САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

ДЗ 1 - Проектирование базы данных

Выполнил:

Сергеев Виктор

K3341

Проверил:
Добряков Д. И.

Санкт-Петербург

2022 г.

Задача

Требуется реализовать модели спроектированные в ДЗ1 с помощью TypeORM и подготовить проект к дальнейшей работы с Express. Конкретные требования:

- Реализовать все модели данных
- Реализовать набор из CRUD-методов для дальнейшей работы Express
- Реализовать API-эндпоинт для получения пользователя по id

Ход работы

Данный проект будет реализован по архитектуре MVCS. Текущая файловая структура проекта изображена на рисунке 1

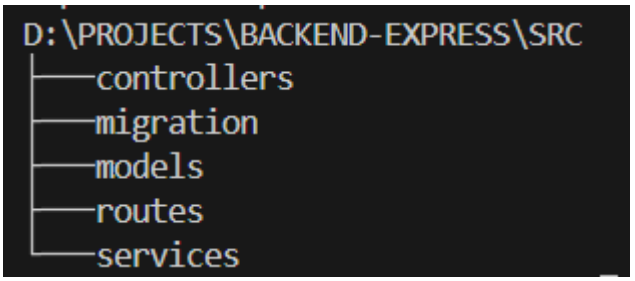

Рисунок 1 - файловая структура проекта

Помимо этого, в директории src находятся файлы index.ts - содержит инициализацию приложения - и data-source.ts - содержит инициализацию настроек базы данных. Назначение каждой директории следующее:

- controllers - содержит контроллеры моделей
- migration - содержит файлы миграции базы данных
- models - содержит описания моделей
- routes - содержит роутеры для каждой модели
- services - содержит сервисы моделей

В этой структуре сервис отвечает за обращения к бд, контроллер для принятие данных, их валидации и взаимодействие с сервисом.

Модели были описаны с помощью TypeORM. На рисунке 2 изображен пример описания модели User - пользователя. Все остальные модели были спроектированы аналогичным образом.

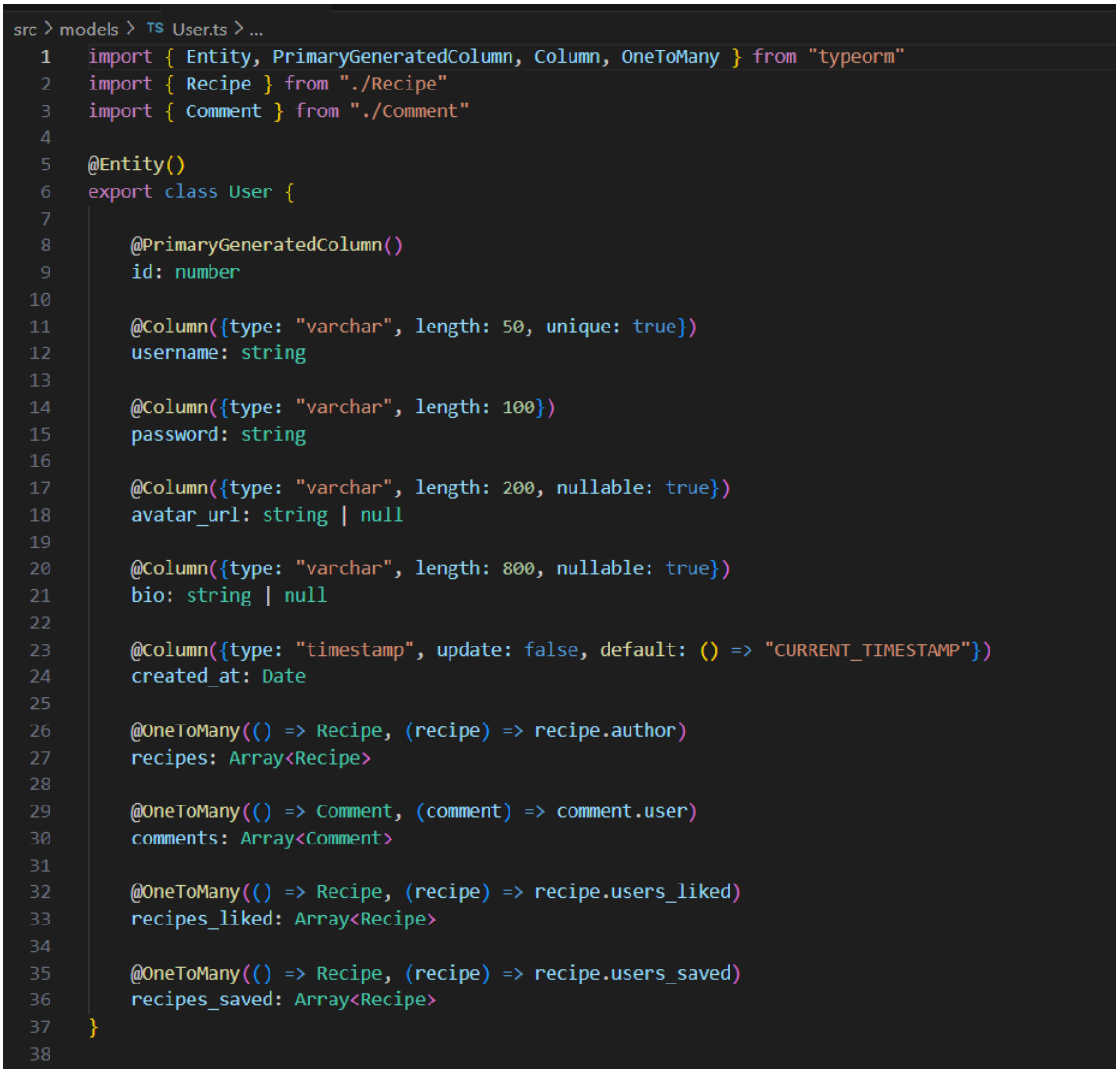

Рисунок 2 - файл User.ts

Далее следовало реализовать CRUD-методы - способ взаимодействовать с базой данных и выполнять основные операции над моделями. Для этого надо было реализовать написать UserService и UserController. На рисунках 3 и 4 представлены примеры UserService и UserController соответственно. Написания сервисов и контроллеров для остальных моделей было выполнено аналогично.

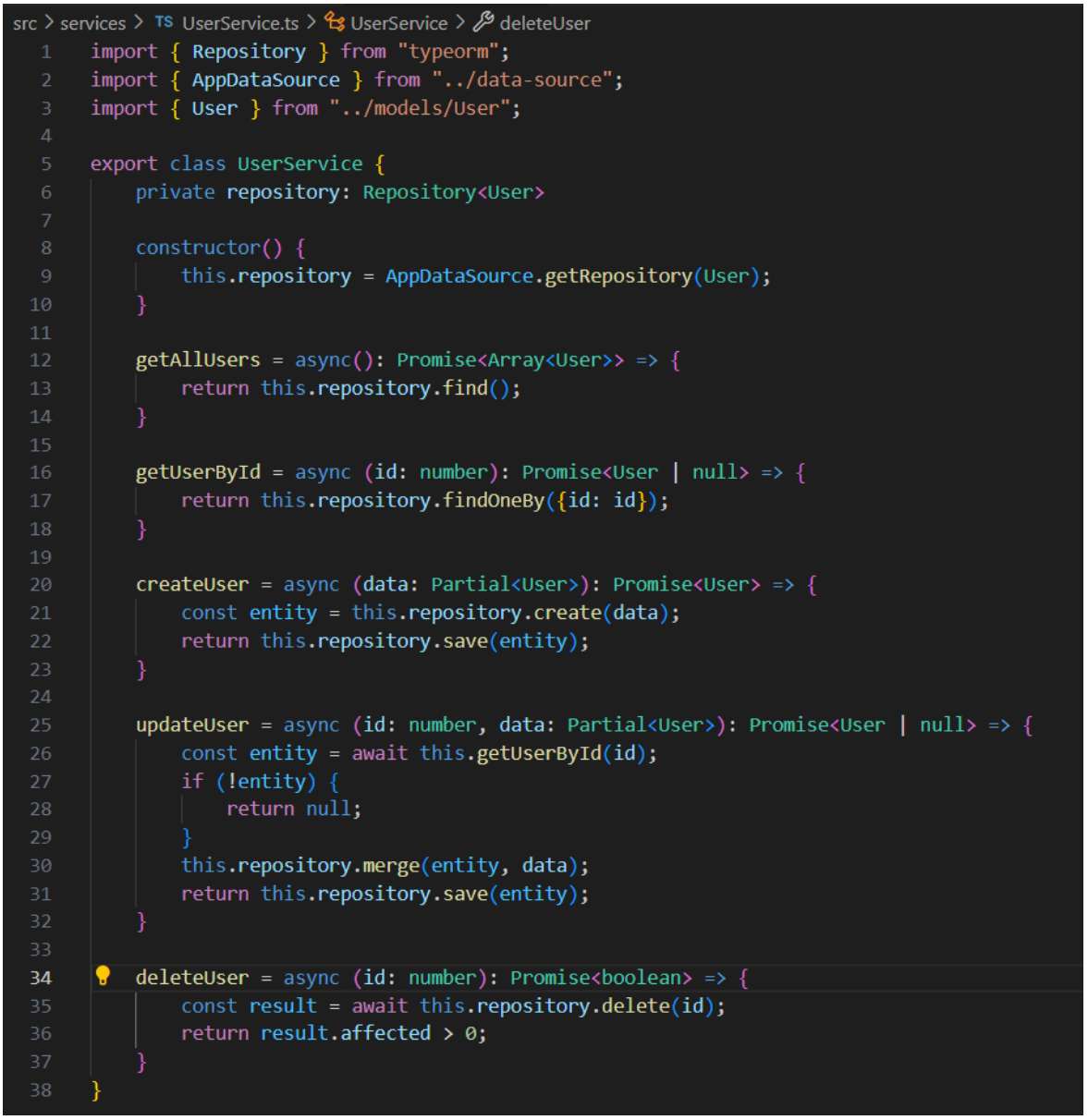

Рисунок 3 - файл UserService.ts

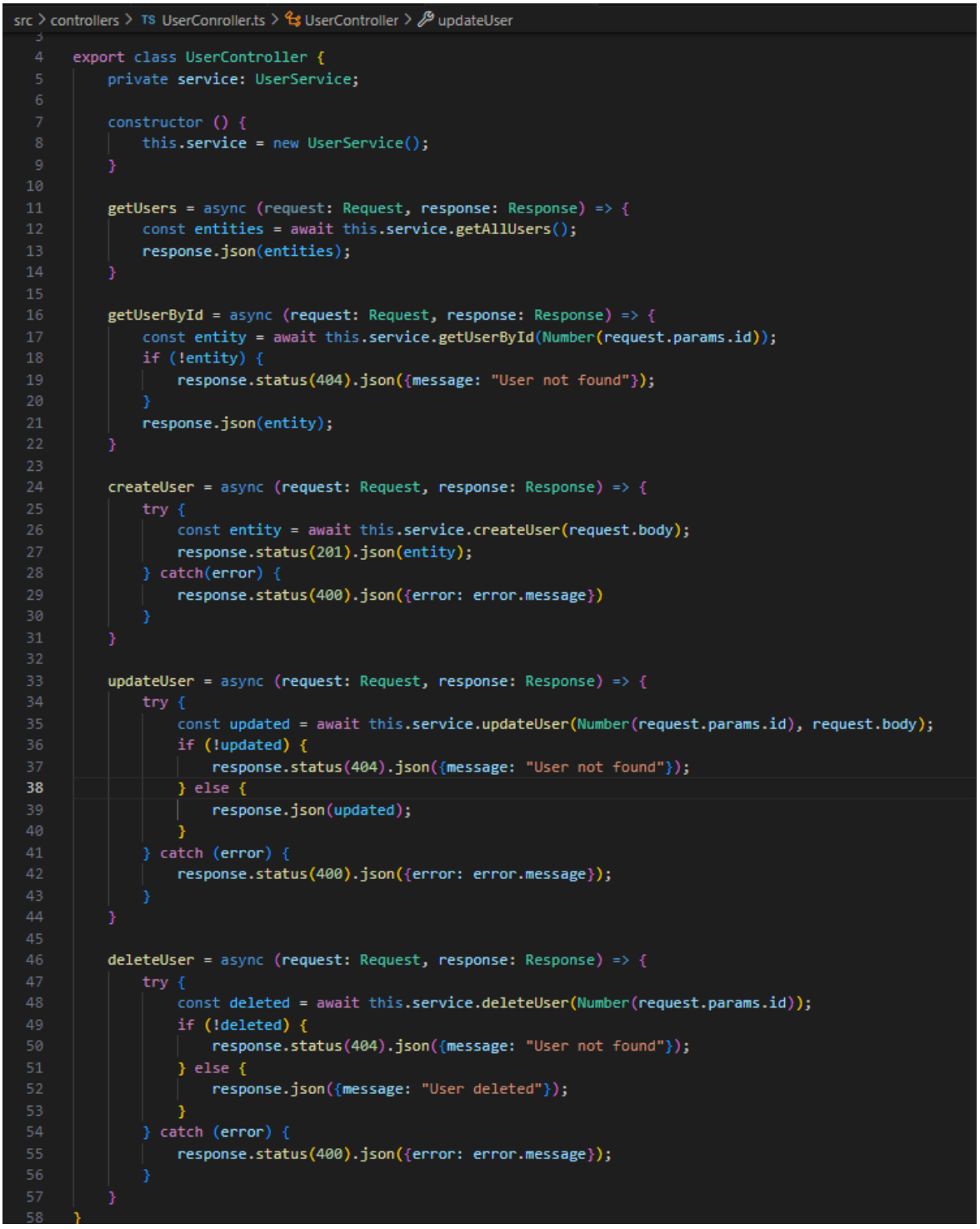

Рисунок 4 - файл UserController.ts

Наконец, был реализован роутер для модели User - UserRouter.ts. Он отвечает за сопоставление методов контроллера URL адресам, по которым идут обращения к серверу. Его код представлен на рисунке 5.

```
src > routes > TS UserRouter.ts > ...
1   import { Router } from "express";
2   import { UserController } from "../controllers/UserConroller";
3
4   const router = Router();
5   const controller = new UserController();
6
7   router.get("/", controller.getUsers);
8   router.get("/:id", controller.getUserById);
9   router.post("/", controller.createUser);
10  router.put("/:id", controller.updateUser);
11  router.delete("/:id", controller.deleteUser);
12
13  export default router;
14
```

Рисунок 5 - файл UserRouter.ts

Наконец, был инициализирован весь проект в файле index.ts. Код представлен на рисунке 6 и на рисунках 7-9 представлена работа api для модели пользователя.

```
src > TS index.ts > ...
1   import "reflect-metadata"
2   import { AppDataSource } from "./data-source"
3   import express = require("express");
4   import bodyParser = require("body-parser");
5   import userRouter from "./routes/UserRouter"
6
7   AppDataSource.initialize()
8   .then(async () => {
9       console.log("db initiated")
10  }).catch(error => console.log(error))
11
12  const app = express();
13  app.use(bodyParser.json())
14  app.use("/api/user", userRouter)
15
16  app.listen(3000);
```

Рисунок 6 - файл index.ts

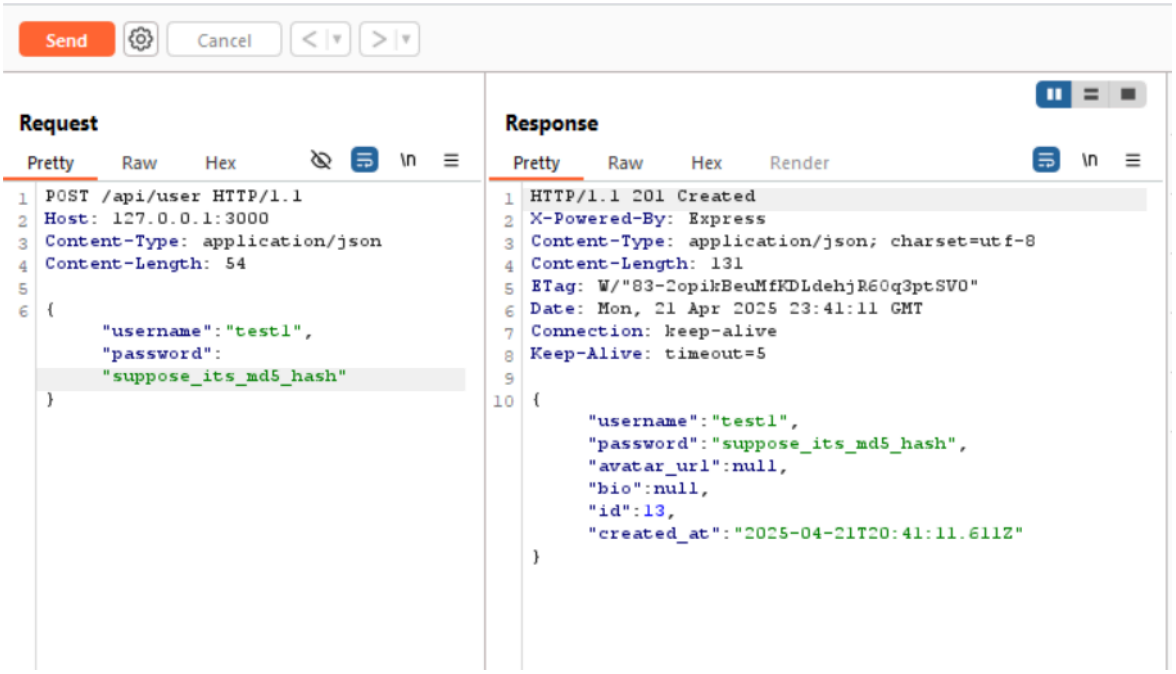

Рисунок 7 - создание пользователя

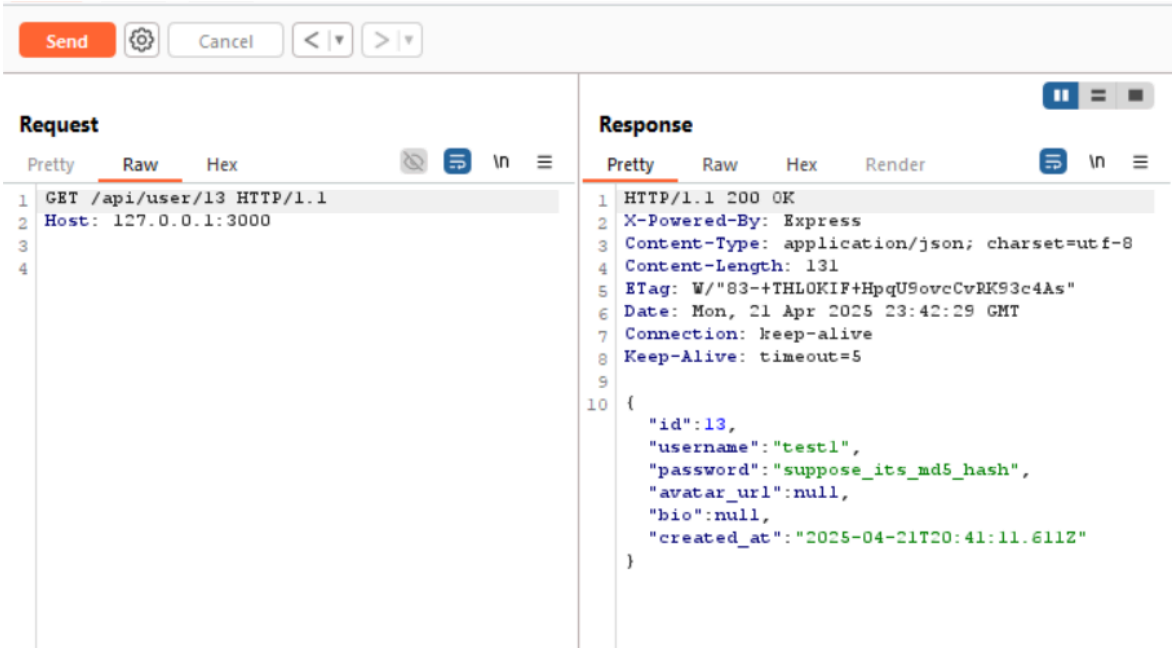

Рисунок 8 - получение пользователя по id

Рисунок 9 - удаление пользователя

Вывод

В процессе работы были реализованы модели базы данных по архитектуре MVCS реализованы сервисы и контроллеры, которые отвечают за взаимодействие с базой данных. Был реализован API для управление моделью User.