Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«Национальный исследовательский университет ИТМО»
(Университет ИТМО)

О Т Ч Е Т

по лабораторной работе №2
«Реализация REST API на основе boilerplate»
курса «Бэкенд разработка»

Выполнили:
Бахарева М.А., K3342
Привалов К.А., K3342
Проверил:
Добряков Д.И.

Санкт-Петербург,
2025

Содержание

Ход работы

Задание

Лабораторная работа заключалась в реализации REST API по выбранной теме (Сервис аренды недвижимости) на основе boilerplate, который был создан в ЛР №1, и моделей, которые были написаны в ДЗ №2.

В процессе работы были выделены основные этапы реализации API:

- Деплой базы данных (PostgreSQL) и написание миграций;
- Рефакторинг моделей, добавление нового функционала в контроллеры для покрытия определенный кейсов (например, загрузка изображений) и сервисов;
- Написание промежуточного ПО для валидации некоторого функционала контроллеров.

Основная часть

Инициализация проекта

Для начала инициализируем Node.JS проект командой npm init -y. Устанавливаем TypeScript, настраиваем в проекте (npm install –save-dev typescript) и создаем файл tsconfig.json со следующим содержанием:

```
{
 "compilerOptions": {
   "target": "ES2020",
   "module": "CommonJS",
   "strict": true,
   "esModuleInterop": true,
   "experimentalDecorators": true,
   "emitDecoratorMetadata": true,
   "outDir": "./dist",
   "rootDir": "./src",
   "strictPropertyInitialization": false,
   "skipLibCheck": true,
   "resolveJsonModule": true,
   "forceConsistentCasingInFileNames": true
 },
 "include": ["src/**/*"],
 "exclude": ["node_modules", "dist"]
}
```

Копируем package.json и package-lock.json для установки базовых зависимостей из boilerplate.

Рефакторинг и добавление новых компонентов

После анализа существующих моделей необходимо было добавить еще одну модель: чат, которую связывала бы сообщения и объект недвижимости. Помимо этого в некоторых моделях необходимо было добавить соединение сущности с другими для эффективных запросов.

Для обеспечения авторизации и аутентификации были реализованы 3 эндпоинта для управления учетными записями:

- Регистрация;
- Логин;
- Смена пароля.

Также важной задачей было создание промежуточного ПО, которое занималось бы валидацией запросов от пользователей.

Итоговая структура проекта выглядит следующим образом:

```
.
├── app.ts
├── config
│   ├── databaseConfig.ts
│   └── ProcessEnv.d.ts
├── controllers
│   ├── BaseController.ts
│   ├── BookingRequestController.ts
│   ├── ChatController.ts
│   ├── ComplaintController.ts
│   ├── FavoriteController.ts
│   ├── PropertyController.ts
│   ├── PropertyImageController.ts
│   ├── RentalController.ts
│   ├── ReviewController.ts
│   └── UserController.ts
├── entities
│   ├── BookingRequest.ts
│   ├── Chat.ts
│   ├── Complaint.ts
│   ├── Favorite.ts
│   ├── Message.ts
│   ├── Property.ts
│   ├── PropertyImage.ts
│   ├── Rental.ts
│   ├── Review.ts
│   └── User.ts
├── index.ts
├── middleware
│   ├── checkJwt.ts
│   ├── errorHandler.ts
│   └── validator
│       ├── validatorBookingRequest.ts
│       ├── validatorChangePassword.ts
│       ├── validatorChat.ts
│       ├── validatorComplaint.ts
│       ├── validatorLogin.ts
│       ├── validatorProperty.ts
│       ├── validatorRegister.ts
│       ├── validatorRental.ts
│       └── validatorUpdateUser.ts
├── migrations
│   ├── 1746898783675-CreateTypeUserRole.ts
│   ├── 1746898803014-CreateTypeBookingRequestStatus.ts
│   ├── 1746898813338-CreateTypeComplaintStatus.ts
│   ├── 1746898826442-CreateTypeRentalStatus.ts
│   ├── 1746898840634-CreateTableUser.ts
│   ├── 1746898848453-CreateTableProperty.ts
│   ├── 1746898852041-CreateTableRental.ts
│   ├── 1746898854512-CreateTableChat.ts
```

```
│   ├── 1746898861382-CreateTableMessage.ts
│   ├── 1746898867125-CreateTableFavorite.ts
│   ├── 1746898873399-CreateTableBookingRequest.ts
│   ├── 1746898884411-CreateTableComplaint.ts
│   ├── 1746898889873-CreateTableReview.ts
│   └── 1746898897868-CreateTablePropertyImages.ts
├── routes
│   ├── booking-requests.ts
│   ├── chats.ts
│   ├── complaints.ts
│   ├── favorites.ts
│   ├── index.ts
│   ├── properties.ts
│   ├── property-images.ts
│   ├── rentals.ts
│   ├── reviews.ts
│   └── users.ts
├── services
│   └── UserService.ts
├── types
│   └── express
│       └── index.d.ts
└── utils
    ├── jwt.ts
    └── password.ts

12 directories, 63 files
```

Вывод

Мы реализовали REST API по нашей теме на основе созданного boilerplate. У нас получилось связать БД и бизнес-логику приложения. В дальнейшем мы планируем перевести загрузку файлов из директории в проекте (контейнере) в S3 хранилище (например, MinIO).