САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Лабораторная работа №2

Выполнил:

Гнеушев Владислав
К3339

Проверил:
Добряков Д. И.

Санкт-Петербург

2025 г.

Задача

По выбранному варианту необходимо будет реализовать RESTful API средствами express + typescript (используя ранее написанный boilerplate).

Ход работы

1. Проведен рефакторинг кода из второй домашней работы. Методы контроллеров теперь принимают на вход DTO-объекты с провалидированными данными.

```
export class CreateCompanyDto {
    @IsNotEmpty({ message: 'Name should not be empty' })
    @IsString({ message: 'Name must be a string' })
    name!: string;

    @IsNotEmpty({ message: 'Employer ID should not be empty' })
    @IsNumber({}, { message: 'Employer ID must be an integer' })
    employerId!: number;

    @IsOptional()
    @IsString({ message: 'Description must be a string' })
    description?: string;
}
```

Рисунок 1 — Пример DTO-класса с правилами валидации

2. Роуты разделены по папкам, в каждой из которых хранятся эндпоинты и схемы тел запросов для этих эндпоинтов.

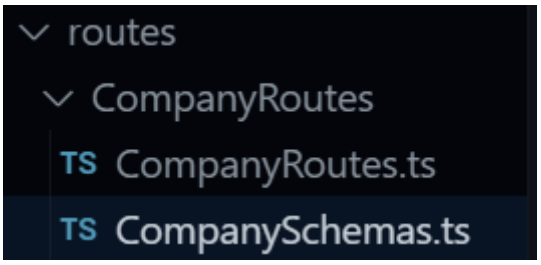
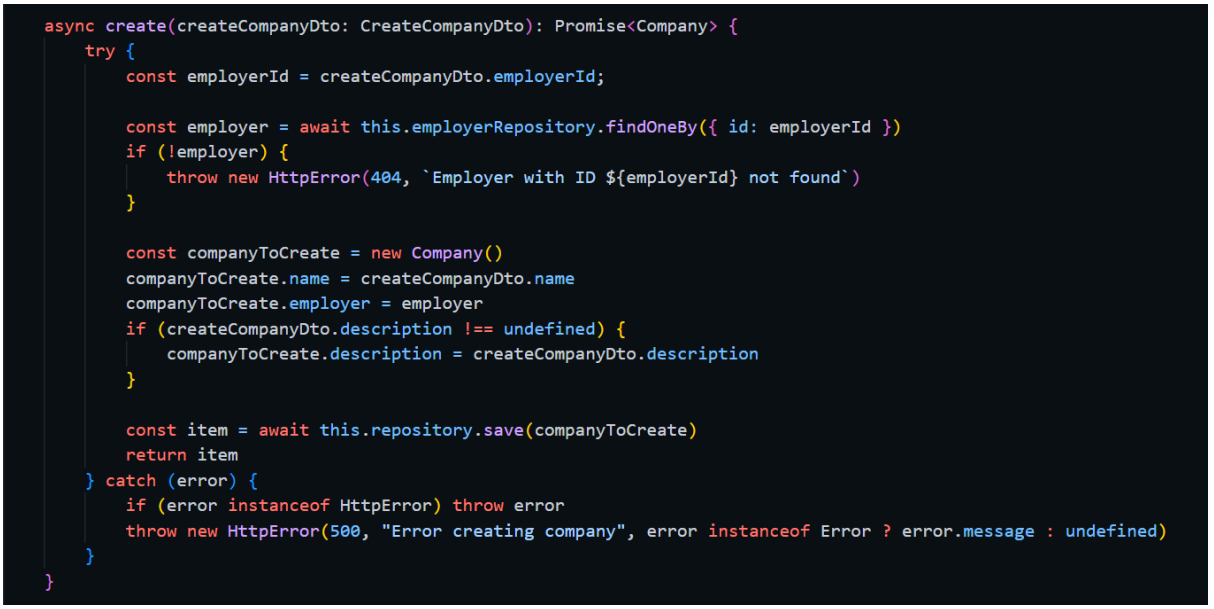


Рисунок 2 — Расположение файлов в директориях

```
router.post(
    '/',
    validationMiddleware(CreateCompanyDto),
    async (req: Request, res: Response, next: NextFunction) => {
        const createCompanyDto = req.body as CreateCompanyDto;
        try {
            const newCompany = await companyController.create(createCompanyDto);
            res.status(201).json(newCompany);
        } catch (error) {
            if (error instanceof HttpError) {
                res.status(error.statusCode).json({
                    message: error.message,
                    ...(error.details && { errors: error.details }),
                });
            } else {
                next(error);
            }
        }
    }
);
```

Рисунок 3 — Пример эндпоинта, добавляющего сущность в базу данных



```
async create(createCompanyDto: CreateCompanyDto): Promise<Company> {
    try {
        const employerId = createCompanyDto.employerId;

        const employer = await this.employerRepository.findOneBy({ id: employerId })
        if (!employer) {
            throw new HttpError(404, `Employer with ID ${employerId} not found`)
        }

        const companyToCreate = new Company()
        companyToCreate.name = createCompanyDto.name
        companyToCreate.employer = employer
        if (createCompanyDto.description !== undefined) {
            companyToCreate.description = createCompanyDto.description
        }

        const item = await this.repository.save(companyToCreate)
        return item
    } catch (error) {
        if (error instanceof HttpError) throw error
        throw new HttpError(500, "Error creating company", error instanceof Error ? error.message : undefined)
    }
}
```

Рисунок 4 — Пример метода контроллера, принимающего на вход DTO-объект с нужными данными

3. Изменена схема регистрации роутов в глобальном роутере Express. Теперь эндпоинты для сущности регистрируются в локальном роутере, а тот уже прикрепляется к основному роутеру.

```
// API Routes
app.use(settings.app.API_PREFIX + '/users', userRoutes);
app.use(settings.app.API_PREFIX + '/companies', companyRoutes);
app.use(settings.app.API_PREFIX + '/job-categories', jobCategoryRoutes);
app.use(settings.app.API_PREFIX + '/job-offers', jobOfferRoutes);
app.use(settings.app.API_PREFIX + '/skills', skillRoutes);
app.use(settings.app.API_PREFIX + '/resumes', resumeRoutes);
app.use(settings.app.API_PREFIX + '/employer-cabinets', employerCabinetRoutes);
app.use(settings.app.API_PREFIX + '/employee-cabinets', employeeCabinetRoutes);
```

Рисунок 5 — Регистрация эндпоинтов в корневом роутере

Вывод

В данной лабораторной работе был проведен рефакторинг старого кода. Теперь проект имеет разделение на модули с роутами, контроллерами и моделями (сущностями). Входные данные для эндпоинтов создания и изменения сущностей валидируются при помощи библиотеки *class-validator.* Обновленные роуты удовлетворяют стандартам REST.