**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ**

**РОССИЙСКОЙ ФЕДЕРАЦИИ**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ**

**ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В.Г.ШУХОВА»**

**(БГТУ им. В.Г. Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

**Расчётно-графическое задание**

По дисциплине: Технологии Web-программирования

Тема: **«Web приложение каталог для хранения комиксов**»

|  |  |
| --- | --- |
|  | Выполнил:  студент группы ПВ-192  Черный Денис  Проверил:  Картамышев С.В. |

Белгород 2022

Оглавление

[HTML. Разработка макетов и верстка шаблонов web-приложения с помощью языков HTML и CSS 3](#_Toc122959066)

[Главная страница 3](#_Toc122959067)

[Контентная страница 4](#_Toc122959068)

[Страница авторизации 4](#_Toc122959069)

[Клиентское программирование 5](#_Toc122959070)

[Главная страница 5](#_Toc122959071)

[Контентная страница 5](#_Toc122959072)

[Страница авторизации 6](#_Toc122959073)

[Серверное программирование 7](#_Toc122959074)

[docker-compose.yaml 7](#_Toc122959075)

[Dockerfile программы сервера 8](#_Toc122959076)

[Dockerfile для nginx 8](#_Toc122959077)

[Запрос 9](#_Toc122959078)

[Разработка и проектирование базы данных 9](#_Toc122959079)

[Схема базы данных 10](#_Toc122959080)

[Код моделей 10](#_Toc122959081)

[REST API 10](#_Toc122959082)

[Схема API (Swagger) 11](#_Toc122959083)

[Работа с HTTP запросами 15](#_Toc122959084)

[Код запроса 15](#_Toc122959085)

[Результат работы запроса 16](#_Toc122959086)

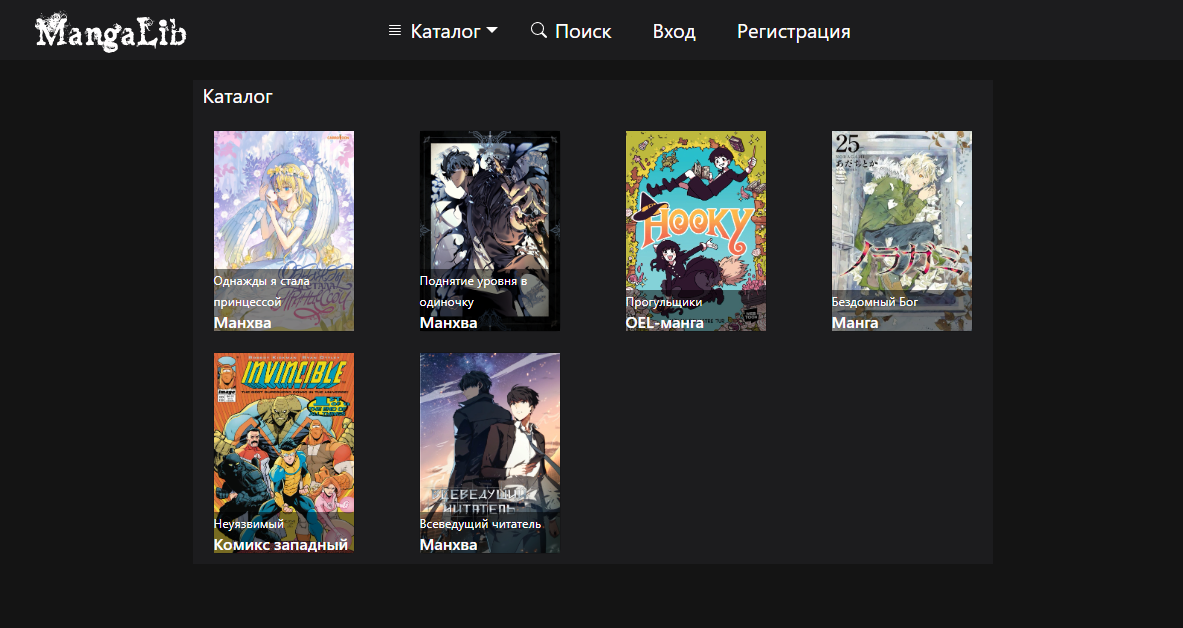
**Цель работы**: создать web приложение каталог с сайтом и сервером для хранения комиксов.

**Ход работы**

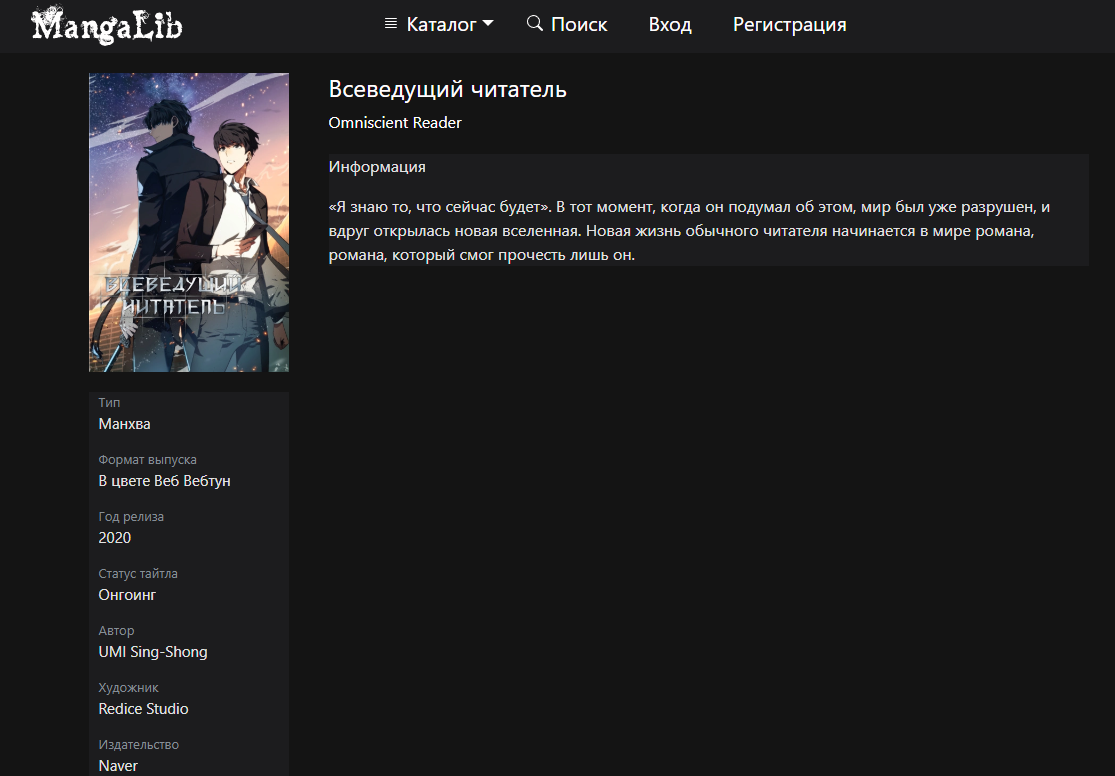
# HTML. Разработка макетов и верстка шаблонов web-приложения с помощью языков HTML и CSS

1. Выбрать шаблон или разработать свой для web-приложения. Шаблон должен включать минимум страницы (главная, контентная страница, страница авторизации/регистрации).
2. Сделать макеты страниц с помощью языка разметки HTML.
3. Стилизовать страницы с помощью языка CSS.
4. Продемонстрировать внешний вид разработанных страниц.

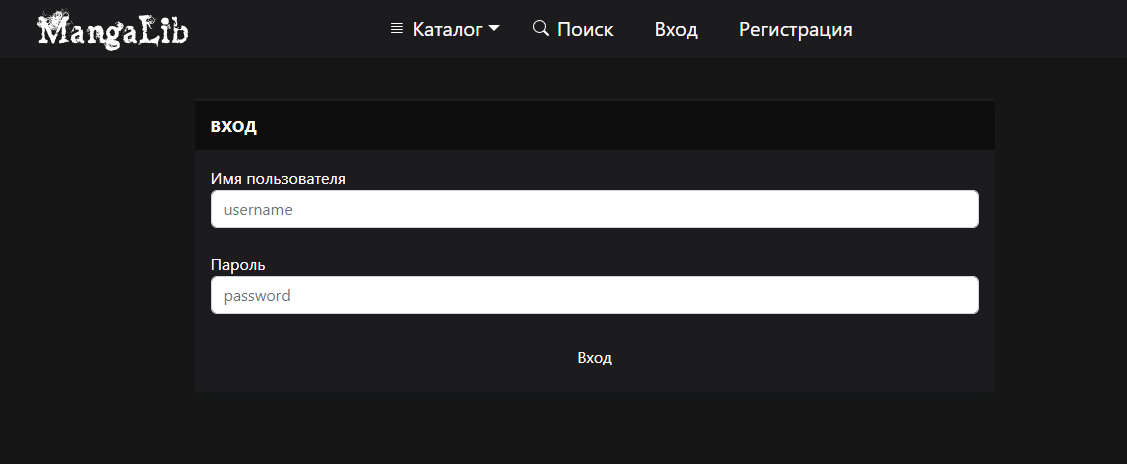
## Главная страница



## Контентная страница



## Страница авторизации



1. Изучить основы разработки на языке JavaScript.
2. Изучить основы разработки frontend-приложения.
3. Развернуть базовое приложение на фреймворке.
4. Добавить необходимые компоненты и перенести в них вёрстку, сделанную в прошлой лабораторной работе.
5. Продемонстрировать работу web-приложения.

# Клиентское программирование

## Главная страница

import '../css/catalog.css';

import { Card } from './card';

import React from 'react';

import { getMangas } from '../api/MangaApi';

import MangaData from '../models/Manga';

export function Catalog() {

    const [manga, setManga] = React.useState<[MangaData]>()

    React.useEffect(()=>{

        getMangas(setManga)

    },[])

    return <>

        <main>

            <div className="catalog">

                <div className="title">

                    Каталог

                </div>

                <div className="content row">

                    {manga?.map(data => <Card{...data} key={data.id} />) }

                </div>

            </div>

        </main>

    </>;

}

## Контентная страница

import { PageSidebar } from "../components/pageSidebar";

import { PageContent } from "../components/pageContent";

import { useParams } from "react-router-dom";

import React from "react";

import { getMangaId } from "../api/MangaApi";

import MangaData from "../models/Manga";

export function PageManga() {

    const {mangaId} = useParams()

    const [manga, setManga] = React.useState<MangaData>({

        id: 0,

        img: '',

        name\_ru: '',

        name\_en: '',

        info: '',

        type: '',

        release\_format: '',

        release\_year: '',

        status: '',

        author: '',

        artist: '',

        publishing: ''

    })

    React.useEffect( () =>{

        getMangaId(setManga, mangaId)

    }, [mangaId])

    return <>

        <main>

            <div className="page">

                <PageSidebar {...manga}/>

                <PageContent {...manga}/>

            </div>

        </main>

    </>

}

## Страница авторизации

import {Field, Form, Formik} from 'formik';

import React from 'react';

import Widget from '../components/ContentWidget';

import 'bootstrap/dist/css/bootstrap.min.css';

import loginValidate from '../validations/loginValidate';

import { postLogin } from '../api/loginApi';

import { useNavigate } from 'react-router-dom';

import authContext from '../components/AuthContext';

export interface LoginValues {

    username: string,

    password: string

}

const Login = () => {

    const navigate = useNavigate();

    const auth\_context = React.useContext(authContext);

    const onSubmit = (values: LoginValues) =>{

        postLogin(values, auth\_context)

        navigate('/')

    }

    return (

        <Widget title='Вход'>

            <Formik

                initialValues={{

                username: '',

                password: '',

                }}

                onSubmit={onSubmit}

                validate={loginValidate}

            >{({ errors, touched }) => (

                <Form>

                    <label htmlFor="username">Имя пользователя</label>

                    <Field className='form-control' id="username" name="username" placeholder="username" />

                    {errors.username && touched.username &&

                        <div className='text-danger'>

                            {errors.username}

                        </div>}

                    <br/>

                    <label htmlFor="password">Пароль</label>

                    <Field type='password' className='form-control' id="password" name="password" placeholder="password" />

                    {errors.password && touched.password &&

                        <div className='text-danger'>

                            {errors.password}

                        </div>}

                    <br/>

                    <div className='text-center'>

                        <button className='btn' type="submit">Вход</button>

                    </div>

                    </Form>

            )}

            </Formik>

        </Widget>

    )

}

export default Login;

# Серверное программирование

1. Развернуть базовое приложение.
2. Настроить конфигурацию работы приложения с docker.
3. Добавить модуль для работы с API.
4. Добавить несколько контроллеров со статическими данными.
5. Продемонстрировать работу API в Postman.

## docker-compose.yaml

version: "3.7"

services:

  db:

    image: postgres

    volumes:

      - postgres\_data:/var/lib/postgresql/data

    env\_file:

      - .env.template

  backend:

    build:

      dockerfile: Dockerfile

      context: ../backend

    env\_file:

      - .env.template

    depends\_on:

      - db

    restart: always

    tty: true

  frontend:

    build:

      dockerfile: Dockerfile

      context: ../frontend

    ports:

      - "81:80"

    depends\_on:

      - backend

volumes:

  postgres\_data:

## Dockerfile программы сервера

FROM python:latest

WORKDIR /

COPY ./requirements.txt .

RUN pip install -r requirements.txt

COPY . .

EXPOSE 8000

CMD ["sh", "run.sh"]

## Dockerfile для nginx

FROM node:lts-alpine as deps

WORKDIR /var/www/frontend

COPY package.json .

RUN npm install

FROM node:lts-alpine as build

WORKDIR /var/www/frontend

COPY --from=deps /var/www/frontend/node\_modules ./node\_modules

COPY . .

RUN npm run build

FROM nginx:stable-alpine

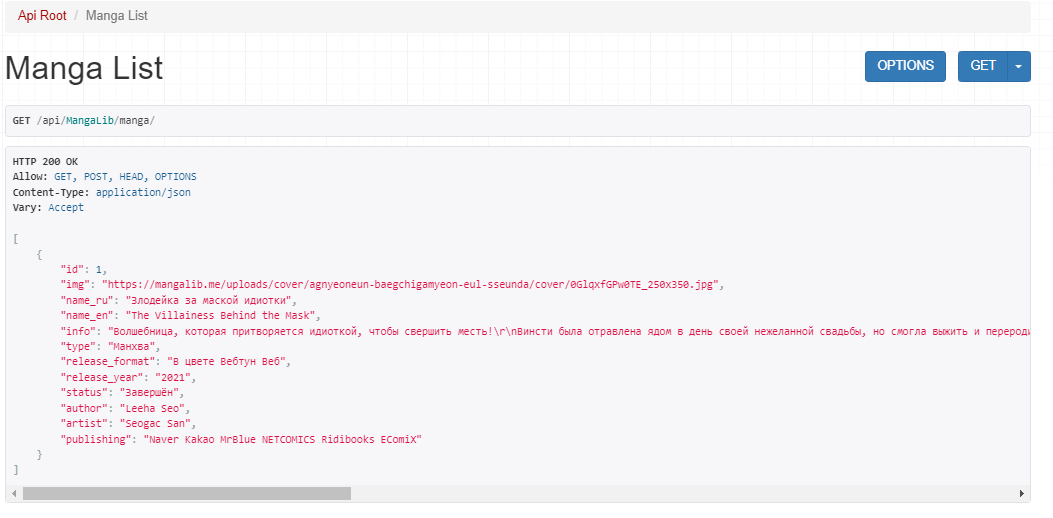
COPY --from=build /var/www/frontend/build /usr/share/nginx/html

COPY --from=build /var/www/frontend/nginx.conf /etc/nginx/conf.d/default.conf

EXPOSE 80

CMD ["nginx", "-g", "daemon off;"]

## Запрос

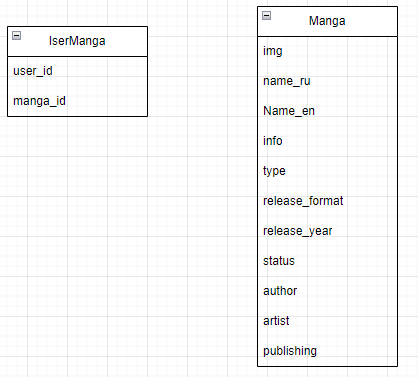


# Разработка и проектирование базы данных

1. Выбрать подходящую СУБД.
2. Изучить методы взаимодействия web-приложения с базой данных.
3. Разработать структуру базы данных.
4. Разработать соответствующие модели в приложении.
5. В отчёт приложить схему базы данных, а также код одной из моделей (на своё усмотрение).

**СУБД - PostrgeSQL**

## Схема базы данных



## Код моделей

class Manga(models.Model):

    img = models.URLField(max\_length=500)

    name\_ru = models.CharField(max\_length=255)

    name\_en = models.CharField(max\_length=255)

    info = models.TextField()

    type = models.CharField(max\_length=255)

    release\_format = models.CharField(max\_length=255)

    release\_year = models.CharField(max\_length=255)

    status = models.CharField(max\_length=255)

    author = models.CharField(max\_length=255)

    artist = models.CharField(max\_length=255)

    publishing = models.CharField(max\_length=255)

class UserMangas(models.Model):

    user\_id = models.ForeignKey(django.contrib.auth.models.User, on\_delete=models.CASCADE)

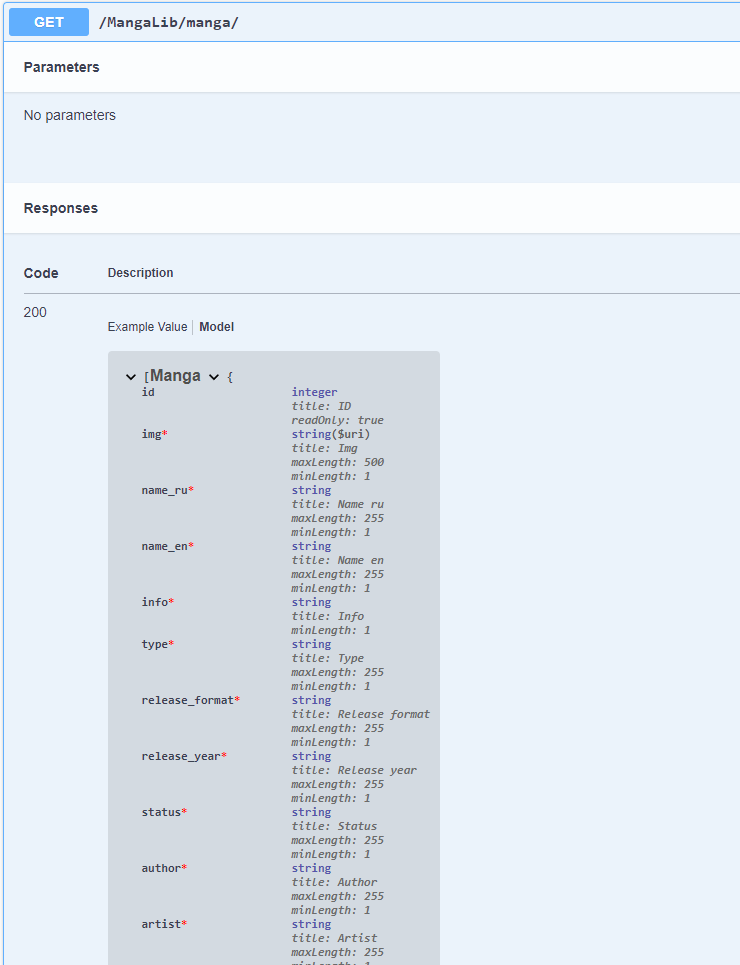
    manga\_id = models.ForeignKey(Manga, on\_delete=models.CASCADE, null = True)

# REST API

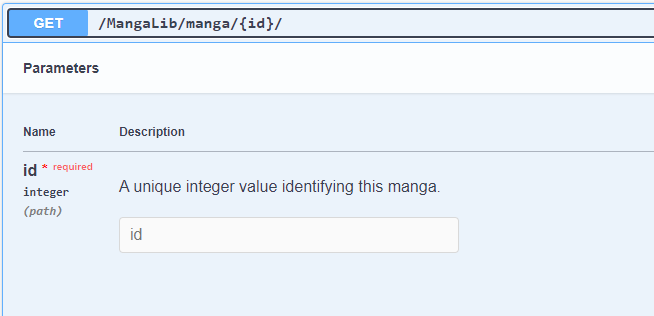
1. Изучить структуру формата представления данных JSON.
2. Изучить типы запросов к API: HEAD, GET, POST, PUT, DELETE.
3. Спроектировать и реализовать собственное REST API (Получение, создание, изменение и удаление каких-либо объектов).
4. В отчёт необходимо предоставить документацию к использованию методов. (Либо словесным описание, либо через Swagger)

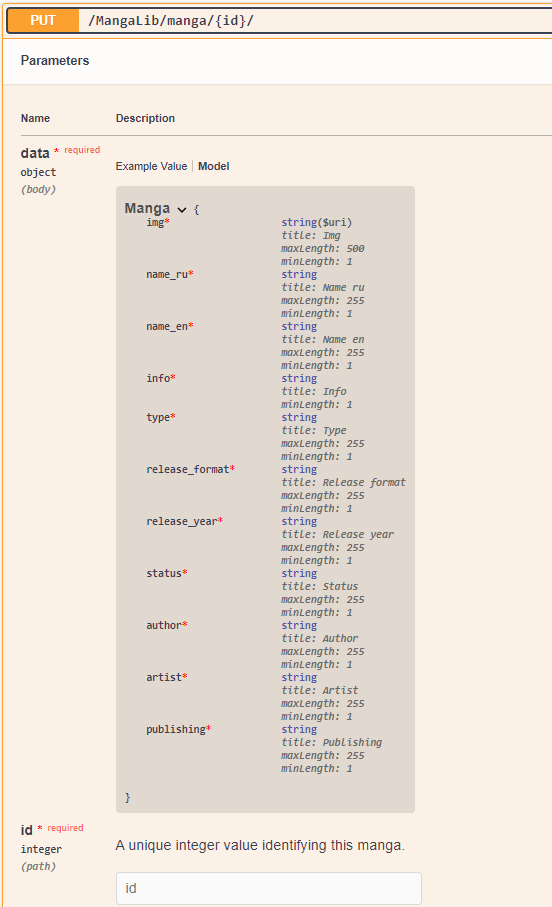
## 

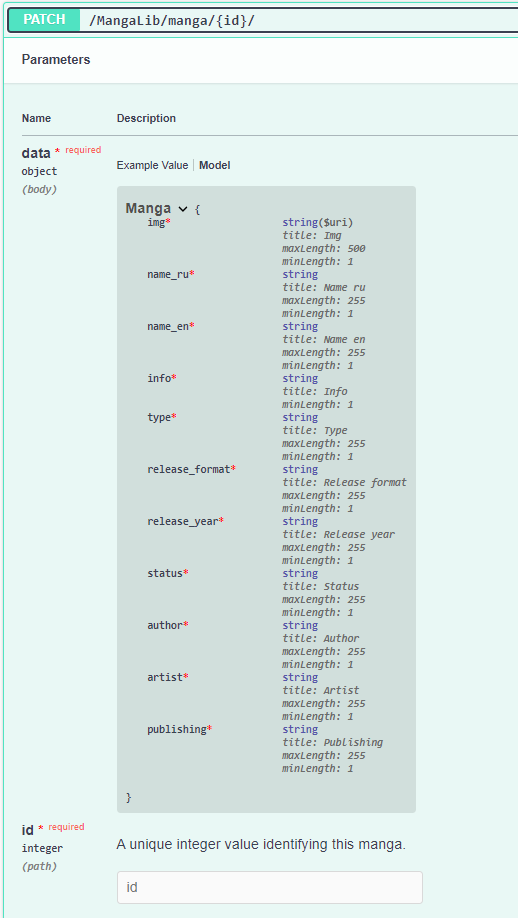
## Схема API (Swagger)













# Работа с HTTP запросами

1. Изучить возможности js для отправки http запросов.
2. Выбрать подходящую библиотеку для работы с запросами.
3. Реализовать взаимодействие фронтенда с REST API, спроектированном в прошлой лабораторной работе.
4. Продемонстрировать работу взаимодействия фронтенд приложения с REST API.

## Код запроса

import axios, { AxiosRequestConfig } from 'axios';

import Axios, {AxiosError, AxiosResponse} from 'axios';

import { AuthContext } from '../components/AuthContext';

import { baseURL } from '../constants';

import MangaData, { UserMangaData } from '../models/Manga';

const mangaPath = 'http://127.0.0.1:8000/api/MangaLib/manga/'

const userMangaPath = 'http://localhost:8000/api/MangaLib/userManga/'

export async function getMangas(resultHandler: (data: any)=>void){

    Axios.get(mangaPath,

        { params:{}, responseType: "json" }

    ).then

    (result => {

        let data: MangaData[] = (result as AxiosResponse<MangaData[]>).data;

        resultHandler(data);

    })

    .catch((error: AxiosError) => {

        alert(error.message);

    });

}

export async function getMangaId(resultHandler: (data: any)=>void, id: string|undefined){

    Axios.get(mangaPath + id + "/",

        { params:{}, responseType: "json" }

    ).then

    (result => {

        let data: MangaData = (result as AxiosResponse<MangaData>).data;

        resultHandler(data);

    })

    .catch((error: AxiosError) => {

        alert(error.message);

    });

}

## Результат работы запроса

