САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Лабораторная работа 2

Выполнила:

Фролова Кристина

Группа 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

Проект был написан на основе boilerplate, реализованного в рамках ЛР1.

На рисунке 1 продемонстрирована структура проекта.

На рисунке 2 продемонстрирован AdvertisementController, который работает с AdvertisementService, продемонстрированного на рисунке 3.

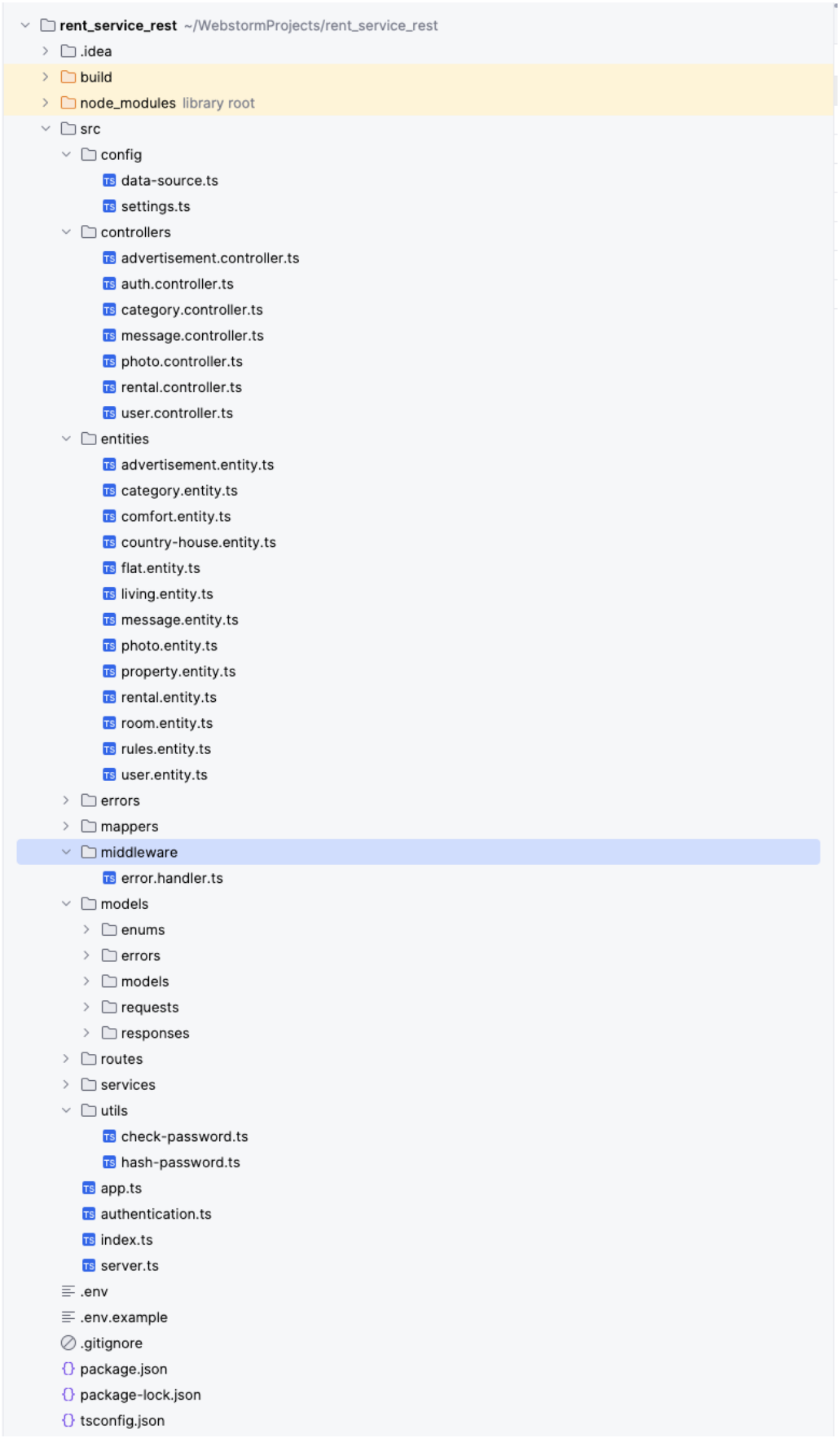

Рисунок 1 – Структура проекта

```
@Route("advertisements")  Show usages
@Tags("Advertisement")
export class AdvertisementController extends Controller {

    @Get()  Show usages
    @SuccessResponse("200", "Ok")
    @Security("jwt")
    public async getAll(): Promise<AdvertisementResponseDto[]> {
        const advertisements : Advertisement[]  = await advertisementService.getAll();
        return advertisements.map(toAdvertisementResponseModel);
    }

    @Get("{advertisementId}")  Show usages
    @SuccessResponse("200", "Ok")
    @Security("jwt")
    @Response<EntityNotFoundError>(404, "Entity not found")
    public async getById(@Path() advertisementId: number): Promise<AdvertisementResponseDto> {
        const advertisement : Advertisement  = await advertisementService.getById(advertisementId);
        return toAdvertisementResponseModel(advertisement);
    }

    @Post()  Show usages
    @SuccessResponse("201", "Created")
    @Security("jwt")
    public async create(
        @Body() body: CreateAdvertisementRequestDto
    ): Promise<AdvertisementResponseDto> {
        const advertisement : Advertisement  = await advertisementService.create(createAdvertisementRequestToModel(body));
        return toAdvertisementResponseModel(advertisement);
    }

    @Put("{advertisementId}")  Show usages
    @SuccessResponse("200", "Updated")
    @Security("jwt")
    @Response<EntityNotFoundError>(404, "Entity not found")
    public async update(
        @Path() advertisementId: number,
        @Body() body: UpdateAdvertisementRequestDto
    ): Promise<AdvertisementResponseDto> {
        const ad : Advertisement  = await advertisementService.update(advertisementId, updateAdvertisementRequestToModel(body));
        return toAdvertisementResponseModel(ad);
    }

    @Delete("{advertisementId}")  Show usages
    @SuccessResponse("204", "Deleted")
    @Security("jwt")
    public async delete(@Path() advertisementId: number): Promise<void> {
        await advertisementService.delete(advertisementId);
    }
```

Рисунок 2 - AdvertisementController

```
class AdvertisementService {  Show usages
    private repo : Repository<AdvertisementEntity>  = dataSource.getRepository(AdvertisementEntity);

    async getAll(): Promise<Advertisement[]> {  Show usages
        const entities : AdvertisementEntity[]  = await this.repo.find({
            relations: [
                "owner",
                "category",
                "property",
                "property.living",
                "property.living.comfort",
                "property.living.flat",
                "property.living.room",
                "property.living.countryHouse",
                "rules",
                "photos"
            ]
        });
        return entities.map(entityToAdvertisement);
    }

    async getRentalsByAdvertisementId(advertisementId: number): Promise<Rental[]> {  Show usages
        return await rentalService.getRentalsByAdvertisementId(advertisementId)
    }

    async getById(id: number): Promise<Advertisement> {  Show usages
        const entity : AdvertisementEntity  = await this.repo.findOne({
            where: {id},
            relations: [
                "owner",
                "category",
                "property",
                "property.living",
                "property.living.comfort",
                "property.living.flat",
                "property.living.room",
                "property.living.countryHouse",
                "rules",
                "photos"
            ]
        });
        if (!entity) throw new EntityNotFoundError(AdvertisementEntity, id, "id");
        return entityToAdvertisement(entity);
    }
```

Рисунок 3 – AdvertisementService

Вывод

В рамках работы был реализован RESTful API на основе реализованного ранее boilerplate.