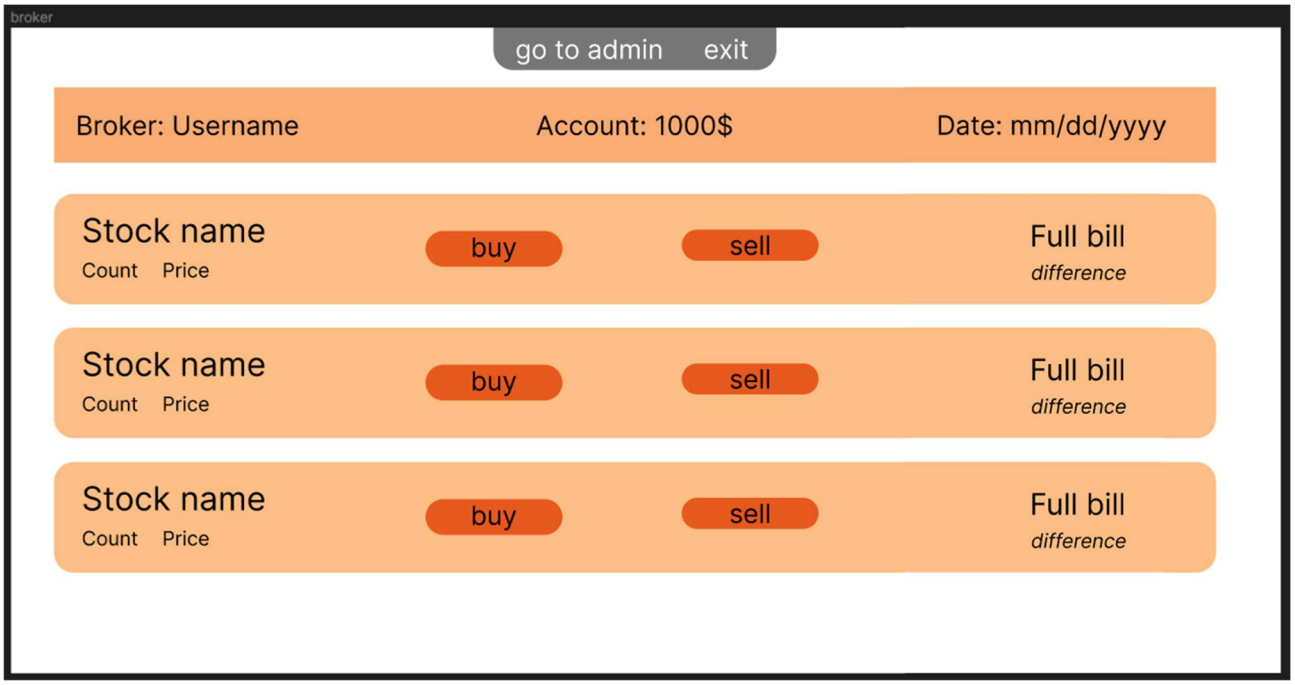
– купля/продажа происходит «мгновенно».

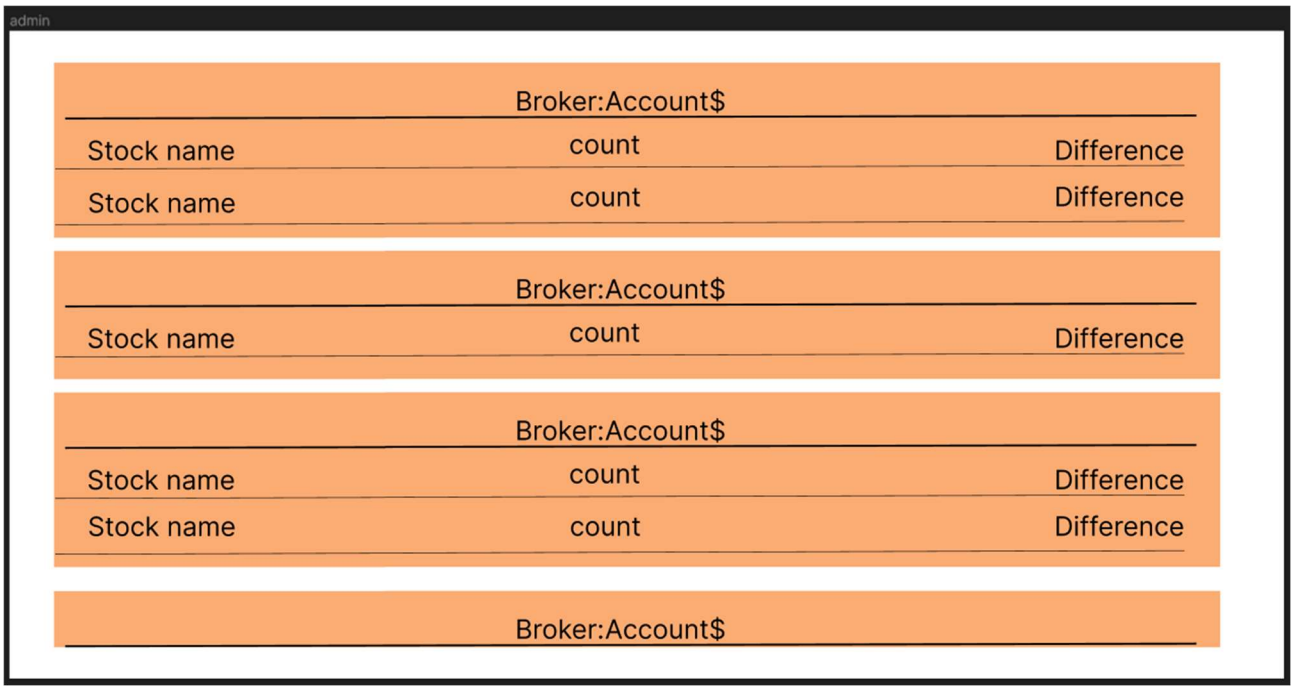
7. Разработаны автоматизированные тесты для проверки корректности работы клиентской части web-приложения с использованием headless-браузера или фреймворка Selenium. Как минимум необходимо проверить, что при покупке/продаже N акций в определенную дату соответствующим образом изменяется баланс средств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

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

Выполнение работы.

1. Созданы макеты страниц в Figma:





2. Для серверной части из прошлой лабораторной был дописан контроллер и сервис-реализующий запросы (реализованы запросы на обработку покупки и продажи акции)

```
@Post("/buyStock")
buyStock(@Body() body: any): string {
  return this.appService.buyStock(body);
}

@Post("/sellStock")
sellStock(@Body() body: any): string {
  return this.appService.sellStock(body);
}
```

Контроллер

```
buyStock(body: any): string {
  const broker = BROKERS.filter((broker)=>{
    return broker.id == body.broker_id;
  })[0];

  const stock = STOCKS.filter((stock) => {
    return stock.id == body.stock_id;
  })[0];

  const stock_price = Number(stock?.data[stock?.data?.length-1-body.index]?.Open.slice(1));

  if (broker && stock) {
    let full_price =  body.stock_count * stock_price;
    if (broker.account > full_price){
      if (!broker.stocks[stock.id]) {
        broker.stocks[stock.id] = {"count": 0, "sum": 0};
      }
      broker.stocks[stock.id].count += body.stock_count;
      broker.stocks[stock.id].sum += full_price;
      broker.account -= full_price;

      fs.writeFile('src/brokers.json', JSON.stringify(BROKERS), (err) => {
        if (err) throw err;
        console.log('The file has been saved!');
      });

      return JSON.stringify({ "mes": "success", "data": broker});
    } else {
      return JSON.stringify({ "mes": "insufficient funds" });
    }
  } else {
    return JSON.stringify({ "mes": "unknown broker" });
  }
}
```

```
sellStock(body: any): string {
  const broker = BROKERS.filter((broker)=>{
    return broker.id == body.broker_id;
  })[0];

  const stock = STOCKS.filter((stock) => {
    return stock.id == body.stock_id;
  })[0];

  const stock_price = Number(stock?.data[stock?.data?.length-1-body.index]?.Open.slice(1));

  if (broker && stock) {
    let full_price =  body.stock_count * stock_price;
    const average_sum = (broker.stocks[stock.id].sum / broker.stocks[stock.id].count) * body.stock_count;

    if (broker.stocks[stock.id]?.count >= body.stock_count){
      broker.stocks[stock.id].count -= body.stock_count;
      broker.stocks[stock.id].sum -= average_sum ;
      broker.account += full_price;

      fs.writeFile('src/brokers.json', JSON.stringify(BROKERS), (err) => {
        if (err) throw err;
        console.log('The file has been saved!');
      });

      return JSON.stringify({ "mes": "success", "data": broker});
    } else {
      return JSON.stringify({ "mes": "not enough shares" });
    }
  } else {
    return JSON.stringify({ "mes": "unknown broker" });
  }
}
```

Сервис

3. На сервере дополнен сокет сервис для уведомления всех клиентов об изменении даты и покупке или продаже акций клиентом (для этого реализован широковещательный запрос)

```
private broadcast(event: string, message: any) {
    const broadCastMessage = JSON.stringify(message);
    for (let client of this.wsClients) {
        client.emit(event, broadCastMessage);
    }
}
```

```
@SubscribeMessage("buy")
handleBuyEvent(@MessageBody() dto: any, @ConnectedSocket() client: any){
    console.log(dto)
    fetch("http://localhost:8081/buyStock", {
        method: "POST",
        body: dto,
        headers: {
            'Content-Type': 'application/json'
        }
    }).then((res)=> {
        res.json().then((answer) => {

            if (answer.mes === "success") {
                //console.log(answer.message.data)
                this.broadcast("broker_buy", answer.data)
            }
        });
    });
}

@SubscribeMessage("sell")
handleSellEvent(@MessageBody() dto: any, @ConnectedSocket() client: any){
    console.log(dto)
    fetch("http://localhost:8081/sellStock", {
        method: "POST",
        body: dto,
        headers: {
            'Content-Type': 'application/json'
        }
    }).then((res)=> {
        res.json().then((answer) => {
            if (answer.mes === "success") {
                //console.log(answer.message.data)
                this.broadcast("broker_sell", answer.data)
            }
        });
    });
}
```

Сокет сервис

4. На клиенте реализован роутер маршрутов

```vue
import { createRouter, createWebHashHistory } from 'vue-router'
import LoginComponent from './components/LoginComponent.vue'
import AdminComponent from './components/AdminComponent.vue'


const routes = [
    { path: '/', redirect: '/login'},
    { path: '/login', name: 'LoginComponent', component: LoginComponent },
    { path: '/broker/:name', name: 'BrokerComponent', component: () => import('../src/components/BrokerComponent.vue') },
    { path: '/admin', name: 'AdminComponent', component: AdminComponent }
]

const router = createRouter({
    history: createWebHashHistory(),
    routes
})

export default router
```

Клиентский роутер

5. Затем было настроено общее хранилище приложения

```vue
import {createStore} from 'vuex'


export default createStore({
    state() {
        return {
            tradingList: [],
            index: 0,
        }
    },
    mutations: {
        setTradingList(state, list){
            state.tradingList = list;
        },

        setIndex(state, newIndex) {
            state.index = newIndex;
        },
    }
})
```

Store.js

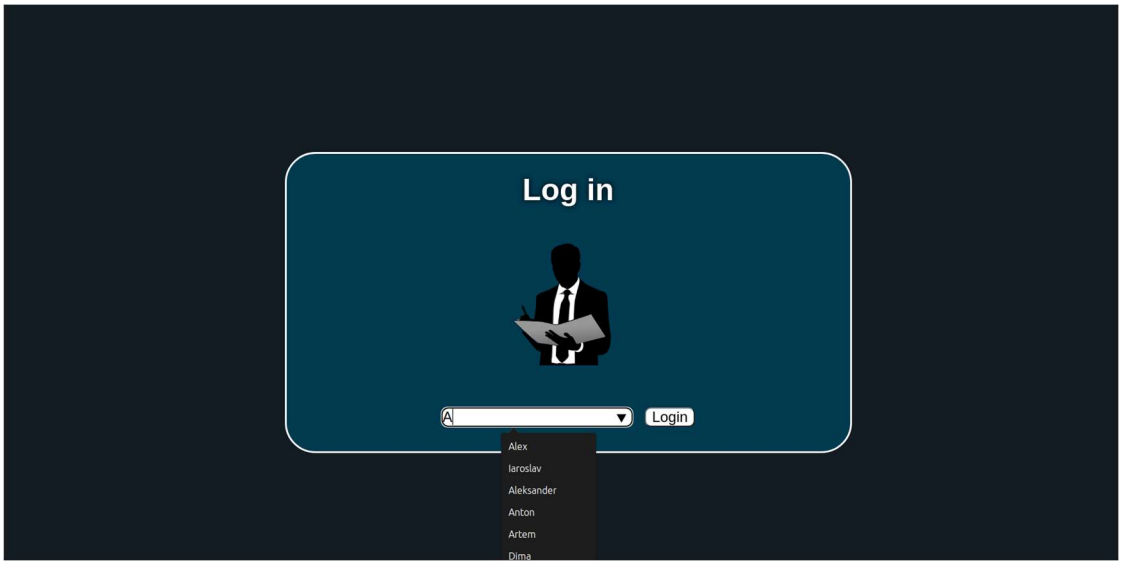
6. Организованы E2E-тесты для проверки корректности покупки и продажи с использованием Selenium

```
async function test_1() {
    let driver = await new Builder().forBrowser('chrome').build();
    try {
        await driver.get('http://localhost:8080/#/broker/Alex');
        let price = null;
        let start_account = null;
        let end_account = null;

        setTimeout(async () => {
            start_account = await driver.findElement(By.xpath("//div[@id='BrokerAccount']")).getText()
            start_account = Number(start_account.slice(9, start_account.length-1))
            console.log('start')
        }, 3000)

        setTimeout(async () => {
            await driver.findElement(By.xpath("//button[@id='APPL_buy_btn']")).click()
            console.log("buy")
        }, 4000)

        setTimeout(async () => {
            await driver.findElement(By.xpath("//input[@id='APPL_buy_inp']")).sendKeys(1)
            console.log("inp")
        }, 5000)

        setTimeout(async () => {
            await driver.findElement(By.xpath("//button[@id='APPL_buy_confirm_btn']")).click()
            price = await driver.findElement(By.xpath("//div[@id='APPL_price']")).getText()
            price = Number(price.slice(0, price.length-1))
            console.log('confirm', price)
        }, 6000)

        setTimeout(async () => {
            end_account = await driver.findElement(By.xpath("//div[@id='BrokerAccount']")).getText()
            end_account = Number(end_account.slice(9, end_account.length-1))
            console.log(start_account)
            console.log(price)
            console.log(end_account)

            console.log(parseFloat(start_account - price), " vs ", parseFloat(end_account))
            if(parseFloat(end_account)===(parseFloat(start_account)-parseFloat(price))){
                console.log("balance right")
            }
        }, 10000)
    } finally {
        //await driver.quit();
    }
};
```

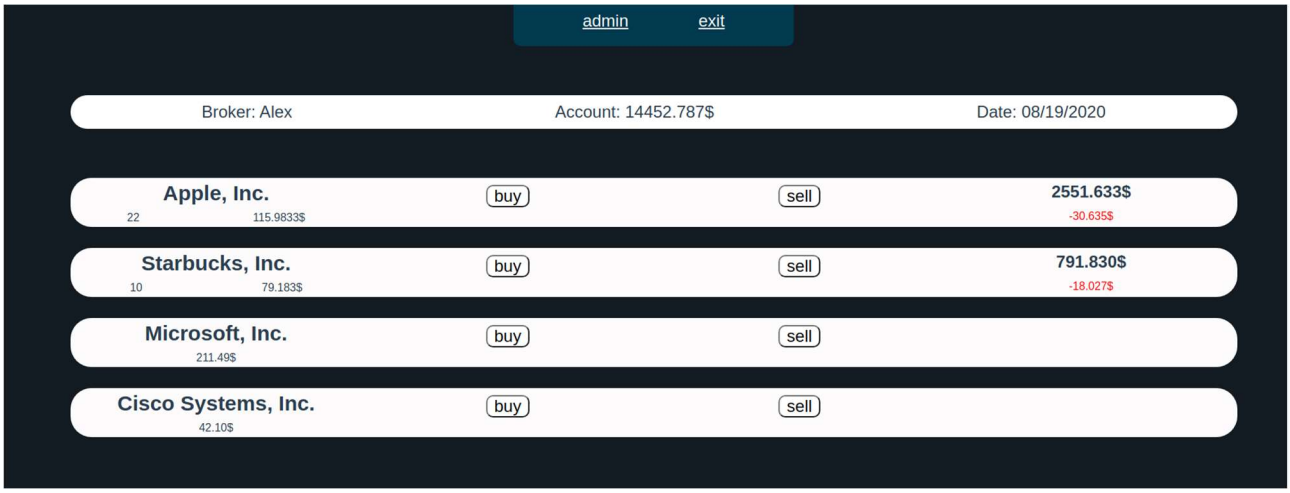
```
async function test_2() {
    let driver = await new Builder().forBrowser('chrome').build();
    try {
        await driver.get('http://localhost:8080/#/broker/Alex');
        let price = null;
        let start_account = null;
        let end_account = null;

        setTimeout(async () => {
            start_account = await driver.findElement(By.xpath("//div[@id='BrokerAccount']")).getText()
            start_account = Number(start_account.slice(9, start_account.length-1))
            console.log('start')
        }, 3000)

        setTimeout(async () => {
            await driver.findElement(By.xpath("//button[@id='APPL_sell_btn']")).click()
            console.log("sell")
        }, 4000)

        setTimeout(async () => {
            await driver.findElement(By.xpath("//input[@id='APPL_sell_inp']")).sendKeys(1)
            console.log("inp")
        }, 5000)

        setTimeout(async () => {
            await driver.findElement(By.xpath("//button[@id='APPL_sell_confirm_btn']")).click()
            price = await driver.findElement(By.xpath("//div[@id='APPL_price']")).getText()
            price = Number(price.slice(0, price.length-1))
            console.log('confirm', price)
        }, 6000)

        setTimeout(async () => {
            end_account = await driver.findElement(By.xpath("//div[@id='BrokerAccount']")).getText()
            end_account = Number(end_account.slice(9, end_account.length-1))
            console.log(start_account)
            console.log(price)
            console.log(end_account)

            console.log((start_account + price).toFixed(3), " vs ", (end_account).toFixed(3))
            if((end_account).toFixed(3)===(start_account+price).toFixed(3)){
                console.log("balance right")
            }
        }, 10000)
    } finally {
        //await driver.quit();
    }
```

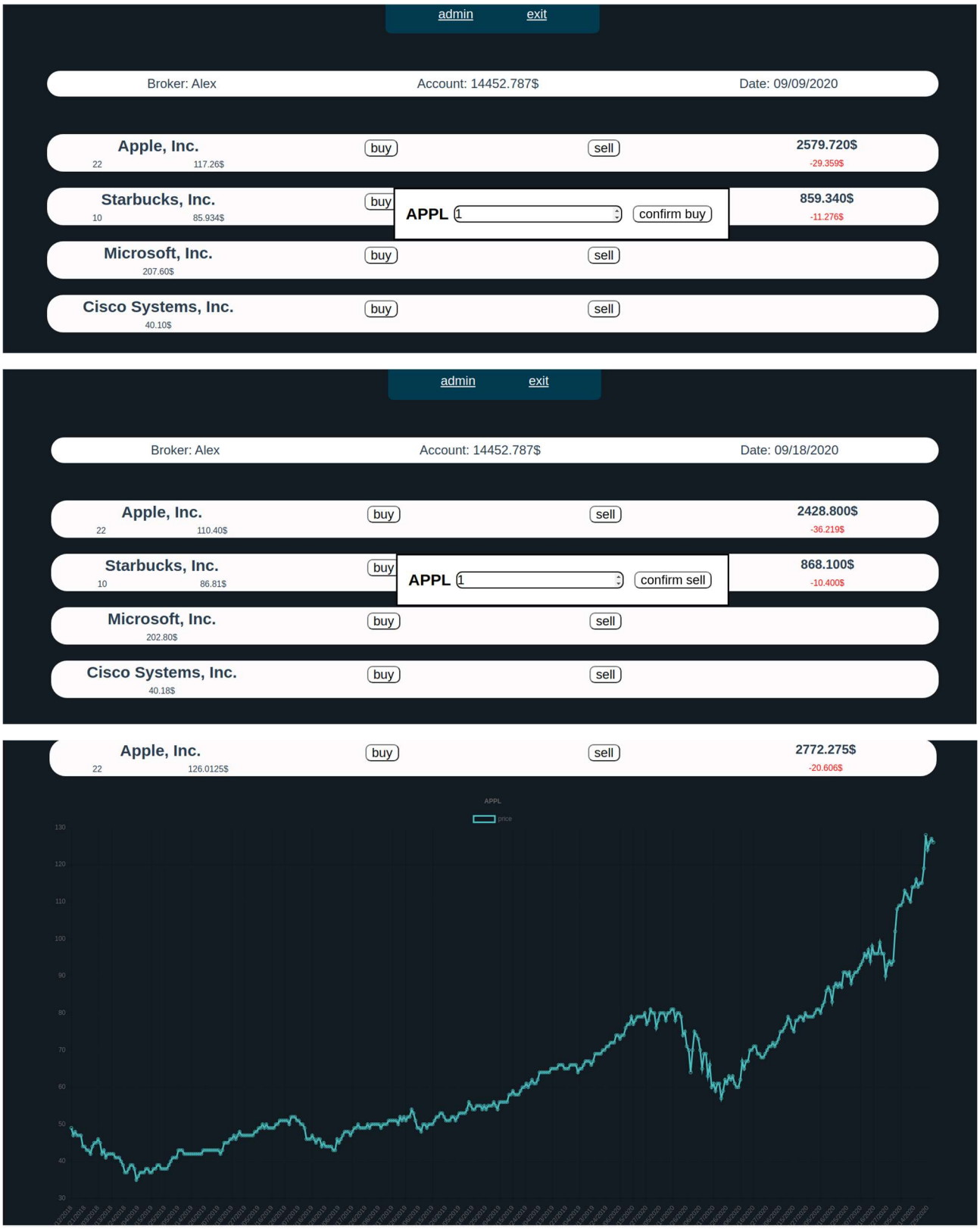
```
start
buy
inp
confirm 117.26
14455.327
117.26
14338.067
14338.067  vs  14338.067
balance right
start
sell
inp
confirm 114.72
14338.067
114.72
14452.787
14452.787  vs  14452.787
balance right
```

7. Итоговый внешний вид приложения:



Страница авторизации с возможностью выбора имени брокера из подсказки





Страница брокера с возможностью покупки, продажи и отслеживания изменения котировок

Страница администратора с возможностью просмотра портфелей каждого брокера

Выводы.

Изучены возможности применения фреймворка Vue (https://v3.ru.vuejs.org/ru/) для разработки интерфейсов пользователя web-приложений и организованы E2E-тесты клиентской части приложения.