**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: **«Фитнес центр**»

|  |  |
| --- | --- |
|  | Выполнила:  студентка группы ПВ-192  Михайлюкова Дарья Владислав  Проверил:  Картамышев С.В. |

Белгород 2022

**Оглавление**

[HTML. Разработка макетов и верстка шаблонов web-приложения с помощью языков HTML и CSS 3](#_Toc122964266)

[Главная страница 3](#_Toc122964267)

[Контентная страница 3](#_Toc122964268)

[Страница авторизации 5](#_Toc122964269)

[Клиентское программирование 5](#_Toc122964270)

[Главная страница 5](#_Toc122964271)

[Контентная страница 6](#_Toc122964272)

[Страница авторизации 7](#_Toc122964273)

[Серверное программирование 9](#_Toc122964274)

[docker-compose.yaml 9](#_Toc122964275)

[Dockerfile программы сервера 10](#_Toc122964276)

[Запрос в Postman 10](#_Toc122964277)

[Разработка и проектирование базы данных 11](#_Toc122964278)

[Схема базы данных в нотации IDEF1X 11](#_Toc122964279)

[Код модели Возможность 11](#_Toc122964280)

[REST API 11](#_Toc122964281)

[Схема API (в Postman) 12](#_Toc122964282)

[Работа с HTTP запросами 13](#_Toc122964283)

[Код апи 13](#_Toc122964284)

[Результат работы 15](#_Toc122964285)

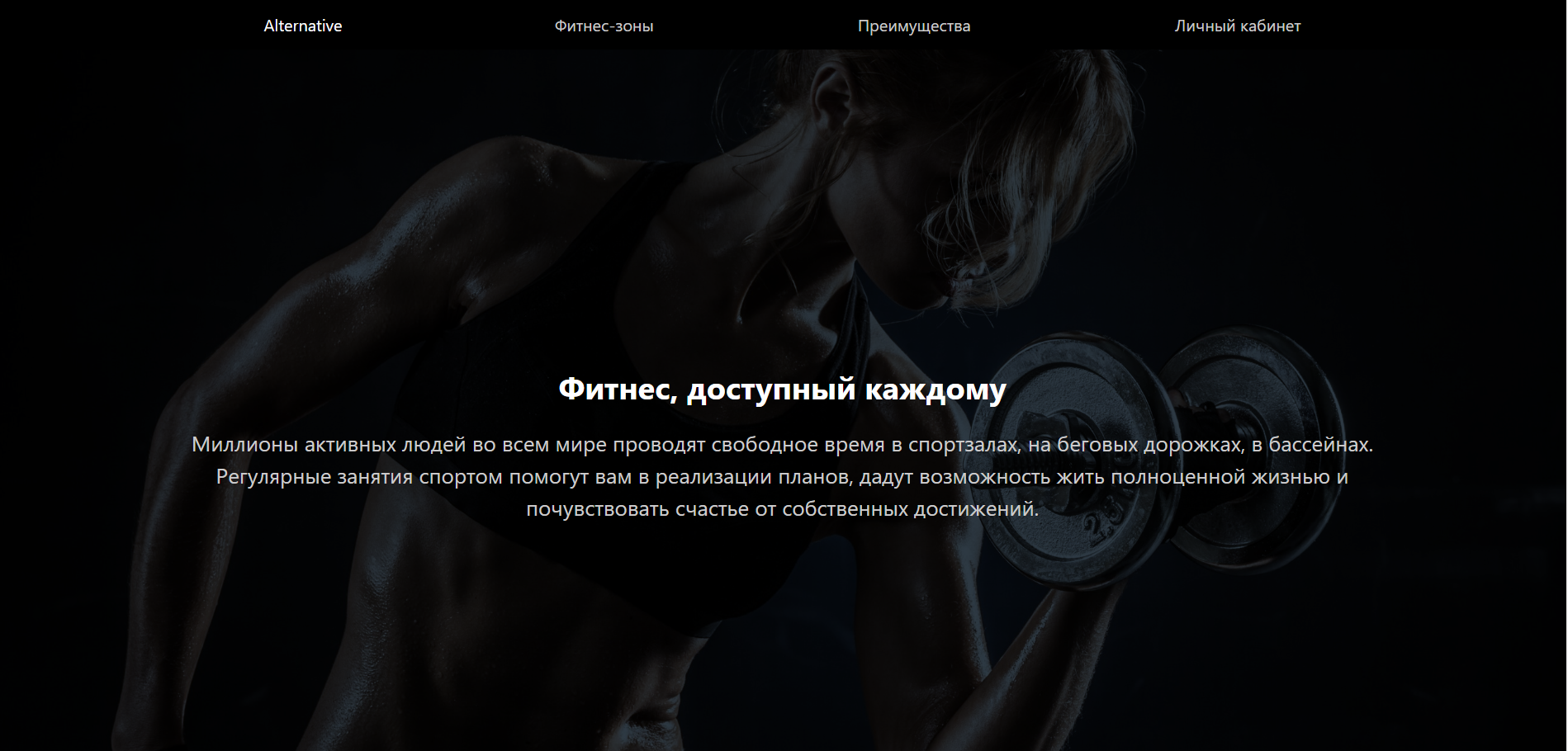
**Цель работы**: создать web приложение с сайтом и сервером для фитнес центра.

**Ход работы**

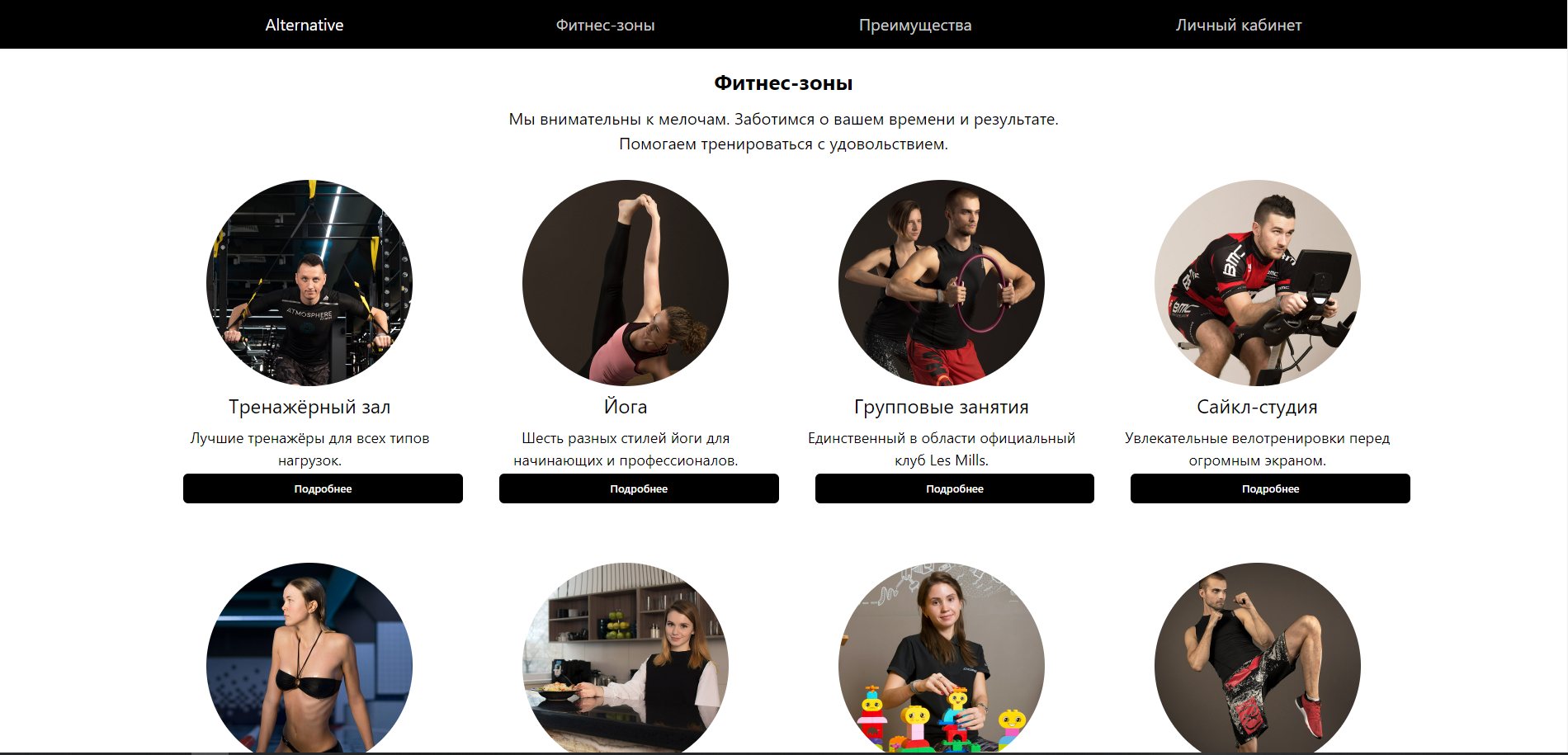
# HTML. Разработка макетов и верстка шаблонов web-приложения с помощью языков HTML и CSS

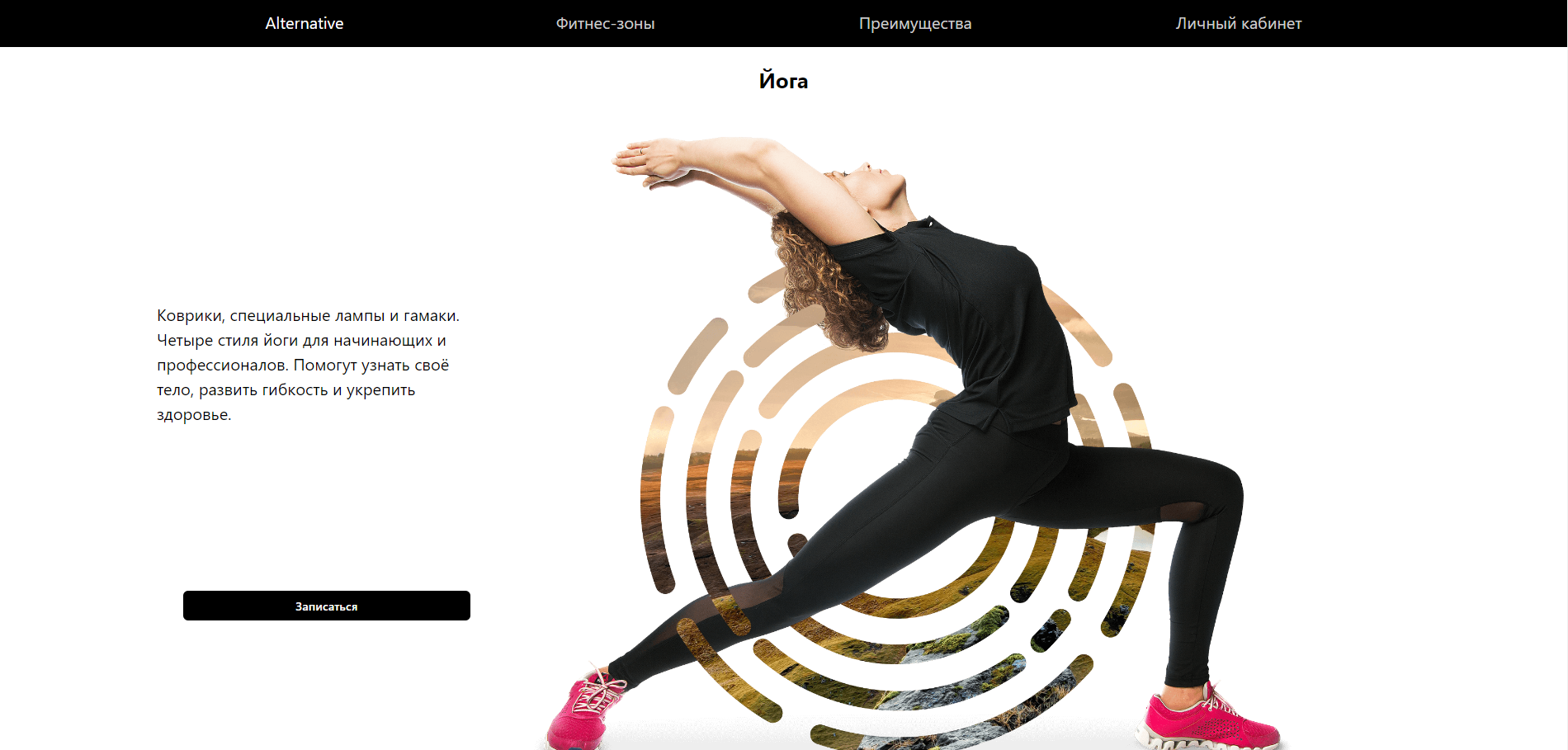
1. Выбрать шаблон или разработать свой для web-приложения. Шаблон должен включать минимум страницы (главная, контентная страница, страница авторизации/регистрации).
2. Сделать макеты страниц с помощью языка разметки HTML.
3. Стилизовать страницы с помощью языка CSS.
4. Продемонстрировать внешний вид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

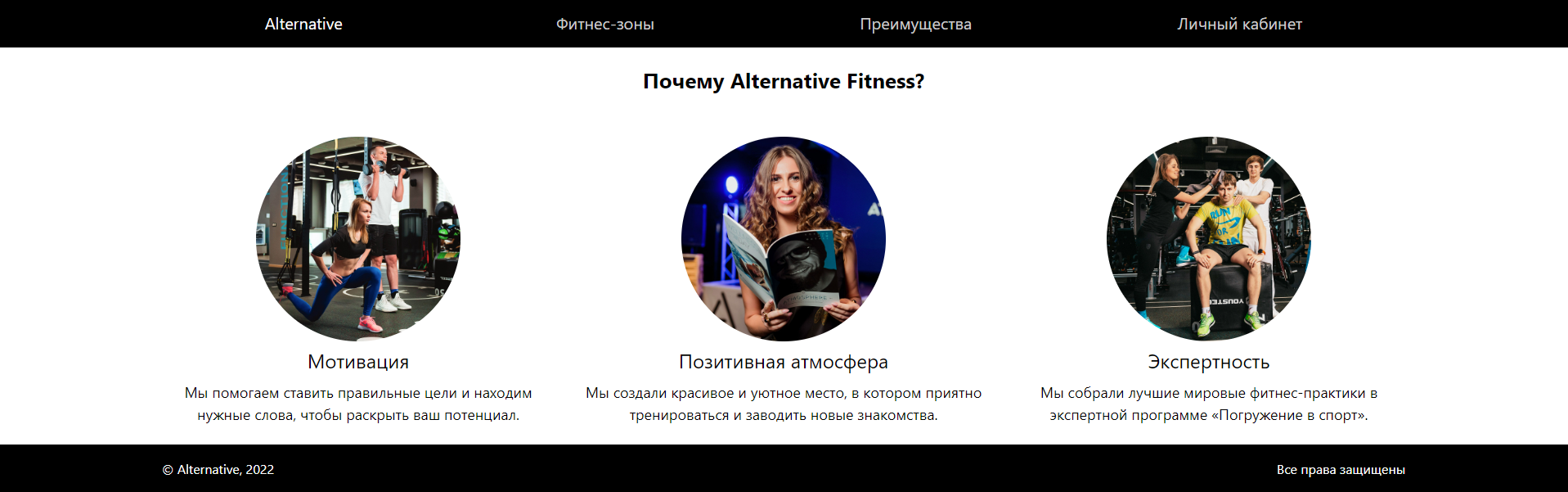
## Главная страница



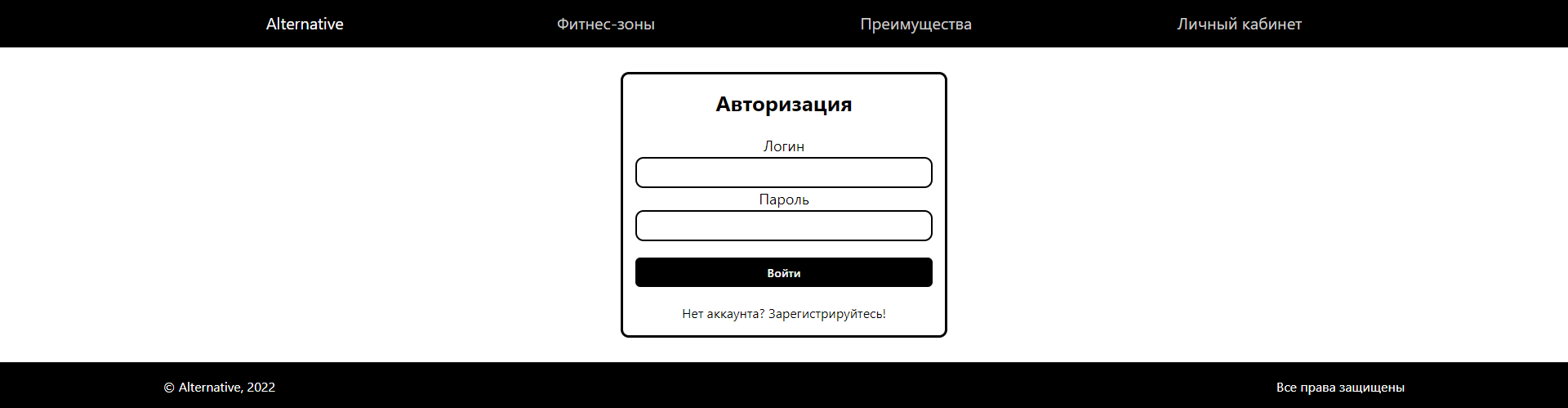
## Контентная страница

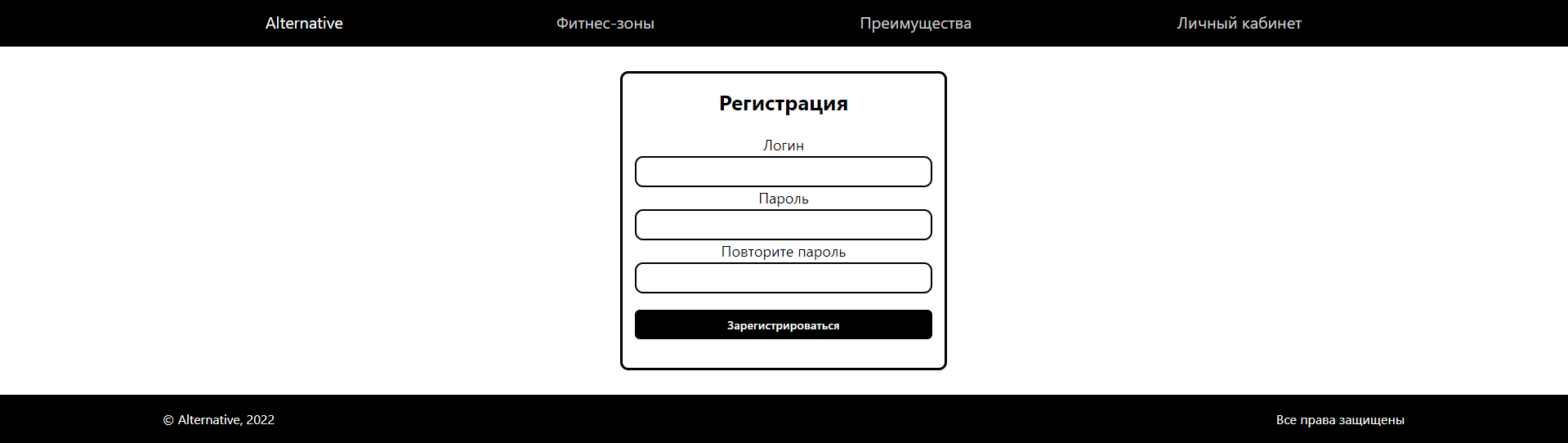






## Страница авторизации





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 React from "react";  
import "./main.css"  
  
export function Main() {  
  
 return(  
 <div id="top">  
 <div className="container">  
 <div className = "title">  
 Фитнес, доступный каждому  
 </div>  
 <div className= "description">  
 Миллионы активных людей во всем мире проводят свободное время в спортзалах, на беговых дорожках, в  
 бассейнах. Регулярные занятия спортом помогут вам в реализации планов, дадут возможность жить  
 полноценной жизнью и почувствовать счастье от собственных достижений.  
 </div>  
 </div>  
 </div>  
 )  
}

## Контентная страница

import React from "react"  
import **advantagesList** from "./advantagesList";  
import Element from "../element/Element";  
import './advantages.css'  
  
export function Advantages() {  
  
 const elementComponents: JSX.Element[] = **advantagesList**.map(op =>  
 <Element  
 key={op.id}  
 id={op.id}  
 img\_name={op.img\_name}  
 title={op.title}  
 description={op.description}  
 />);  
  
 return(  
 <div className="advantages-page">  
 <div className="title">  
 Почему Alternative Fitness?  
 </div>  
 <main className="container">  
 <section className="grid grid-cols-3 gap-6">  
 {elementComponents}  
 </section>  
 </main>  
 </div>  
 )  
}

import React from "react"  
import **capabilitiesList** from "./capabilitiesList";  
import './capabilities.css'  
import ElementWithButton from "../element\_with\_button/ElementWithButton";  
import ElementWithInfoData from "../../data/ElementWithInfoData";  
import {getCapabilities} from "../../api/CapabilitiesApi";  
  
export function Capabilities() {  
  
 const [capabilities, setCapabilities] = React.useState<ElementWithInfoData[]>(**capabilitiesList**);  
  
 React.useEffect(() => {  
 const get\_capabilities = async () => {  
 const result: ElementWithInfoData[] | null = await getCapabilities();  
 if (result !== undefined && result !== null) {  
 setCapabilities(result)  
 }  
 }  
 get\_capabilities()  
 },  
 []  
 );  
  
 const elementComponents: JSX.Element[] = capabilities.map(op =>  
 <ElementWithButton  
 key={op.id}  
 id={op.id}  
 img\_name={op.img\_name}  
 title={op.title}  
 description={op.description}  
 more\_info={op.more\_info}  
 info\_image={op.info\_image}  
 />);  
  
 return(  
 <div className="capabilities-page">  
 <div className="title">  
 Фитнес-зоны  
 </div>  
 <div className= "description">  
 Мы внимательны к мелочам. Заботимся о вашем времени и результате. Помогаем тренироваться с удовольствием.  
 </div>  
 <section className="grid grid-cols-4 gap-3">  
 {elementComponents}  
 </section>  
 </div>  
 )  
}

## Страница авторизации

import React from "react";  
  
import { useNavigate, NavigateFunction } from "react-router-dom";  
import {User} from "../../data/User";  
import **AuthApi** from "../../api/AuthApi";  
import './registration.css'  
  
function Registration() {  
 const navigate:NavigateFunction = useNavigate();  
 const [login, setLogin] = React.useState<string>('');  
 const [password, setPassword] = React.useState<string>('');  
 const [passwordCopy, setPasswordCopy] = React.useState<string>('');  
 const [isPressed, setPressed] = React.useState<boolean>(false);  
 const [user, setUser] = React.useState<User>({ token: '', refresh\_token:'' });  
 const [error, setError] = React.useState<string | null>(null);  
 React.useEffect(() => {  
 if(user.token === ''){  
 return;  
 }  
 **localStorage**.setItem('token', user.token);  
 **localStorage**.setItem('refresh\_token', user.refresh\_token);  
 navigate('/profile')  
  
 },  
 [user]  
 );  
  
 React.useEffect(() => {  
 if(error === null){  
 return;  
 }  
 alert(error)  
  
 },  
 [error]  
 );  
  
  
 function handleSubmit(event: any) {  
 event.preventDefault()  
 setError(null)  
 setPressed(true)  
 if(password !== passwordCopy){  
 alert('Пароли должны совпадать!')  
 }  
 if (login !== '' && password !== '') {  
 **AuthApi**.postAuth(setUser, setError, login, password)  
 }  
 }  
  
 function handleLogin(event: any){  
 event.preventDefault()  
 setLogin(event.target.value.trim())  
 }  
  
 function handlePassword(event: any){  
 event.preventDefault()  
 let value = event.target.value.trim()  
 if (event.target.name === 'password') {  
 setPassword(value)  
 } else {  
 setPasswordCopy(value)  
 }  
 }  
  
 return (  
 <React.Fragment>  
 <main className="registration">  
 <section className="reg-form">  
 <div className="title">  
 <h1>Регистрация</h1>  
 </div>  
 <form method="post">  
 <div className="text">  
 <div><label>Логин</label></div>  
 <div><input type="text" className="text-input" name="login" required onChange={evt => handleLogin(evt)} /></div>  
 {(isPressed && login === '') && <div className="errors">заполните поле</div>}  
 </div>  
 <div className="text">  
 <div><label>Пароль</label></div>  
 <div><input type="password" className="text-input" name="password" required onChange={evt => handlePassword(evt)} /></div>  
 {(isPressed && password === '') && <div className="errors">введите пароль</div>}  
 </div>  
 <div className="text">  
 <div><label>Повторите пароль</label></div>  
 <div><input type="password" className="text-input" name="password-copy" required onChange={evt => setPasswordCopy(evt.target.value)} /></div>  
 {(isPressed && password === '') && <div className="errors">повторите пароль</div>}  
 {(password !== passwordCopy) && <div className="errors">пароли не совпадают</div>}  
 </div>  
 {error !== null &&  
 <div className="reg-tip">  
 <div className="errors">Что-то пошло не так!</div>  
 </div>  
 }  
 <div className="btn-container pointer-events-auto ml-8 rounded-md bg-black py-2 px-3 text-[0.9rem] font-semibold leading-5 text-white hover:bg-gray-900">  
 <button type="submit" onClick={(event)=>handleSubmit(event)} className="btn button-default btn-reg">Зарегистрироваться</button>  
 </div>  
 </form>  
 </section>  
 </main>  
 </React.Fragment>  
 );  
}  
  
  
  
export default Registration;

# 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.

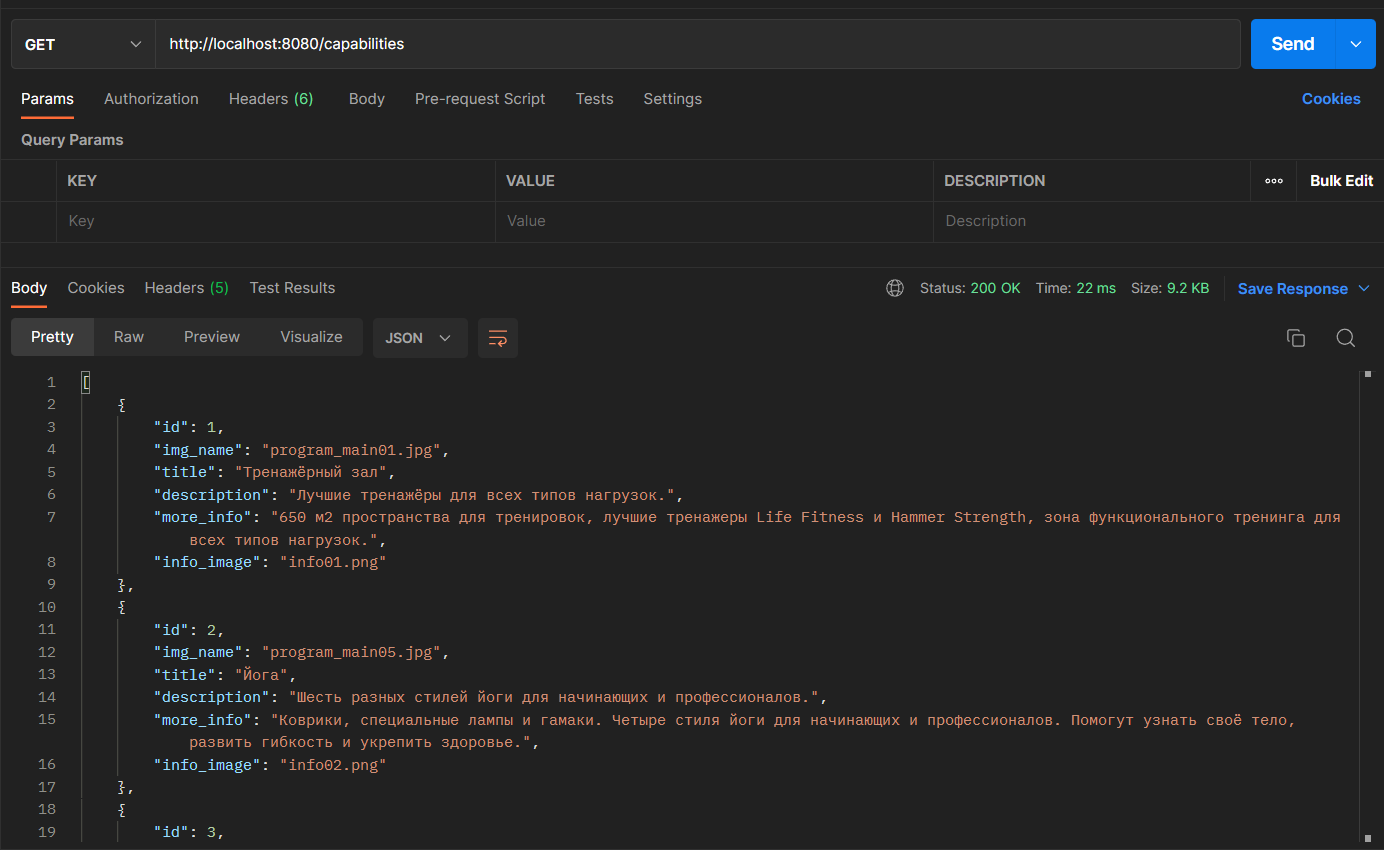
## docker-compose.yaml

version: "3"  
  
services:  
 postgres\_db:  
 image: postgres:12.13  
 container\_name: postgres\_database  
 restart: always  
 env\_file:  
 - .env  
 volumes:  
 - postgres\_data:/var/lib/postgresql/data/  
 ports:  
 - "5432:5432"  
  
 aiohttp\_server:  
 build: ./server  
 container\_name: aiohttp\_server  
 depends\_on:  
 - postgres\_db  
 env\_file:  
 - .env  
 restart: on-failure  
 ports:  
 - "8081:8080"  
  
 web:  
 build: ./nginx  
 container\_name: web-nginx  
 depends\_on:  
 - aiohttp\_server  
 volumes:  
 - ./nginx/conf.d/:/etc/nginx/conf.d/  
 ports:  
 - "8080:80"  
  
   
  
volumes:  
 postgres\_data:

## Dockerfile программы сервера

FROM python:3.9  
  
WORKDIR /  
RUN /usr/local/bin/python -m pip install --upgrade pip  
  
COPY requirements.txt /  
RUN pip3 install --upgrade pip -r requirements.txt  
  
COPY . .  
  
CMD ["python","main.py"]

## Запрос в Postman

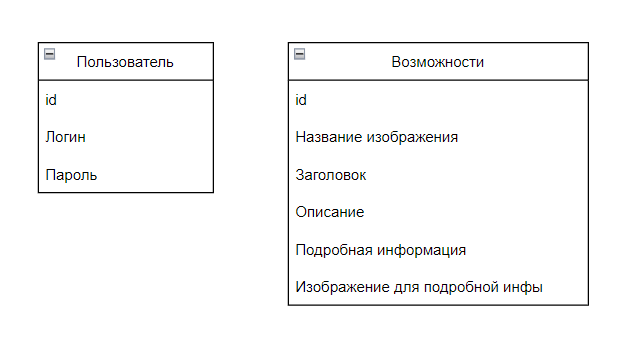


# Разработка и проектирование базы данных

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 так же код одной из моделей (на своё усмотрение).

**СУБД - PostrgeSQL**

## Схема базы данных в нотации IDEF1X



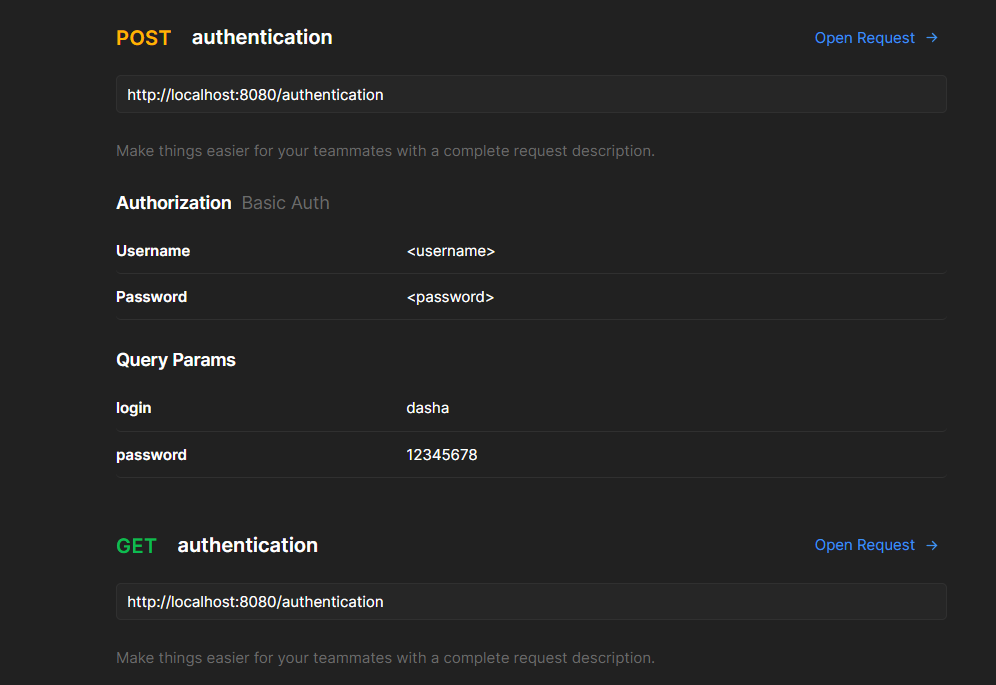
## Код модели Возможность

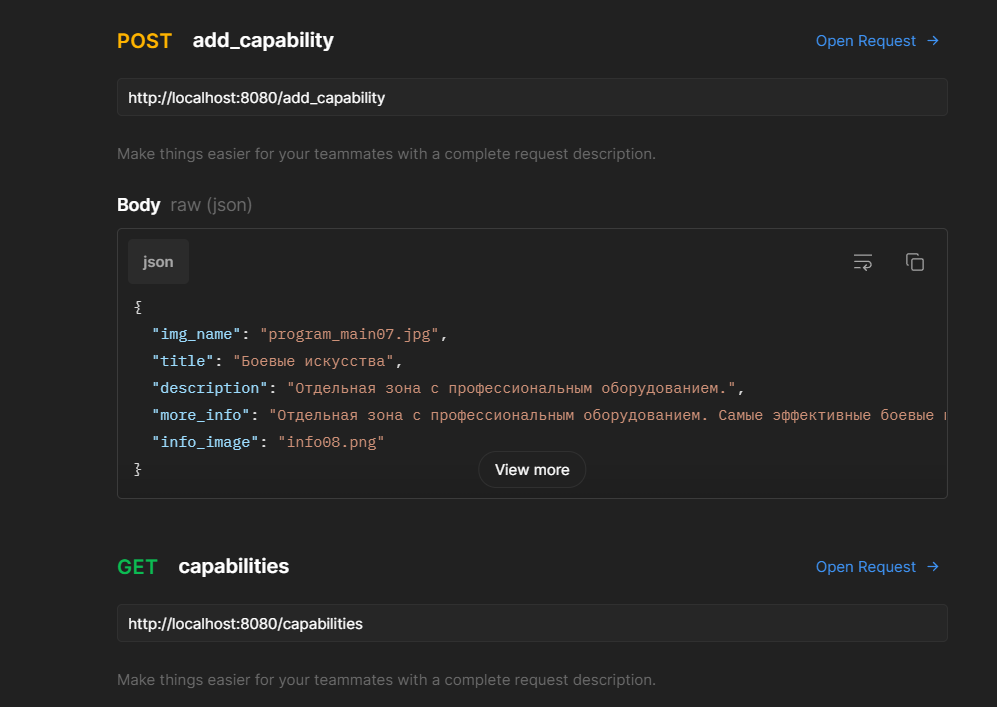
class Capability(Base):  
 \_\_tablename\_\_ = "capability"  
 id = sa.Column(sa.Integer, primary\_key=True)  
 img\_name = sa.Column(sa.String(255))  
 title = sa.Column(sa.String(255))  
 description = sa.Column(sa.String(255))  
 more\_info = sa.Column(sa.String(255))  
 info\_image = sa.Column(sa.String(255))  
  
  
capability\_table = Capability.\_\_table\_\_

# 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 (в Postman)





# 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 апи

import **Axios**, {AxiosError} from "axios";  
import {User} from "../data/User";  
  
const authenticationBasePath: string = 'http://localhost:8080/authentication';  
const refreshBasePath: string = 'http://localhost:8080/refresh';  
const isTokenValidPath: string = 'http://localhost:8080/check-token';  
  
export function getAuth\_(resultHandler: (data: any) => void, errorHandler: (data: any) => void, login: string, password: string) {  
  
 **Axios**.get(authenticationBasePath,  
 {  
 auth: {  
 username: login,  
 password: password  
 },  
 responseType: 'json'  
 }  
 ).then  
 (response => {  
 **console**.log(response);  
 resultHandler((oldData: Object) => ({...oldData, ...response.data}));  
 })  
 .catch((error: AxiosError) => {  
 errorHandler(error.message);  
 });  
  
}  
  
export function postAuth\_(resultHandler: (data: any) => void, errorHandler: (data: any) => void, login: string, password: string) {  
  
 **Axios**.post(authenticationBasePath,  
 null,  
 {  
 auth: {  
 username: login,  
 password: password  
 },  
 responseType: 'json'  
 }  
 ).then  
 (response => {  
 resultHandler((oldData: Object) => ({...oldData, ...response.data}));  
 })  
 .catch((error: AxiosError) => {  
 errorHandler(error.message);  
 });  
  
}  
  
export function getRefresh\_() {  
 const token = **localStorage**.getItem('refresh\_token')  
 return **Axios**.get(refreshBasePath,  
 {  
 headers: {'Authorization': '' + token},  
 responseType: 'json'  
 }  
 ).then  
 (response => {  
 const data: User = response.data;  
 **localStorage**.setItem('token', data.token);  
 return true;  
 })  
 .catch((error: AxiosError) => {  
 **localStorage**.setItem('token', '');  
 **localStorage**.setItem('refresh\_token', '');  
 return null;  
 });  
  
}  
  
export function checkToken\_() {  
  
 const token = **localStorage**.getItem('token')  
 return **Axios**.get(isTokenValidPath,  
 {  
 headers: { 'Authorization': ''+ token},  
 responseType: 'json'  
 }  
 ).then  
 (r => {  
 return r.data;  
 })  
 .catch((error: AxiosError) => {  
 if(error.response!.status === 401){  
 return **AuthApi**.getRefresh();  
 }  
 alert(error.message);  
 return null;  
 });  
  
}  
  
const **AuthApi** = {  
 getAuth: getAuth\_,  
 postAuth: postAuth\_,  
 getRefresh: getRefresh\_,  
 checkToken: checkToken\_  
}  
  
export default **AuthApi**;

## Результат работы

