САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бек-энд разработка

Отчет

Лабораторная работа №3

Выполнила:
Самчук Анита
К34402

Проверил:
Добряков Д. И.

Санкт-Петербург

2023 г.

Задача

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

Ход работы

Было решено отделить пользователя, там самым у нас образовалось три микросервиса:

- User
- Post, которые также в себе содержат комментарии
- Gateway – соединяющая сущность

Для обеспечения связи между двумя микросервисами, был реализован шлюз, который принимает запросы от клиентов и маршрутизирует их к соответствующим микросервисам. Это позволило клиентам взаимодействовать с обоими микросервисами через одну точку входа.

```
const app : Express  = express();
const port : 5555  = 5555
const host : "localhost"  = 'localhost'

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

app.all( path: `/api/v2/users/*`, handlers: async (request: any, response: any) : Promise<void>  => {
    try {
        await fetch( input: `http://localhost:5556/api/v2/`)
        console.log(`http://localhost:5556${request.originalUrl}`)
        response.redirect(307, `http://localhost:5556${request.originalUrl}`)
    } catch (e) {
        response.status(500).send('Internal Server Error');
    }
})

app.all( path: `/api/v2/posts/*`, handlers: async (request: any, response: any) : Promise<void>  => {
    try {
        await fetch( input: `http://localhost:5557/api/v2/healthcheck`)
        console.log(`http://localhost:5557${request.originalUrl}`)
        response.redirect(307, `http://localhost:5557${request.originalUrl}`)
    } catch (e) {
        response.status(500).send('Internal Server Error');
    }
})


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

Для микросервиса, отвечающего за пользователей (User), мы переработали и оптимизировали код, удалив все, что не относится к этой функциональности. То же самое мы сделали для микросервиса, управляющего постами и комментариями (Posts, Comments).

То есть между ними пропала связь один-ко-многим, и поле userId у постов и комментарием стало просто полем

Поэтому, что получит userId, была релизована утилита getUser:

```
export const getUser = async (token?: string) : Promise<any>  => {
    if (token){
        const user : Response  = await fetch(
            input: `${process.env.USERAPI}/account`,
            init: {
                headers: {authorization: token}
            })
        if (user.status == 200) {
            const userjson = await user.json()
            return userjson.id
        } else {
            throw new Error("Unauthorized")
        }
    }
    throw new Error("Unauthorized")
}
```

А так теперь будет выглядеть ее вызов в контроллере, на примере обновления поста:

```
    updatePost = async (request: Request, response: Response) : Promise<void>  => {
        const {body} = request
        const {id : string } = request.params

        try {
            const userId = await getUser(request.headers.authorization)
            const post: Post | PostError = await this.postService.updatePost(+id, body, userId)

            response.status( code: 201).json(post)
        } catch (error: any) {
            response.status( code: 404).json( body: {error: error.message})
        }
    }
```

Вывод

В результате выполнения данной лабораторной работы мы успешно разделили функциональность нашего приложения на два отдельных микросервиса: пользователи и посты с комментариями. Использование шлюза позволило нам обеспечить взаимодействие между микросервисами через единую точку входа. Это улучшило масштабируемость и управляемость приложением, а также подготовило его к дальнейшей разработке и расширению функциональности.