

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

Дисциплина: Фронт-энд разработка

Отчет

Лабораторная работа 2: взаимодействие с внешним API

**Выполнил:
Тимофеев Н. А.**

**Группа:
К33402**

**Проверил:
Добряков Д. И.**

Санкт-Петербург

2022 г.

Задача

Привязать страницу из ЛР1 к внешнему API средствами fetch/axios/xhr

Ход работы

Был выбран **fetch()**, тк он входит в стандартную библиотеку JavaScript (не требует установки извне, в отличие от axios) и обладает простым интерфейсом.

Был реализован сервис на языке Python с использованием веб-фреймворка FastAPI для регистрации, авторизации и взаимодействий с криптовалютами. Курсы валют (например, BTC) берутся из открытого API криптовалютной биржи Kraken.

Swagger моего сервиса доступен по адресу <http://194.87.248.78:8000/docs>

А реализованный сайт который взаимодействует с этим API доступен по адресу <http://symptomatic-mailbox.surge.sh>



New here?

Take a quick tour into investment world

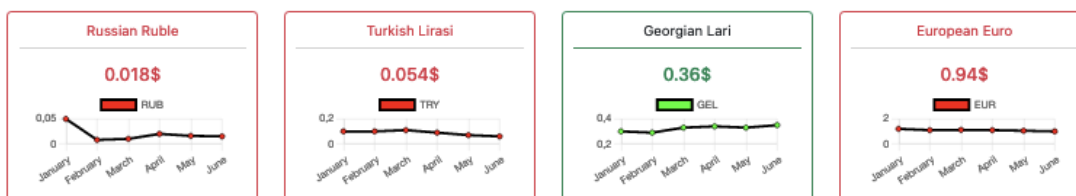
[Join us now](#)

The global cryptocurrency market cap today is **\$949.09B**, a **+0.14%** change from 24 hours ago.

Trending positions:



Worldwide currencies:



TI LLC

[Home](#)
[Stocks](#)
[Crypto](#)
[Legal](#)
[About](#)

Buy Crypto

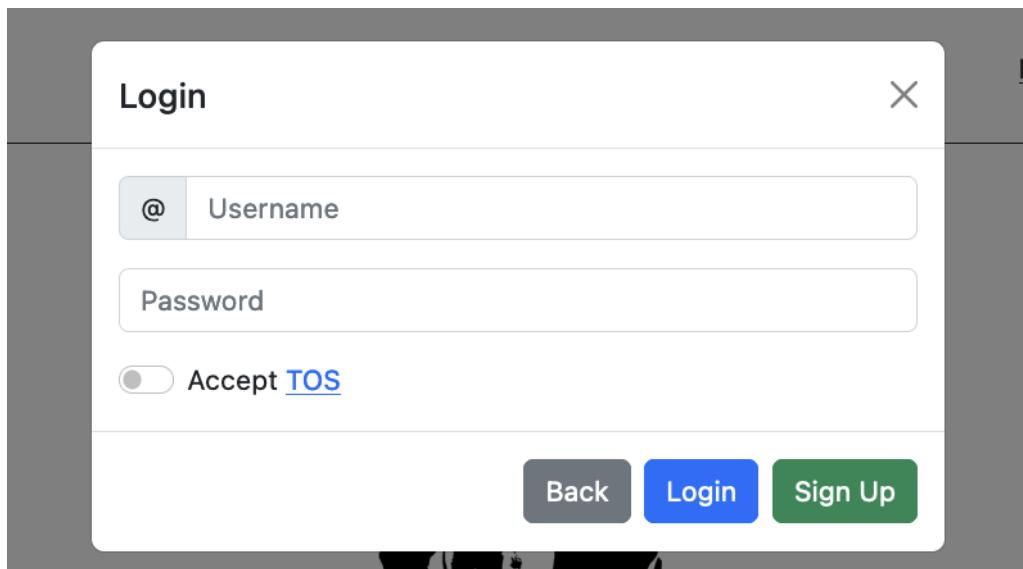
[Buy Ethereum](#)
[Buy Bitcoin](#)
[Buy USDT](#)

Subscribe to our newsletter

Monthly digest of what's new and exciting from us.

[Subscribe](#)

Модальное окно авторизации:

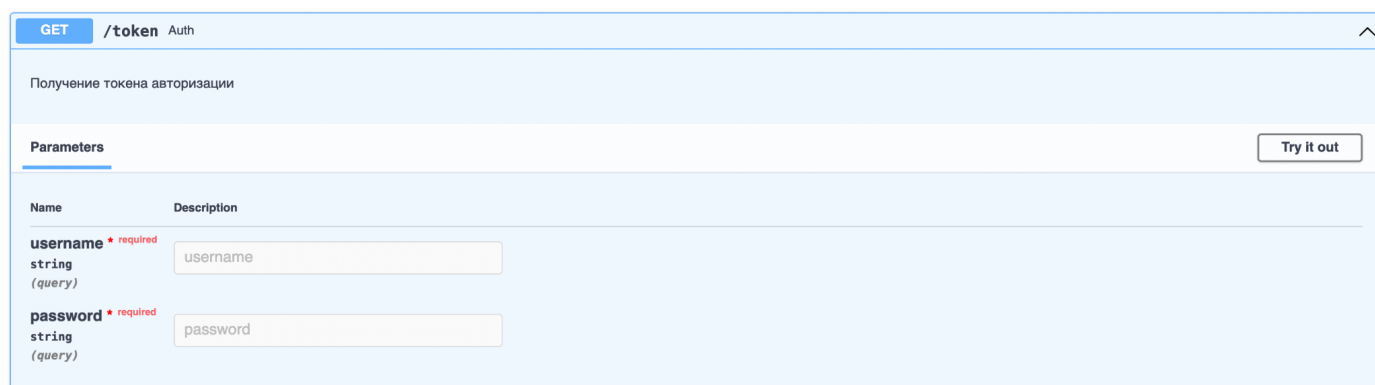


A login modal window titled "Login" with a close button (X) in the top right corner. It contains two input fields: "Username" with a placeholder "@" and "Password". Below the password field is a toggle switch labeled "Accept [TOS](#)". At the bottom, there are three buttons: "Back" (grey), "Login" (blue), and "Sign Up" (green).

Скрипт привязанный к этому модальному окну:

```
loginUser = (username, password) => {  
  fetch(`${backendUrl}/token?username=${username}&password=${password}`)  
    .then((res) => {  
      if (res.status !== 200) {  
        throw "Wrong username or password!";  
      }  
      return res.json();  
    })  
    .then((data) => {  
      console.log(data);  
      localStorage.setItem("token", data);  
      document.getElementById("closeLoginModal").click();  
      window.location.reload();  
    })  
    .catch((e) => {  
      document.getElementById("loginErrorMessage").innerHTML = e;  
    });  
};
```

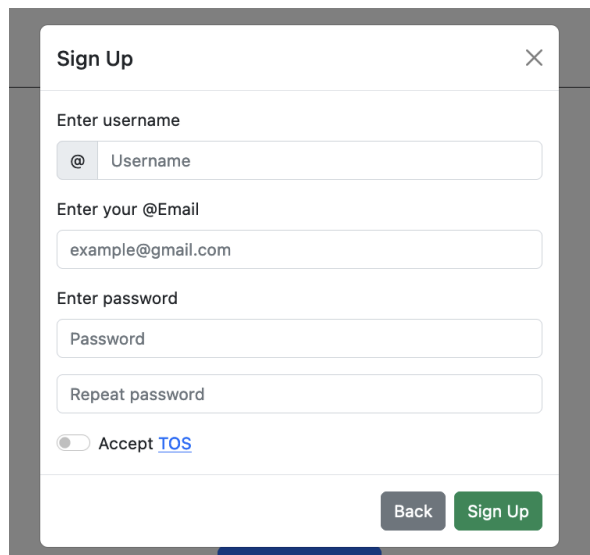
API Endpoint:



API endpoint documentation for GET /token (Auth). The description is "Получение токена авторизации". The parameters section shows two required query parameters: "username" (string) and "password" (string). Each parameter has a corresponding input field.

Name	Description
username * required string (query)	username
password * required string (query)	password

Модальное окно регистрации:



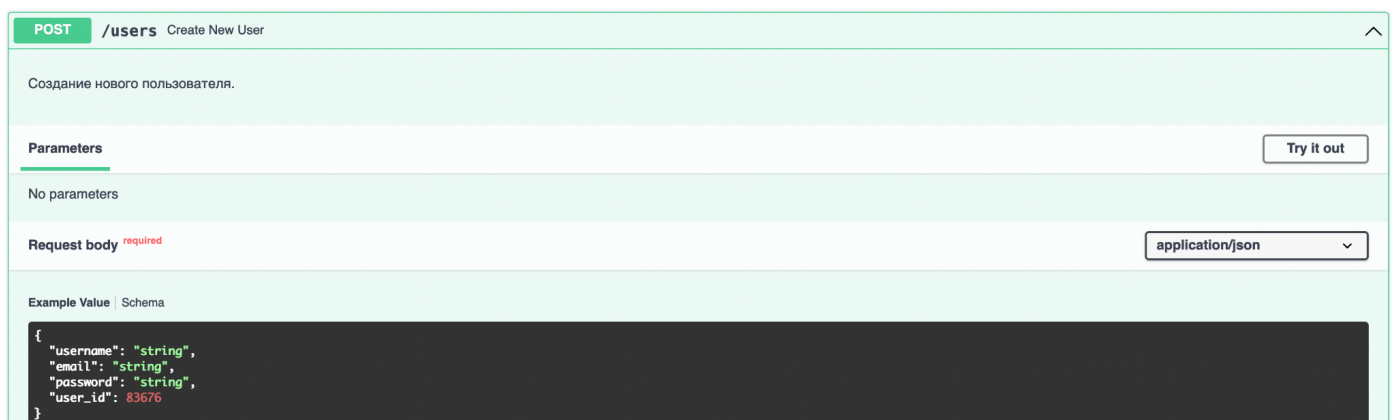
A modal window titled "Sign Up" with a close button (X) in the top right corner. The form contains the following fields and elements:

- "Enter username" label with a text input field containing the placeholder "Username".
- "Enter your @Email" label with a text input field containing the placeholder "example@gmail.com".
- "Enter password" label with a text input field containing the placeholder "Password".
- A second text input field labeled "Repeat password".
- A checkbox labeled "Accept [TOS](#)".
- "Back" and "Sign Up" buttons at the bottom right.

Скрипт привязанный к модальному окну:

```
registerUser = (username, password, email) => {
  fetch(`${backendUrl}/users`, {
    // ?username=${username}&password=${password}
    method: "POST",
    body: JSON.stringify({
      username,
      password,
      email,
    }),
    headers: {
      Accept: "application/json, text/plain, */*",
      "Content-Type": "application/json",
    },
  })
  .then(() => {
    document.getElementById("closeSignUpModal").click();
  })
  .catch((e) => alert(e));
};
```

API Endpoint:



API endpoint documentation for the **POST /users** endpoint, titled "Create New User".

Создание нового пользователя.

Parameters Try it out

No parameters

Request body required application/json

Example Value | Schema

```
{
  "username": "string",
  "email": "string",
  "password": "string",
  "user_id": 83676
}
```

Страница криптовалюты:



Bitcoin (BTC) is a cryptocurrency, a virtual currency designed to act as money and a form of payment outside the control of any one person, group, or entity, and thus removing the need for third-party involvement in financial transactions. It is rewarded to blockchain miners for the work done to verify transactions and can be purchased on several exchanges.

TI LLC

Home
Stocks
Crypto
Legal
About

Buy Crypto

Buy Ethereum
Buy Bitcoin
Buy USDT

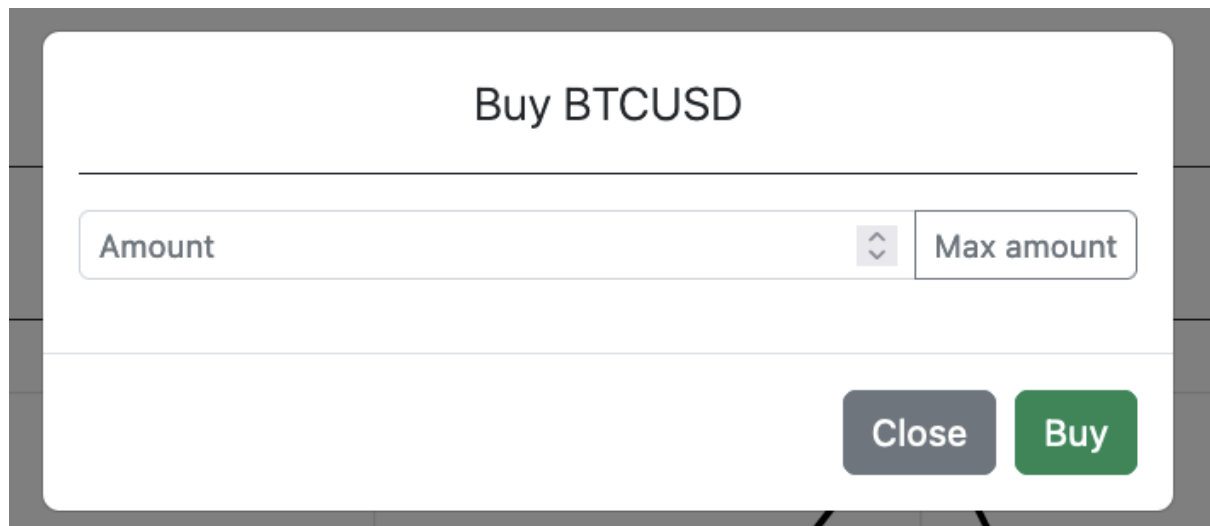
Subscribe to our newsletter

Monthly digest of what's new and exciting from us.

Получение цены с Public API Kraken:

```
processPrices = (pair, func, mode) => {
  let resp = fetch(`https://api.kraken.com/0/public/Ticker?pair=${pair}`)
    .then((response) => response.json())
    .then((data) => {
      console.log(`Price for ${pair} is ` + Number(Object.values(data.result)[0].a[0]));
      console.log(Number(Object.values(data.result)[0].a[0]));
      switch (mode) {
        case "single":
          console.log(mode);
          func(Object.values(data.result)[0].a[0].a);
        case "all":
          func(Object.values(data.result));
        default:
          break;
      }
    })
    .catch(() => {});
  return resp;
};
```

Модальное окно покупки валюты:



The image shows a modal window titled "Buy BTCUSD". Inside the modal, there is a horizontal input field labeled "Amount" with a small dropdown arrow on its right side. To the right of the input field is a button labeled "Max amount". At the bottom right of the modal, there are two buttons: a grey "Close" button and a green "Buy" button.

Скрипт привязанный к модальному окну:

```
buyCrypto = () => {
  const amount = document.getElementById("cryptoAmount").value;
  const token = localStorage.getItem("token");
  if (!amount || !token) {
    return;
  }
  fetch(`${backendUrl}/crypto?currency=BTC&amount=${amount}`, {
    headers: {
      Accept: "application/json",
      Token: token,
    },
    method: "POST",
  })
    .then((res) => {
      return res.json();
    })
    .then((data) => {
      Toastify({
        text: "Success! Bought " + amount + " BTC!",
        duration: 3000,
      }).showToast();
      document.getElementById("closeBuyCrypto").click();
    })
    .catch((e) => {
      Toastify({
        text: "Error!",
        duration: 3000,
      }).showToast()
    });
};
```

API Endpoint:

POST

/crypto

Buy Crypto

⌵

Записать транзакцию о покупке криптовалюты

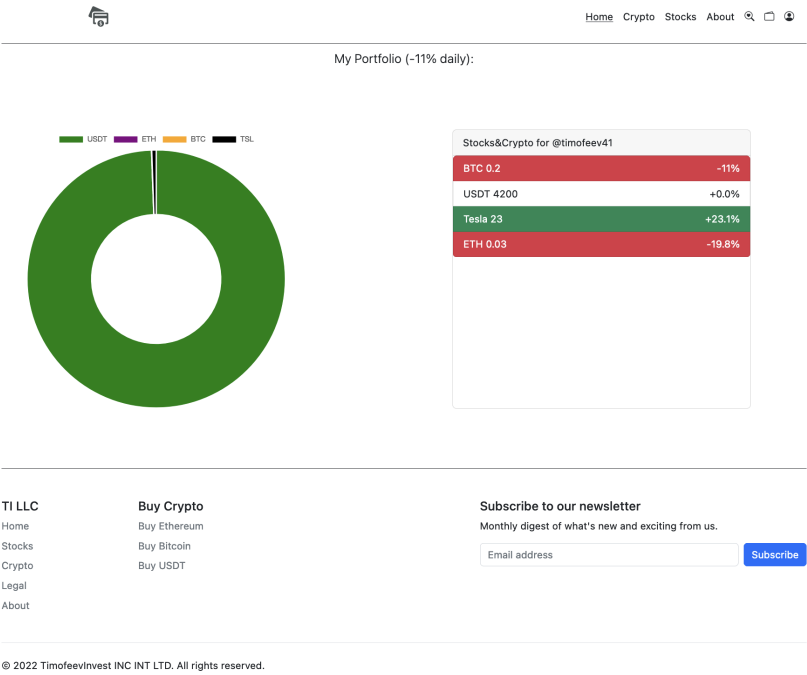
Parameters

Cancel

Name	Description
<div><div>currency</div><div>string</div><div>(query)</div></div> <div>required</div>	<input type="text" value="currency"/>
<div><div>amount</div><div>number</div><div>(query)</div></div> <div>required</div>	<input type="text" value="amount"/>
<div><div>token</div><div>string</div><div>(header)</div></div> <div>required</div>	<input type="text" value="token"/>

Execute

Страница с портфолио пользователя:



Скрипт для получения данных:

```
getPortfolio = () => {
  const graph = document.getElementById("portfolioChart");
  const stats = document.getElementById("listPortfolio");
  const token = localStorage.getItem("token");
  if (!graph || !stats || !token) {
    return;
  }

  fetch("http://194.87.248.78:8000/crypto", {
    headers: {
      Accept: "application/json",
      Token: token,
    },
  })
    .then((res) => {
      stats.innerHTML = "";
      return res.json();
    })
    .then((data) => {
      const labels = []
      const values = []
      for (const [key, value] of Object.entries(data)) {
        console.log(`${key}: ${value}`);
        labels.push(key);
        values.push(value);
        stats.innerHTML += `<li class="list-group-item bg-danger text-white d-inline-flex justify-content-between">
          <span>${key} ${value}</span><span>-${Math.round(Math.random() * 85)}%</span>
        </li>`;
      }

      const chartData = {
        labels: labels,
        datasets: [
          {
            label: "Portfolio",
            data: values,
            backgroundColor: ["green", "purple", "orange", "black"],
            hoverOffset: 1,
          },
        ],
      },

      const config = {
        type: "doughnut",
        data: chartData,
      };

      const portfolioChart = new Chart(document.getElementById("portfolioChart"), config);
    });
};
```

API Endpoint:

The screenshot shows an API client window with the following details:

- Method:** GET
- URL:** /crypto
- Header:** Get Portfolio
- Parameters:** A table with two columns: Name and Description. The first row contains 'token' (marked as required), 'string', and '(header)'. The value 'admin+admin' is entered in the input field.
- Buttons:** 'Execute' (blue) and 'Clear' (white) buttons are at the bottom.
- Cancel:** A red 'Cancel' button is in the top right corner.

Графики были реализованы с помощью библиотеки ChartJS, для верстки использовался фреймворк Bootstrap 5. Для вывода toast-уведомлений использована библиотека Toastify JS. Все нужные скрипты и стили были подключены через CDN. Для деплоя использована утилита Surge.

Backend написан на языке Python с фреймворком FastAPI, деплой выполнялся с помощью docker-compose, исходный код см. в Приложении 1.

Вывод

В ходе работы я научился использовать `fetch()` для взаимодействия с внешними API, отправлять GET и POST запросы, модифицировать HTML страницу с помощью JavaScript. В работе я не использовал JQuery, так как он усложняет поддержку проекта и читабельность кода.

Приложение 1. Исходный код сервиса

app.py

```
import random

from fastapi import FastAPI, Header, HTTPException, Depends
from pydantic import BaseModel, parse_obj_as
from collections import defaultdict
from fastapi.middleware.cors import CORSMiddleware

app = FastAPI()

origins = ["*"]

app.add_middleware(
    CORSMiddleware,
    allow_origins=origins,
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)

# my personal postgres clickhouse mongo redis db O(1) runtime speed 100% no cap

users = defaultdict(lambda: defaultdict(lambda: '***'))

crypto = defaultdict(lambda: defaultdict(lambda: 0))

@app.get('/users')
async def get_all_users() -> dict:
    return users

class NewUser(BaseModel):
    username: str
    email: str | None = None
    password: str
    user_id: int = random.choice(range(1, 100001))

@app.post('/users')
async def create_new_user(user: NewUser) -> dict:
```

```

    """Создание нового пользователя."""

    if users.get(user.username):

        raise HTTPException(status_code=500, detail='User already exist')

    users[user.username] = user.dict() # type: ignore

    return users[user.username]

@app.get('/token')
async def auth(username: str, password: str) -> str:

    """Получение токена авторизации"""

    if user := users.get(username):

        if user['password'] == password:

            return f'{username}+{password}'

        raise HTTPException(status_code=500, detail='User not found!')

async def get_user(token: str = Header()) -> NewUser:

    """Депенденси для авторизации"""

    if user := users.get(token.split('+')[0]):

        return parse_obj_as(NewUser, user)

    raise HTTPException(status_code=500, detail='User not found!')

@app.get('/me')
async def get_my_data(user = Depends(get_user)) -> NewUser:

    """Получение данных пользователя по токenu"""

    return user

@app.post('/crypto')
async def buy_crypto(currency: str, amount: float, user: NewUser = Depends(get_user)) -> int:

    """Записать транзакцию о покупке криптовалюты"""

    crypto[user.user_id][currency] += int(amount)

    return crypto[user.user_id][currency]

@app.get('/crypto')
async def get_portfolio(user: NewUser = Depends(get_user)) -> dict:

    return crypto[user.user_id]

```

```
@app.on_event('startup')

async def startup():

    users['admin'] = NewUser(user_id=1, username='admin', email='amogus',
password='admin').dict() # type: ignore

    crypto[1]['BTC'] = 123

    crypto[1]['ETH'] = 300

    crypto[1]['DOGE'] = 12312
```

Docker-compose.yml:

version: '3'

services:

web:

build: .

command: uvicorn app:app --host 0.0.0.0

ports:

- 8000:8000

Dockerfile:

FROM python:3.10

WORKDIR /code

COPY ./requirements.txt /code/requirements.txt

RUN pip install --no-cache-dir --upgrade -r /code/requirements.txt

COPY ./app.py /code/app.py