МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ "НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО"

ЛАБОРАТОРНАЯ РАБОТА №2

REST, RESTful, SOAP, GraphQL

РАБОТУ ВЫПОЛНИЛА:
Штрейх Анна Андреевна

РАБОТУ ПРОВЕРИЛ:
Добряков Давид Ильич

Санкт-Петербург
2022 г.

Цель работы:

По выбранному варианту необходимо будет реализовать RESTful API средствами express + typescript (используя ранее написанный boilerplate).

Выполнение:

Для реализации выбран вариант сервиса онлайн-кинотеатра. К модели пользователя добавилась модель фильма.

```
interface FilmAttributes {
    id: number;
    title: string;
    country: string;
    duration: number;
    genre: string;
    ageLimit: number;
    description: string;
};

interface FilmCreationAttributes
    extends Optional<FilmAttributes, 'id'> { }

interface FilmInstance
    extends Model<FilmAttributes, FilmCreationAttributes>,
    FilmAttributes {
    createdAt?: Date;
    updatedAt?: Date;
}
```

Рисунок 1 – Интерфейсы для модели фильма

Для фильмов реализованы сервисы, контроллеры и пути.

```
services > TS filmService.ts > FilmService > getByTitle
import Film from '../db/models/film';

class FilmService {
    async getAll(){
        const films = await Film.findAll()

        if (films) return films
        else return { "message": "films not found" }

    }
    async getById(id: number) {
        const film = await Film.findByPk(id)

        if (film) return film

        throw new Error(`film with id ${id} not found`)
    }

    async getByTitle(title: string) {
        const film = await Film.findOne({ where: { title: title } })
```

Рисунок 2 – Сервисы фильмов

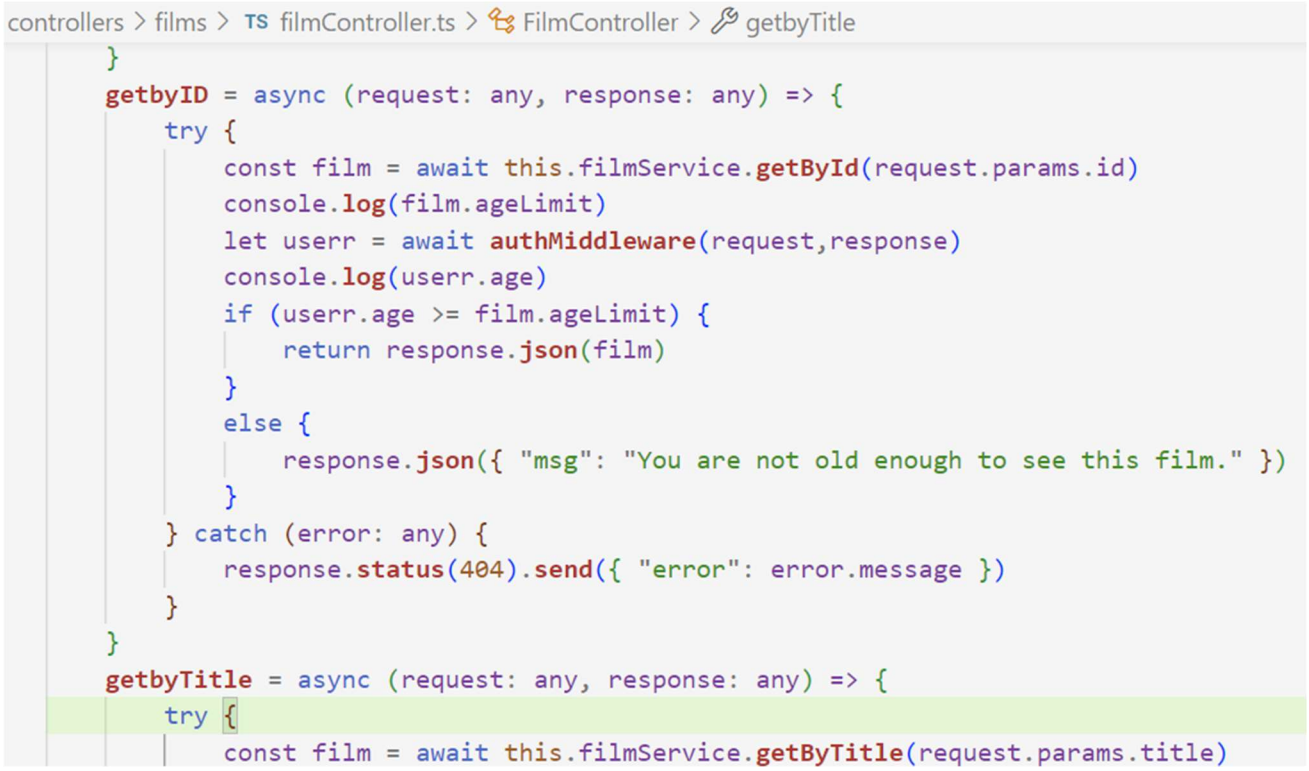

Рисунок 3 – Контроллеры фильмов

```
import express from "express"
import FilmController from "../controllers/films/filmController"


const router = express.Router()

const filmController = new FilmController()

router
  .route('/films')
  .get(filmController.get)

router
  .route('/films/id/:id')
  .get(filmController.getbyID)

router
  .route('/films/genre/:genre')
  .get(filmController.getbyGenre)

router
  .route('/films/country/:country')
  .get(filmController.getbyCountry)
```

Рисунок 4 – Пути фильмов

Для пользователей реализована регистрация. При выводе пользователя пароль будет захэширован.

```
registration = async (request: any, response: any) => {
    try {
        const userData = request.body;

        const hashedPassword = await bcrypt.hash(userData.password, 10);
        const user = await this.userService.create({
            ...userData,
            password: hashedPassword,
        });
        user.password = 'password';
        console.log(user)

        const tokenData = this.userService.createToken(user);
        console.log(tokenData)
        response.setHeader('Set-Cookie', [this.createCookie(tokenData)]);
        response.send(user);
    }
    catch (error: any) {
        response.status(404).send({ "error": error.message })
    }
}
```

Рисунок 5 – Регистрация

```
createCookie(tokenData: any) {
    return `Authorization=${tokenData.token}; HttpOnly; Max-Age=${tokenData.expiresIn};path=/v1`;
}
```

Рисунок 6 – Создание куки

Также реализованы вход и выход из аккаунта и просмотр самого профиля.

```
loggingIn = async (request: any, response: any) => {
    const logInData = request.body;
    try {
        const user = await this.userService.getByEmail(logInData.email)
            const isPasswordMatching = await bcrypt.compare(logInData.password, user.password);
            if (isPasswordMatching) {
                user.password = 'password';
                const tokenData = this.userService.createToken(user);
                response.setHeader('Set-Cookie', [this.createCookie(tokenData)]);
                response.send(user);
            } else {
                response.send({ "error": "Wrong password." })
            }
    } catch (error: any) {
        response.status(404).send({ "error": error.message })
    }
}

loggingOut = (request: any, response: any) => {
    response.setHeader('Set-Cookie', ['Authorization=;Max-age=0']);
    response.status(200).send({ "massage": "You are logged out." });
  }
```

Рисунок 7 – Вход и выход

```
createToken(user: any) {
    const expiresIn = 60 * 60;
    const secret = 'giveme678G563/';
    const dataStoredInToken: DataStoredInToken = {
      _id: user.id,
    };
    return {
      expiresIn,
      token: jwt.sign(dataStoredInToken, secret, { expiresIn }),
    };
  }
```

Рисунок 8 – Сервис создания токена

Для определения, какой пользователь сейчас авторизован, реализован authMiddleware.

```
async function authMiddleware(request: any, response: any) {
  const cookies = request.cookies;
  if (cookies && cookies.Authorization) {
    const secret = 'giveme678G563/';
    try {
      const verificationResponse = jwt.verify(cookies.Authorization, secret) as DataStoredInToken;
      const id = verificationResponse._id;
      const user = await userService.getById(id)
      request.user = user;
      return user;

    } catch (error: any) {
      throw new Error(error)
    }
  } else {
    throw new Error(`You are not authorized.`)
  }
}
```

Рисунок 9 – Проверка авторизации

```
showProfile = async (request: any, response: any) => {
    try {
        const user = await authMiddleware(request,response)
        return response.json(user);
    } catch (e: any) {
        response.status(404).send({ "error": e.message })
    }
}
```

Рисунок 9 – Функция возвращения данных о текущем пользователе

Вывод:

API успешно реализовано.