

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

Дисциплина: Бэк-энд разработка

Отчет

Домашняя работа №3
Документирование API

Выполнил:
Даньшин Семён
К3340

Проверил:
Добряков Д. И.

Санкт-Петербург

2025 г.

Задача

Реализовать автодокументирование средствами swagger и документацию API средствами Postman.

Ход работы

1. Реализация Swagger документации

В проекте используется автоматическая генерация Swagger документации из protobuf файлов с помощью protoc-gen-openapi2.

Генерация OpenAPI спецификации:

```
protoc --openapi2_out=./api \
  --openapi2_opt=logtostderr=true \
  --openapi2_opt=json_names_for_fields=false \
  api/workouts/workouts.proto
```

2. Структура API документации

Swagger документация включает следующие разделы:

Основная информация:

```
title: Fitness Trainer API
version: 1.0.0
baseUrl: /api
description: API for fitness Trainer service
```

Сервисы и эндпоинты:

AuthService - аутентификация:

- POST /v1/auth/login - вход в систему
- POST /v1/auth/refresh - обновление токена
- POST /v1/auth/logout - выход из системы

UserService - управление пользователями:

- GET /v1/users/me - получение текущего пользователя
- POST /v1/users - создание пользователя
- PUT /v1/users - обновление данных пользователя
- GET /v1/users/{userId} - получение пользователя по ID

ExerciseService - управление упражнениями:

- GET /v1/exercises - получение списка упражнений
- POST /v1/exercises - создание нового упражнения
- GET /v1/exercises/{exerciseId} - получение деталей упражнения

- GET /v1/exercises/{exerciseId}/alternatives - альтернативные упражнения
- GET /v1/exercises/{exerciseId}/history - история выполнения

RoutineService - управление планами тренировок:

- GET /v1/routines - получение списка рутин
- POST /v1/routines - создание новой рутины
- GET /v1/routines/{routineId} - получение деталей рутины
- PUT /v1/routines/{routineId} - обновление рутины
- DELETE /v1/routines/{routineId} - удаление рутины

WorkoutService - управление тренировками:

- GET /v1/workouts - получение списка тренировок
- POST /v1/workouts - начало новой тренировки
- GET /v1/workouts/active - активные тренировки
- GET /v1/workouts/{workoutId} - получение тренировки
- POST /v1/workouts/{workoutId}/complete - завершение тренировки

3. Модели данных в Swagger

Документация включает полное описание всех моделей данных:

Пример модели User:

```
WorkoutUser:
  type: object
  properties:
    id:
      type: string
    createdAt:
      type: string
      format: date-time
    email:
      type: string
    firstName:
      type: string
    lastName:
      type: string
    height:
      type: number
      format: float
    weight:
      type: number
      format: float
```

4. Безопасность в документации

Документированы методы аутентификации:

- Bearer token authentication
- Refresh token mechanism
- Session management

5. Postman коллекция

Создана полная Postman коллекция с примерами запросов:

Структура коллекции:

```
Fitness Trainer API/  
├── Auth/  
│   ├── Login  
│   ├── Refresh Token  
│   └── Logout  
├── Users/  
│   ├── Get Current User  
│   ├── Create User  
│   ├── Update User  
│   └── Get User by ID  
├── Exercises/  
│   ├── Get Exercises  
│   ├── Create Exercise  
│   ├── Get Exercise Details  
│   └── Get Exercise History  
├── Routines/  
│   ├── Get Routines  
│   ├── Create Routine  
│   ├── Get Routine Details  
│   └── Update Routine  
└── Workouts/  
    ├── Start Workout  
    ├── Get Active Workouts  
    ├── Get Workout Details  
    └── Complete Workout
```

6. Примеры запросов в Postman

Создание пользователя:

```
POST /v1/users  
{  
  "email": "user@example.com",  
  "password": "securepassword",  
  "firstName": "John",  
  "lastName": "Doe"  
}
```

Начало тренировки:

```
POST /v1/workouts  
{  
  "routineId": "uuid-here",  
  "generateWorkout": false  
}
```

7. Переменные окружения в Postman

Настроены переменные для разных окружений:

- `baseUrl` - базовый URL API

- `accessToken` - токен доступа
- `userId` - ID текущего пользователя

8. Автотесты в Postman

Добавлены базовые автотесты для проверки:

- Статус кодов ответов
- Структуры ответов
- Наличия обязательных полей

ExerciseService			^
GET	/v1/exercises	Метод для получения списка всех упражнений	🔒
POST	/v1/exercises	Метод для создания нового упражнения	🔒
GET	/v1/exercises/{exerciseId}	Метод для получения деталей об упражнении	🔒
GET	/v1/exercises/{exerciseId}/alternatives	Метод для получения альтернативных упражнений по exercise_id	🔒
GET	/v1/exercises/{exerciseId}/history	Метод для получения истории выполнения упражнения	🔒
GET	/v1/muscle_groups	Метод для получения списка групп мышц	🔒
RoutineService			^
GET	/v1/routines	Метод для получения списка доступных пользователю рутин	🔒
POST	/v1/routines	Создание новой рутины	🔒
GET	/v1/routines/{routineId}	Получение информации о рутине по ID	🔒
DELETE	/v1/routines/{routineId}	Удаление рутины по ID	🔒
PUT	/v1/routines/{routineId}	Обновление рутины по ID	🔒
POST	/v1/routines/{routineId}/exercise_instances/order	Метод для установки порядка упражнений в рутине	🔒
GET	/v1/routines/{routineId}/exercise_instances/{exerciseInstanceId}	Получить информацию об упражнении в рутине	🔒
DELETE	/v1/routines/{routineId}/exercise_instances/{exerciseInstanceId}	Удаление упражнения из рутины	🔒
POST	/v1/routines/{routineId}/exercise_instances/{exerciseInstanceId}/sets	Метод для добавления сета в упражнение	🔒
DELETE	/v1/routines/{routineId}/exercise_instances/{exerciseInstanceId}/sets/{setId}	Метод для удаления сета из упражнения	🔒
PUT	/v1/routines/{routineId}/exercise_instances/{exerciseInstanceId}/sets/{setId}	Метод для обновления сета в упражнении	🔒
POST	/v1/routines/{routineId}/exercises	Добавление упражнения в рутину	🔒

WorkoutService			^
GET	/v1/workouts	Метод для получения списка всех тренировок	🔒
POST	/v1/workouts	Метод для начала новой тренировки	🔒
GET	/v1/workouts/active	Метод для получения списка активных тренировок	🔒
GET	/v1/workouts/{workoutId}	Метод для получения состояния тренировки	🔒
DELETE	/v1/workouts/{workoutId}	Удалить тренировку	🔒
POST	/v1/workouts/{workoutId}/comment	Метод для добавления комментария к тренировке	🔒
POST	/v1/workouts/{workoutId}/complete	Метод для завершения тренировки	🔒
POST	/v1/workouts/{workoutId}/log/exercise	Метод для создания записи о выполнении упражнения	🔒
GET	/v1/workouts/{workoutId}/log/exercise/{exerciseLogId}	Метод для получения записи о выполнении упражнения	🔒
DELETE	/v1/workouts/{workoutId}/log/exercise/{exerciseLogId}	Удалить запись о выполнении упражнения	🔒
POST	/v1/workouts/{workoutId}/log/exercise/{exerciseLogId}/notes	Добавление заметки к выполнению упражнения	🔒
POST	/v1/workouts/{workoutId}/log/exercise/{exerciseLogId}/power_rating	Добавить оценку усилий при выполнении упражнения	🔒
POST	/v1/workouts/{workoutId}/log/exercise/{exerciseLogId}/set	Метод для создания записи о выполнении подхода	🔒
DELETE	/v1/workouts/{workoutId}/log/exercise/{exerciseLogId}/set/{setId}	Удалить запись о выполнении подхода	🔒
PUT	/v1/workouts/{workoutId}/log/exercise/{exerciseLogId}/set/{setId}	Метод для изменения записи о выполнении подхода	🔒
POST	/v1/workouts/{workoutId}/rate	Метод для установки оценки тренировки	🔒
GET	/v1/workouts/{workoutId}/report	Метод для получения отчета о тренировке	📄 🔒
UserService			^
POST	/v1/users	Метод для создания пользователя	🔒
PUT	/v1/users	Метод для обновления данных пользователя	🔒
GET	/v1/users/me	Метод для получения текущего пользователя	🔒
GET	/v1/users/workout_generation_settings	Метод для получения настроек генерации тренировок	🔒
PUT	/v1/users/workout_generation_settings	Метод для обновления настроек генерации тренировок	🔒
GET	/v1/users/{userId}	Метод для получения пользователя по ID	🔒
AuthService			^
POST	/v1/auth/login		🔒
POST	/v1/auth/logout		🔒
POST	/v1/auth/refresh		🔒
FileService			^
POST	/v1/files/presign		🔒

9. Доступ к документации

- Swagger UI доступен по адресу: `/api/docs`
- OpenAPI спецификация: `/api/openapi.json`
- Postman коллекция экспортирована в формате JSON

Вывод

Реализована полная документация API проекта:

1. **Автоматическая генерация** Swagger документации из protobuf схем
2. **Полное покрытие** всех эндпоинтов с описанием параметров и ответов
3. **Детальное описание** моделей данных и их валидации
4. **Postman коллекция** с примерами всех запросов и автотестами
5. **Настройка окружений** для разработки и продакшена
6. **Документирование безопасности** и методов аутентификации

Документация обеспечивает удобную разработку и интеграцию с API, а также служит справочным материалом для разработчиков.