

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

Дисциплина: Бэк-энд разработка

Отчет

Лабораторная работа 5

Выполнили:

Прокопец Семен

Группа К3439

**Проверил:
Добряков Д. И.**

Санкт-Петербург

2025 г.

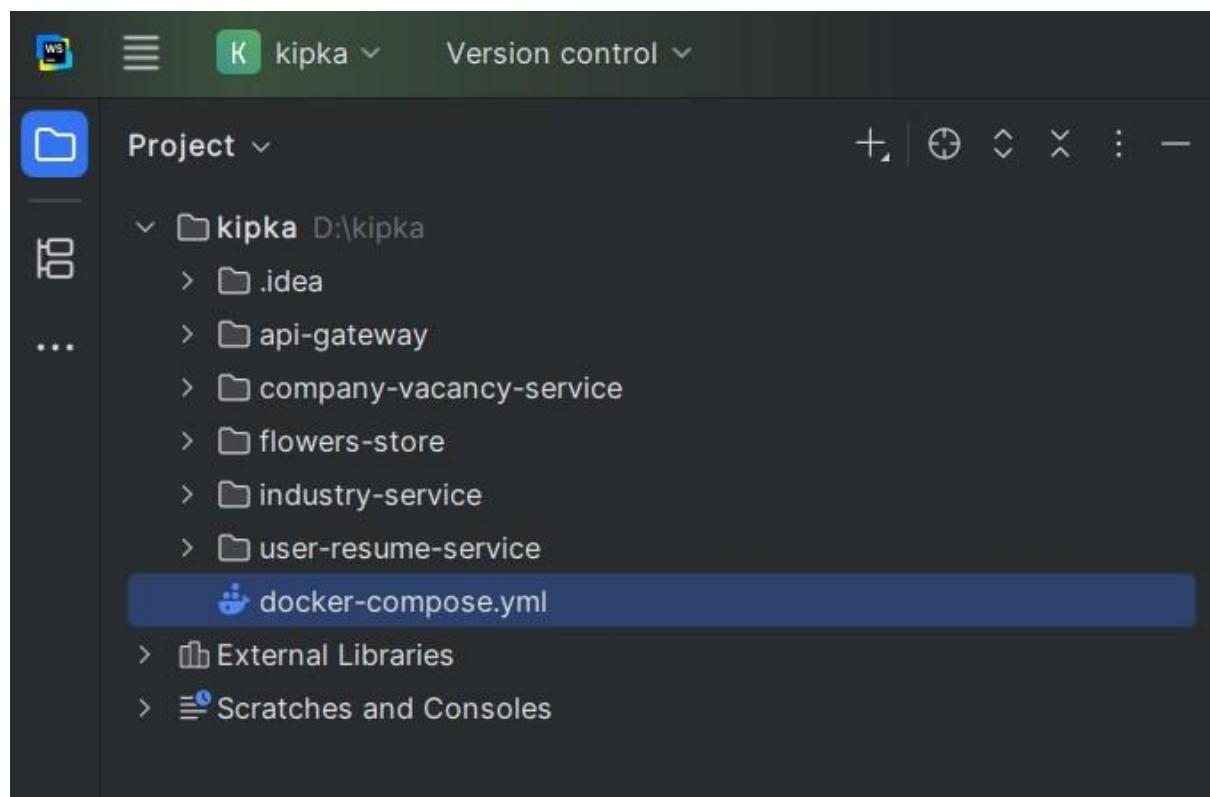
Задача

Реализация Микросервисной Архитектуры

Ход работы

Нужно реализовать Микросервисы на nest.js + PrismaORM

Структура:



Gateway:

```
import {
  Controller,
  Req,
  Res,
  Get,
  Post,
  Body,
  HttpStatus,
  ValidationPipe,
} from '@nestjs/common';
import { HttpService } from '@nestjs/axios';
import { Request, Response } from 'express';
import { ApiTags, ApiOperation, ApiBody, ApiResponse } from
  '@nestjs/swagger';

@Controller()
export class AppController {
  private readonly services = {
    user: 'http://localhost:3001',
    company: 'http://localhost:3002',
    industry: 'http://localhost:3003',
  };

  constructor(private readonly httpService: HttpService) {}
}
```

```
@Get(':service/*')

@ApiTags('gateway')
@ApiOperation({ summary: 'Proxy GET-запросы к микросервисам' })
@ApiResponse({ status: 200, description: 'Успешный ответ от микросервиса' })
@ApiResponse({ status: 404, description: 'Сервис не найден' })
async get(@Req() req: Request, @Res() res: Response) {
    const { service } = req.params;
    this.logRequest(req, service);

    if (!this.services[service]) {
        return res
            .status(HttpStatus.NOT_FOUND)
            .json({ error: `Service "${service}" not found` });
    }

    const url = this.buildUrl(req, service);
    try {
        const response = await this.httpService.axiosRef.get(url, {
            headers: req.headers,
        });
        this.proxyResponse(res, response);
    } catch (error) {
        this.handleError(res, error);
    }
}

@Post(':service/*')
@ApiTags('gateway')
@ApiOperation({ summary: 'Proxy POST-запросы к микросервисам' })
@ApiBody({ description: 'Данные для отправки в микросервис' })
@ApiResponse({ status: 201, description: 'Ресурс успешно создан' })
@ApiResponse({ status: 400, description: 'Ошибка валидации или запроса' })
async post(
    @Req() req: Request,
    @Res() res: Response,
    @Body(new ValidationPipe({ transform: true })) body: any,
) {
    const { service } = req.params;
    this.logRequest(req, service, body);

    if (!this.services[service]) {
        return res
            .status(HttpStatus.NOT_FOUND)
            .json({ error: `Service "${service}" not found` });
    }
}
```

```
const url = this.buildUrl(req, service);
try {
  const response = await this.httpService.axiosRef.post(url, body, {
    headers: req.headers,
  });
  this.proxyResponse(res, response);
} catch (error) {
  this.handleError(res, error);
}
}

private buildUrl(req: Request, service: string): string {
  const baseUrl = this.services[service];
  let path = req.url.replace(`api/${service}`, '');

  if (!path || path === '/') path = '';

  const finalUrl = `${baseUrl}${path}`;
  console.log(`Proxying to: ${finalUrl}`);
  return finalUrl;
}

private proxyResponse(res: Response, response: any) {
  const setCookies = response.headers['set-cookie'];
  if (setCookies) {
    res.header('Set-Cookie', setCookies);
  }
  res.status(response.status).json(response.data);
}

private handleError(res: Response, error: any) {
  const status = error.response?.status || 500;
  const data = error.response?.data || { error: 'Internal Server Error' };
  console.error(` Error in gateway:`, error.message);
  res.status(status).json(data);
}

private logRequest(req: Request, service: string, body?: any) {
  console.log(` ${req.method} request to service: ${service}`);
  console.log(`   URL: ${req.originalUrl}`);
  if (body) {
    console.log(`   Body:`, JSON.stringify(body, null, 2));
  }
}
```

```
@Controller('health')
export class AppHealthController {
  @Get()
  check() {
    return {
      status: 'Gateway is running',
      timestamp: new Date(),
    };
  }
}
```

Вывод: Реализовали Микросервисную Архитектуру