

Национальный исследовательский университет ИТМО



Лабораторная работа №2
«Взаимодействие с внешним API»

По дисциплине
«Фронт-энд разработка»

Выполнил:
Шевченко Максим
Группа:
К33401
Преподаватель:
Добряков Д.И.

Санкт-Петербург
2023 г

ЗАДАНИЕ

Сайт криптобиржи/инвестиционной платформы

Нужно привязать сайт, сверстанный в ЛР1, к внешнему API средствами fetch/axios/xhr.

ВЫПОЛНЕНИЕ

Запуск:

- `python3 -m http.server 3000` для запуска сайта на 3000 порту;
- Запустить сервер, написанный с помощью DRF: `python manage.py runserver`.

1. Начальная страница








Добро пожаловать на нашу криптовалютную биржу!

Тут вы можете безопасно и удобно покупать, продавать и обменивать свои любимые криптовалюты. Мы предлагаем широкий спектр цифровых активов, конкурентоспособные обменные курсы и удобную платформу, чтобы лишить вашу торговлю проблем. Независимо от того, являетесь ли вы новичком в мире криптовалют или опытным трейдером, мы здесь, чтобы поддержать вас на каждом этапе пути!

Начать →



Найти валюту

 \$19291.8 ✓ 10% This week	 \$1312.23 ✗ -5% This week
 \$0.9993 ✓ 12% This week	 \$272.21 ✗ -3.5% This week
 \$0.0593 ✓ 6.8% This week	

support@company.com (123) 456-7890

Чтобы загрузить криптовалюты, будем при загрузке страницы запрашивать их у сервера:

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

```

}))

async function loadCoins(searchString="", sortString="") {
  let url = "http://127.0.0.1:8000/coins/"
  document.querySelector("#coins").innerHTML = ""

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

  if (sortString) {
    url = addSearchParam(url, 'ordering', sortString)
  }

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

  for (const coin of responseJSON) {
    document.querySelector("#coins").innerHTML += makeCard(coin)
  }
}

```

Функция `makeCard` будет создавать код для карточек и вставлять его в тег, для которого мы указали `id`, равный `coin`

```
function makeCard({id, name, price, weekly_changes, image}) { ...
```

```

<div class="container py-5">
  <div class="row" id="coins"></div>
</div>

```

Для сортировки и фильтрации достаточно отправить на сервер запросы с соответствующей `query`-строкой. Элементы в `url` добавляются с помощью следующей функции:

```

function addSearchParam(url, key, value) {
  let newUrl = new URL(url)
  newUrl.searchParams.set(key, value)

  return newUrl.toString()
}

```

А нажатие на сортировку или кнопку поиска вызывает функции, которые создают новый `url` и запрашивают загрузку валюты:

```

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

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

async function sort(sortString) {
  await loadCoins("", sortString)
}

```

2. Вход

Для авторизации будем доставать данные из input-форм и отправлять их на 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://127.0.0.1:8000/auth/token/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["auth_token"]

    window.location.href = "http://127.0.0.1:3000/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.appendChild(wrapper)
}
```

```
    alertPlaceholder.append(wrapper)
  }
```

3. Регистрация

Логика такая же, как и во входе, но отличается только адрес запроса на сервер:

'http://localhost:3000/register'

4. 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 *makeCoinlibChart*.

```
async function loadChart() {
  const queryString = window.location.search
  const urlParams = new URLSearchParams(queryString)
  const id = urlParams.get('id')

  const url = `http://127.0.0.1:8000/coins/${id}/`

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

  document.querySelector("#coin").innerHTML = makeCoinlibChart()
}
}
```

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

Если нажата кнопка купить/продать, срабатывает функция *transaction*, которая проверяет, есть ли в хранилище токен авторизации и в зависимости от того, есть ли соответствующее отношение между пользователем и валютой, отправляет либо РАТСН-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(type) {
  await checkAuth()
  const ownership = await loadOwnership()

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

  if (ownership) {
    [requestUrl, requestData] = makeTransactionPatchRequest(type, ownership)
    method = "PATCH"
  } else {
    [requestUrl, requestData] = makeTransactionPostRequest(type)
    if (requestUrl) {
      method = "POST"
    }
  }

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

    if (response.status === 200 || response.status === 201) {
      await showToast()
    }
  }
}
```

5. Профиль пользователя

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

```
document.addEventListener('DOMContentLoaded', () => {
  checkAuth()
  loadCoins()
  loadTitle()
})
```

Проверка аутентификации:

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

Чтобы найти валюты пользователя, нужно запросить отношения пользователя с валютой

```
async function loadCoins() {
  document.querySelector("#coins").innerHTML = ""

  const possessionsURL = `http://127.0.0.1:8000/possessions`
  const possessionResponse = await fetch(possessionsURL, {
    headers: {
      "Authorization": `Token ${localStorage.accessToken}`
    }
  })
  const possessionResponseJSON = await possessionResponse.json()

  for (const ownership of possessionResponseJSON) {
    if (ownership["count"] > 0) {
      document.querySelector("#coins").innerHTML += makeCard(ownership,
ownership["currency"])
    }
  }
}
```

Чтобы загрузить заголовок, необходимо получить информацию о пользователе:

```
async function loadTitle() {
  const userURL = `http://127.0.0.1:8000/auth/users/me`
  const userResponse = await fetch(userURL, {
    headers: {
      "Authorization": `Token ${localStorage.accessToken}`
    }
  })
  const userResponseJSON = await userResponse.json()

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

6. Смена пароля

Созданы 3 формы: старый пароль, два раза новый

Нажатие на кнопку с типом submit вызывает функцию changePassword.

Которая отправляет все три поля и токен аутентификации на сервер.

Если ответ пустой, пароль был изменен успешно, поэтому отправим пользователя на страницу профиля, иначе, выведем алерт

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

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



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

    const response = await
fetch(`http://127.0.0.1:8000/auth/users/set_password/`, {
  method: "POST",
  body: JSON.stringify(data),
  headers: {
    'Authorization': `Token ${localStorage.accessToken}`,
    'Content-Type': 'application/json'
  }
})

    try {
      const responseJson = await response.json()
      if (response.status !== 200) {
        const alertPlaceholder =
document.getElementById('alertPlaceholder')
        alert(alertPlaceholder, responseJson[0], "danger")
        return
      }
    } catch (err) {
      window.location.href = "http://localhost:3000/profile.html"
    }
  }
}
```

ВЫВОДЫ

В процессе работы я подключил ранее сверстанный сайт к API, познакомился с языком JS и методом `fetch`, с помощью которого делал запросы.

В качестве API использовался DRF-server. В итоге мною были реализованы запросы для регистрации, аутентификации, создания записей, просмотра записей и сортировки.