САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Лабораторная работа №3

Выполнил:
Кривцов Павел
Группа K33402

Проверил:
Добряков Д. И.

Санкт-Петербург
2023 г.

Задача

Необходимо реализовать отдельный микросервис, выполняющий какую-либо содержательную функцию из всего арсенала функций приложения.

Ход работы

Разделим приложение на три микросервиса:

- auth – операции по авторизации
- movies – все, что связано с описанием фильма и watchlist’a
- gateway – соединяющая сущность

Gateway содержит в себе один index.ts и имеет вид:

```
import axios from "axios";

const app = express();
const port = 8000
const host = 'localhost'

const microservices = {
    users: 8001,
    kinopoisk: 8002
}

app.use(express.urlencoded( options: {extended: true}))
app.use(express.json())
app.use(cors())

for (const m in microservices) {
    app.all( path: `/${m}/*`,  handlers: async (req: any, res: any) => {
        const url = `http://${host}:${microservices[m]}${req.originalUrl}`;
        try {
            const response = await axios({
                method: req.method,
                url: url,
                data: req.body,
                headers: {
                    auth: req.headers.auth
                }
            });
            res.status(response.status).send(response.data);
        } catch (e) {
            if (e.response) {
                res.status(e.response.status).send(e.response.data);
            } else {
                res.status(500).send('Internal Server Error');
            }
        }
    });
}

app.listen(port,  callback: () => {
    console.log(`Running gateway on  http://${host}:${port}`)
})

```

Из изменений к предыдущей работе, watchlist теперь содержит в поле username строку, а не сущность User

```
@Unique( name: 'movie_user_constraint', fields: ["movie", "username"])
@Entity()
export class Watchlist {
    @PrimaryGeneratedColumn()
    id: number

    @Column( options: {
        nullable: true,
    })
    rate: number;

    @ManyToOne( typeFunctionOrTarget: () => Movie, inverseSide: (movie : Movie ) => movie.watchlist)
    movie: Movie;

    @Column()
    username: string;
}
```

Поэтому, чтобы получить username по токену в контроллере watchlist, напишем утилиту getUsername(request):

```
import 'dotenv/config'

export const getUsername = async (request: any) => {
    console.log(`${process.env.AUTH}/me`)
    const response = await fetch( input: `${process.env.AUTH}/me`, init: {
        headers: {
            auth: request.headers.auth
        }
    })
    if (response.status == 200){
        const responseJSON = await response.json()
        return responseJSON['username']
    }
    throw new Error("Unauthorized")
}
```

Вот так будет выглядеть ее вызов в контроллере:

```
27    getAllByUsername = async (request: Request, response: Response) => {
28        try {
29            const username = await getUsername(request)
30            const watchlists = await watchlistService.getAllByUsername(username)
31            return response.send(watchlists)
32        } catch (error: any) {
33            return response.status( code: 404).send( body: {error: error.message})
34        }
35
36    }
37
```

В остальном, сервисы и контроллеры микросервисов остаются неизменными, так как составляющие микросервисов были связаны между собой только по watchlist’y.

Вывод

В ходе работы реализованное ранее монолитное приложение было разделено на микросервисы и была сохранена связь между сущностями.