

**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 для соревнований в прохождении компьютерных
игр**»

Выполнил:
студент группы ПВ-192
Орлов-Курेशи Максим
Проверил:
Картамышев С.В.

Белгород 2022

Оглавление

HTML. Разработка макетов и верстка шаблонов web-приложения с помощью языков HTML и CSS	3
Главная страница	3
Контентная страница	4
Страница авторизации	4
Клиентское программирование	5
Главная страница	5
Контентная страница	5
Страница авторизации	6
Серверное программирование	7
docker-compose.yaml	8
Dockerfile программы сервера	8
Dockerfile для nginx	9
Запрос в Postman	9
Разработка и проектирование базы данных	9
Схема базы данных	10
Код модели Game	10
REST API	10
Схема API (Swagger)	11
Работа с HTTP запросами	15
Код запроса	15
Результат работы запроса	16

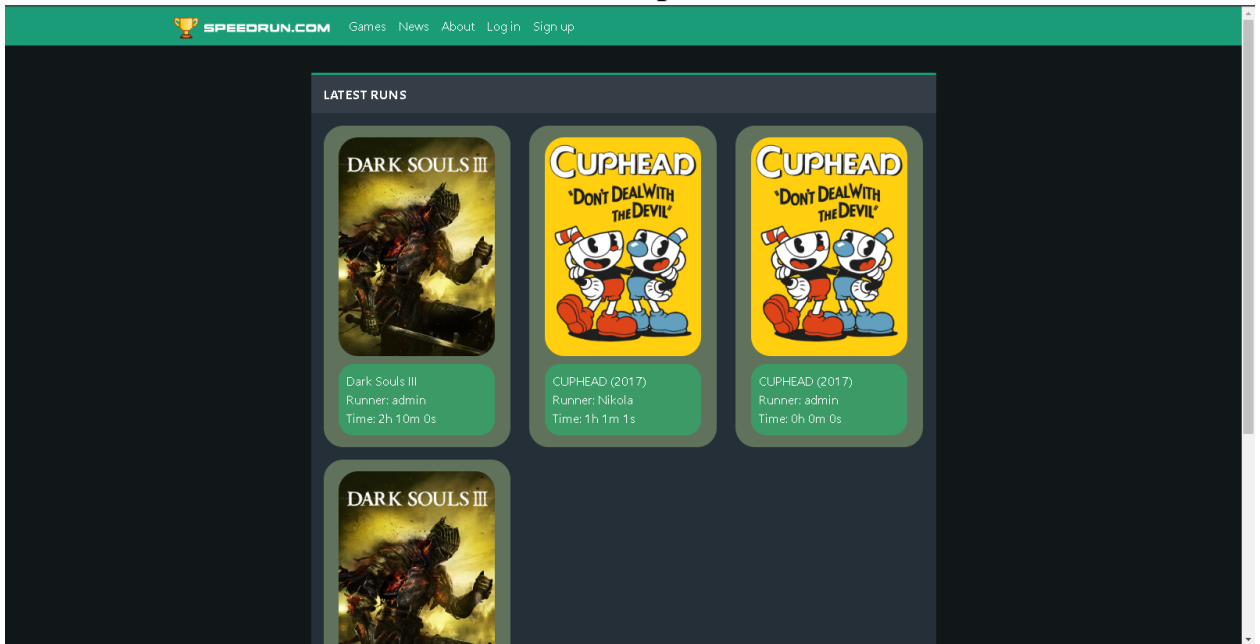
Цель работы: создать web приложение с сайтом и сервером для соревнований в прохождении компьютерных игр.

Ход работы

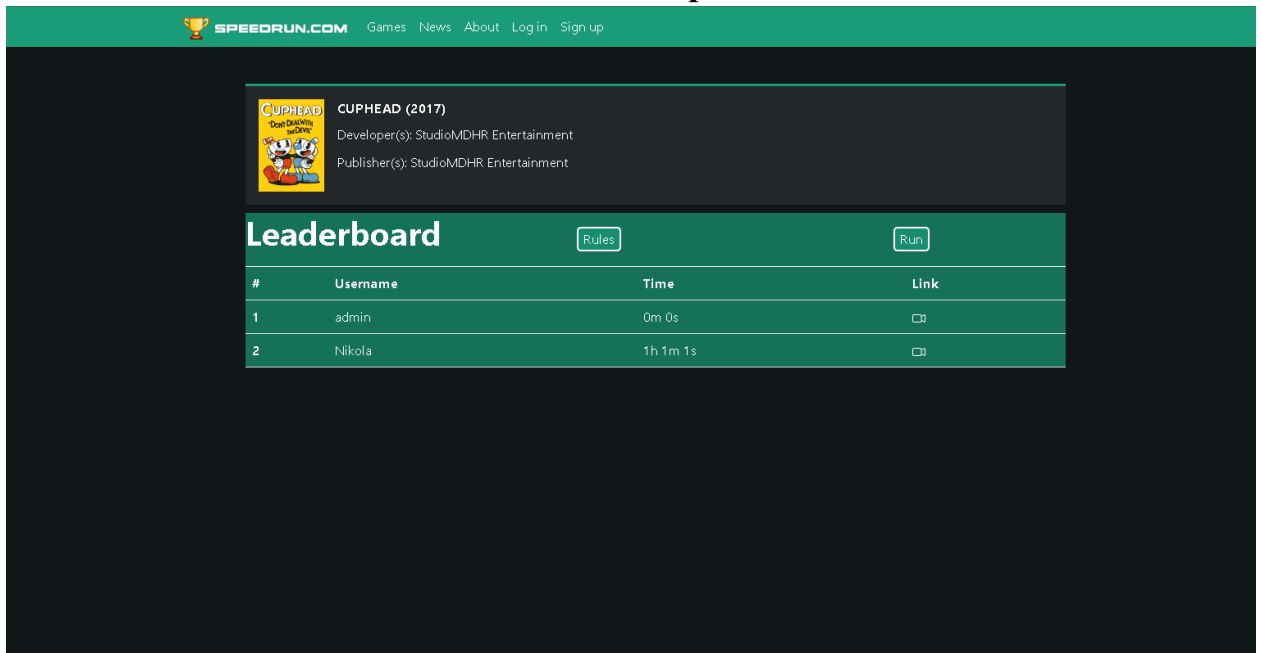
HTML. Разработка макетов и верстка шаблонов web-приложения с помощью языков HTML и CSS

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



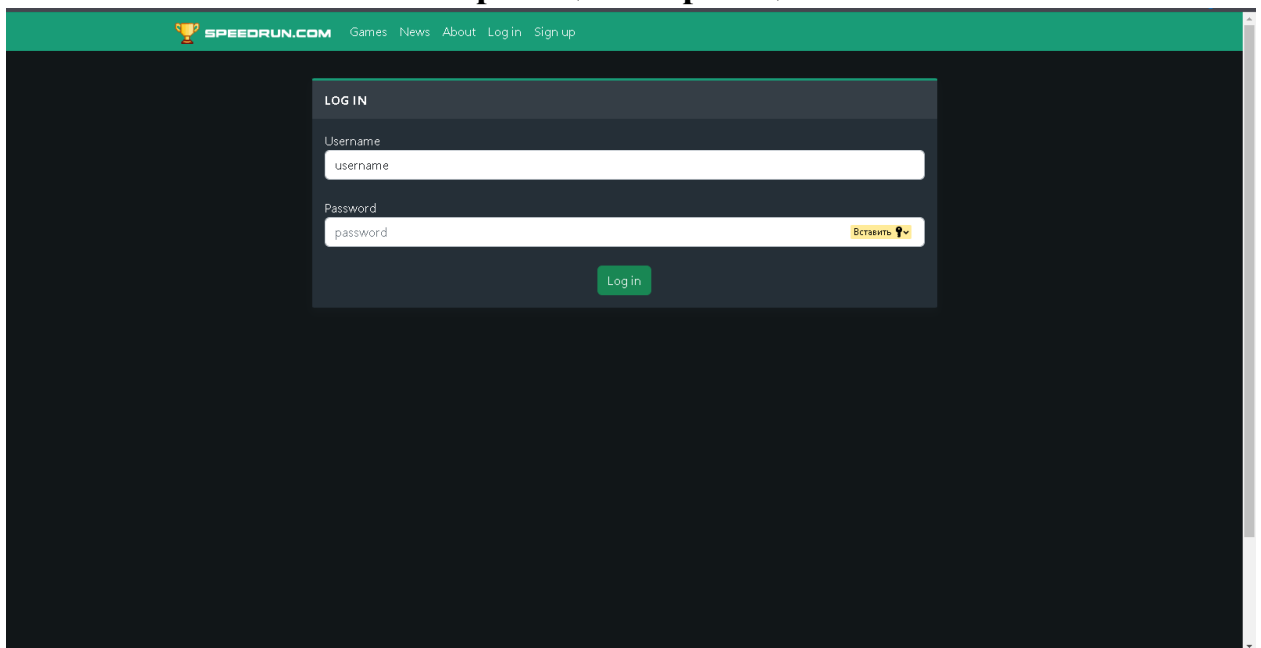
SPEEDRUN.COM Games News About Log in Sign up

CUPHEAD (2017)
Developer(s): StudioMDHR Entertainment
Publisher(s): StudioMDHR Entertainment

Leaderboard [Rules](#) [Run](#)

#	Username	Time	Link
1	admin	0m 0s	🏆
2	Nikola	1h 1m 1s	🏆

Страница авторизации



SPEEDRUN.COM Games News About Log in Sign up

LOG IN

Username

Password
 [Вставить 🔑](#)

[Log in](#)

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, { useEffect } from 'react';
import 'bootstrap/dist/css/bootstrap.min.css';
import RunCard from '../components/RunCard';
import RunData from '../models/RunData';
import { getLatestRuns } from '../apis/homeApi';
import Widget from '../components/ContentWidget';

const Home = () => {
  const [runData, setRunData] = React.useState<RunData[]>([]);
  useEffect( () =>{
    getLatestRuns(setRunData);
  }, [])

  return (
    <Widget title='LATEST RUNS'>
      <div className='row'>
        { runData.map(run => <RunCard {...run} key={run.id} />) }
      </div>
    </Widget>
  )
}

export default Home;
```

Контентная страница

```
import React, { useEffect } from 'react';
import 'bootstrap/dist/css/bootstrap.min.css';
import GameData from '../models/GameData';
import { getGames } from '../apis/gameApi';
import { useParams } from 'react-router-dom';
import GamePreview from '../components/GamePreview';
import Leaderboard from '../components/Leaderboard';

const Game = () => {
  const {gameId} = useParams()
  const [gameData, setGameData] = React.useState<GameData>({
    id: 0, name: '', img: '', publisher: '', developer: '', rule: ''});
  useEffect( () =>{
```

```

        getGames(setGameData, gameId)
      }, [gameId])
    return (
      <>
        <GamePreview {...gameData}/>
        <Leaderboard game_id={gameData.id} rule={gameData.rule} />
      </>
    )
  }
}

export default Game;

```

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 '../apis/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='Log in'>
      <Formik
        initialValues={{
          username: '',
          password: '',
        }}

```

```

        onSubmit={onSubmit}
        validate={loginValidate}
      >({{ errors, touched }} => (
        <Form>
          <label htmlFor="username">Username</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">Password</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 btn-success' type="submit">Log
in</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.yml

```
version: "3.7"

services:
  db:
    image: postgres
    volumes:
      - postgres_data:/var/lib/postgresql/data
    env_file:
      - .env.template

  backend:
    build:
      dockerfile: Dockerfile
      context: ../backend
    volumes:
      - ../backend:/var/www
    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 /var/www/
COPY ./requirements.txt .
RUN pip install -r requirements.txt
COPY ./ .
ENTRYPOINT ./run.sh
```


Дockerfile 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 в Postman

The screenshot shows the Postman interface with a GET request to `http://localhost:8000/api/speedrun/runs`. The response is a JSON array of two speedrun entries. The first entry has id 34, data '2822-11-18', hours 1, minutes 18, seconds 15, video 'qw.qw', username 'admin', and game 1. The second entry has id 35, data '2822-11-11', hours 2, minutes 18, seconds 8, video 'qwerty.ru', username 'admin', and game 1.

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body: Cookies (1) Headers (10) Test Results

Status: 200 OK Time: 608 ms Size: 771 B Save Response

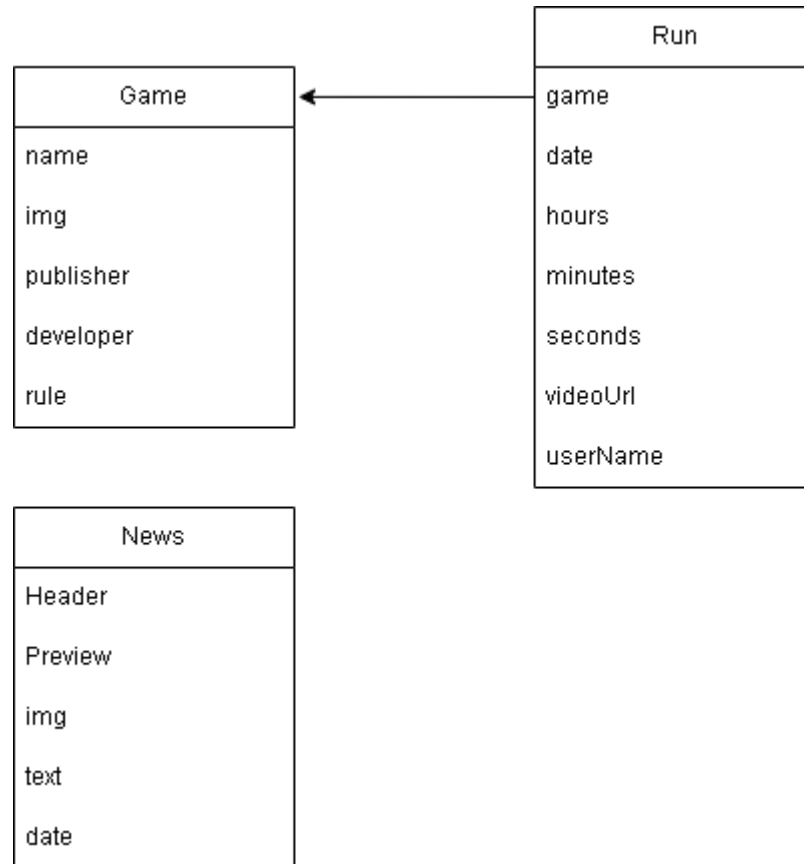
```
1 {
2   {
3     "id": 34,
4     "data": "2822-11-18",
5     "hours": 1,
6     "minutes": 18,
7     "seconds": 15,
8     "video": "qw.qw",
9     "username": "admin",
10    "game": 1
11  },
12  {
13    "id": 35,
14    "data": "2822-11-11",
15    "hours": 2,
16    "minutes": 18,
17    "seconds": 8,
18    "video": "qwerty.ru",
19    "username": "admin",
20    "game": 1
21  }
22 }
```

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 - PostgreSQL

Схема базы данных



Код модели Game

```
class Game(models.Model):
    name = models.CharField(max_length=255)
    img = models.URLField(max_length=500)
    publisher = models.CharField(max_length=255)
    developer = models.CharField(max_length=255)
    rule = models.TextField()
```

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 описанием, либо через Swagger)

Cxema API (Swagger)

GET	/speedrun/games/
Parameters	
No parameters	
Responses	
Code	Description
200	<div>Example Value Model</div> <div><pre>▼ [Game ▼ { id integer title: ID readOnly: true name* string title: Name maxLength: 255 minLength: 1 img* string(\$uri) title: Img maxLength: 500 minLength: 1 publisher* string title: Publisher maxLength: 255 minLength: 1 developer* string title: Developer maxLength: 255 minLength: 1 rule* string title: Rule minLength: 1 }]</pre></div>

POST

/speedrun/games/

Parameters

Name	Description
<div><div><div>data</div><div><div>* required</div></div></div><div><div>object</div><div>(body)</div></div></div>	<div>Example Value Model</div> <div><div>Game</div><div><div><div>Game</div><div><div>name*</div><div>string</div><div>title: Name</div><div>maxLength: 255</div><div>minLength: 1</div></div><div><div>img*</div><div>string(\$uri)</div><div>title: Img</div><div>maxLength: 500</div><div>minLength: 1</div></div><div><div>publisher*</div><div>string</div><div>title: Publisher</div><div>maxLength: 255</div><div>minLength: 1</div></div><div><div>developer*</div><div>string</div><div>title: Developer</div><div>maxLength: 255</div><div>minLength: 1</div></div><div><div>rule*</div><div>string</div><div>title: Rule</div><div>minLength: 1</div></div></div></div></div>

GET

/speedrun/games/{id}/

Parameters

Name	Description
<div><div><div>id</div><div><div>* required</div></div></div><div><div>integer</div><div>(path)</div></div></div>	<div>A unique integer value identifying this game.</div> <div><div>id</div></div>

PUT

/speedrun/games/{id}/

Parameters

Name	Description
------	-------------

data * required

object

(body)

Example Value | Model

```
Game {
  name* string
    title: Name
    maxLength: 255
    minLength: 1
  img* string($uri)
    title: Img
    maxLength: 500
    minLength: 1
  publisher* string
    title: Publisher
    maxLength: 255
    minLength: 1
  developer* string
    title: Developer
    maxLength: 255
    minLength: 1
  rule* string
    title: Rule
    minLength: 1
}
```

id * required

integer

(path)

A unique integer value identifying this game.

id

PATCH

/speedrun/games/{id}/

Parameters

Name	Description
<div><div><div><div>data</div><div>*</div><div>required</div></div><div>object</div><div>(body)</div></div><div><div>Example Value</div><div>Model</div></div></div> <div><div>Game</div><div><div>name*</div><div>img*</div><div>publisher*</div><div>developer*</div><div>rule*</div></div><div><div>string</div><div>title: Name</div><div>maxLength: 255</div><div>minLength: 1</div><div>string(\$uri)</div><div>title: Img</div><div>maxLength: 500</div><div>minLength: 1</div><div>string</div><div>title: Publisher</div><div>maxLength: 255</div><div>minLength: 1</div><div>string</div><div>title: Developer</div><div>maxLength: 255</div><div>minLength: 1</div><div>string</div><div>title: Rule</div><div>minLength: 1</div></div></div>	

id

*

required

integer

(path)

 A unique integer value identifying this game. id |

DELETE

/speedrun/games/{id}/

Parameters

Name	Description
<div><div><div><div>id</div><div>*</div><div>required</div></div><div>integer</div><div>(path)</div></div><div>A unique integer value identifying this game.</div><div>id</div></div>	

Responses

Code	Description
204	

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, {AxiosError, AxiosResponse} from 'axios';
import { baseUrl } from '../constants';
import GameData from '../models/GameData';

const gamePath = baseUrl + '/speedrun/games/'

export async function getGames(resultHandler: (data: any)=>void, id:string='') {
  if (!id.length){
    Axios.get(gamePath,
      { params:{}, responseType: "json" }
    ).then
    (result => {
      let data: GameData[] = (result as AxiosResponse<GameData[]>).data;
      resultHandler(data);
    })
    .catch((error: AxiosError) => {
      alert(error.message);
    });
  }
  else{
    Axios.get(gamePath+id+'/',
      { params:{}, responseType: "json" }
    ).then
    (result => {
      let data: GameData[] = (result as AxiosResponse<GameData[]>).data;
      resultHandler(data);
    })
    .catch((error: AxiosError) => {
      alert(error.message);
    });
  }
}
```

Результат работы запроса

GAMES



Dark Souls III



CUPHEAD (2017)



CUPHEAD (2017)

Developer(s): StudioMDHR Entertainment

Publisher(s): StudioMDHR Entertainment

Leaderboard

[Rules](#)[Run](#)

#	Username	Time	Link
1	admin	0m 0s	
2	Nikola	1h 1m 1s	