САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Лабораторная работа №1
Реализация boilerplate

Выполнил:
Оспельников Алексей

Группа К3340

Проверил:
Добряков Д. И.

Санкт-Петербург

2025 г.

Задача

Задание:

Нужно написать свой boilerplate на express + TypeORM + typescript.
Должно быть явное разделение на:

1) модели
2) контроллеры
3) роуты

Ход работы

Пример сущности:

```
import { Entity, PrimaryGeneratedColumn, Column, ManyToOne,
OneToMany, CreateDateColumn } from "typeorm";
import { User } from "./User";
import { Property } from "./Property";
import { Review } from "./Review";
@Entity()
export class Rent {
    @PrimaryGeneratedColumn()
    id: number;
    @Column()
    guest_count: number;
   @Column()
    status: string;
    @Column()
    start_rent: Date;
    @Column()
    end_rent: Date;
    @ManyToOne(() => User, user => user.tenants)
    tenant: User;
    @ManyToOne(() => Property, property => property.rents)
    estate: Property;
    @OneToMany(() => Review, review => review.rent)
    review: Review[];
    @CreateDateColumn()
    created_at: Date;
}
```

Пример Роутера:

```
import { Router } from "express";
import { MessageController } from "../controller/MessageController";
const messageRouter = Router();
messageRouter.get("/message", MessageController.all);
messageRouter.post("/message", MessageController.create);
messageRouter.get("/message/:id", MessageController.findOne);
messageRouter.put("/message/:id", MessageController.update);
messageRouter.delete("/message/:id", MessageController.delete);
export default userRouter;
```

Пример Контроллера:

```
import { Request, Response } from "express";
import { userRepository } from "../repository";
export class UserController {
  static async all(request: Request, response: Response) {
    const data = await userRepository.findAll();
    return response.status(200).send(data);
  }
  static async byEmail(request: Request, response: Response) {
    const data = await userRepository.findByEmail(request.params.email);
    return response.status(200).send(data);
  }
  static async create(request: Request, response: Response) {
    const data = await userRepository.createUser(request.body);
    return response.status(201).send(data);
  }
  static async findOne(request: Request, response: Response) {
    const id = Number(request.params.id);
    const data = await userRepository.findOne(id);
    return response.send(data);
  }
  static async update(request: Request, response: Response) {
    const id = Number(request.params.id);
    const data = await userRepository.updateUser(id, request.body);
    return response.send(data);
  }
  static async delete(request: Request, response: Response) {
    const id = Number(request.params.id);
    const data = await userRepository.delete(id);
    return response.send(data);
  }
}
```

Пример сервиса

```
import { Repository } from "typeorm";
import { Message } from "../entity/Message";

export class MessageService {
  constructor(private readonly messageRepository: Repository<Message>) {}
  async findAll() {
    const messages = await this.messageRepository.find();
    return messages;
  }
  async findOne(id: number) {
    const messages = await this.messageRepository.findOne({ where: { id } });
    return messages;
  }

  async createMessage(newmessage: Message) {
    const message = this.messageRepository.create(newmessage);
    await this.messageRepository.save(message);
    return message;
  }

  async updateMessage(id: number, data: Partial<Message>) {
    const message = await this.messageRepository.findOne({ where: { id } });
    if (message) {
      this.messageRepository.merge(message, data);
      await this.messageRepository.save(message);
      return message;
    } else {
      return { message: "Message not found" };
    }
  }

  async delete(id: number) {
    const message = await this.messageRepository.findOne({ where: { id } });

    if (message) {
      await this.messageRepository.remove(message);
      return { message: "Message Deleted successfully" };
    } else {
      return { message: "Message not found" };
    }
  }
}
```

Вывод

За время работы были сделаны модели, контроллеры, роутеры и сервисы для приложения для аренды недвижимости