САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Лабораторная работа №1

Выполнил:

Кузнецов Артур

Группа К3340

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

Сначала я инициализировал проект и установил базовые зависимости. После этого через `npx tsc --init` создал `tsconfig.json`, включил там поддержку декораторов и указал папки `src` и `dist` для исходников и собранных файлов. В файле `package.json` были настроены скрипты: `dev` для запуска в режиме разработки, `build` для компиляции TypeScript и `start` для запуска скомпилированного приложения. Для форматирования кода был добавлен файл `.prettierrc` с настройками, чтобы код выглядел аккуратно и единообразно.

Затем добавил в корень файл `.env`, в котором прописал параметры подключения к базе: хост, порт, имя пользователя, пароль и имя базы, а также секрет для JWT. Дальше в `src/config/database.ts` подключил `dotenv.config()` и настроил DataSource TypeORM.

В папке `src/models` определил модель User. Эта модель описывает пользователя с полями: идентификатор, email, пароль, имя, фамилия, фото профиля, номер телефона, пол, биография и дата создания. Поля были настроены с учётом типов данных и ограничений, например, email должен быть уникальным, а пароль обязательным. После этого в `src/controllers` описал простейшие CRUD-функции для модели пользователя. Каждая функция получает в аргументах `Request` и `Response`, обрабатывает ошибки и возвращает нужные HTTP-коды.

Далее в `src/routers` связал URL-пути с контроллерами: создал `userRouter`, куда вынес все эндпойнты, и подключил его в точке входа `src/app.ts`. В `app.ts` также импортировал `reflect-metadata`, включил JSON-парсер и запустил сервер после успешного подключения к БД.

Наконец, проверил работу шаблона: через Postman отправил запросы на создание, получение, обновление и удаление записи в таблице users и убедился, что все операции корректно выполняются.

Вывод

В ходе лабораторной работы был создан готовый boilerplate для API на основе Express + TypeORM + TypeScript. Полученный шаблон обладает модульной структурой с разделением на модели, контроллеры и маршруты, содержит конфигурацию подключения к PostgreSQL через `.env` и позволяет быстро развивать проект: добавлять новые сущности, расширять контроллеры и защищать маршруты, не тратя время на повторную настройку окружения.