САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Фронт-энд разработка

Отчет

Лабораторная работа №2

Выполнил:

Таякин Даниил

Группа:
К33392

Проверил:
Добряков Д. И.

Санкт-Петербург

2023 г.

Цель

На примере собственного готовящегося коммерческого проекта составить отчет по ранее реализованным элементам в web-приложении. Поскольку в лабораторной работе №2 требовалось привязать к внешнему API, средствами fetch/axios/xhr. То сформируем задание по нашему проекту так:

1. описать как подгружается данные о тестовых аукционах.

Идея проекта

Dekast – это бесплатная платформа, которая облегчает сделки с коллекционными предметами между пользователями. Она поддерживает прямые, безопасные и анонимные продажи, а также публичные аукционы, в которых каждый может принять участие в течение определенного периода времени. Для прозрачного предоставления этой услуги Dekast использует все преимущества TON, технологии блокчейн, изначально разработанной Telegram.

Примечание

Данный проект все еще находится на стадии разработки. Некоторые функции не реализованы до конца, а конечная версия релиза может отличаться от текущей.

Ход работы

Внешний API является также моей разработкой. Перед тем, как запрашиваются данные о проходящих аукционах из базы, сначала отправляется запрос на одноразовый токен доступа к API. Для обращения к API был написан отдельный класс, который можно увидеть на рисунке 1 и 2.

```
export class API {

  static baseURL = process.env.PEWPEE_API_ENDPOINT;

  headers = new Headers({
    'Content-Type': 'application/x-www-form-urlencoded'
  });

  constructor(headers?: Headers) {
    if (headers) { this.headers = headers }
  }

  async getAccessToken(): Promise<string> {
    let tokenData = await fetch(`${process.env.PEWPEE_OAUTH_ENDPOINT}/token.php`, {
      method: 'POST',
      headers: this.headers,
      body: new URLSearchParams({
        'client_id': process.env.PEWPEE_CLIENT_ID!,
        'grant_type': "client_credentials",
        'client_secret': process.env.PEWPEE_CLIENT_SECRET!
      }),
    }).then(response => response.json());

    return tokenData['access_token'];
  }
```

Рис. 1 – Первая часть класса API

```
  async call(url: string, method: string = 'GET', params?: Record<string, string>): Promise<any> {
    const accessToken = await this.getAccessToken();

    let setURL = API.baseURL + url
    if (method == 'GET') {
      setURL += '?' + new URLSearchParams(params).toString()
    }

    console.log(setURL);

    const response = await fetch(setURL, {
      method: method,
      headers: {
        'Authorization': `Bearer ${accessToken}`,
      },
      body: method != 'GET' ? new URLSearchParams(params) : null
    }).then(response => response.json());

    return response;
  }

}
```

Рис. 2 – Вторая часть класса API

На рисунке 2 представлена асинхронная функция call, которая упрощает весь процесс запроса данных к базе, так как она запрашивает токен, и формирует запрос с нужными параметрами.

Рис. 3 – Структура API в проекте.

На рисунке 3 представлена структура того, как располагается данный код в папке проекта.

```
import { API } from './common';

export class Username {

  constructor () {
    // nothing yet
  }

  async getTopAuctions(params?: Record<string, string>) {
    return await new API().call('/usernames/usernames.php', 'GET', params);
  }
}
```

Рис. 4 – Использование класса API в классе Username

На рисунке 4 представлен код класса Usernames, который использует класс API для получения топовых проходящих аукционов.

```
export async function getServerSideProps() {
  let usernameAPI = new Username();
  let data = await usernameAPI.getTopAuctions({'that': 'top_auctions'});

  const toncoinPrice = await getToncoinPrice();

  await data.auctions.map((username: any) => {
    username.usd_price = toncoinPrice * username.min_bid;

    username.end_time = username.auction_end_time;
    username.end_time_locale_string = new Date(username.end_time).toLocaleString();

    return username;
  });

  console.log(data);

  return { props: { data } };
}
```

Рис. 5 – Запрос getTopAuctions в index.tsx

На рисунке 5 представлен код файла index.tsx - главной страницы web-приложения. Функция getServerSideProps является частью фреймворка Next.js, которая посылает запрос на backend и возвращает странице HomePage сформированные props.

Вывод

Был представлен код того, как приложение работает с API запросами. Демо версию проекта можно посмотреть по адресу: https://dekast.io.