**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

**Дисциплина:** Бэк-энд разработка

Отчет

Домашнее задание №2

Выполнил:

Соловьева П.А.

Группа К3344

Проверил:

Добряков Д. И.

Санкт-Петербург

2025 г.

**Задача**

1) Реализовать все модели данных, спроектированные в рамках ДЗ1.

2) Реализовать набор из CRUD-методов для работы с моделями данных средствами Express + TypeScript.

3) Реализовать API-эндпоинт для получения пользователя по id/email.

**Ход работы**

### **1. Модели данных (Entities)**

Реализованы следующие сущности:

**User (Пользователь)**

import { Entity, PrimaryGeneratedColumn, Column, OneToMany, CreateDateColumn, UpdateDateColumn } from "typeorm";

import { Property } from "./Property";

import { Rental } from "./Rental";

import { Message } from "./Message";

import { Review } from "./Review";

export type UserRole = 'owner' | 'tenant' | 'admin';

@Entity()

export class User {

@PrimaryGeneratedColumn("uuid")

id!: string;

@Column()

first\_name!: string;

@Column()

last\_name!: string;

@Column({ unique: true })

email!: string;

@Column({ nullable: true })

phone\_number?: string;

@Column()

password\_hash!: string;

@Column({ type: "text", default: "tenant" })

role!: UserRole;

@CreateDateColumn()

created\_at!: Date;

@UpdateDateColumn()

updated\_at!: Date;

@OneToMany(() => Property, (p) => p.owner)

properties!: Property[];

@OneToMany(() => Rental, (r) => r.tenant)

rentals!: Rental[];

@OneToMany(() => Message, (m) => m.sender)

sent\_messages!: Message[];

@OneToMany(() => Message, (m) => m.receiver)

received\_messages!: Message[];

@OneToMany(() => Review, (rev) => rev.reviewer)

reviews!: Review[];

}

**Property (Объект недвижимости)**

**import { Entity, PrimaryGeneratedColumn, Column, ManyToOne, OneToMany } from "typeorm";**

**import { User } from "./User";**

**import { Rental } from "./Rental";**

**import { PropertyAmenity } from "./PropertyAmenity";**

**@Entity()**

**export class Property {**

**@PrimaryGeneratedColumn("uuid")**

**id: string;**

**@ManyToOne(() => User, (user) => user.properties)**

**owner: User;**

**@Column()**

**title: string;**

**@Column("text")**

**description: string;**

**@Column()**

**type: string;**

**@Column()**

**location: string;**

**@Column("decimal")**

**price\_per\_month: number;**

**@Column("simple-array")**

**photos: string[];**

**@Column({ type: "date" })**

**available\_from: Date;**

**@Column({ type: "date" })**

**available\_to: Date;**

**@Column({ type: "datetime", default: () => "CURRENT\_TIMESTAMP" })**

**created\_at: Date;**

**@Column({ type: "datetime", default: () => "CURRENT\_TIMESTAMP", onUpdate: "CURRENT\_TIMESTAMP" })**

**updated\_at: Date;**

**@OneToMany(() => PropertyAmenity, (pa) => pa.property)**

**amenities: PropertyAmenity[];**

**@OneToMany(() => Rental, (rental) => rental.property)**

**rentals: Rental[];**

**}**

**Rental (Аренда)**

**import { Entity, PrimaryGeneratedColumn, Column, ManyToOne, OneToMany } from "typeorm";**

**import { User } from "./User";**

**import { Property } from "./Property";**

**import { Review } from "./Review";**

**@Entity()**

**export class Rental {**

**@PrimaryGeneratedColumn("uuid")**

**id: string;**

**@ManyToOne(() => Property, (property) => property.rentals)**

**property: Property;**

**@ManyToOne(() => User, (user) => user.rentals)**

**tenant: User;**

**@Column({ type: "date" })**

**start\_date: Date;**

**@Column({ type: "date" })**

**end\_date: Date;**

**@Column()**

**status: string;**

**@Column({ type: "datetime", default: () => "CURRENT\_TIMESTAMP" })**

**created\_at: Date;**

**@OneToMany(() => Review, (review) => review.rental)**

**reviews: Review[];**

**}**

**Message (Сообщение)**

**import { Entity, PrimaryGeneratedColumn, Column, ManyToOne } from "typeorm";**

**import { User } from "./User";**

**import { Rental } from "./Rental";**

**@Entity()**

**export class Message {**

**@PrimaryGeneratedColumn("uuid")**

**id: string;**

**@ManyToOne(() => User, (user) => user.sent\_messages)**

**sender: User;**

**@ManyToOne(() => User, (user) => user.received\_messages)**

**receiver: User;**

**@ManyToOne(() => Rental, (rental) => rental.id)**

**rental: Rental;**

**@Column("text")**

**content: string;**

**@Column({ type: "datetime", default: () => "CURRENT\_TIMESTAMP" })**

**timestamp: Date;**

**}**

**Review (Отзыв)**

**import { Entity, PrimaryGeneratedColumn, Column, ManyToOne } from "typeorm";**

**import { Rental } from "./Rental";**

**import { User } from "./User";**

**@Entity()**

**export class Review {**

**@PrimaryGeneratedColumn("uuid")**

**id: string;**

**@ManyToOne(() => Rental, (rental) => rental.reviews)**

**rental: Rental;**

**@ManyToOne(() => User, (user) => user.reviews)**

**reviewer: User;**

**@Column("int")**

**rating: number;**

**@Column("text")**

**comment: string;**

**@Column({ type: "datetime", default: () => "CURRENT\_TIMESTAMP" })**

**created\_at: Date;**

**}**

**Amenity (Удобства)**

**import { Entity, PrimaryGeneratedColumn, Column, OneToMany } from "typeorm";**

**import { PropertyAmenity } from "./PropertyAmenity";**

**@Entity()**

**export class Amenity {**

**@PrimaryGeneratedColumn("uuid")**

**id: string;**

**@Column()**

**name: string;**

**@Column({ nullable: true })**

**description: string;**

**@OneToMany(() => PropertyAmenity, (pa) => pa.amenity)**

**propertyAmenities: PropertyAmenity[];**

**}**

**PropertyAmenity (Удобства объекта недвижимости)**

import { Entity, PrimaryGeneratedColumn, ManyToOne } from "typeorm";

import { Property } from "./Property";

import { Amenity } from "./Amenity";

@Entity()

export class PropertyAmenity {

@PrimaryGeneratedColumn("uuid")

id: string;

@ManyToOne(() => Property, (property) => property.amenities)

property: Property;

@ManyToOne(() => Amenity, (amenity) => amenity.propertyAmenities)

amenity: Amenity;

}

### **2. API Endpoints**

Реализованы следующие RESTful API endpoints:

**Пользователи**

* GET /users — получить всех пользователей
* GET /users/:id — получить пользователя по ID
* GET /users?email=example@mail.com — получить пользователя по email
* POST /users — регистрация нового пользователя
* PUT /users/:id — обновить данные пользователя
* DELETE /users/:id — удалить пользователя

**Объекты недвижимости (Property)**

* GET /properties — получить все объекты (с возможностью фильтрации)
* GET /properties/:id — получить объект по ID
* POST /properties — создать новый объект
* PUT /properties/:id — обновить объект
* DELETE /properties/:id — удалить объект

**Аренда (Rental)**

* GET /rentals — получить все аренды
* GET /rentals/:id — получить аренду по ID
* POST /rentals — создать аренду
* PUT /rentals/:id — обновить аренду
* DELETE /rentals/:id — удалить аренду

**Сообщения (Message)**

* GET /messages — получить все сообщения
* GET /messages/:id — получить сообщение по ID
* POST /messages — создать сообщение
* PUT /messages/:id — обновить сообщение
* DELETE /messages/:id — удалить сообщение

**Отзывы (Review)**

* GET /reviews — получить все отзывы
* GET /reviews/:id — получить отзыв по ID
* POST /reviews — создать отзыв
* PUT /reviews/:id — обновить отзыв
* DELETE /reviews/:id — удалить отзыв

### **3. Функциональность**

**Вход и регистрация**

* POST /users — регистрация

**Личный кабинет**

* GET /users/:id — информация о пользователе и связанных арендах/отзывах

**Поиск недвижимости**

* GET /properties с фильтрацией по цене, типу и местоположению

**Управление объектами**

* CRUD для Property и Rental
* Возможность загрузки фотографий через статические файлы

**История сделок**

* Rental связан с пользователем и объектом недвижимости
* Отслеживание статусов: PENDING, ACTIVE, FINISHED

**Вывод**

В ходе выполнения домашнего задания был успешно реализован RESTful API для сервиса аренды недвижимости на базе Express.js и TypeScript.

Достигнутые результаты:

1. Реализована регистрация и вход пользователей
2. Создан личный кабинет
3. Реализован поиск недвижимости с фильтрацией
4. CRUD для объектов недвижимости и аренды
5. История сделок через сущность Rental
6. Подготовлена возможность расширения функционала (сообщения, отзывы, дополнительные фильтры)

Проект готов к использованию и дальнейшему расширению.