

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

Специальность:

09.03.03 Мобильные и сетевые технологии

Проверил:

Добряков Д. И. _____

Дата: «__» _____ 202__ г.

Оценка _____

Выполнил:

студенты группы К33401

Ковалев В. М.

Санкт-Петербург

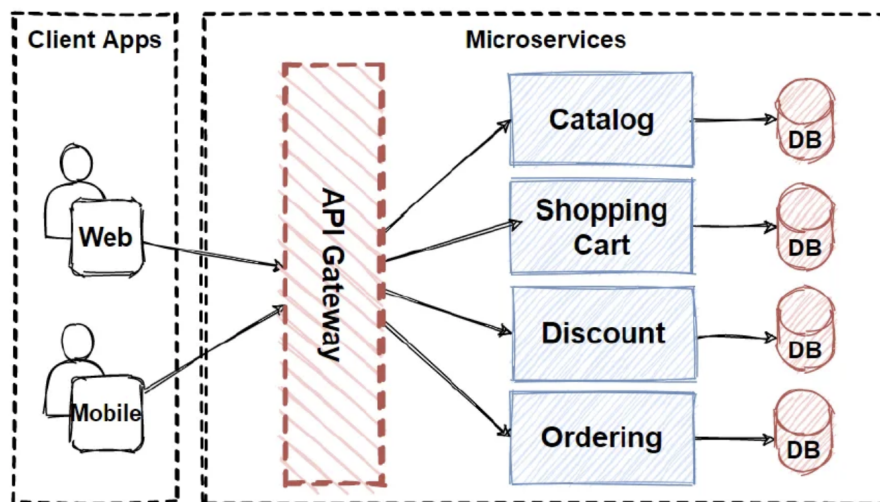
2023

ЦЕЛЬ РАБОТЫ



Разбить монолитное приложение на микросервисы

ВЫПОЛНЕНИЕ

В качестве паттерна для микросервисной архитектуры своего приложения я использовал API Gateway.



Структура проекта:

- ▼ **lw_3** ~/Code/GitHub/ITMO-ICT-Backend-2023/labs/K3
 - > **auth-service**
 - > **games-service**
 - > **gateway-service**
 -  **docker-compose.yml**
 -  **External Libraries**

Роутинг на gateway сервере:

```
import express from "express"
import GamesRouter from "./games";
import AuthRouter from "./auth"

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

router.use('/games', GamesRouter)
router.use('/auth', AuthRouter)

export default router
```

Переадресация запросов на auth-сервис:

```
router.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 router;
```

Переадресация запросов на games-сервис:

```
router.route( prefix: '/' )
  .all(authMiddleware, async (req: Request, res: Response) => {
    try {
      const response = await axios({
        method: req.method,
        url: GAMES_SERVICE_URL + req.url,
        headers: { "user-id": req.user?.id },
        data: req.body ? { ...req.body, user: req.user } : 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 router;
```

Middleware для авторизации:

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

export const authMiddleware = async (req: Request, res: Response, next: NextFunction) => {
  try {
    const authorization = req.headers.authorization
    try {
      const response1 = await axios.get( url: AUTH_SERVICE_URL + "/users/me", config: { headers: { Authorization: auth
      const user = response1.data
      req.user = user
      next()
    } catch (e) {
      return res.status( code: 403 ).json( body: { "error": "Access Denied" } )
    }
  } catch (e) {
    console.log(e)
    return res.status( code: 500 ).json( body: { "error": "Gateway server internal error" } )
  }
}
```

Аналогичная логика будет, если к примеру нужно будет добавить сервис, который доступен только авторизованным пользователям. Сначала запрос обрабатывает middleware и кладет в request данные пользователя. Далее в роуте можно взять ID пользователя и добавить в headers, которые

уйдут в микросервис. Далее он уже может с ним работать. Адреса сервисов получаются из переменных окружения:

```
export const AUTH_SERVICE_URL = process.env.AUTH_SERVICE ? `http://${process.env.AUTH_SERVICE}` : ""
export const GAMES_SERVICE_URL = process.env.AUTH_SERVICE ? `http://${process.env.GAMES_SERVICE}` : ""
```

Пример запроса к auth сервису:

The screenshot shows a REST client interface with the following components:

- Request Method:** GET
- Request URL:** {{url}}/auth/users/me
- Request Headers:** Authorization: d636a48c-45c8-
- Response Body:** {"id": 3, "email": "admin@gmail.com", "name": null}

Key	Value
Authorization	d636a48c-45c8-

```
{
  "id": 3,
  "email": "admin@gmail.com",
  "name": null
}
```

Пример запросы к games сервису с токеном:

GET ▼ `{{url}}/games?offset=20&count=20&publisher=Ubisoft&sortByPrice=desc&developer=Ubisoft`

Params ● Authorization **Headers (7)** Body Pre-request Script Tests Settings

Headers 👁 6 hidden

	Key	Value
<input checked="" type="checkbox"/>	Authorization	d636a48c-45c8-400c-8a0b-eea5dba3902a

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON ▼ ⇌

```

1  {
2    "total": 75,
3    "offset": "20",
4    "count": "20",
5    "result": [
6      {
7        "id": 460920,
8        "name": "Steep™",
9        "releaseDate": "2016-12-02T00:00:00.000Z",
10       "developer": "Ubisoft Annecy;Ubisoft Montpellier;Ubisoft Kiev",
11       "publisher": "Ubisoft"
12     }
13   ]
14 }

```

Пример запросы к games сервису без токена:

GET ▼ `{{url}}/games?offset=20&count=20&publisher=Ubisoft&sortByPrice=desc&developer=Ubisoft`

Params ● Authorization **Headers (7)** Body Pre-request Script Tests Settings

Headers 👁 6 hidden

	Key	Value
<input type="checkbox"/>	Authorization	d636a48c-45c8-400c-8a0b-eea5dba3902a

Body Cookies Headers (7) Test Results 🌐

Pretty Raw Preview Visualize JSON ▼ ⇌

```

1  {
2    "error": "Access Denied"
3  }

```

ВЫВОД

В результате выполнения лабораторной работы я получил приложение, которое разделено на микросервисы.