САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Лабораторная работа 1: Реализация boilerplate

Выполнил:

Лазебный Всеволод

Группа К3344

Проверил:
Добряков Д. И.

Санкт-Петербург

2025 г.

Задача

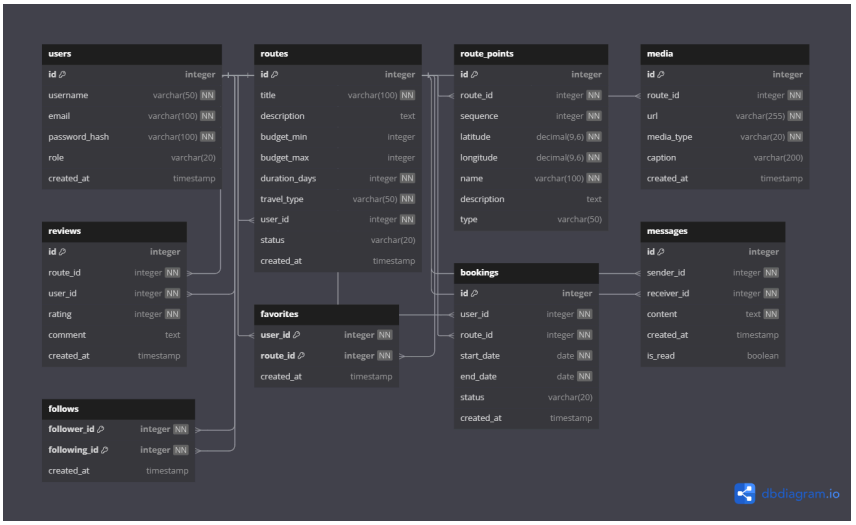
Нужно написать свой boilerplate на express + TypeORM + typescript.

Должно быть явное разделение на:

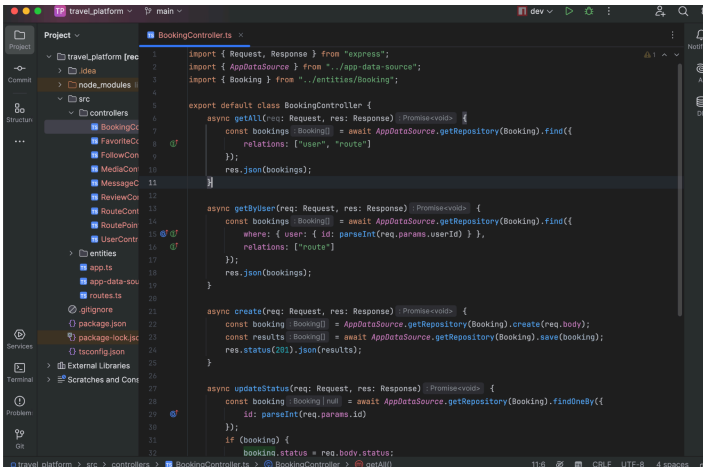
- модели
- контроллеры
- роуты

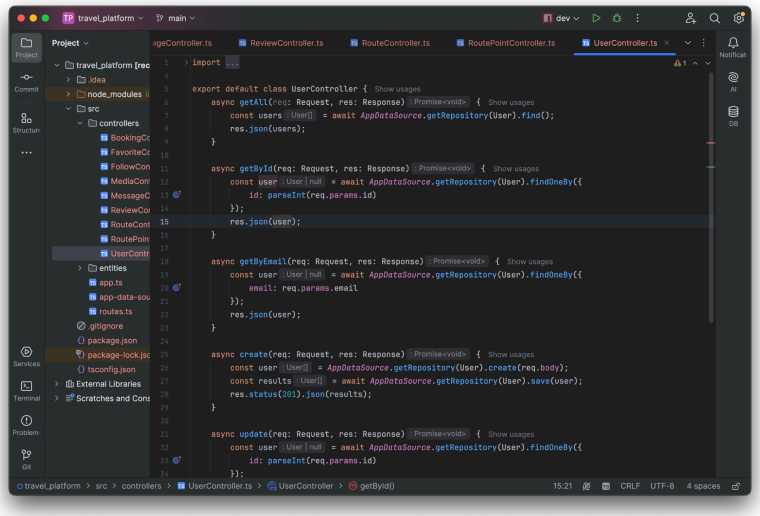
Ход работы

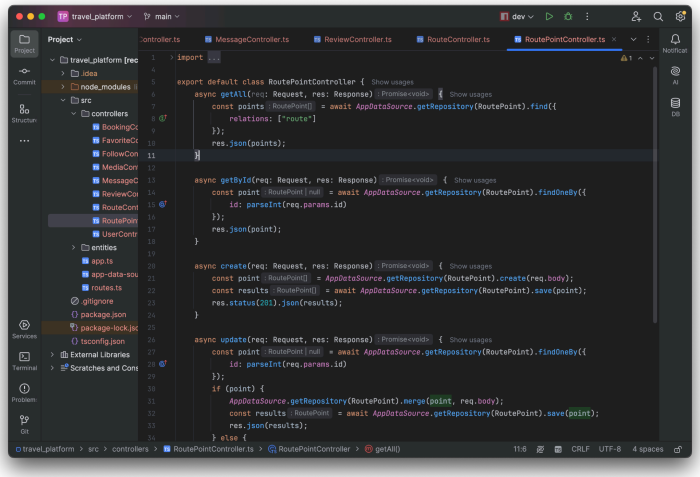
В рамках ДЗ1 была спроектирована следующая схема данных:



Также в рамках ДЗ2 были прописаны все модели и сервисы. Поэтому в этой лабораторной работе я начал работу с контроллеров. Создал соответствующую папку и, основываясь на существующие сервисы, сделал контроллеры. Примеры можно видеть ниже:







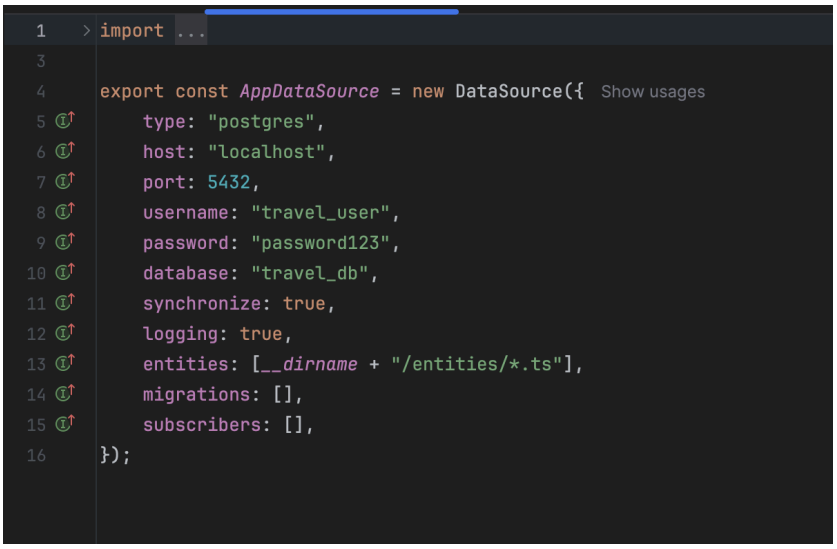
Также были настроены все маршруты:

```
66
67  // Route routes
68  router.get("/routes", routeController.getAll.bind(routeController));
69  router.get("/routes/:id", routeController.getById.bind(routeController));
70  router.post("/routes", routeController.create.bind(routeController));
71  router.put("/routes/:id", routeController.update.bind(routeController));
72  router.delete("/routes/:id", routeController.delete.bind(routeController));
73
74  // RoutePoint routes
75  router.get("/route-points", routePointController.getAll.bind(routePointController));
76  router.get("/route-points/:id", routePointController.getById.bind(routePointController));
77  router.post("/route-points", routePointController.create.bind(routePointController));
78  router.put("/route-points/:id", routePointController.update.bind(routePointController));
79  router.delete("/route-points/:id", routePointController.delete.bind(routePointController));
80
81  // Media routes
82  router.get("/media", mediaController.getAll.bind(mediaController));
83  router.get("/media/:id", mediaController.getById.bind(mediaController));
84  router.post("/media", mediaController.create.bind(mediaController));
```

И также произведены обновления в дополнительных файлах:

```
{
  "name": "travel_platform",
  "version": "1.0.0",
  "description": "Travel platform backend",
  "main": "dist/app.js",
  "scripts": {
    "start": "ts-node src/app.ts",
    "build": "tsc",
    "dev": "nodemon src/app.ts"
  },
  "dependencies": {
    "express": "^4.18.2",
    "pg": "^8.11.3",
    "reflect-metadata": "^0.1.13",
    "typeorm": "^0.3.17"
  },
  "devDependencies": {
    "@types/express": "^4.17.21",
    "@types/node": "^20.11.7",
    "nodemon": "^3.0.3",
    "ts-node": "^10.9.2",
    "typescript": "^5.3.3"
  }
}
```

Также был создан app-data-source:



```
import ...

export const AppDataSource = new DataSource({  Show usages
    type: "postgres",
    host: "localhost",
    port: 5432,
    username: "travel_user",
    password: "password123",
    database: "travel_db",
    synchronize: true,
    logging: true,
    entities: [__dirname + "/entities/*.ts"],
    migrations: [],
    subscribers: [],
});
```

Вывод

В ходе этой лабораторной были реализованы все контроллеры, пути для дополнения структуры из одних моделей. Также был разработан свой boilerplate.