САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Лабораторная работа №1

Выполнил:

Шурубова Прасковья

Группа: К3343

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

Модели и сервисы были написаны в рамках ДЗ2, поэтому мы начали прописывать контроллеры. Создали папку Controllers, на основе сервисов написали контроллеры.

Пример userController:

```
import { Request, Response } from "express";
import { AppDataSource } from "../app-data-source";
import { User } from "../entities/User";

export const userController = {
 async getAllUsers(req: Request, res: Response): Promise<void>
{
   const users = await
AppDataSource.getRepository(User).find();
   res.json(users);
 },

 async getUserById(req: Request, res: Response): Promise<void>
{
   const user = await
AppDataSource.getRepository(User).findOneBy({
     user_id: Number(req.params.id),
   });

   if (!user) {
     res.status(404).json({ message: "User not found" });
     return;
   }

   res.json(user);
 },

 async getUserByEmail(req: Request, res: Response):
Promise<void> {
   const user = await
AppDataSource.getRepository(User).findOneBy({
     email: req.params.email,
   });

   if (!user) {
```

```
      res.status(404).json({ message: "User not found" });
      return;
    }

    res.json(user);
  },

  async createUser(req: Request, res: Response): Promise<void>
{
  const data = req.body;

  if (data.user_id && isNaN(Number(data.user_id))) {
    res.status(400).json({ message: "Invalid user_id" });
    return;
  }

  const repo = AppDataSource.getRepository(User);
  const newUser = repo.create(data);
  const result = await repo.save(newUser);
  res.status(201).json(result);
  },

  async updateUser(req: Request, res: Response): Promise<void>
{
    const repo = AppDataSource.getRepository(User);
    const user = await repo.findOneBy({ user_id:
Number(req.params.id) });

    if (!user) {
      res.status(404).json({ message: "User not found" });
      return;
    }

    repo.merge(user, req.body);
    const result = await repo.save(user);
    res.json(result);
  },

  async deleteUser(req: Request, res: Response): Promise<void>
{
    const result = await
AppDataSource.getRepository(User).delete({
      user_id: Number(req.params.id),
    });
    res.json(result);
  }
};
```

Также прописали все маршруты, решили разместить их в одном файле:

```
// Users
router.get("/users", userController.getAllUsers);
router.get("/users/:id", userController.getUserById);
router.get("/users/email/:email", userController.getUserByEmail);
router.post("/users/create", userController.createUser);
router.put("/users/update/:id", userController.updateUser);
router.delete("/users/delete/:id", userController.deleteUser);
// Recipes
router.get("/recipes", recipeController.getAllRecipes);
router.get("/recipes/:id", recipeController.getRecipeById);
router.post("/recipes/create", recipeController.createRecipe);
router.put("/recipes/update/:id", recipeController.updateRecipe);
router.delete("/recipes/delete/:id", recipeController.deleteRecipe);
```

В package.json добавили следующие параметры:

```
{
  "name": "backend",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "start": "tsx src/server.ts"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": "",
  "dependencies": {
    "dotenv": "^16.5.0",
    "express": "^5.1.0",
    "pg": "^8.15.6",
    "tsx": "^4.19.4",
    "typeorm": "^0.3.22"
  },
  "devDependencies": {
    "@types/express": "^5.0.1",
    "@types/node": "^22.15.12"
  }
```

start - для запуска сервера с автоперезапуском.

Также был создан app-data-source.ts:

```
import { DataSource } from "typeorm";
import 'dotenv/config';

const AppDataSource = new DataSource({
   type: 'postgres',
   host: process.env.DB_HOST,
   port: parseInt(process.env.DB_PORT || '5432'),
   username: process.env.DB_USERNAME,
```

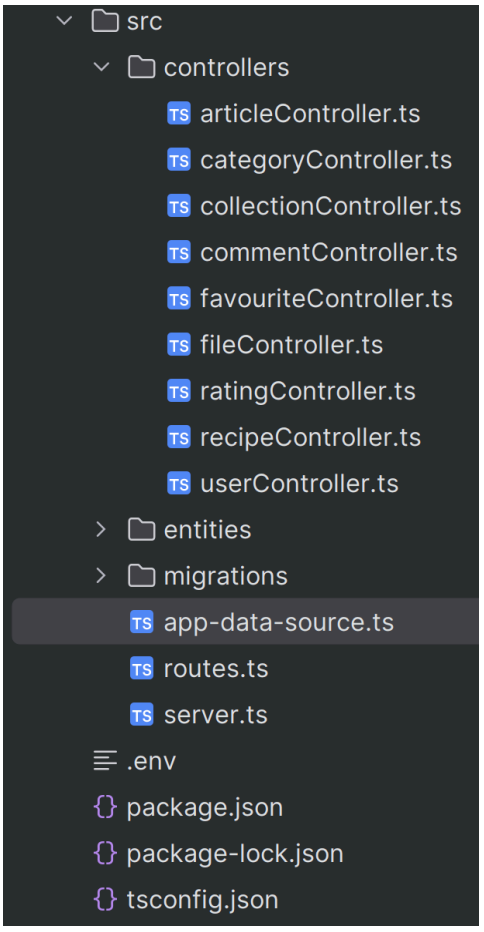
```
    password: process.env.DB_PASSWORD,
    database: process.env.DB_NAME,
    entities: ["src/entities/*.ts"],
    logging: true,
    synchronize: false,
    migrations: ["src/migrations/*.ts"],
    subscribers: [],
});

export default AppDataSource;
```

synchronize: для синхронизации схемы БД.

logging: логирование SQL-запросов.

Информация о параметрах записана в .env.

Структура нашего проекта:



Вывод: в ходе работы была разделена бизнес-логика нашего проекта: были созданы контроллеры, маршруты, параметры для подключения к базе данных. Проект готов к дальнейшему расширению.