

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

Дисциплина: Бэк-энд разработка

Отчет

Лабораторная работа

Выполнил:

Прокопец Семен

Группа К3339

Проверил:

Добряков Д. И.

Санкт-Петербург

2025 г.






























Задача

Реализация стиля RESTful

Ход работы

Нужно реализовать стиль RESTful на nest.js + PrismaORM

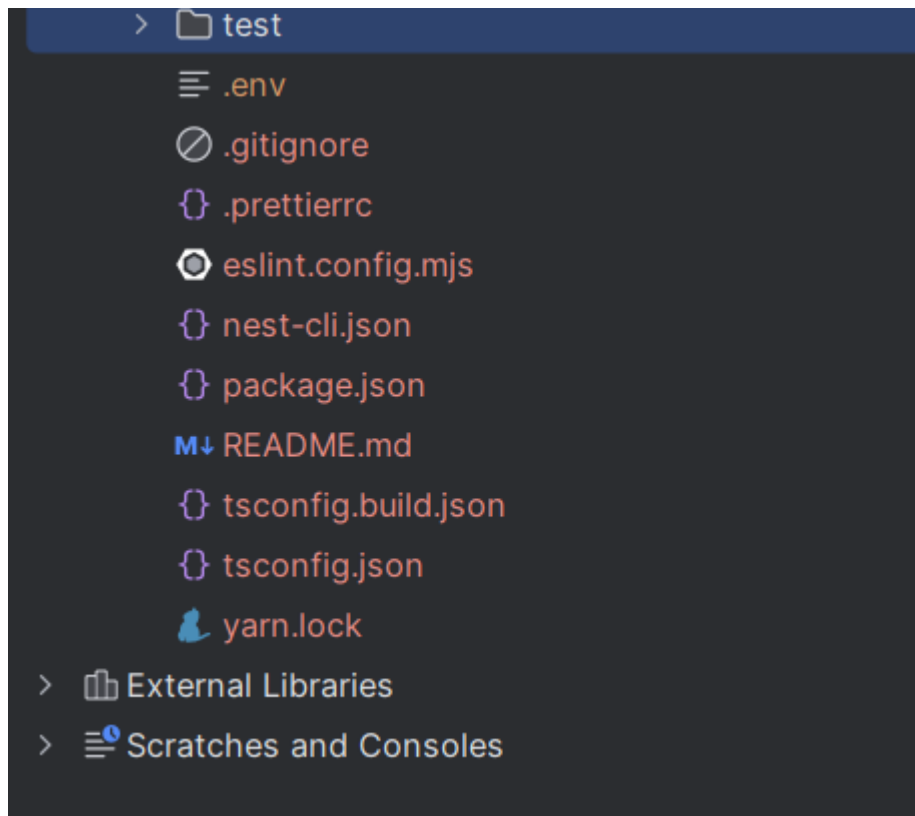
Структура:

- >  dist
- >  node_modules library root
- ✓  prisma
 -  schema.prisma
- ✓  src
 - ✓  application
 -  application.controller.ts
 -  application.dto.ts
 -  application.module.ts
 -  application.service.ts
 - ✓  company
 -  company.controller.ts
 -  company.dto.ts
 -  company.module.ts
 -  company.service.ts
 - ✓  conception
 -  guard.ts
 -  middleware.ts
 -  pipe.ts
 - ✓  education
 -  education.controller.ts
 -  education.dto.ts
 -  education.module.ts
 -  education.service.ts
 - ✓  Industry
 -  industry.controller.ts
 -  industry.dto.ts
 -  industry.module.ts
 -  industry.service.ts

- ▼ resume
 - TS resume.controller.ts
 - TS resume.dto.ts
 - TS resume.module.ts
 - TS resume.service.ts
- ▼ users
 - TS users.controller.ts
 - TS users.dto.ts
 - TS users.module.ts
 - TS users.service.ts
- ▼ vacancy
 - TS vacancy.controller.ts
 - TS vacancy.dto.ts
 - TS vacancy.module.ts
 - TS vacancy.service.ts
- > workExperiences
 - TS app.module.ts
 - TS main.ts
 - TS prisma.service.ts
 - TS types.ts

> test

- ≡ .env
- 🔗 .gitignore
- 📄 .prettierrc
- ⚙️ eslint.config.mjs
- 📄 nest-cli.json
- 📄 package.json
- M↓ README.md
- 📄 tsconfig.build.json



для каждой сущности были реализованы контроллеры, сервисы и модули:

Рассмотрим сущность резюме

Контроллер:

```
import {  
  
  Body,  
  
  Controller,  
  
  Delete,  
  
  Get,  
  
  Param,  
  
  Post,  
  
  Put,  
  
  Query,  
  
  UseGuards,  
  
  UsePipes,
```

```
ValidationPipe,

} from '@nestjs/common';

import { ApiTags, ApiOperation, ApiResponse, ApiBearerAuth, ApiBody, ApiParam }
from '@nestjs/swagger';

import { ResumeService } from '../resume.service';

import { ParseIntPipe } from '../conception/pipe';

import { CreateResumesDto, TUpdateResumesDto } from '../resume.dto';

@ApiTags('Resumes')

@Controller('resumes')

export class ResumeController {

  constructor(private readonly resumesService: ResumeService) {}

  @Get()

  @ApiOperation({ summary: 'Получить все резюме' })

  @ApiResponse({ status: 200, description: 'Список резюме успешно получен' })

  @ApiBearerAuth()

  findAll() {

    return this.resumesService.resumeFindAll();

  }

  @Get('/:id')

  @ApiOperation({ summary: 'Получить резюме по ID' })

  @ApiResponse({ status: 200, description: 'Резюме успешно получено' })

  @ApiResponse({ status: 404, description: 'Резюме не найдено' })

  @ApiParam({ name: 'id', type: 'number', description: 'ID резюме' })

  getResume(@Param('id', ParseIntPipe) id: number) {

    return this.resumesService.resumeGetById(id);

  }

}
```

```

@Post()

@UsePipes(new ValidationPipe())

@ApiOperation({ summary: 'Создать новое резюме' })
@ApiResponse({ status: 201, description: 'Резюме успешно создано' })
@ApiResponse({ status: 400, description: 'Неверные данные' })
@ApiBody({ type: CreateResumesDto })
@ApiBearerAuth()

create(@Body() dto: CreateResumesDto) {

    return this.resumesService.resumeCreate(dto);
}

@Put('/:id')

@UsePipes(new ValidationPipe())

@ApiOperation({ summary: 'Обновить резюме' })
@ApiResponse({ status: 200, description: 'Резюме успешно обновлено' })
@ApiResponse({ status: 400, description: 'Неверные данные' })
@ApiResponse({ status: 404, description: 'Резюме не найдено' })
@ApiParam({ name: 'id', type: 'number', description: 'ID резюме' })
@ApiBearerAuth()

update(

    @Param('id', ParseIntPipe) id: number,

    @Body() dto: TUpdateResumesDto,
) {

    return this.resumesService.resumeUpdate(id, dto);
}

@Delete('/:id')

@UsePipes(new ValidationPipe())

```

```

@ApiOperation({ summary: 'Удалить резюме' })

@ApiResponse({ status: 200, description: 'Резюме успешно удалено' })

@ApiResponse({ status: 404, description: 'Резюме не найдено' })

@ApiParam({ name: 'id', type: 'number', description: 'ID резюме' })

@ApiBearerAuth()

delete(@Param('id', ParseIntPipe) id: number) {

    return this.resumesService.resumeDelete(id);

}

}

```

Сервис:

```

import {Injectable, NotFoundException} from '@nestjs/common';

import {PrismaService} from '../prisma.service';

import {CreateResumesDto, TUpdateResumesDto} from './resume.dto';

import {ConfigService} from '@nestjs/config';

import {EnumAppMode} from '../types';

@Injectable()

export class ResumeService {

    constructor(private readonly prisma:PrismaService, private readonly
configService:ConfigService) {

    }

    resumeFindAll() {

        console.log(this.configService.get<EnumAppMode>('MODE'))

        return this.prisma.resume.findMany()

    }

    resumeGetById(id: number) {

        const resume = this.prisma.resume.findUnique({

```



```

        where: { id },
    });

    if (!resume) {
        throw new NotFoundException('resume not found');
    }

    return resume;
}

resumeCreate(dto: CreateResumesDto){
    return this.prisma.resume.create({
        data: dto,
    })
}

resumeUpdate(id: number, dto: TUpdateResumesDto) {
    return this.prisma.resume.update({
        where: { id },
        data: dto,
    });
}

resumeDelete(id: number) {
    return this.prisma.resume.delete({
        where: { id },
    });
}
}

```

DTO:

```
import { IsString, IsOptional, IsNumber, ValidateNested } from "class-validator";
```

```
import { Type } from "class-transformer";
```

```
class SkillDto {
```

```
    @IsString()
```

```
    name: string;
```

```
    @IsString()
```

```
    @IsOptional()
```

```
    level?: string;
```

```
}
```

```
export class CreateResumesDto {
```

```
    @IsString()
```

```
    title: string;
```

```
    @IsString()
```

```
    @IsOptional()
```

```
    experience_summary?: string;
```

```
    @IsNumber()
```

```
    @IsOptional()
```

```
    salary_expectations?: number;
```

```
    @IsString()
```

```
    @IsOptional()
```

```
    skills?: string;
```

```
    @IsNumber()
```

```
    user_id: number;
```

```
}  
  
export type TUpdateResumesDto = Partial<CreateResumesDto>;
```

Бодуль:

```
import { Module } from '@nestjs/common';  
  
import { ResumeService } from './resume.service';  
  
import { ResumeController } from './resume.controller';  
  
import { PrismaService } from '../prisma.service';  
  
import { ConfigService } from '@nestjs/config';  
  
@Module({  
  controllers: [ResumeController],  
  providers: [ResumeService, PrismaService, ConfigService],  
})  
  
export class ResumeModule {}
```

Вывод

Реализовали стиль RESTful