Национальный исследовательский университет ИТМО

Лабораторная работа №2
«Взаимодействие с внешним API»

По дисциплине
«Фронт-энд разработка»

Выполнил:
Кривцов П.А.
Группа:
K33402
Преподаватель:
Добряков Д.И.

Санкт-Петербург
2022 г

ЗАДАНИЕ

Сайт криптобиржи/инвестиционной платформы

Нужно привязать сайт, свёрстанный в ЛР1, к внешнему API средствами fetch/axios/xhr. Страницы, требующие модификации:

- Начальная страница – отображение первых трех криптовалют;
- Вход;
- Регистрация;
- Список доступных криптовалют – поиск, сортировка, отображение действующего списка;
- Графики роста криптовалют – покупка/продажа валюты, загрузка нужного графика;
- Профиль пользователя – кнопка для смены пароля, отображение купленных валют;
- Смена пароля.

ВЫПОЛНЕНИЕ

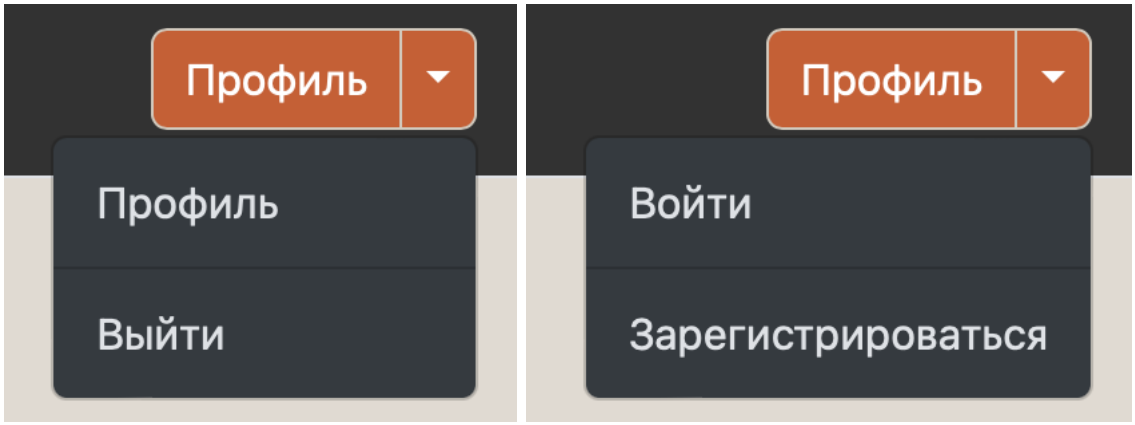
Исходный код: https://github.com/upaffy/ITMO-ICT-Frontend-2022/tree/main/labs/K33402/Krivtsov Pavel/Lab2

Запуск:

- python3 -m http.server 8000 для запуска сервера на 8000 порту;
- json-server jsonServer/db.json -m ./node_modules/json-server-auth для запуска json server’a

1. Начальная страница

   Здесь и в страницах, использующих navbar, импортируется файл *profileButton.js*, скрипты в котором определяют вид кнопки “Профиль”:



Чтобы изменения применялись к кнопке, нужно определить ее id как

```
id="profileButtonGroup"
```

profileButton.js

```
function getButtonHTML() {
    if (localStorage.accessToken) {
        return `
            <a href="./profile.html" class="btn btn-orange btn-md"
role="button">
                Профиль
            </a>

            <button type="button" class="btn btn-md btn-orange dropdown-
toggle dropdown-toggle-split" data-bs-toggle="dropdown" aria-
expanded="false">
                <span class="visually-hidden">Toggle Dropdown</span>
            </button>
            <ul class="dropdown-menu dropdown-menu-dark">
                <li><a class="dropdown-item"
href="./profile.html">Профиль</a></li>
                <li><hr class="dropdown-divider"></li>
                <li><a class="dropdown-item"
onclick="logout()">Выйти</a></li>
            </ul>
        `
    } else {
        return `
            <a href="./signup.html" class="btn btn-orange btn-md"
role="button">
                Профиль
            </a>

            <button type="button" class="btn btn-md btn-orange dropdown-
toggle dropdown-toggle-split" data-bs-toggle="dropdown" aria-
expanded="false">
                <span class="visually-hidden">Toggle Dropdown</span>
            </button>
            <ul class="dropdown-menu dropdown-menu-dark">
                <li><a class="dropdown-item"
href="./signin.html">Войти</a></li>
                <li><hr class="dropdown-divider"></li>
                <li><a class="dropdown-item"
href="./signup.html">Зарегистрироваться</a></li>
            </ul>
        `
    }
}

async function loadProfileButton() {
    document.querySelector("#profileButtonGroup").innerHTML = getButtonHTML()
}

async function logout() {
    localStorage.clear()
    window.location.href = "/"
}

document.addEventListener('DOMContentLoaded', () => {
    loadProfileButton()
})
```

Чтобы загрузить первые три криптовалюты, запросим у сервера валюты, id которых меньше 3:

```
async function loadPopularCurrencies() {
    document.querySelector("#currencies").innerHTML = ""

    const url = "http://localhost:3000/currencies?id_lte=3"
    const response = await fetch(url)
    const responseJSON = await response.json()

    for (const counter in responseJSON) {
        currency = responseJSON[counter]
        document.querySelector("#currencies").innerHTML +=
getCardHTML(currency)
    }
}

document.addEventListener('DOMContentLoaded', () => {

    loadPopularCurrencies()
})
```

Функция *getCardHTML* создает HTML-код карточки, основываясь на значении ее свойств:

```
function getCardHTML({ id, name, abbreviation, price, count, daily_changes,
weekly_changes, date_added, image}) {
    return `…`
}
```

2. Вход

   Для авторизации достанем данные из input-форм и отправим их на сервер POST-запросом, затем дождемся ответа и либо адресуем пользователя на начальную страницу, либо выведем alert message

```
async function login(event) {
    event.preventDefault()

    const inputs = Array.from(event.target.querySelectorAll('input'))
    const loginData = {}

    for (const input of inputs) {
        loginData[input.name] = input.value
    }

    const response = await fetch('http://localhost:3000/login', {
        method: "POST",
        body: JSON.stringify(loginData),
        headers: {
            'Content-Type': 'application/json'
        }
    })
    const responseJson = await response.json()

    if (response.status !== 200) {
```

```
        const alertPlaceholder = document.getElementById('alertPlaceholder')
        alert(alertPlaceholder, responseJson, "danger")
    } else {
        localStorage.accessToken = responseJson["accessToken"]
        localStorage.userId = responseJson["user"].id

        window.location.href = "http://localhost:8000/index.html"
    }
}
```

Для вывода alert messages в коде страницы есть контейнер с соответствующим id:

```
<div id="alertPlaceholder"></div>
```

Скрипт:

```
const alert = (alertPlaceholder, message, type) => {
    const wrapper = document.createElement('div')
    wrapper.innerHTML = `
        <div class="alert alert-${type} alert-dismissible" role="alert">
            <div>${message}</div>
            <button type="button" class="btn-close" data-bs-dismiss="alert"
aria-label="Close"></button>
        </div>
    `

    alertPlaceholder.append(wrapper)
}
```

3. Регистрация

   Схожа с Входом, отличается только адрес запроса на сервер: *'http://localhost:3000/register'*

4. Список доступных криптовалют

   Для отображения валют в коде страницы создан контейнер с id=”currencies”

```
<div class="container py-5">
    <div class="row row-cols-1 g-4">
        <div class="col" id="currencies">
        </div>
    </div>
</div>
```

Функция *loadCurrencies* загружает валюты с сервера так, чтобы они соответствовали параметрам поиска/сортировки/порядка сортировки.

Изначально эти параметры не выставлены и функция загружает все валюты (getCardHTML отдает код карточки в зависимости от ее свойств, addSearchParam возвращает новый url с добавлением переданных параметров):

```
async function loadCurrencies(searchString="", sortString="", orderString="")
{
    let url = "http://localhost:3000/currencies"
    document.querySelector("#currencies").innerHTML = ""

    if (searchString) {
        url = addSearchParam(url, 'q', searchString)
    }

    if (sortString && orderString) {
        url = addSearchParam(url, '_sort', sortString)
        url = addSearchParam(url, '_order', orderString)
    }

    const response = await fetch(url)
    const responseJSON = await response.json()

    for (const currency of responseJSON) {
        document.querySelector("#currencies").innerHTML +=
getCardHTML(currency)
    }
}

document.addEventListener('DOMContentLoaded', () => {
    loadCurrencies()
})
```

Если пользователь использует поиск или сортировку, вызываются функции, которые передают соответствующие параметры в функцию *loadCurrencies*:

```
async function search(event) {
    event.preventDefault()

    const searchString = event.target.querySelector('input').value
    await loadCurrencies(searchString)
}

async function sort(event) {
    event.preventDefault()

    if (event.target.id === "sort_by_date") {
        const orderString = sort_by_date_desc?"desc":"asc"
        sort_by_date_desc = !sort_by_date_desc

        await loadCurrencies("", "date_added", orderString)
    } else if (event.target.id === "sort_by_price") {
        const orderString = sort_by_price_desc?"desc":"asc"
        sort_by_price_desc = !sort_by_price_desc

        await loadCurrencies("", "price", orderString)
    }
```

```
    setSortButtonsHTML()
}
```

Функция *setSortButtonsHTML* определяет вид кнопок сортировки в зависимости от состояния глобальных переменных:

```
function setSortButtonsHTML() {
    const date_sign = sort_by_date_desc?"↑":"↓"
    const price_sign = sort_by_price_desc?"↑":"↓"

    document.querySelector("#sort_by_date").innerHTML = `Сортировать по дате
${date_sign}`
    document.querySelector("#sort_by_price").innerHTML = `Сортировать по цене
${price_sign}`
}
```

Функция *getCardHTML* добавляет в код карточки ссылку для перехода по тапу. Эта ссылка имеет вид "./currency.html?id=${id}", где передается id валюты, чтобы отображать информацию о ней на следующей странице

5. График роста валюты, покупка/продажа

   При загрузке страницы вызывается функция *LoadGraph*, которая по переданному в query url'a id валюты получает информацию о ней с сервера и формирует url для взаимодействия с внешним API сайта *Coinlib*, используя при этом функцию *getCurrencyFrame.*

```
function getCurrencyFrame() {
    return `<iframe
src="https://widget.coinlib.io/widget?type=chart&theme=light&coin_id=${curren
cy["coinlib_id"]}&pref_coin_id=1505" width="100%" height="536px" border="0"
style="border:0; margin:0; padding:0; line-height:14px;"></iframe>`
}

async function loadGraph() {
    let url = "http://localhost:3000/currencies"

    const queryString = window.location.search
    const urlParams = new URLSearchParams(queryString)
    const id = urlParams.get('id')

    url = addSearchParam(url, "id", id)

    const response = await fetch(url)
    const responseJSON = await response.json()
    currency = responseJSON[0]
```

```
    document.querySelector("#currency").innerHTML = getCurrencyFrame()
}
```

Если нажата кнопка купить/продать, срабатывает функция transaction, которая проверяет, аутентификацию пользователя и в зависимости от того, есть ли соответствующее отношение между пользователем и валютой, отправляет либо PATCH-запрос (увеличивая количество валюты), либо POST-запрос, создавая соответствующую запись в таблице, либо не отправляет ничего, если нажата кнопка “продать”, но валюты нет в портфеле пользователя

```
async function transaction(event) {
    event.preventDefault()
    await checkAuth()
    const ownership = await getOwnership()

    let requestUrl = ""
    let requestData = {}
    let method = ""

    if (ownership) {
        [requestUrl, requestData] = configureTransactionPatchRequest(event,
ownership)
        method = "PATCH"
    } else {
        [requestUrl, requestData] = configureTransactionPostRequest(event)
        if (requestUrl) {
            method = "POST"
        }
    }

    if (method) {
        const response = await fetch(requestUrl, {
            method: method,
            body: JSON.stringify(requestData),
            headers: {
                'Content-Type': 'application/json'
            }
        })

        if (response.status === 200) {
            await showToast()
        }
    }
}
```

6. Профиль пользователя

   При переходе на эту страницу проверяется аутентификация пользователя, загружаются его валюты и username для заголовка

```
document.addEventListener('DOMContentLoaded', () => {
    checkAuth()
    loadProfileCurrencies()
```

```
    loadProfileTitle()
})
```

Проверка аутентификации:

```
async function checkAuth() {
    if (!localStorage.accessToken) {
        window.location.href = "http://localhost:8000/signin.html"
    }
}
```

Загрузка портфеля пользователя происходит в 2 этапа. Сначала загружаются все отношения между конкретным пользователем и валютой. Затем, зная id, загружаем каждую валюту отдельно.

```
async function loadProfileCurrencies() {
    document.querySelector("#profileTitle").innerHTML = ""
    document.querySelector("#profileCurrencies").innerHTML = ""

    const ownershipURL =
`http://localhost:3000/ownership?userId=${localStorage.userId}`
    const ownershipResponse = await fetch(ownershipURL)
    const ownershipResponseJSON = await ownershipResponse.json()

    for (const ownership of ownershipResponseJSON) {
        if (ownership["count"] > 0) {
            const currencyURL =
`http://localhost:3000/currencies?id=${ownership["currency_id"]}`
            const currencyResponse = await fetch(currencyURL)
            const currencyResponseJSON = await currencyResponse.json()

            document.querySelector("#profileCurrencies").innerHTML +=
getCardHTML(ownership, currencyResponseJSON[0])
        }
    }
}
```

Загрузка заголовка:

```
async function loadProfileTitle() {
    const userURL = `http://localhost:3000/users?id=${localStorage.userId}`
    const userResponse = await fetch(userURL)
    const userResponseJSON = await userResponse.json()

    document.querySelector("#profileTitle").innerHTML = `Профиль пользователя
${userResponseJSON[0]["username"]}`
}
```

7. Смена пароля

   Созданы 3 формы: старый пароль, два раза новый

   Нажатие на кнопку с типом submit вызывает функцию changePassword. Которая получает email пользователя по хранящемуся в localStorage его id, аутентифицирует его, используя значения из поля “старый пароль”.

Если аутентификация проходит успешно, оба новых пароля совпадают, создается PATCH-запрос для изменения на сервере.

Если на каком-то этапе встречена ошибка, появляется alert с соответствующим содержанием

```
async function changePassword(event) {
    event.preventDefault()

    const email = await getEmail()
    const inputs = Array.from(event.target.querySelectorAll('input'))

    let passwords = {}
    for (const input of inputs) {
        passwords[input.id] = input.value
    }

    const loginData = {"email": email, "password": passwords["oldPassword"]}

    const response = await fetch('http://localhost:3000/login', {
        method: "POST",
        body: JSON.stringify(loginData),
        headers: {
            'Content-Type': 'application/json'
        }
    })

    const responseJson = await response.json()
    if (response.status !== 200) {
        const alertPlaceholder = document.getElementById('alertPlaceholder')
        alert(alertPlaceholder, responseJson, "danger")
        return
    }

    if (passwords["password1"] !== passwords["password2"]) {
        const alertPlaceholder = document.getElementById('alertPlaceholder')
        alert(alertPlaceholder, "Новые пароли должны совпадать", "danger")
        return
    }

    const [patchURL, patchData] =
configureUserPatchRequest(passwords["password1"])
    const patchResponse = await fetch(patchURL, {
        method: "PATCH",
        body: JSON.stringify(patchData),
        headers: {
            'Content-Type': 'application/json'
        }
    })

    if (patchResponse.status === 200) {
        window.location.href = "http://localhost:8000/profile.html"
    } else {
        const jsonPatchResponse = await patchResponse.json()
        const alertPlaceholder = document.getElementById('alertPlaceholder')
        alert(alertPlaceholder, jsonPatchResponse, "danger")
    }
}
```

ВЫВОДЫ

В процессе работы я познакомился с языком JS, взаимодействуя с внешним API. В качестве API использовался json-server. В итоге мною были реализованы запросы для регистрации, аутентификации, создания записей, просмотра записей и сортировки.