Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное
учреждение высшего образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»
Факультет инфокоммуникационных технологий

ОТЧЕТ

по Лабораторной работе № 3

по теме: «Микросервисы»
по дисциплине: Бек-энд разработка

Специальность:
09.03.03 Мобильные и сетевые технологии

Проверил:
Добряков Д. И. _____
Дата: «__» _____ 202__г.
Оценка _____

Выполнил:
студент группы K33401
Чернов Е. К.

Санкт-Петербург
2023

ЦЕЛЬ РАБОТЫ

Получить практические навыки по созданию микросервисной архитектуры.

ВЫПОЛНЕНИЕ

1 Создание архитектуры

За основу микросервисной архитектуры возьмем следующие (рисунок 1):

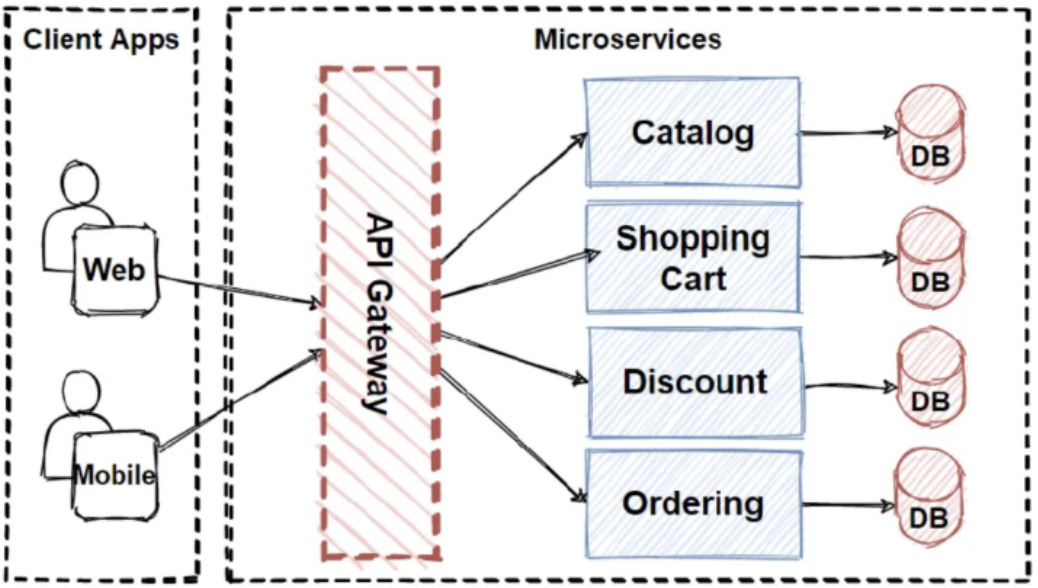

Рисунок 1 - Архитектура

Здесь принцип работы заключается в том, что клиент обращается по адресу сервиса *Gateway*, который перенаправляет запрос в дальнейшие микросервисы. В нашем случае мы выделяем два сервиса, это *Auth-service* и *Coins-service*.

2 Сервисы

Gateway-service отвечает за распределение адресации к интересующим сервисам.

Auth-service отвечает за авторизацию и регистрацию пользователя. Доступ к нему осуществляется следующим образом (рисунок 2):

AuthRouter.ts

```ts
import express, {Request, Response} from "express"
import axios, {isAxiosError} from "axios";
import {AUTH_SERVICE_URL} from "../index";


const authRouter: express.Router = express.Router()

authRouter.route( prefix: '/*')
    .all( handlers: async (req: Request, res: Response) => {
        try {
            const authorization = req.headers.authorization
            const response = await axios({
                method: req.method,
                url: AUTH_SERVICE_URL + req.url,
                headers: authorization ? {"Authorization": authorization} : undefined,
                data: req.body ? req.body : undefined
            })
            return res.json(response.data)
        } catch (e) {
            if (isAxiosError(e) && e.response) {
                return res.status(e.response.status).json(e.response.data)
            } else {
                console.log(e)
                return res.status( code: 500).json( body: {"error": "Gateway server internal error"})
            }
        }
    })

export default authRouter;
```

Рисунок 2 - AuthRouter

Coins-service отвечает за взаимодействие клиента с интерфейсом монет и портфелей монет пользователя (рисунок 3):

CoinRouter.ts

```ts
import axios, {isAxiosError} from "axios"
import express, {Request, Response} from "express"
import {COINS_SERVICE_URL} from "../index";


const coinRouter: express.Router = express.Router()

coinRouter.route( prefix: '/*')
    .all( handlers: async (req: Request, res: Response) => {
        try {
            const authorization = req.headers.authorization
            const response = await axios({
                method: req.method,
                url: COINS_SERVICE_URL + req.url,
                headers: authorization ? {"Authorization": authorization} : undefined,
                data: req.body ? {...req.body} : undefined
            })
            return res.json(response.data)
        } catch (e) {
            if (isAxiosError(e) && e.response) {
                return res.status(e.response.status).json(e.response.data)
            } else {
                console.log(e)
                return res.status( code: 500).json( body: {"error": "Gateway server internal error"})
            }
        }

    })

export default coinRouter
```

Рисунок 3 - CoinRouter

3 Разделение на сервисы

Когда мы разделяли на сервисы, то функционал регистрации и работы с пользователем вынесли в отдельную папку и функционал работы с монетами и портфелями в отдельную папку. Сделали таким образом, чтобы их можно было запускать отдельно (рисунок 4):

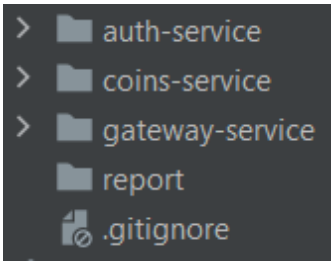

Рисунок 4 - Структура проекта

ВЫВОД

В ходе работы получили практические навыки по созданию микросервисной архитектуры.