САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Домашнее задание 2: Работа с ТуреORM

Выполнил:

Лазебный Всеволод

Группа К3344

Проверил: Добряков Д. И.

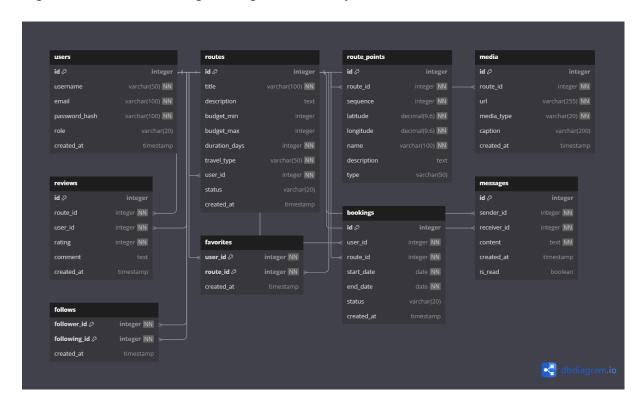
Санкт-Петербург

Задача

- 1. Реализовать все модели данных, спроектированные в рамках Д31
- 2. Реализовать набор из CRUD-методов для работы с моделями данных средствами Express + TypeScript
- 3. Реализовать API-эндпоинт для получения пользователя по id/email

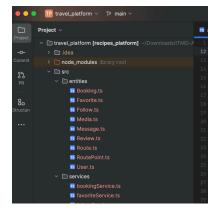
Ход работы

В рамках Д31 была спроектирована следующая схема данных:



В ходе выполнения второй работы я пошел последовательно по каждому пункту.

В папке entities были созданы все модели данных:



Для примера прикладываю скрипт реализации сущности Route:

```
import { Entity, PrimaryGeneratedColumn, Column, ManyToOne, OneToMany } from "typeorm";
import { User } from "./User";
import { RoutePoint } from "./RoutePoint";
import { Media } from "./Review";
import { Review } from "./Review";
import { Eavorite } from "./Review";
import { Booking } from "./Booking";

@Entity() Show usages
export class Route {
@PrimaryGeneratedColumn()
id: number;

decolumn({ length: 100 })
title: string;

@Column('text')
description: string;

@Column('int')
budget_min: number;

@Column('int')
budget_max: number;
```

```
10  export class Route {
26     @Column('int')
27     duration_days: number;
28
29     @Column({ length: 50 })
30     travel_type: string;
31
32     @Column({ default: 'draft' })
33     status: string;
34
45     @Column('timestamp', { default: () :string => 'CURRENT_TIMESTAMP' })
36     created_at: Date;
37
38     @ManyToOne(():typeof User => User, user: User => user.routes)
39     user: User;
40
41     @OneToMany(():typeof RoutePoint => RoutePoint, point:RoutePoint => point.route)
42     points: RoutePoint[];
43
44     @OneToMany(():typeof Media => Media, media:Media => media.route)
45     media: Media[];
46
47     @OneToMany(():typeof Review => Review, review:Review => review.route)
48     reviews: Review[];
```

Аналогично были спроектированы и прочие сущности.

Набор из CRUS-методов находится в папке services, не отходя от Route, прикрепил реализацию routeService

```
import { Route } from "../entities/Route";
        router.get("/", async (req: Request, res: Response) : Promise<void> => {
            const routes : Route[] = await AppDatαSource.getRepository(Route).find({
                relations: ["user", "points", "media"]
            res.json(routes);
        router.get("/:id", async (req: Request, res: Response) : Promise < void> => {
            const route : Route | null | = await AppDataSource.getRepository(Route).findOne({
16 6  1
                where: { id: parseInt(req.params.id) },
                relations: ["user", "points", "media", "reviews"]
        router.post("/", async (req: Request, res: Response) : Promise<void> => {
            const route : Route[] = AppDataSource.getRepository(Route).create(req.body);
            const results : Route[] = await AppDataSource.getRepository(Route).save(route);
         router.post("/", async (req: Request, res: Response) :Promise<void> => {
    res.status(∠⊍⊥).]son(results);
         router.put("/:id", async (req: Request, res: Response) : Promise<void> => {
             const route : Route | null = await AppDataSource.getRepository(Route).findOneBy({
                  id: parseInt(req.params.id)
             if (route) {
                  AppDataSource.getRepository(Route).merge(route, req.body);
                  const results : Route = await AppDαtαSource.getRepository(Route).save(route);
                  res.json(results);
             } else {
                  res.status(404).json({ message: "Route not found" });
         router.delete("/:id", async (req: Request, res: Response) : Promise<void> => {
             const results : DeleteResult = await AppDataSource.getRepository(Route).delete(req.params.id);
             res.json(results);
```

import express, { Request, Response } from "express";

Для получения пользователя по id/email был реализован API-эндпоинт

export default router;

```
import express, { Request, Response } from "express";
       import { AppDataSource } from "../app-data-source";
       import { User } from "../entities/User";
      const router : Router = express.Router();
      router.get("/", async (req: Request, res: Response) : Promise<void> => {
          const users : User[] = await AppDataSource.getRepository(User).find();
          res.json(users);
      router.get("/:id", async (req: Request, res: Response) : Promise < void> => {
           const user : User | null = await AppDataSource.getRepository(User).findOneBy({
14 6
               id: parseInt(req.params.id)
          res.json(user);
      router.get("/email/:email", async (req: Request, res: Response) : Promise<void> => {
          const user : User | null = await AppDataSource.getRepository(User).findOneBy({
               email: req.params.email
          res.json(user);
```

Таким образом, получилось создать следующую структуру:

```
TP travel_platform > 9 main >
                                                                                                                                                     ■ dev ∨ ▷ ₺ :
                                                                                                                                                                                             24 Q 🐯

■ app.ts ×

         travel_platform [recipes_platform] ~/Download
-0-
               node_modules library roo
                                                                                             .then(() : void => console.log("Data Source has been initialized!"))
.catch(err : any => console.error("Error during initialization:", err));

→ entities

                    Booking.ts
                                                                                       const app :Express = express();
app.use(express.json());

■ Favorite.ts

80
                                                                                       app.use("/users", userRouter);
app.use("/routes", routeRouter);
                     Review.ts
                                                                                       app.use("/media", mediαRouter);
app.use("/reviews", reviewRouter);
                     Route.ts
                     RoutePoint.ts
                     s User.ts

    □ services

                                                                                       res.status(200).json({ status: "ok" });
});
                     reviewService.ts
                                                                                       const PORT :string|5000 = process.env.PORT || 5000;
app.listen(PORT, () :void => {
   console.log(`Server running on http://localhost:${PORT}`);
                     s routePointService.ts
                     routeService.ts
                    userService.ts
                  s app-data-source.ts
travel_platform > src > 📧 app.ts
                                                                                                                                                        12:1 Ø ₨ CRLF UTF-8 4 spaces ♂
```

Вывод

В ходе работы научился работать с новыми средствами и освоил ТуреORM. Также составил четкую структуру для дальнейшей работы.