САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет
по лабораторной работе №3
«Микросервисы »

Выполнила:

Киреева М.С.

Группа
К3333

Проверил:
Добряков Д. И.

Санкт-Петербург

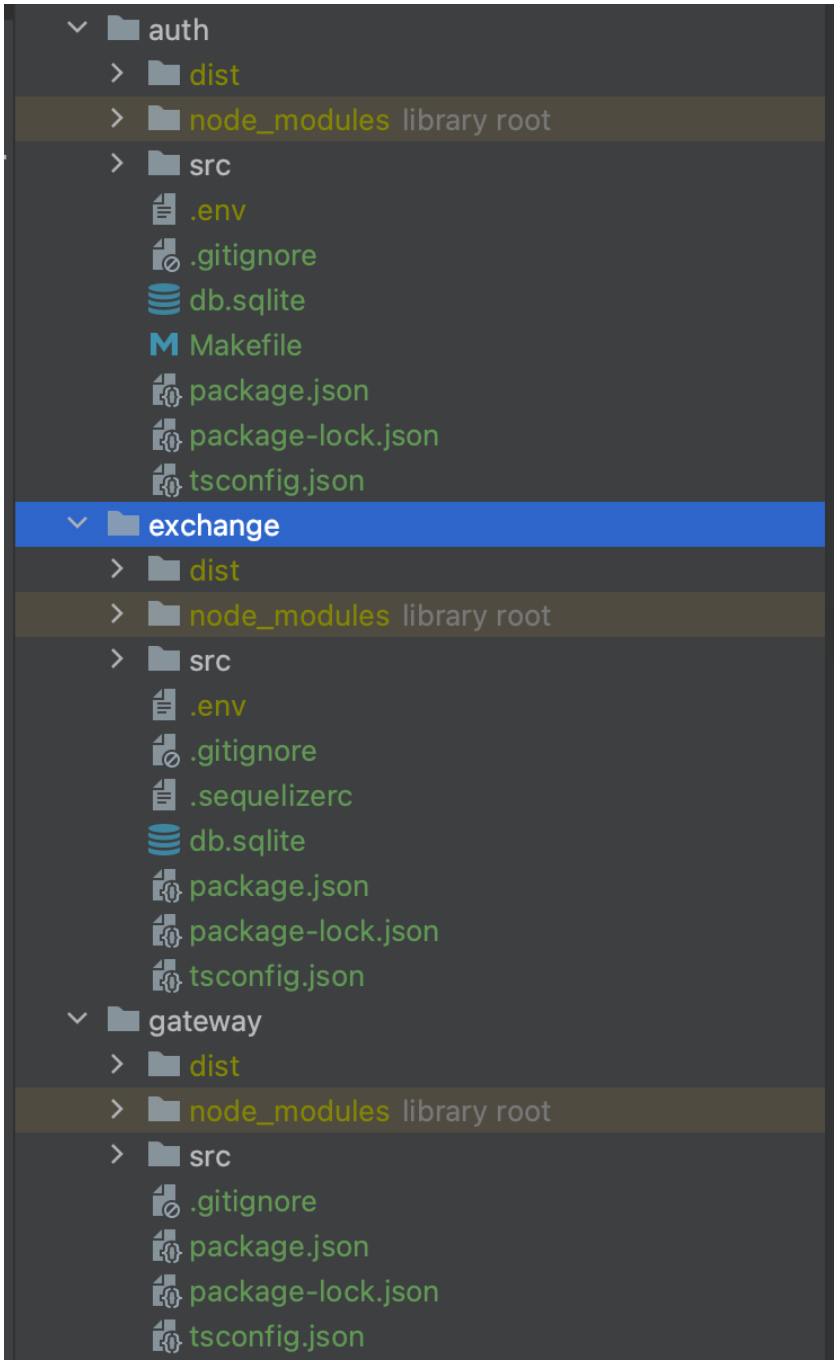
2023 г.

Задача

Реализовать отдельный микросервис, выполняющий функцию авторизации на криптобирже.

Ход работы

Структура проекта



Реализация proxy

```
import express from "express";
import { createProxyMiddleware } from "http-proxy-middleware";

const app = express();

app.use("/users", createProxyMiddleware({ target: "http://localhost:8001", changeOrigin: true }));

app.use("/exchange", createProxyMiddleware({ target: "http://localhost:8002", changeOrigin: true }));

new *
app.listen( port: 8000, callback: () => {
    console.log("Gateway server is running on port 8000");
});
```

Реализация роутов на exchange

```
const router: express.Router = express.Router()

const controller: CurrencyController = new CurrencyController()

//Добавление валюты на биржу
router.route( prefix: '/add_currency')
    .post(passport.authenticate( strategy: 'bearer', options: { session: false }), controller.post)

//Удаление валюты
router.route( prefix: '/delete/:id')
    .delete(passport.authenticate( strategy: 'bearer', options: { session: false }), controller.deleteById)

//Все валюты на бирже
router.route( prefix: '/all')
    .get(passport.authenticate( strategy: 'bearer', options: { session: false }), controller.getAll)

//Фильрация по дате
router.route( prefix: '/date_filter')
    .get(passport.authenticate( strategy: 'bearer', options: { session: false }), controller.byDate)

//Информация о валюте
router.route( prefix: '/name_filter')
    .get(passport.authenticate( strategy: 'bearer', options: { session: false }), controller.byName)

2 usages
```

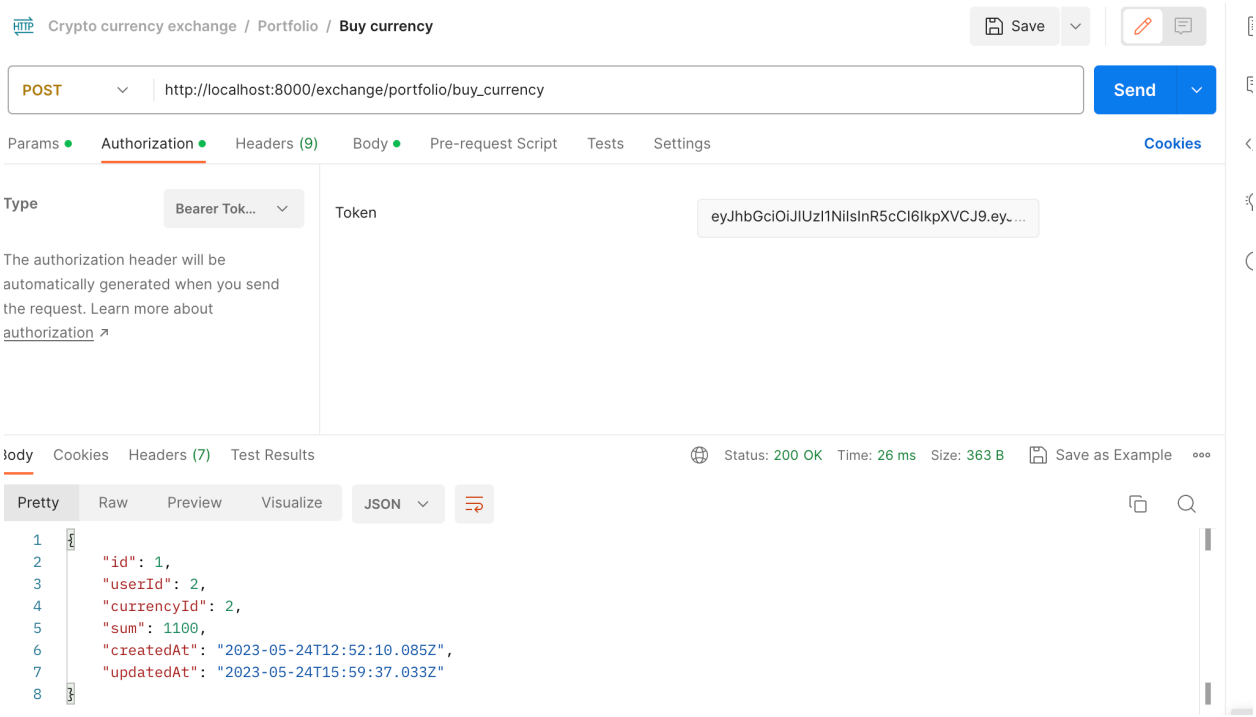
Middleware для проверки токена

```
import passport from 'passport'
import {Strategy} from 'passport-http-bearer'
import axios from 'axios'



const customJwtStrategy = new Strategy( verify: async function (token: any, done: any) {
    axios.post(
        url: `http://localhost:8001/users/validateToken`,
        data: {'accessToken': token}
    ).then((resp : AxiosResponse<any> ) => {
        if (resp.status == 200 && resp.data.valid) {
            const user = resp.data.user
            done(null, user)
        } else {
            done(null, false)
        }
    }).catch((error) => {
        done(error)
    })
})

passport.use(customJwtStrategy)

5+ usages  new *
export default passport
```

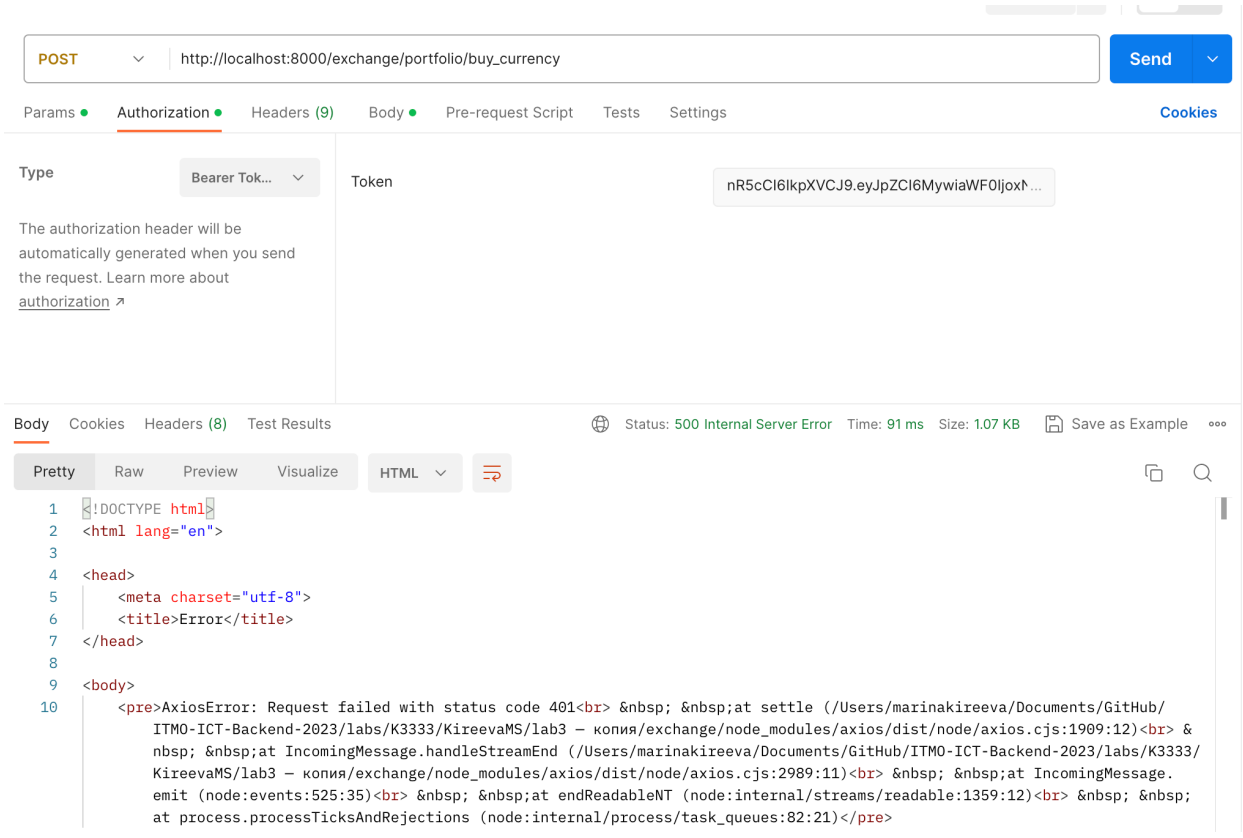
Контроллер для проверки токена на стороне микросервиса users

```
    3 usages  new *
    validateToken = async (request: any, response: any) => {
        const {body} = request
        const {accessToken} = body
        try {
            const payload = jwt.verify(accessToken, jwtOptions.secretOrKey)
            // @ts-ignore
            const user = await this.userService.getById(payload.id)
            response.send({'valid': true, 'user': user})
        } catch (e: any) {
            response.status(401).send({'valid': false})
        }
    }
```

Проверка работы с токеном



Работа с невалидным токеном



Вывод

В ходе данной работы было реализовано разделение исходного функционала приложения на микросервисы с использованием схемы с gateway.