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
По выбранному варианту необходимо реализовать RESTful API средствами express + typescript (используя ранее написанный boilerplate).

Ход работы

Мой вариант:

Платформа для фитнес-тренировок и здоровья

- Вход
- Регистрация
- Личный кабинет пользователя (трекинг прогресса, планы тренировок)
- Поиск тренировок с фильтрацией по уровню, типу (кардио, силовые) и продолжительности
- Страница тренировки с видео, описанием и инструкциями
- Блог о здоровье и питании

Для выполнения работы был взят ранее написанный boilerplate.

Функциональность приложения была значительно расширена путем добавления новых сущностей и соответствующих им CRUD-операций:

- BlogPost: модель для создания и управления записями в блоге.
- Exercise: модель для описания различных упражнений.
- Workout: модель, представляющая собой одну тренировку.
- WorkoutPlan: модель для составления полноценных планов тренировок.
- ExerciseWorkout: модель, связывающая упражнения с тренировками.
- WorkoutInPlan: модель, связывающая тренировки с планами тренировок.
- CurrentProgress: модель для отслеживания общего прогресса пользователя.
- ExerciseProgress: модель для отслеживания прогресса пользователя по конкретным упражнениям.
- PlanProgress: модель для отслеживания прогресса пользователя по планам тренировок.

Для каждой из этих моделей были реализованы:

- Сервисы (*.service.ts): отдельный слой бизнес-логики, отвечающий за взаимодействие с базой данных и обработку данных.
- Контроллеры (*.controller.ts): обработчики запросов, взаимодействующие с сервисами и отправляющие ответы клиенту.
- Маршруты (*.routes.ts): определены HTTP-эндпоинты для каждой сущности, включая стандартные CRUD-операции.
- Для некоторых моделей были созданы enum-перечисления (папка src/enums).

Были добавлены кастомные методы для получения данных, связанных с текущим пользователем:

- getMyCurrentProgress (для CurrentProgress).
- getAllMyExerciseProgresses и getMyExerciseProgressesByExercise (для ExerciseProgress).
- getAllMyPlanProgresses и getMyPlanProgressesByPlan (для PlanProgress).
- getAllMyWorkoutPlans (для WorkoutPlan).

Реализована функция поиска и фильтрации планов тренировок (searchWorkoutPlans) по таким критериям, как уровень сложности (level), тип тренировки (type), минимальная и максимальная длительность (durationMin, durationMax).

В процессе разработки были решены ключевые проблемы, связанные с типизацией и корректной работой Express.js и TypeScript:

- Проблема с req.userId: изначально TypeScript не распознавал свойство userId на объекте Request после добавления его в authMiddleware. Эта проблема была решена путем расширения стандартного интерфейса Request из Express.js через создание декларационного файла src/types/express.d.ts и соответствующей настройки tsconfig.json.
- Проблема типизации контроллеров (No overload matches this call): некоторые методы контроллеров, использующие return res.status().json() для раннего завершения ответа, вызывали ошибки типизации в VS Code, так как их возвращаемый тип (Promise<Response | undefined>) не соответствовал ожидаемому Express.js (void | Promise<void>). Это было исправлено путем

преобразования методов контроллеров в стрелочные функции без явного указания возвращаемого типа, что позволяет TypeScript корректно выводить его. Для случаев, когда требовалось явное утверждение типа, использовалось приведение as RequestHandler в файлах роутов.

Вывод

Таким образом, в рамках данной лабораторной работы было реализовано RESTful API средствами express + typescript, используя ранее написанный boilerplate.