САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Лабораторная работа

Выполнил:

Чернышев Миша

Группа К3341

Проверил:
Добряков Д. И.

Санкт-Петербург

2025 г.

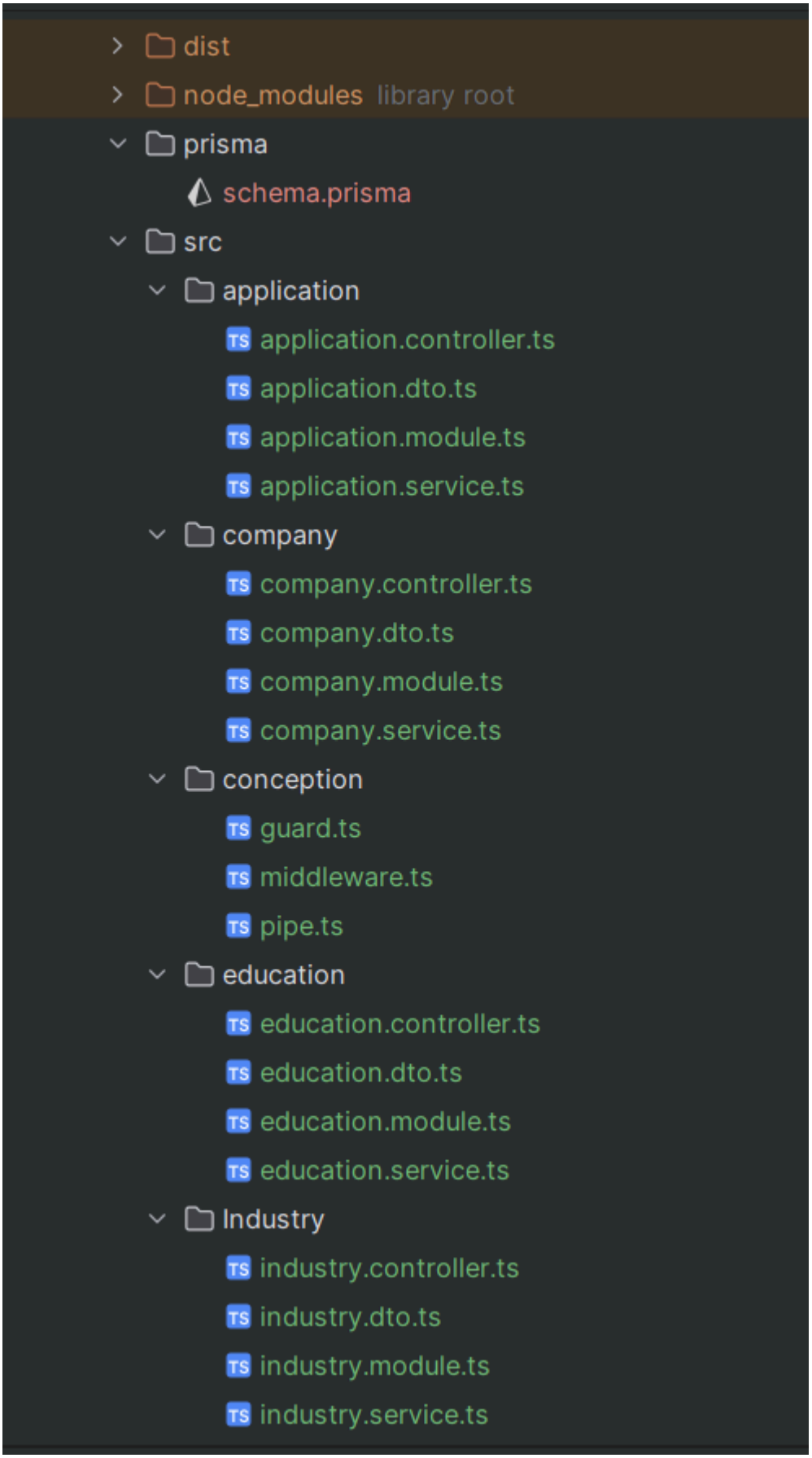
Задача

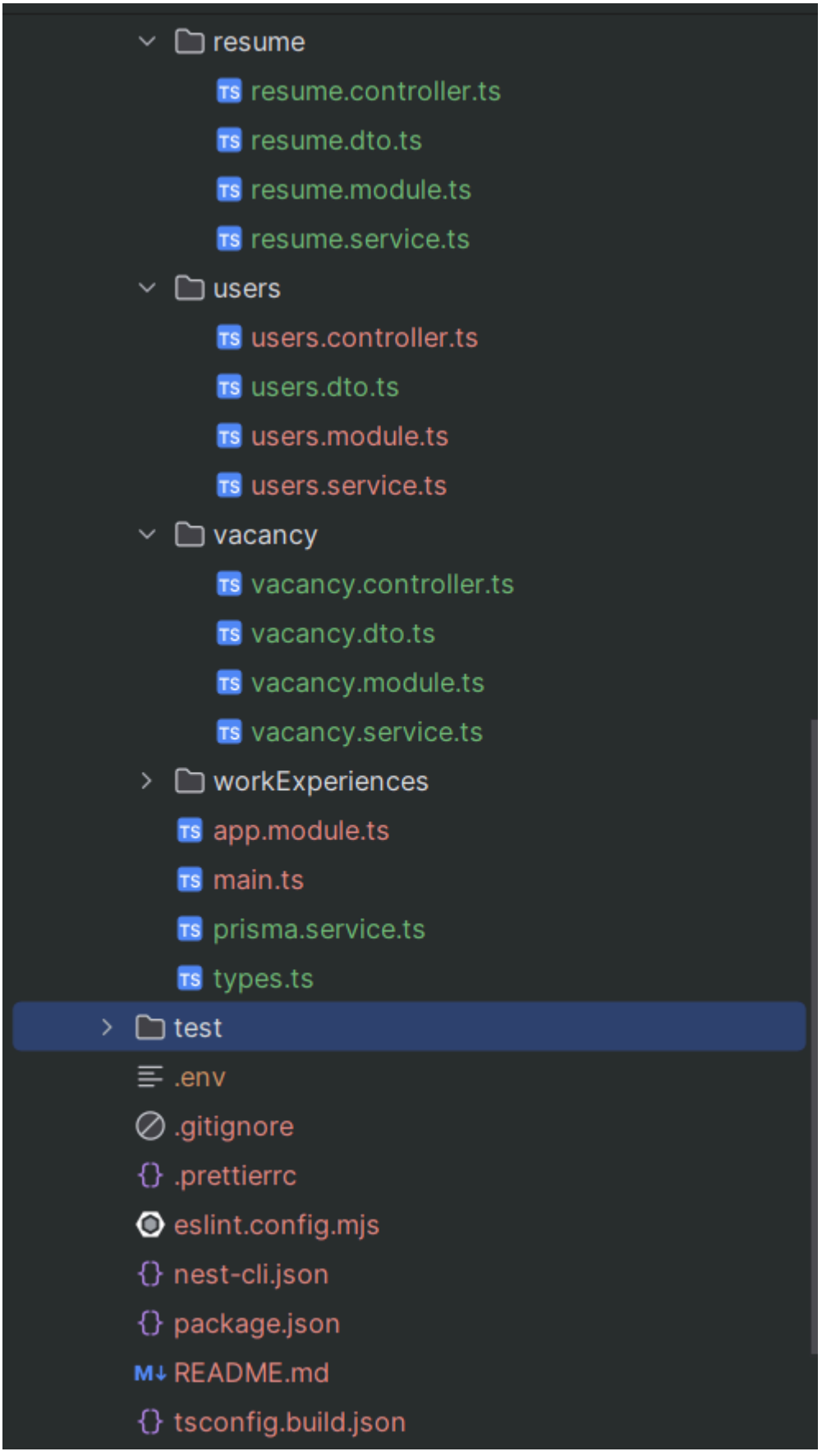
Реализация boilerplate

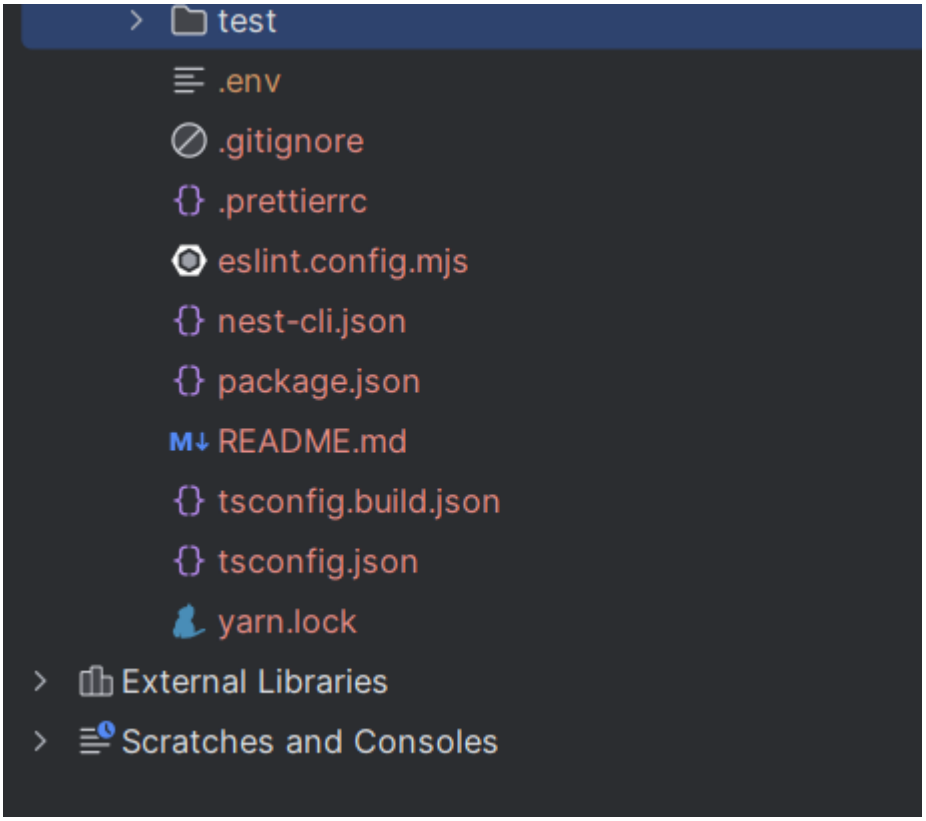
Ход работы

Нужно написать свой bouilerplate 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
resume.controller.ts
resume.dto.ts
resume.module.ts
resume.service.ts
users
users.controller.ts
users.dto.ts
users.module.ts
users.service.ts
vacancy
vacancy.controller.ts
vacancy.dto.ts
vacancy.module.ts
vacancy.service.ts
workExperiences
app.module.ts
main.ts
prisma.service.ts
types.ts
test
.env
.gitignore
.prettierrc
eslint.config.mjs
nest-cli.json
package.json
README.md
tsconfig.build.json



После ревью:

Добавил дто для авторизации:

login.dto.ts

```ts
import { IsEmail, IsString } from 'class-validator';
import { ApiProperty } from '@nestjs/swagger';

export class LoginDto {  Show usages
    @ApiProperty({
        example: 'mem@ka.com',
        description: 'Email пользователя',
    })
    @IsEmail()
    email: string;

    @ApiProperty({
        example: '12345678',
        description: 'Пароль пользователя',
    })
    @IsString()
    password: string;
}
```

получение токена:

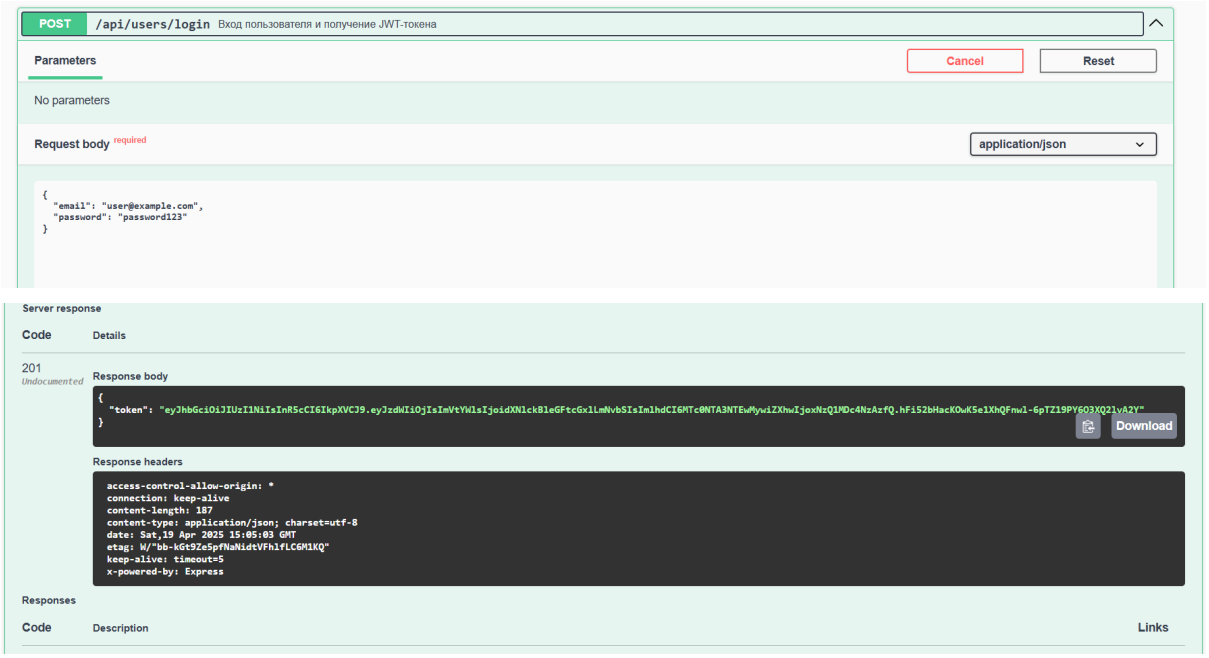
```ts
  @Post( path: 'login')  no usages
  @ApiOperation({ summary: 'Вход пользователя и получение JWT-токена' })
  @ApiResponse({ status: 200, description: 'JWT-токен успешно получен' })
  @ApiResponse({ status: 401, description: 'Неверный email или пароль' })
  @ApiBody({ type: LoginDto })
  async login(@Body() dto: LoginDto): Promise<{ token: string }> {
    const user : {id: number; email: string; password: st...  = await this.usersService.validateUser(dto.email, dto
    const token : string  = this.usersService.generateToken(user);
    return { token };
  }
}
```

Проверка токена в middlewere:

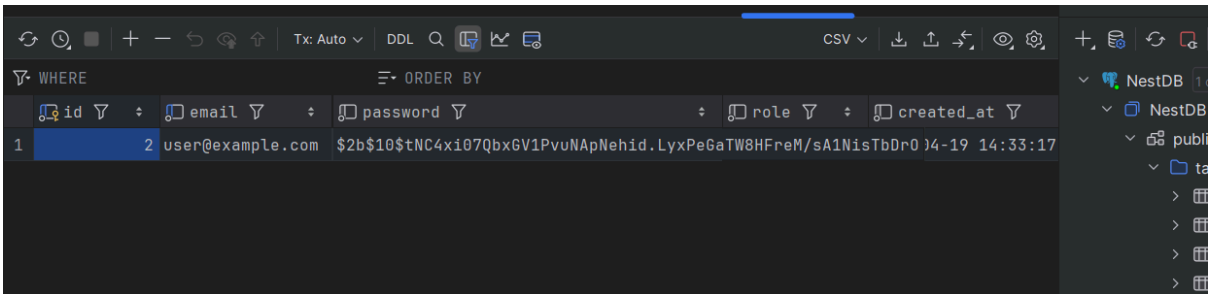
```ts
import {Injectable, UnauthorizedException} from "@nestjs/common";

import {JwtService} from "@nestjs/jwt";
```

```
@Injectable()
export class JwtMiddleware {
   constructor(private readonly jwtService: JwtService) {}

   use(req: any, res: any, next: () => void) {
       console.log(`Request URL: ${req.url}`);

       const authHeader = req.headers.authorization;

       if (!authHeader || !authHeader.startsWith('Bearer ')) {
           console.log('Token is missing or invalid');
           throw new UnauthorizedException('Отсутствует или неверный токен');
       }

       const token = authHeader.split(' ')[1];

       try {
           const payload = this.jwtService.verify(token, { secret: 'secret' });
           console.log('Token verified:', payload);
           req.user = payload;
           next();
       } catch (error) {
           console.error('Token verification failed:', error.message);
           throw new UnauthorizedException('Неверный или просроченный токен');
       }
   }
}
```

Получение токена по почте и паролю:



Хеширование пароля:



Вывод

Спроектировали и реализовали классное ПО