САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Лабораторная работа

Выполнила:

Гусейнова Марьям

БР 1.2

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 данной работы был создан базовый boilerplate для веб-приложения, использующего Express.js, TypeORM и TypeScript.

Работа началась с инициализации проекта и установки необходимых зависимостей.

Далее была определена структура проекта, включающая разделение на директории для моделей (models), контроллеров (controllers) и маршрутов (routes).

Затем была создана модель User с определением полей (имя, фамилия, email, хеш пароля).

После этого был разработан контроллер user.controller с базовыми методами для выполнения CRUD-операций (получение всех пользователей, получение пользователя по ID, создание пользователя, обновление пользователя и удаление пользователя).

Были определены маршруты в файле user.routes.ts и связаны с соответствующими методами контроллера.

Для настройки подключения к базе данных был создан файл data-source.ts, в котором были определены параметры подключения. Также была был использован .env файл для хранения конфигурационных переменных. Пример такого файла можно найти в .env.example.

Ключевым элементом является файл index.ts, который служит точкой входа в приложение. В этом файле выполняются следующие основные действия:

- Импортируются необходимые модули: express для создания веб-сервера, userRoutes для подключения маршрутов, и

AppDataSource для инициализации подключения к базе данных TypeORM.
- Создается экземпляр приложения Express.
- Определяется порт, на котором будет запущен сервер (с использованием переменной окружения PORT из .env файла или значения по умолчанию).
- Используется middleware express.json() для обработки входящих запросов с телом в формате JSON.
- Подключаются определенные ранее маршруты из userRoutes по корневому пути (/).
- Происходит инициализация подключения к базе данных TypeORM с помощью AppDataSource.initialize(). После успешного подключения запускается HTTP-сервер Express, который начинает прослушивать входящие запросы на указанном порту.

Завершающим этапом было добавление системы авторизации пользователей с использованием JWT (JSON Web Token).
Были реализованы эндпоинты регистрации и логина. При регистрации создаётся новый пользователь, и его пароль сохраняется в базе в зашифрованном виде с использованием bcrypt. При логине происходит проверка email и пароля. В случае успешной аутентификации генерируется JWT-токен, содержащий идентификатор пользователя и срок действия (1 час), который возвращается клиенту.

Для защиты маршрутов был реализован middleware authMiddleware, который:

- Проверяет наличие и корректность токена в заголовке Authorization.
- Расшифровывает токен и получает из него идентификатор пользователя.
- Производит запрос к базе данных, чтобы убедиться в существовании пользователя.
- Если проверка проходит успешно, пользователь сохраняется в объекте запроса и передаётся дальше по цепочке обработки.

Благодаря этому токены становятся безопасными: если пользователь будет удалён, старый токен перестанет работать.

Вывод

Таким образом, в рамках лабораторной работы был создан полностью работоспособный шаблон (boilerplate) серверного приложения с авторизацией, CRUD для пользователей и корректной структурой проекта на базе Express.js, TypeORM и TypeScript.