Министерство науки и высшего образования РФ

Государственное образовательное учреждение высшего профессионального образования

Белгородский Государственный Технологический Университет им В. Г. Шухова

Кафедра программного обеспечения вычислительной техники и

автоматизированных систем.

Расчётно-графическое задание по дисциплине

«Технологии web-программирования»

Разработка микросервиса «Movieblog».

Выполнил:
студент группы ПВ-42
Волобуев И.К.
Проверил:
Картамышев С.В.

Белгород 2020 г.

Цель работы.

Целью работы является разработка REST API приложения по поиску и оценки кинофильмов. Пользователь может просматривать информацию о кинофильме: его название, дата релиза, бюджет, сборы, длительность, синопсис, людей, которые участвовали в съёмках, а также посмотреть трейлер. Если пользователь зарегистрирован и авторизовался в сервисе, то он может поставит фильму оценку по десятибалльной шкале. Это оценка будет влиять на общий балл картины, который формируется в зависимости от оценок остальных пользователей. Пользователь также может заходить в профили других пользователей, в которой будет информация об оценённых ими фильмах.

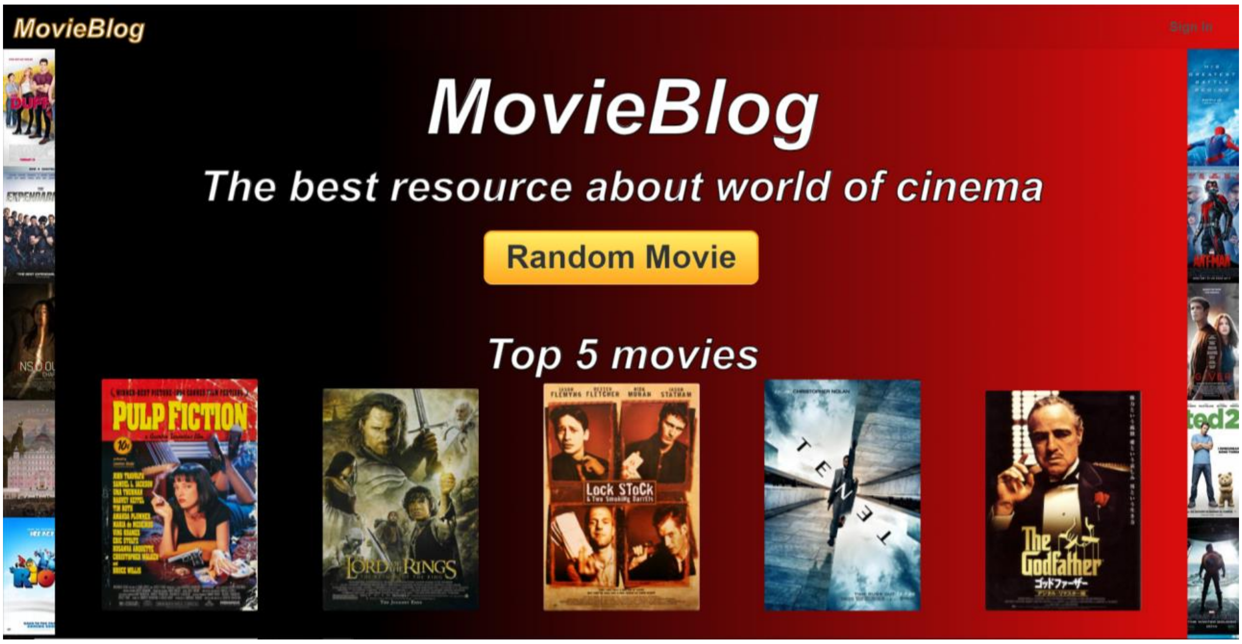
Для реализации задачи разработка была разбита две составляющих:

1. Фронтенд – приложение, использующееся на стороне клиента, с которого будут делаться определённые запросы на сервер. Вёрстка компонентов была выполнена при помощи фреймворка Vue js, а для ajax-запросов библиотека axios.
2. Бекенд – приложение, которое будет принимать запросы с клиента и обрабатывать их. Для реализации использовался объектно-ориентированный язык программирования Java вместе с фреймворком Spring. А в качестве СУБД выбор был сделан в пользу Postgresql.

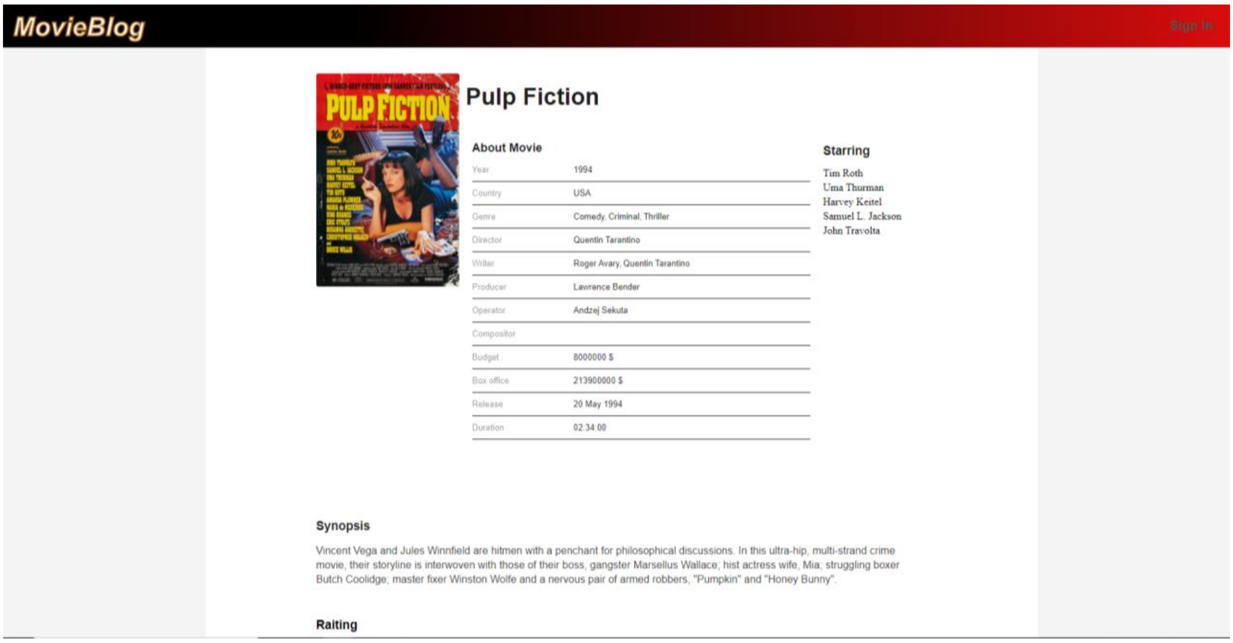
Ход работы.

Внешний вид приложения.

Главная страница.



Страница фильма.



Pulp Fiction

About Movie

Year	1994
Country	USA
Genre	Comedy, Criminal, Thriller
Director	Quentin Tarantino
Writer	Roger Avary, Quentin Tarantino
Producer	Lawrence Bender
Operator	Andzej Sekuta
Compositor	
Budget	8000000 $
Box office	213900000 $
Release	20 May 1994
Duration	02:34:00

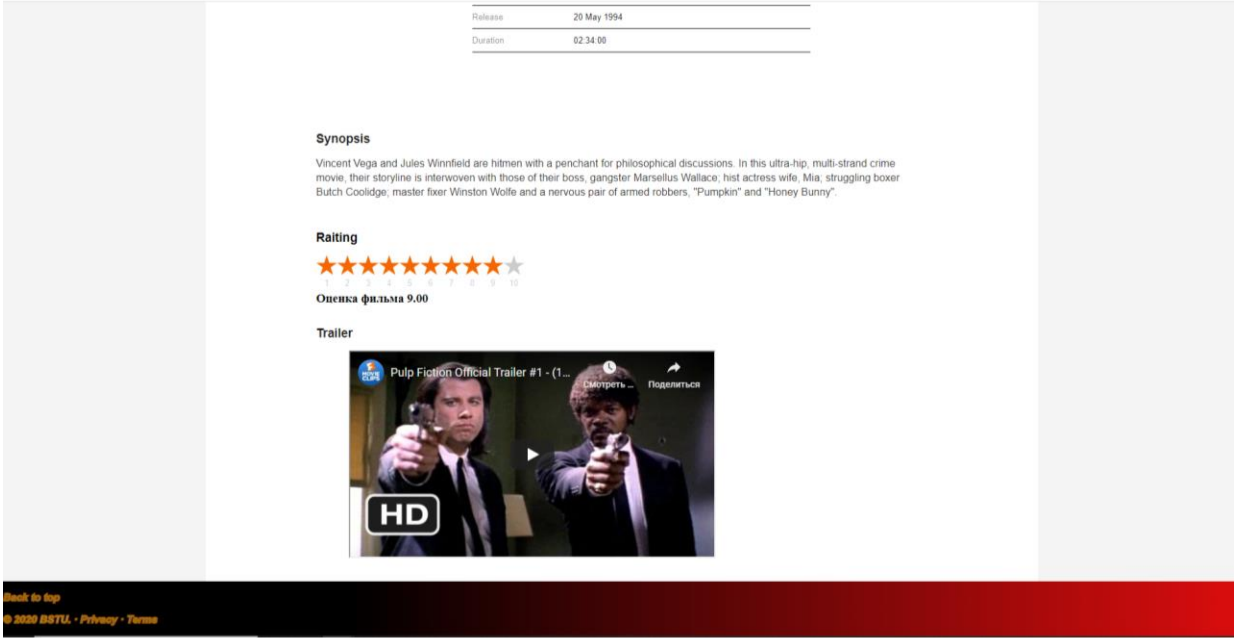
Starring

Tim Roth
Uma Thurman
Harvey Keitel
Samuel L. Jackson
John Travolta

Synopsis

Vincent Vega and Jules Winnfield are hitmen with a penchant for philosophical discussions. In this ultra-hip, multi-strand crime movie, their storyline is interwoven with those of their boss, gangster Marsellus Wallace; hist actress wife, Mia; struggling boxer Butch Coolidge; master fixer Winston Wolfe and a nervous pair of armed robbers, "Pumpkin" and "Honey Bunny".

Raiting



Release	20 May 1994
Duration	02:34:00

Synopsis

Vincent Vega and Jules Winnfield are hitmen with a penchant for philosophical discussions. In this ultra-hip, multi-strand crime movie, their storyline is interwoven with those of their boss, gangster Marsellus Wallace; hist actress wife, Mia; struggling boxer Butch Coolidge; master fixer Winston Wolfe and a nervous pair of armed robbers, "Pumpkin" and "Honey Bunny".

Raiting

★★★★★★★★★☆

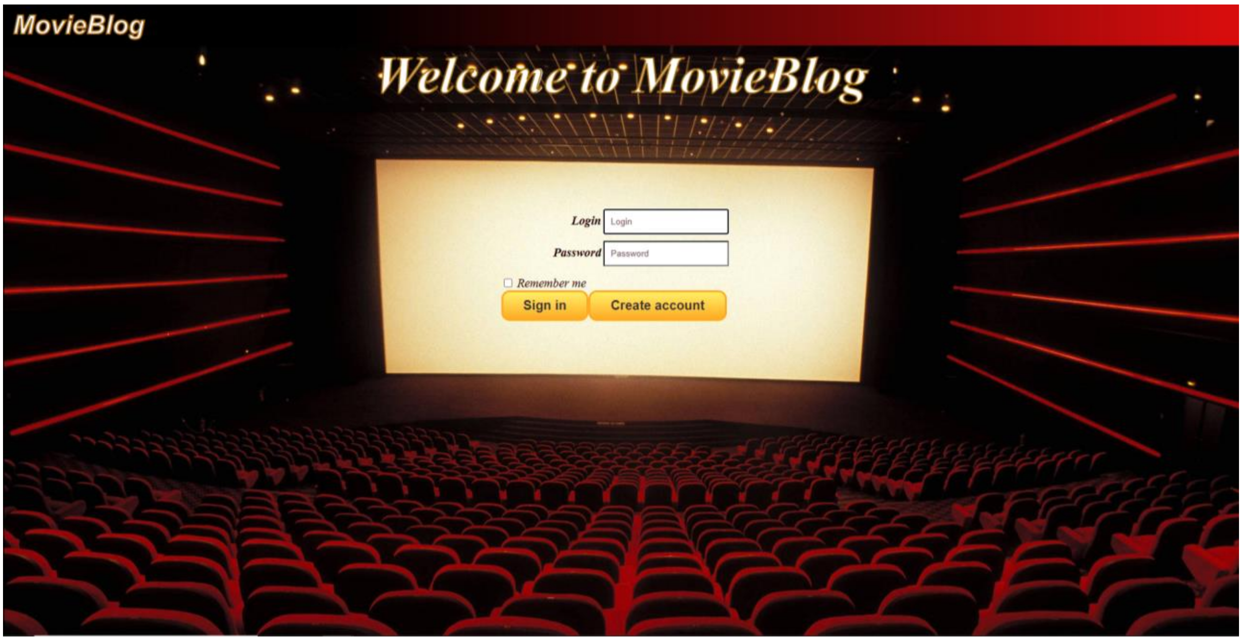
Оценка фильма 9.00

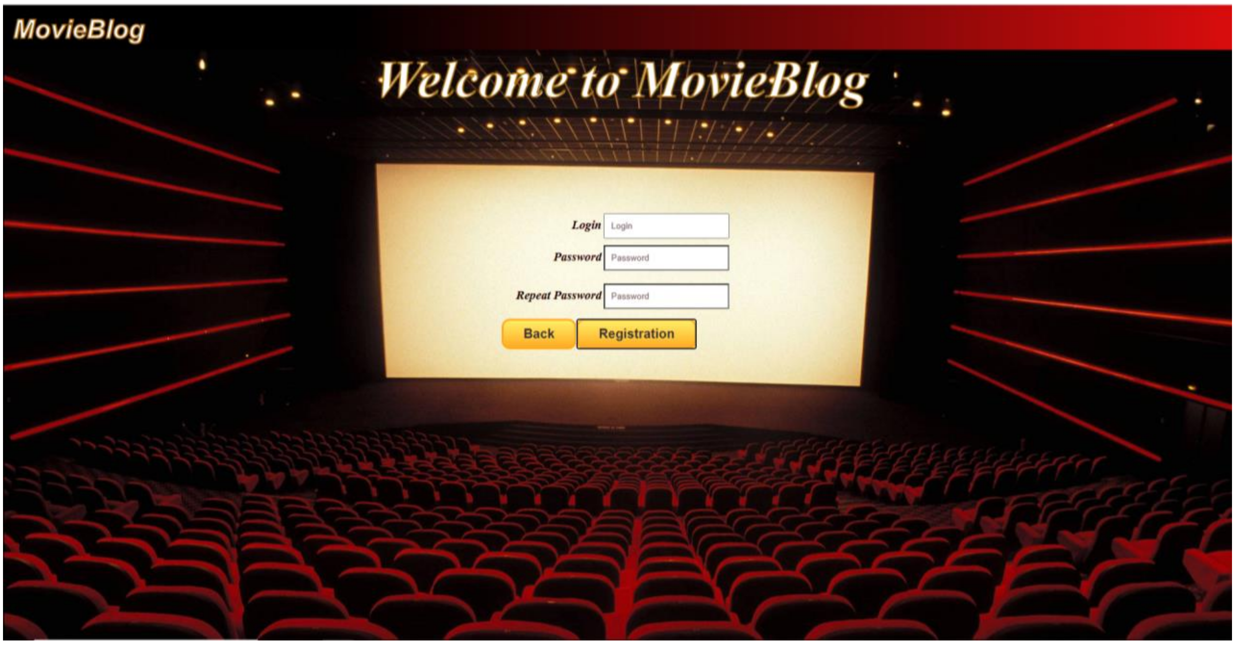
Trailer

Back to top

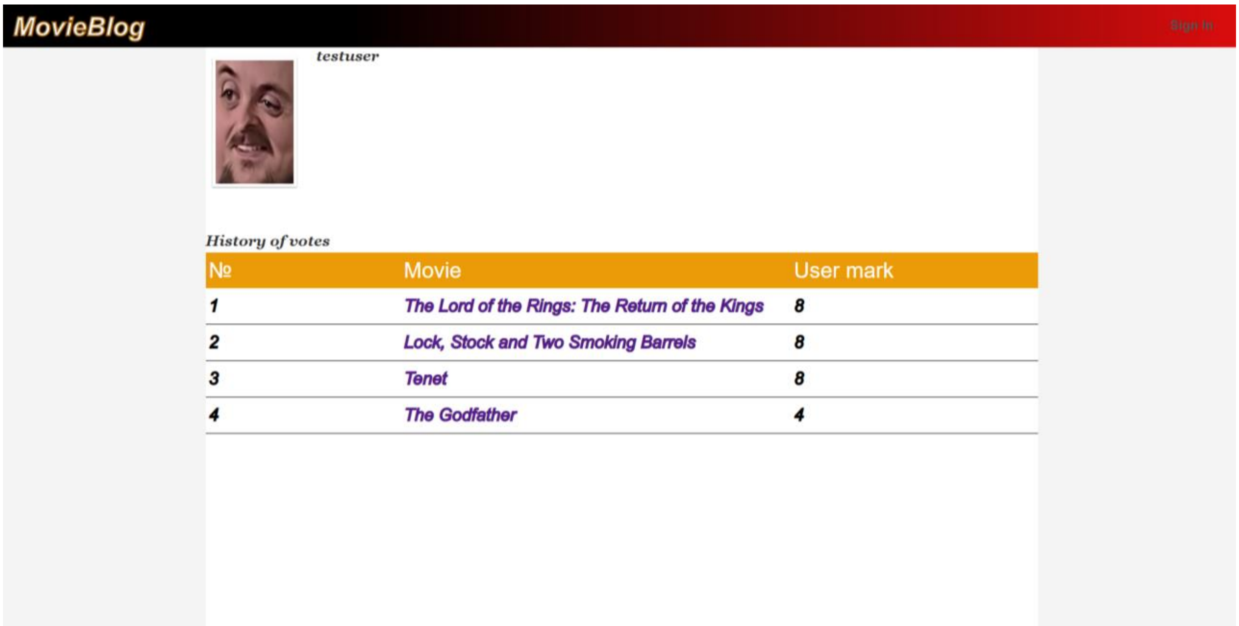
© 2020 BSTU. · Privacy · Terms

Страница регистрации и авторизации.





Страница пользователя.



testuser

History of votes

№	Movie	User mark
1	The Lord of the Rings: The Return of the Kings	8
2	Lock, Stock and Two Smoking Barrels	8
3	Tenet	8
4	The Godfather	4

Фронтенд.

Home.vue

```
<template>
  <div>
    <div class="background_container">
    <div class="content_container">
      <div class="style_welcome_head__container">
        <div class="style_welcome_head__style">
          <h1 class="style_welcome_head__title
style_welcome_head__title_margin">MovieBlog</h1>
          <h2 class="style_welcome_head__text">The best resource about world
of cinema</h2>
        </div>
      </div>
      <div class="style_random_button__container">
```

```
        <button class="myButton_Home" v-on:click="getRandomMovie">Random
Movie</button>
      </div>
      <div class="style_top_5__container">
        <h2 class="style_welcome_head__style style_welcome_head__text">Top 5
movies</h2>
        <div class="style_top_5__poster_container">
            <div class="styles_posterColumn__width" v-for="item in items">
              <router-link :to="{path: '/film/' + item.id}">
                <img class="film-poster styles_pic__size styles_pic__back
image"
                     :src="getPoster(item.id)"/>
              </router-link>
            </div>
          </div>
        </div>
      </div>
    </div>
    </div>
</template>

<script>
import api from './backend-api'

export default {
  name: 'Home',
  data() {
    return {
      items: [],
      posters: []
    }
  },
  created() {
    api.getMovieWithHigherRating()
        .then(response => {
          this.items = response.data
          for (let i = 0; i < this.items.length; i++){
            api.getMoviePoster(this.items[i].id)
                .then(response => {
                  this.posters.push({id:this.items[i].id, img: 'data:' +
response.headers['content-type'] + ';base64,' + response.data}))
            }
          })
  },
  methods: {
    getRandomMovie: function () {
      api.getRandomMovieId().then((response => {
        window.location.href  = '/film/' + response.data.toString()
      }))
    },
    getPoster: function (id){
      for (let i = 0; i < this.posters.length; i++)
        if (this.posters[i].id === id)
          return this.posters[i].img
      return ''
    }
  }
}
</script>
```

Content.vue

```
<template>
  <div class="styles_root__main_container">
    <div class="styles_contentContainer__main_props">
      <div class="styles_container__props">
        <div class="styles_container__props_margin">
          <div class="styles_posterColumn__width styles_column__el_position">
            <div class="styles_poster__position">
              <div class="styles_poster__pic_padding">
                <a>
                  <img class="film-poster styles_pic__size styles_pic__back
image"
                       :src="post_img"/>
                </a>
              </div>
            </div>
          </div>
          <div class="styles_column__el_position styles_info_container__pos">
            <div class="styles_container_info__padding">
              <div class="styles_container__props_margin">
                <div class="styles_title__pos">
                  <h1 class="styles_title__margin styles_title__font
styles_title__color" itemprop="name">{{movie.name}}</h1>
                </div>
              </div>
              <div class="styles_container__props_margin">
                <div class="styles_column__el_position styles_info__pos">
                  <h3 class="styles_title_info__pos
styles_title_info__font_size styles_title_info__font_style
styles_title__color">
                    About Movie
                  </h3>
                  <div>
                    <div class="styles_rowInformation__color
styles_rowInformation__position">
                      <div class="styles_title__props">Year</div>
                      <div class="styles_value__props">{{movie.year}}</div>
                    </div>
                    <div class="styles_rowInformation__color
styles_rowInformation__position">
                      <div class="styles_title__props">Country</div>
                      <div v-for="country in movie.country"
:key="country.id">
                        <div v-if="movie.country.indexOf(country) !==
movie.country.length-1">
                          <div
class="styles_value__props">{{country.name}},</div>
                        </div>
                        <div v-else>
                          <div
class="styles_value__props">{{country.name}}</div>
                        </div>
                      </div>
                    </div>
                    <div class="styles_rowInformation__color
styles_rowInformation__position">
                      <div class="styles_title__props">Genre</div>
                      <div v-for="genre in movie.genre"  :key="genre.id">
                        <div v-if="movie.genre.indexOf(genre) !==
movie.genre.length-1">
                          <div
class="styles_value__props">{{genre.name}},</div>
                        </div>
                        <div v-else>
                          <div
```

```
class="styles_value__props">{{genre.name}}</div>
                        </div>
                      </div>
                    </div>
                    <div class="styles_rowInformation__color
styles_rowInformation__position">
                      <div class="styles_title__props">Director</div>
                      <div v-for="director in movie.director"
:key="director.id">
                        <div v-if="movie.director.indexOf(director) !==
movie.director.length-1">
                          <div
class="styles_value__props">{{director.name}},</div>
                        </div>
                        <div v-else>
                          <div
class="styles_value__props">{{director.name}}</div>
                        </div>
                      </div>
                    </div>
                    <div class="styles_rowInformation__color
styles_rowInformation__position">
                      <div class="styles_title__props">Writer</div>
                      <div v-for="writer in movie.writer" :key="writer.id">
                        <div v-if="movie.writer.indexOf(writer) !==
movie.writer.length-1">
                          <div class="styles_value__props">{{writer.name}},
</div>
                        </div>
                        <div v-else>
                          <div
class="styles_value__props">{{writer.name}}</div>
                        </div>
                      </div>
                    </div>
                    <div class="styles_rowInformation__color
styles_rowInformation__position">
                      <div class="styles_title__props">Producer</div>
                      <div v-for="producer in movie.producer"
:key="producer.id">
                        <div v-if="movie.producer.indexOf(producer) !==
movie.producer.length-1">
                          <div class="styles_value__props">{{producer.name}},
</div>
                        </div>
                        <div v-else>
                          <div
class="styles_value__props">{{producer.name}}</div>
                        </div>
                      </div>
                    </div>
                    <div class="styles_rowInformation__color
styles_rowInformation__position">
                      <div class="styles_title__props">Operator</div>
                      <div v-for="operator in movie.operator"
:key="operator.id">
                        <div v-if="movie.operator.indexOf(operator) !==
movie.operator.length-1">
                          <div
class="styles_value__props">{{operator.name}},</div>
                        </div>
                        <div v-else>
                          <div
class="styles_value__props">{{operator.name}}</div>
```

```
                        </div>
                      </div>
                    </div>
                    <div class="styles_rowInformation__color
styles_rowInformation__position">
                      <div class="styles_title__props">Compositor</div>
                      <div v-for="compositor in movie.compositor"
:key="compositor.id">
                        <div v-if="movie.compositor.indexOf(compositor) !==
movie.compositor.length-1">
                          <div
class="styles_value__props">{{compositor.name}},</div>
                        </div>
                        <div v-else>
                          <div
class="styles_value__props">{{compositor.name}}</div>
                        </div>
                      </div>
                    </div>
                    <div class="styles_rowInformation__color
styles_rowInformation__position">
                      <div class="styles_title__props">Budget</div>
                      <div class="styles_value__props">{{movie.budget}}
$</div>
                    </div>
                    <div class="styles_rowInformation__color
styles_rowInformation__position">
                      <div class="styles_title__props">Box office</div>
                      <div class="styles_value__props">{{ movie.box_office }}
$</div>
                    </div>
                    <div class="styles_rowInformation__color
styles_rowInformation__position">
                      <div class="styles_title__props">Release</div>
                      <div
class="styles_value__props">{{movie.release}}</div>
                    </div>
                    <div class="styles_rowInformation__color
styles_rowInformation__position">
                      <div class="styles_title__props">Duration</div>
                      <div
class="styles_value__props">{{movie.duration}}</div>
                    </div>
                  </div>
                </div>
                <div class="styles_column__el_position styles_actors__pos">
                  <div class="styles_actors__pad ">
                    <h3 class="styles_actors_title__margin
styles_title_info__font_style styles_title__color">
                      Starring
                    </h3>
                    <ul class="styles_actor_list__margin">
                          <li class="styles_actor__props" v-
bind:key="star.id" v-for="star in movie.actor">{{star.name}}</li>
                    </ul>
                  </div>
                </div>
              </div>
            </div>
          </div>
        </div>


      </div>
      <div class="styles_container__back styles_container__disp">
        <div class="styles_container__props">
          <h3 class="styles_actors_title__margin
styles_title_info__font_style styles_title__color">Synopsis</h3>
          <div class="styles_synopsis__font">
            <p>{{movie.synopsis}}</p>
          </div>
        </div>
      </div>
      <div class="styles_raiting__marg styles_container__props">
        <h3 class="styles_actors_title__margin styles_title_info__font_style
styles_rootDark__QT0qE">Raiting</h3>
        <form class="styles_form__disp " id="star_list">
          <label class="styles_radio__star_container
styles_root__star_container"  v-bind:key="index" v-for="index in 10" v-
bind:data-value=index>
            <input type="radio" class="styles_input__disp" name="star" v-
bind:value="index" v-on:click="send_mark(index)">
            <span class="styles_iconContainer__pos"
@mouseover="over_star(index)" @mouseleave="out_star()">
              <span v-if="index <= movie.rating && actual_mark === 0"
class="styles_icon__marg styles_defaultIcon__star styles_baseIcon__star
styles_filledIcon__star">
                <full_star></full_star>
              </span>
              <span v-else-if="index <= actual_mark" class="styles_icon__marg
styles_defaultIcon__star styles_baseIcon__star styles_hoveredIcon__star">
                <hovered_star></hovered_star>
              </span>
              <span v-else class="styles_icon__marg styles_defaultIcon__star
styles_baseIcon__star styles_defaultIcon__star">
                <empty_star></empty_star>
              </span>
            </span>
          </label>
        </form>
        <h3 style="margin-top: 5px; margin-bottom: 5px">Оценка фильма
{{movie.rating.toFixed(2)}}</h3>
        <div v-if="user.isAuth() && movie.user_mark !== 0">Ваша оценка
{{movie.user_mark}}</div>
      </div>
      <div class="styles_raiting_container__margin styles_container__back
styles_container__disp">
        <div class="styles_container__props">
          <h3 class="styles_actors_title__margin
styles_title_info__font_style styles_title__color">Trailer</h3>
          <div class="styles_trailerContainer__margin_align">
            <iframe width="560" height="315" v-bind:src=movie.trailer
allow="accelerometer; autoplay; clipboard-write; encrypted-media; gyroscope;
picture-in-picture" allowfullscreen></iframe>
          </div>
        </div>
      </div>
    </div>
  </div>
</template>

<script>
import User from "../components/user/user";
import Empty_star from "@/components/stars/Empty_star";
import Full_star from "@/components/stars/Full_star";
import Hovered_star from "@/components/stars/Hovered_star";

import api from './backend-api'
import user from "../components/user/user";

export default {
  name: "Content",
  components: {Hovered_star, Full_star, Empty_star},
  methods: {
    send_mark(mark){
      console.log('Отправлена оценка ' + mark)
      api.putMovieRatingByUser(this.movie.id, mark,
localStorage.getItem('access-token'))
          .then(response => this.movie.rating = response.data)
      this.movie.user_mark = mark
    },
    over_star(star_index){
      this.actual_mark = star_index
    },
    out_star(){
      this.actual_mark = 0
    },
  },
  created() {
    api.getMovie(this.$route.params.id).then((response) => {
      this.movie = response.data
      let release_date = response.data.release.slice(0, 10).split('-')
      const monthNames = ["January", "February", "March", "April", "May",
"June",
        "July", "August", "September", "October", "November", "December"
      ];
      this.movie.year = release_date[0]
      this.movie.release = parseInt(release_date[2]) + " " +
monthNames[parseInt(release_date[1])-1] + " " + release_date[0]
    })
    api.getMoviePoster(this.$route.params.id).then((response) => {
      this.post_img = 'data:' + response.headers['content-type'] + ';base64,'
+ response.data
    })
    if (user.isAuth()){
      api.getUserMovieRating(this.$route.params.id,
localStorage.getItem('access-token'))
      .then(response => {

        this.movie.user_mark = response.data
        console.log(this.movie.user_mark)
      })
    }
  },
  data() {
  return{
      user: User,
      actual_mark: 0,
      empty_star: "url('/src/assets/Empty_star.png')",
      full_star: "url('/src/assets/Full_star.png')",
      hovered_star: "url('/src/assets/Hovered_star.png')",
      post_img: '',
      movie: {
        id: 0,
        name: '',
        year: '',
        release: '',
        country: [],
        genre: [],
        director: [],
        writer: [],
        producer: [],
        operator: [],
        compositor: [],
        actor: [],
        budget: 0,
        box_office: 0,
        duration: '',
        synopsis: '',
        rating: 0.0,
        trailer: ''
      }
    }
  }
}
</script>
```

AuthReg.vue

```
<template>
  <div class="styles_bg styles_page">
  <div class="myFont">
  <div>
    <h1 class="welcome">Welcome to MovieBlog</h1>
  </div>
  <form class="form-signin">
      <div v-if="current_state === 1">
        <div class="input_box">
          <div class="input">
            <label for="inputEmail">Login</label>
            <input type="text" id="inputEmail" v-model="login" class="form-
control" style="margin-left: 3px"
                   placeholder="Login" required autofocus>
          </div>
          <div class="input">
            <label for="inputPassword">Password</label>
            <input type="password" id="inputPassword" v-model="password"
class="form-control" style="margin-left: 3px"
                   placeholder="Password" required>
          </div>
        </div>
        <div class="checkbox mb-3">
          <label>
            <input type="checkbox" value="remember-me"> Remember me
          </label>
        </div>
        <button class="myButton" type="button" v-
on:click="signIn_button_clicked">Sign in</button>
        <button class="myButton" type="button" v-
on:click="create_acc_button_clicked">Create account</button>
        </div>
      <div v-else>
        <div class="input_box">
          <div class="input">
            <label for="inputEmail">Login</label>
            <input type="text" id="inputNewEmail" @change="checkUsername" v-
model="registrationUsername" class="form-control" style="margin-left: 3px"
placeholder="Login" required autofocus>
            <p v-if="isUsernameExist" style="font-size: 12pt; margin-top:
2px; margin-bottom: 2px">Пользователь уже существует</p>
          </div>
          <div class="input">
            <label for="inputPassword">Password</label>
            <input type="password" id="inputNewPassword" v-
```

```
model="registrationPassword" class="form-control" style="margin-left: 3px"
placeholder="Password" required>
          </div>
          <div class="input">
            <label for="inputPassword">Repeat Password</label>
            <input type="password" id="inputNewPasswordRepeat" v-
model="registrationPasswordRepeat" class="form-control" style="margin-left:
3px"  placeholder="Password" required>
          </div>
        </div>
        <button class="myButton" type="submit" v-
on:click="back_button_clicked">Back</button>
        <button class="myButton" type="button" v-
on:click="registration_button_clicked">Registration</button>
      </div>
  </form>
  </div>
  </div>
</template>

<script>
import api from "../views/backend-api"

export default {
  name: "AuthReg",
  methods: {
    create_acc_button_clicked(){
      this.current_state = 2
    },
    back_button_clicked(){
      this.current_state = 1
    },
    registration_button_clicked(){
      if (this.validate_registration()){
        api.sendUserRegistrationData(this.registrationUsername,
this.registrationPassword)
        .then((response) => {
          if (response.data)
            api.authorizationUser(this.registrationUsername,
this.registrationPassword)
          .then((auth_response) =>{
            localStorage.setItem('access-token',
auth_response.data['access_token'])
            localStorage.setItem('username', this.registrationUsername)
            window.location.href = '/'
          })
          .catch(() => console.log('Registration error'))
        })
      }
    },
    signIn_button_clicked(){
      if (this.validate()){
        console.log('Auth started')
        api.authorizationUser(this.login, this.password).then((response) => {
          localStorage.setItem('access-token', response.data['access_token'])
          localStorage.setItem('username', this.login)
          window.location.href = '/'
        }
        ).catch(() => console.log('No access'))
      }
    },
    validate(){
      this.errors = {}
      if (this.login.trim().length === 0)
```

```
        this.errors.email = 'Заполните Email.'
      if (this.password.trim().length === 0)
        this.errors.password = 'Заполните Пароль.'
      console.log(this.errors)
      return Object.keys(this.errors).length === 0
    },
    validate_registration(){
      this.errors = {}
      if (this.registrationUsername.trim().length === 0)
        this.errors.registrationUsername = 'Заполните имя пользователя'
      if (this.registrationPassword.trim().length === 0 ||
this.registrationPasswordRepeat.trim().length === 0)
        this.errors.registrationPassword = 'Заполните пароль'
      if (this.registrationPassword !== this.registrationPasswordRepeat)
        this.errors.registrationPasswordRepeat = 'Пароли не совпадают'
      if (this.isUsernameExist)
        this.errors.isUsernameExist = 'Имя пользователя уже существует'
      return Object.keys(this.errors).length === 0
    },
    async checkUsername(){
      console.log(this.registrationUsername)
      await
api.ajaxUsernameNotExists(this.registrationUsername).then(response => {
        this.isUsernameExist = response.data;
      })
    }
  },
  data(){
    return {
      login: '',
      password: '',
      errors: {},
      current_state: 1,
      registrationUsername: '',
      isUsernameExist: false,
      registrationPassword: '',
      registrationPasswordRepeat: ''
    }
  },
}
</script>
```

Profile.vue

```
<template>
  <div class="styles_root__main_container__profile">
    <div class="styles_profileContainer__main_props">
      <div class="styles_containerProfile_props">
        <div class="styles_container__props_margin">
          <div class="profileInfoWrapLeft">
            <div class="styles_avatar__user_box">
              <img :src="avatar" style="height: 100%; width: 100%">
            </div>
            <div v-if="isThisActualUser()" style="margin-left: 20px; border:
groove; height: 20px; width: 90px">
              <label for="file" class="btn">Update avatar</label>
              <input id="file" type="file" @change="uploadImage"
style="visibility: hidden">
            </div>
            </div>
          <div class="profileInfoWrapRight">
            <h2 class="styles_font__nickname">{{user.username}}</h2>
          </div>
        </div>
      </div>
      <div class="styles_history_votes">
          <h2 class="styles_font__nickname">History of votes</h2>
          <div class="styles_profile_film_list">
            <div class="styles_profile_top_list">
              <div class="styles_profile_list_num">№</div>
              <div class="styles_profile_list_name">Movie</div>
              <div class="styles_profile_list_mark">User mark</div>
            </div>
              <div v-for="movie in user.movies"  :key="movie.id"
class="styles_item">
                <div
class="styles_profile_list_num">{{user.movies.indexOf(movie)+1}}</div>
                <router-link :to="/film/ + movie.movie.id">
                  <div
class="styles_profile_list_name">{{movie.movie.name}}</div>
                </router-link>
                <div class="styles_profile_list_mark">{{movie.rating}}</div>
            </div>
          </div>
      </div>
    </div>

  </div>
</template>

<script>
import api from '../views/backend-api'

export default {
name: "Profile",
  created() {
    api.getUser(this.$route.params.id).then(response =>{
      console.log(response.data)
      this.user = response.data
      this.user.movies = response.data.movie
    })
    api.getUserAvatar(this.$route.params.id).then(response => {
      this.avatar = 'data:' + response.headers['content-type'] + ';base64,' +
response.data
    })
  },
  data(){
  return{
    avatar: '',
    user: {
        username: '',
        movies: [],
        selectedFile: null
    }
    }
  },
  methods: {
    isThisActualUser(){
      return this.user.username === localStorage.getItem('username')
    },
    uploadImage(event){
      console.log(event)
      this.selectedFile = event.target.files[0]
      let formData = new FormData()
      formData.append('avatar', this.selectedFile)
      console.log(this.selectedFile)
      if (this.selectedFile.name.split('.')[1] === 'jpg')
        api.postUserAvatar(this.$route.params.id, formData)
        .then(() => api.getUserAvatar(this.$route.params.id)
          .then(response => this.avatar = 'data:' +
response.headers['content-type'] + ';base64,' + response.data))
    }
  }
}
</script>
```

router.js

```
import Vue from 'vue'
import Router from 'vue-router'
import Home from '../views/Home'
import Content from "../views/Content";
import AuthReg from "../views/AuthReg";
import Profile from "../views/Profile";

Vue.use(Router)

const router = new Router({
    mode: 'history',
    routes: [
        {
            path: '/',
            name: 'Home',
            component: Home
        },
        {
            path: '/film/:id',
            name: 'Content',
            component: Content
        },
        {
            path: '/auth',
            name: 'AuthReg',
            component: AuthReg
        },
        {
            path: '/profile/:id',
            name: 'Profile',
            component: Profile
        }
    ]
})

export default router
```

App.vue

```
<template>
  <div id="app">
    <header>
      <nav class="styles_header__props styles_headerContainer__color"
style="--header-height:108px">
          <div class="styles_header__inner_props">
            <div class="styles_header_width_height">
              <div class="styles_position__logo_Header">
                <router-link class="styles_font__logo_Header" v-
bind:to="{name: 'Home'}">MovieBlog</router-link>
```

```
        <router-link v-if="!user.isAuth() &&
$router.currentRoute.path !== '/auth'" class="styles_font__signIn_Header
styles__signIn_Button" :to="{name: 'AuthReg'}">Sign In</router-link>
        <div v-else-if="user.isAuth()"
class="styles_font__signIn_Header styles__signIn_Button">
         <div class="styles_font__signIn_Header
styles__profile_logout_Button" v-on:click="user.logout()">Logout</div>
         <router-link class="styles_font__signIn_Header
styles__profile_logout_Button" :to="{path: '/profile/' +
user.id}">Profile</router-link>
        </div>
       </div>
      </div>
     </div>
   </nav>
  </header>

  <main role="main">
   <router-view/>
  </main>

  <footer class="container styles_footer__props
styles_headerContainer__color">
   <div class="styles_footer__font">
    <p class="float-right"><a href="#">Back to top</a></p>
    <p>&copy; 2020 BSTU. &middot; <a href="#">Privacy</a> &middot; <a
href="#">Terms</a></p>
   </div>
  </footer>
 </div>
</template>

<script>
import User from "./components/user/user";

export default {
 name: 'app',
 created() {
  this.user.loadCurrentUser()
 },
 data(){
  return{
   user: User
  }
 }
}
</script>
```

vue.config.js

```
module.exports = {
  devServer: {
    proxy: {
      '/api': {
        target: 'http://localhost:8082',
        ws: true,
        changeOrigin: true,
        secure: false
      }
    }
    }
    outputDir: 'target/dist',
    assetsDir: 'static'
};
```

user.js

```
import api from '../../views/backend-api'

export default {
    login: null,
    accessToken: null,
    id: 0,
    isAuth() {
        this.accessToken = localStorage.getItem('access-token')
        return this.accessToken !== null
    },
    logout() {
        localStorage.removeItem('access-token')
        localStorage.removeItem('username')
        this.id = 0
    },
    loadCurrentUser() {
        let data = localStorage.getItem('username')
        console.log(data)
        if (data !== null && data !== '') {
            api.getUserIdByUsername(data).then(response => {
                console.log(response)
                this.id = response.data
            })
        }
    }
}
```

main.js

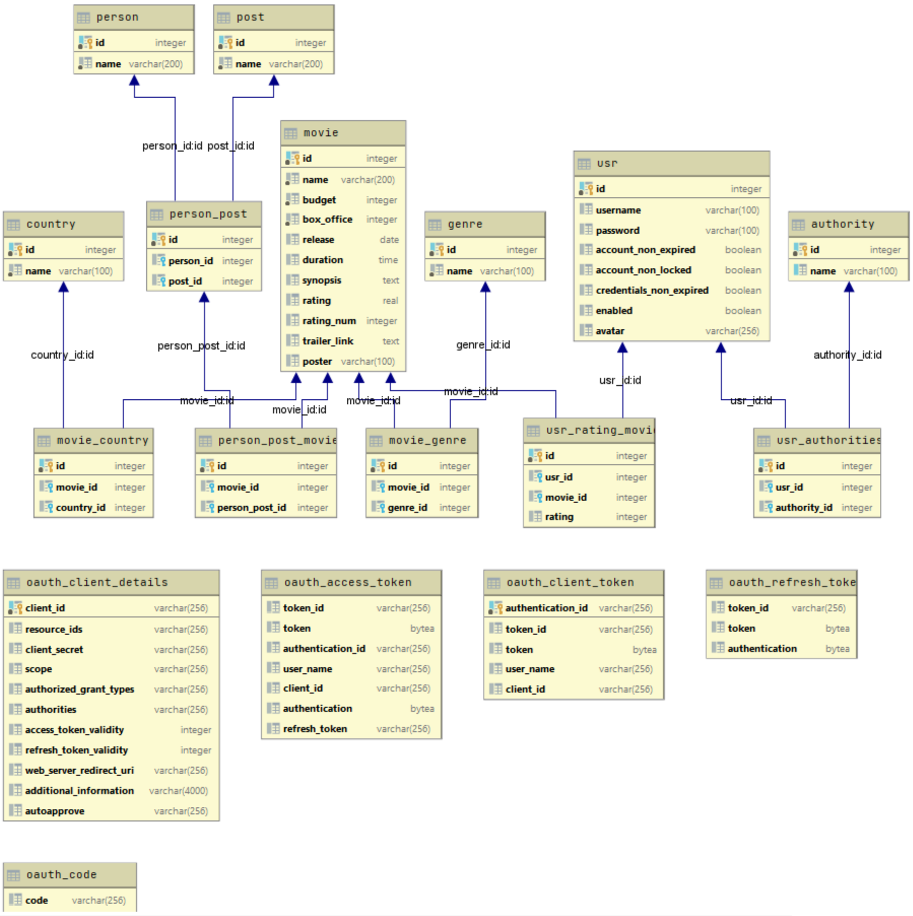
```
import Vue from 'vue'
import App from './App'
import router from "./router";

Vue.config.productionTip = false

new Vue({
    router,
    render: h => h(App)
}).$mount('#app')
```

Бекенд.

Схема базы данных.



Сущности ORM.

Movie.java

```
@Entity(name = "movie")
@Data
@FieldDefaults(level = AccessLevel.PRIVATE)
public class Movie implements Serializable {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    Long id;

    @NotNull
    @ToString.Include
    String name;

    @NotNull
    Long budget;

    @NotNull
    Long box_office;

    @NotNull
    GregorianCalendar release;
    @NotNull
    Time duration;

    @NotNull
    String synopsis;

    @NotNull
    Double rating;

    @NotNull
    Integer rating_num;

    @NotNull
    String trailer_link;

    @NotNull
    String poster;

    @ManyToMany
    @JoinTable(
            name = "movie_country",
            joinColumns = @JoinColumn(name = "movie_id", referencedColumnName
= "id"),
            inverseJoinColumns = @JoinColumn(name = "country_id",
referencedColumnName = "id")

    )
    Set<Country> countries;

    @ManyToMany
    @JoinTable(
            name = "movie_genre",
            joinColumns = @JoinColumn(name = "movie_id", referencedColumnName
= "id"),
            inverseJoinColumns = @JoinColumn(name = "genre_id",
referencedColumnName = "id")
            )
    Set<Genre> genres;

    @ManyToMany
    @JoinTable
            (
                    name = "person_post_movie",
                    joinColumns = @JoinColumn(name = "movie_id",
referencedColumnName = "id"),
                    inverseJoinColumns = @JoinColumn(name = "person_post_id",
referencedColumnName = "id")
            )
    Set<PersonPost> personPosts;

    @OneToMany(mappedBy = "movie", cascade = CascadeType.ALL)
    Set<UserRatingMovie> usersRating;

    public void updateRatingWithNewUser(Long userRating){
        rating = (rating * (rating_num - 1) + userRating) / rating_num;
    }

    public void updateRatingWithExistUser(Long userRating, Long oldRating){
        rating = (rating * rating_num - oldRating + userRating) / rating_num;
    }
}
```

Genre.java

```java
@Entity(name = "genre")
@Getter
@Setter
@FieldDefaults(level = AccessLevel.PRIVATE)
public class Genre implements Serializable {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    Long id;

    @NotNull
    String name;

    @ManyToMany
    @ToString.Exclude
    @EqualsAndHashCode.Exclude
    Set<Movie> movies;
}
```

Country.java

```java
@Entity(name = "country")
@Getter
@Setter
@FieldDefaults(level = AccessLevel.PRIVATE)
public class Country implements Serializable {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    Long id;

    @NotNull
    String name;

    @ToString.Exclude
    @EqualsAndHashCode.Exclude
    @ManyToMany
    Set<Movie> movies;
}
```

Person.java

```java
@Entity(name = "person")
@Data
@FieldDefaults(level = AccessLevel.PRIVATE)
public class Person implements Serializable {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    Long id;

    String name;

    @OneToMany(mappedBy = "person")
    Set<PersonPost> personPosts;

}
```

Post.java

```
@Entity(name = "post")
@Data
@FieldDefaults(level = AccessLevel.PRIVATE)
public class Post implements Serializable {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    Long id;

    @NotNull
    String name;

    @OneToMany(mappedBy = "post")
    Set<PersonPost> personPosts;
}
```

PersonPost.java

```
@Entity(name = "person_post")
@Getter
@Setter
@FieldDefaults(level = AccessLevel.PRIVATE)
public class PersonPost implements Serializable {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    Long id;

    @ManyToOne(fetch = FetchType.EAGER)
    @JoinColumn(name = "person_id")
    Person person;

    @ManyToOne(fetch = FetchType.EAGER)
    @JoinColumn(name = "post_id")
    Post post;

    @ManyToMany(mappedBy = "personPosts")
    Set<Movie> movies;
}
```

UserAccount.java

```
@Entity(name = "usr")
@Data
@FieldDefaults(level = AccessLevel.PRIVATE)
public class UserAccount implements UserDetails, Serializable {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    Long id;

    @NotNull
    String username;

    @NotNull
    String password;

    String avatar;
    boolean accountNonExpired;

    boolean accountNonLocked;

    boolean credentialsNonExpired;

    boolean enabled;

    @ManyToMany(fetch = FetchType.EAGER)
    @JoinTable(name = "usr_authorities",
            joinColumns = @JoinColumn(name = "usr_id", referencedColumnName =
"id"),
            inverseJoinColumns = @JoinColumn(name = "authority_id",
referencedColumnName = "id"))
    Set<Authority> authorities;

    @OneToMany(cascade = CascadeType.ALL, mappedBy = "user", fetch =
FetchType.EAGER)
    Set<UserRatingMovie> movie;

    @Override
    public Collection<Authority> getAuthorities() {
        return authorities;
    }

    @Override
    public boolean isAccountNonExpired() {
        return accountNonExpired;
    }

    @Override
    public boolean isAccountNonLocked() {
        return accountNonLocked;
    }

    @Override
    public boolean isCredentialsNonExpired() {
        return credentialsNonExpired;
    }

    @Override
    public boolean isEnabled() {
        return enabled;
    }

    public void rateMovie(Movie newMovie, Long rating){
        UserRatingMovie userRate = new UserRatingMovie();
        movie.add(userRate);
        newMovie.getUsersRating().add(userRate);
        userRate.setUser(this);
        userRate.setMovie(newMovie);
        userRate.setRating(rating);
    }
}
```

Authority.java

```
@Entity(name = "authority")
@FieldDefaults(level = AccessLevel.PRIVATE)
public class Authority implements GrantedAuthority, Serializable {
    @Id
    @Getter
    @Setter
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    Long id;

    String name;

    @Override
    public String getAuthority() {
        return name;
    }
}
```

UserRating.java

```
@Entity(name = "usr_rating_movie")
@Data
@EqualsAndHashCode(onlyExplicitlyIncluded = true)
@FieldDefaults(level = AccessLevel.PRIVATE)
public class UserRatingMovie implements Serializable {

    @Id
    @EqualsAndHashCode.Include
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    Long id;

    @ManyToOne(fetch = FetchType.EAGER, cascade = CascadeType.ALL)
    @JoinColumn(name = "movie_id")
    Movie movie;

    @ManyToOne(fetch = FetchType.EAGER, cascade = CascadeType.ALL)
    @JoinColumn(name = "usr_id")
    UserAccount user;

    @NotNull
    @Column(name = "rating")
    Long rating;
}
```

Репозитории.

MovieRepository.java

```
public interface MovieRepository extends JpaRepository<Movie, Long> {
    Movie getMovieById(Long id);

    List<Movie> getAllBy(Sort sort);

    Long countMovieBy();
}
```

AuthorityRepository.java

```
public interface AuthorityRepository extends JpaRepository<Authority, Long> {
    Authority getAuthorityByName(String authorityName);
}
```

UserRatingRepository.java

```
public interface UserRatingMovieRepository extends
JpaRepository<UserRatingMovie, Long> {
    Optional<UserRatingMovie> getUserRatingMovieByMovieIdAndUserId(Long
movieId, Long userId);
}
```

UserRepository.java

```
public interface UserRepository extends JpaRepository<UserAccount, Long> {
    Optional<UserAccount> getUserAccountByUsername(String username);

    Optional<UserAccount> getUserAccountById(Long id);

    Boolean existsByUsername(String username);
}
```

Сервисы.

MovieService.java

```
public interface MovieService {
    MovieResponse getMovie(Long id);

    ImageResponse getMoviePoster(Long id);
}
```

MovieServiceImpl.java

```
@Service
@RequiredArgsConstructor
public class MovieServiceImpl implements MovieService{

    private final MovieRepository movieRepository;
    private final MovieMapper movieMapper;
    private final ImageDownloader imageDownloader;

    public MovieResponse getMovie(Long id){
        return movieMapper.toDTO(movieRepository.getMovieById(id));
    }

    @Override
    public ImageResponse getMoviePoster(Long id) {
        return
imageDownloader.getImage(movieRepository.getMovieById(id).getPoster());
    }
}
```

MainPageService.java

```
public interface MainPageService {
    List<MainPageMoviePosterResponse> getMoviePosters();

    Long getRandomMovieId();
}
```

MainPageServiceImpl.java

```
@Service
@RequiredArgsConstructor
public class MainPageServiceImpl implements MainPageService {

    private final MovieRepository movieRepository;

    private final MainPageMapper mainPageMapper;

    public List<MainPageMoviePosterResponse> getMoviePosters() {
        return
mainPageMapper.toDTOs(movieRepository.getAllBy(Sort.by(Sort.Direction.DESC,
"rating"))
                .stream().limit(5).collect(Collectors.toList()));
    }

    public Long getRandomMovieId() {
        Long countOfMovie = movieRepository.countMovieBy();
        Random random = new Random();
        return (long) random.nextInt(countOfMovie.intValue())+1;
    }
}
```

UserAccountService.java

```
public interface UserAccountService {
    Boolean findUserByUsername(String username);

    Double rateMovie(Principal principal, Long movieId, Long rating);

    UserResponse getUser(Long id);

    boolean registrationUser(UserRegistrationRequest
userRegistrationRequest);

    AuthorizationToken authenticationUser(RequestAuth requestAuth, String
authorization);

    Long getUserIdByUsername(String username);

    Long getUserMovieRating(Principal principal, Long movieId);

    ImageResponse getUserAvatar(Long id);

    void updateUserAvatar(Long id, MultipartFile multipartFile);
}
```

UserAccountServiceImpl.java

```
@RequiredArgsConstructor
@Service
public class UserAccountServiceImpl implements UserAccountService {

    private final UserRepository userRepository;

    private final MovieRepository movieRepository;

    private final UserMapper userMapper;
    private final PasswordEncoder passwordEncoder;

    private final AuthorityRepository authorityRepository;

    private final UserRatingMovieRepository userRatingMovieRepository;

    private final ImageDownloader imageDownloader;

    private final ImageCreator imageCreator;

    private final String avatarUsersDirectoryPath = "D:\\movie-library-
images\\users-avatar\\";

    @Override
    public Boolean findUserByUsername(String username) {
        return userRepository.existsByUsername(username);
    }

    @Override
    @PreAuthorize("hasAuthority('ROLE_USER')")
    public Double rateMovie(Principal principal, Long movieId, Long rating) {
        UserAccount user =
userRepository.getUserAccountByUsername(principal.getName()).get();
        Optional<UserRatingMovie> userRatingMovie =

userRatingMovieRepository.getUserRatingMovieByMovieIdAndUserId(movieId,
user.getId());
        Movie movie = movieRepository.getMovieById(movieId);
        if (userRatingMovie.isPresent()) {
            Long oldRating = userRatingMovie.get().getRating();

            userRatingMovie.get().setRating(rating);
            userRatingMovieRepository.save(userRatingMovie.get());

            movie.updateRatingWithExistUser(rating, oldRating);
        } else {
            user.rateMovie(movie, rating);
            userRepository.save(user);

            movie.setRating_num(movie.getRating_num() + 1);
            movie.updateRatingWithNewUser(rating);
        }
        movieRepository.save(movie);
        return movie.getRating();
    }

    @Override
    public UserResponse getUser(Long id){
        return
userMapper.toDTO(userRepository.getUserAccountById(id).orElseThrow(
                () -> new UsernameNotFoundException("User not found")));
    }

    @Override
    public Long getUserIdByUsername(String username) throws
UsernameNotFoundException {
        UserAccount userAccount =
userRepository.getUserAccountByUsername(username).orElseThrow(
                () -> new UsernameNotFoundException("Username not found"));
        return userAccount.getId();
    }

    @Override
    @PreAuthorize("hasAuthority('ROLE_USER')")
    public Long getUserMovieRating(Principal principal, Long movieId) {
        UserAccount user =
userRepository.getUserAccountByUsername(principal.getName()).get();
        Optional<UserRatingMovie> userRatingMovie =

userRatingMovieRepository.getUserRatingMovieByMovieIdAndUserId(movieId,
user.getId());
        return userRatingMovie.isPresent() ?
userRatingMovie.get().getRating() : 0L;
    }

    @Override
    public ImageResponse getUserAvatar(Long id){
        return
imageDownloader.getImage(userRepository.getUserAccountById(id).get().getAvata
r());
    }

    @Override
    public void updateUserAvatar(Long id, MultipartFile multipartFile) {
        UserAccount userAccount = userRepository.getUserAccountById(id)
                .orElseThrow(() -> new UsernameNotFoundException("User not
found"));
        String avatarFilename = userAccount.getUsername() + "-avatar." +
multipartFile.getContentType()
                .split("/")[1];
        imageCreator.createImage(avatarUsersDirectoryPath + avatarFilename,
multipartFile);
        userAccount.setAvatar(avatarUsersDirectoryPath + avatarFilename);
        userRepository.save(userAccount);
    }

    @Override
    public boolean registrationUser(UserRegistrationRequest
userRegistrationRequest) {
        if
(!userRepository.existsByUsername(userRegistrationRequest.getUsername())){
            userRepository.save(createUser(userRegistrationRequest));
            return true;
        }
        return false;
    }

    private UserAccount createUser(UserRegistrationRequest
userRegistrationRequest){
        UserAccount userAccount = new UserAccount();
        userAccount.setUsername(userRegistrationRequest.getUsername());

userAccount.setPassword(passwordEncoder.encode(userRegistrationRequest.getPas
sword()));
        Authority authority =
authorityRepository.getAuthorityByName("ROLE_USER");
        Set<Authority> authorities = new HashSet<>();
        authorities.add(authority);
        userAccount.setAuthorities(authorities);
        userAccount.setEnabled(true);
        userAccount.setAccountNonLocked(true);
        userAccount.setAccountNonExpired(true);
        userAccount.setCredentialsNonExpired(true);
        userAccount.setAvatar(avatarUsersDirectoryPath + "user-unknown-
avatar.jpg");
        return userAccount;
    }
    @Override
    public AuthorizationToken authenticationUser(RequestAuth requestAuth,
String authorization) {
        String uri = "http://localhost:8082/oauth/token";
        RestTemplate restTemplate = new RestTemplate();
        restTemplate.getMessageConverters().add(new
ObjectToUrlEncodedConverter(new ObjectMapper()));
        HttpEntity<Map<String, Object>> request =
                new HttpEntity<>(createBody(requestAuth),
createHeaders(authorization));
        return restTemplate.postForObject(uri, request ,
AuthorizationToken.class);
    }

    private HttpHeaders createHeaders(String authorization){
            HttpHeaders headers = new HttpHeaders();
            headers.setBasicAuth(authorization.split(" ")[1]);
            headers.setContentType(MediaType.APPLICATION_FORM_URLENCODED);
            return headers;
    }

    private Map<String, Object> createBody(RequestAuth requestAuth){
        Map<String, Object> bodyRequest = new HashMap<>();
        bodyRequest.put("grant_type", requestAuth.getGrant_type());
        bodyRequest.put("username", requestAuth.getUsername());
        bodyRequest.put("password", requestAuth.getPassword());
        return bodyRequest;
    }
}
```

UserDetailsServiceImpl.java

```
@Service
@RequiredArgsConstructor
public class UserDetailServiceImpl implements UserDetailsService {

    private final UserRepository userRepository;

    @Override
    public UserDetails loadUserByUsername(String s) throws
UsernameNotFoundException {
        return userRepository.getUserAccountByUsername(s)
                .orElseThrow(() -> new UsernameNotFoundException("User not
found"));
    }
}
```

Контроллеры.

MainPageController.java

```
@RestController
@RequiredArgsConstructor
public class MainPageController {

    private final MainPageService mainPageService;

    @GetMapping
    List<MainPageMoviePosterResponse> getMoviesTopFive(){
        return mainPageService.getMoviePosters();
    }

    @GetMapping("/random")
    Long getRandomMovieID(){
        return mainPageService.getRandomMovieId();
    }
}
```

ContentPageController.java

```
@RequiredArgsConstructor
@RestController
@RequestMapping("/movie")
@Slf4j
public class ContentPageController {

    private final MovieService movieService;

    @GetMapping("/{id}")
    MovieResponse getMovie(@PathVariable Long id){
          return movieService.getMovie(id);
      }

    @GetMapping("/{id}/img")
    ResponseEntity<String> getMoviePoster(@PathVariable Long id){
        ImageResponse imageResponse = movieService.getMoviePoster(id);
        String encodeImage =
Base64.getEncoder().encodeToString(imageResponse.getByteArrayResource().getBy
teArray());
        return ResponseEntity
                .ok()
                .cacheControl(CacheControl.maxAge(300, TimeUnit.SECONDS))
                .contentLength(encodeImage.length())
                .contentType(imageResponse.getMediaType())
                .body(encodeImage);
    }
}
```

AuthorizationRegistrationPage.java

```
@RestController
@RequestMapping("/auth")
@RequiredArgsConstructor
@Slf4j
public class AuthorizationRegistrationPage {

    private final UserAccountService userAccountService;

    @PostMapping("/registration/ajax")
    Boolean authorizationUser(@RequestBody UsernameAjaxRequest username){
        Boolean exist =
userAccountService.findUserByUsername(username.getUsername());
        log.info("User {} is {}", username, exist);
        return exist;
    }

    @PostMapping("/registration")
    ResponseEntity<Boolean> registrationUser(@RequestBody
UserRegistrationRequest userRegistrationRequest){
        return
ResponseEntity.ok().body(userAccountService.registrationUser(userRegistration
Request));
    }

    @PostMapping( "/request")
    ResponseEntity<AuthorizationToken> authorizationRequest(RequestAuth
requestAuth,
                                                                @RequestHeader
String authorization){
        log.info("{}, {}", requestAuth, authorization);
        return
ResponseEntity.ok().body(userAccountService.authenticationUser(requestAuth,
authorization));
    }
}
```

UserPageController.java

```
@RestController
@RequestMapping("/user")
@RequiredArgsConstructor
@Slf4j
public class UserPageController {

    private final UserAccountService userAccountService;

    @GetMapping("/{id}")
    UserResponse getUser(@PathVariable Long id){
        return userAccountService.getUser(id);
    }

    @GetMapping("/{id}/avatar")
    public ResponseEntity<String> getUserAvatar(@PathVariable Long id){
        ImageResponse imageResponse = userAccountService.getUserAvatar(id);
        String encodeImage =
Base64.getEncoder().encodeToString(imageResponse.getByteArrayResource().getBy
teArray());
        return ResponseEntity.ok()
                .cacheControl(CacheControl.maxAge(300, TimeUnit.SECONDS))
                .contentLength(encodeImage.length())
                .contentType(imageResponse.getMediaType())
                .body(encodeImage);
    }

    @PostMapping("/{id}/avatar")
    @ResponseStatus(HttpStatus.NO_CONTENT)
    public void userUpdateAvatar(@PathVariable Long id, @RequestParam
MultipartFile avatar){
        userAccountService.updateUserAvatar(id, avatar);
    }

    @GetMapping("/current")
    Long getUserByUsername(@RequestParam String username){
        return userAccountService.getUserIdByUsername(username);
    }

    @GetMapping("/movie/{id}/current_user")
    Long getUserMovieRating(Principal principal, @PathVariable Long id){
        return userAccountService.getUserMovieRating(principal, id);
    }

    @PutMapping("/movie/{id}")
    Double rateMovieByUser(@PathVariable Long id, @RequestBody RequestRating
rating, Principal principal){
        log.info("{}, {}, {}", principal.getName(), id, rating);
        return userAccountService.rateMovie(principal, id,
rating.getRating());
    }
}
```

DTO.

MovieResponse.java

```
@Data
@FieldDefaults(level = AccessLevel.PRIVATE)
public class MovieResponse {
    Long id;
    String name;
    GregorianCalendar release;
    List<CountryResponse> country;
    List<GenreResponse> genre;
    List<PersonResponse> director;
    List<PersonResponse> writer;
    List<PersonResponse> producer;
    List<PersonResponse> operator;
    List<PersonResponse> compositor;
    List<PersonResponse> actor;
    Long budget;
    Long box_office;
    String duration;
    String synopsis;
    Double rating;
    String trailer;
}
```

AuthorizationToken.java

```
@Data
@FieldDefaults(level = AccessLevel.PRIVATE)
public class AuthorizationToken {
    String access_token;
    String token_type;
    Integer expires_in;
    String scope;
}
```

UserResponse.java

```
@Data
@FieldDefaults(level = AccessLevel.PRIVATE)
public class UserResponse {
    String username;
    List<UserMoviesRatingResponse> movie;
}
```

UserRegistrationRequest.java

```
@Data
@FieldDefaults(level = AccessLevel.PRIVATE)
```

```
public class UserRegistrationRequest {
    String username;
    String password;
}
```

UserMovieRatingResponse.java

```
@Data
@FieldDefaults(level = AccessLevel.PRIVATE)
public class UserMoviesRatingResponse {
    Long id;
    UserMovieResponse movie;
    Integer rating;
}
```

RequestAuth.java

```
@Data
@FieldDefaults(level = AccessLevel.PRIVATE)
public class RequestAuth {
    String grant_type;
    String username;
    String password;
}
```

Конфигурация приложения.

MvcConfig.java

```
@Configuration
public class MvcConfig implements WebMvcConfigurer {

    @Override
    public void addCorsMappings(CorsRegistry registry){
        registry.addMapping("/**")
                .allowCredentials(true)
                .allowedHeaders("Authorization", "Cache-Control", "Content-Type", "Accept", "Origin")
                .allowedMethods("HEAD", "OPTIONS", "GET", "POST", "PUT", "DELETE", "PATCH")
                .allowedOrigins("*");
    }
}
```

WebSecurityConfiguration.java

```
@Configuration
@EnableWebSecurity
@RequiredArgsConstructor
public class WebSecurityConfiguration extends WebSecurityConfigurerAdapter {

    private final UserDetailsService userDetailsService;

    private final PasswordEncoder passwordEncoder;

    @Bean
    @Override
    public AuthenticationManager authenticationManagerBean() throws Exception
{
        return super.authenticationManagerBean();
    }

    @Override
    protected void configure(AuthenticationManagerBuilder auth) throws
Exception {

auth.userDetailsService(userDetailsService).passwordEncoder(passwordEncoder);
    }
}
```

ResourceServerConfiguration.java

```
@Configuration
@EnableResourceServer
public class ResourceServerConfiguration extends
ResourceServerConfigurerAdapter {

    @Override
    public void configure(ResourceServerSecurityConfigurer resources) throws
Exception {
        resources.resourceId("cinema-search-resource-server");
    }

    @Override
    public void configure(HttpSecurity http) throws Exception {
        http.authorizeRequests()
                .antMatchers(HttpMethod.PUT,
"/movie/**").access("#oauth2.hasScope('write')")
                .anyRequest().permitAll();
    }
}
```

AuthorizationConfig.java

```
@Configuration
@EnableAuthorizationServer
@RequiredArgsConstructor
@EnableGlobalMethodSecurity(prePostEnabled = true)
public class AuthorizationServerConfiguration  extends
AuthorizationServerConfigurerAdapter {

    private final DataSource dataSource;

    private final AuthenticationManager authenticationManager;

    private final UserDetailsService userDetailsService;

    private final PasswordEncoder passwordEncoder;

    @Bean
    public TokenStore tokenStore(){
        return new JdbcTokenStore(dataSource);
    }

    @Bean
    public OAuth2AccessDeniedHandler oAuth2AccessDeniedHandler(){
        return new OAuth2AccessDeniedHandler();
    }
    @Override
    public void configure(ClientDetailsServiceConfigurer clients) throws
Exception {
        clients.jdbc(dataSource);
    }

    @Override
    public void configure(AuthorizationServerSecurityConfigurer security)
throws Exception {
        security
                .tokenKeyAccess("permitAll()")
                .checkTokenAccess("isAuthenticated()")
                .passwordEncoder(passwordEncoder);
    }

    @Override
    public void configure(AuthorizationServerEndpointsConfigurer endpoints)
throws Exception {
        endpoints
                .tokenStore(tokenStore())
                .authenticationManager(authenticationManager)
                .userDetailsService(userDetailsService);
    }
}
```

PasswordEncodingConfiguration.java

```
@Configuration
public class PasswordEncodingConfiguration {

    @Bean
    public PasswordEncoder passwordEncoder(){
        return new BCryptPasswordEncoder();
    }
}
```

Вывод.

В результате выполнения данной работы получилось полноценное веб-приложение задуманным ранее функционалом. Началось всё с HTML CSS вёрстки основных страниц сайта, затем она была перенесена в компоненты фреймворка Vue. Далее была спроектирована база данных, прописаны точки входа в сервер, и наконец проработана бизнес-логика в сервисах.