

Enunciado de Práctica: Backend API Restful para "Gym Tracker"

1. Descripción del Proyecto

El objetivo de esta práctica es desarrollar el **backend** de una aplicación "Full Stack" para el seguimiento de rendimiento deportivo utilizando **Laravel 12**. El sistema debe gestionar datos jerárquicos (categorías de ejercicios) y una arquitectura de relaciones compleja donde **los usuarios comparten y se suscriben a rutinas**, exponiendo una API robusta y segura.

2. Requisitos Técnicos Mínimos (Obligatorios)

Para considerar la práctica apta para evaluación, se deben implementar los siguientes elementos estructurales:

2.1. Configuración e Infraestructura

- Inicializar un proyecto en Laravel 12.
- Ejecutar `php artisan install:api` para configurar **Laravel Sanctum**.
- Configurar la autenticación basada en tokens.

2.2. Diseño de la Base de Datos (Migraciones)

Se deben crear las migraciones necesarias para generar el siguiente esquema relacional. Aunque Laravel proporciona una migración por defecto para usuarios, se debe verificar/modificar para cumplir con la descripción:

1. **users:** (automática al instalar Sanctum)
 - Debe contener: id, name, email (único), email_verified_at (nullable), password y remember_token.
 - Esta tabla gestionará la autenticación y la propiedad de las suscripciones a rutinas.
 - También tiene que implementar los métodos para crear los tokens.
2. **categories:**
 - Debe incluir id, name (único) y icon_path (string).
3. **exercises:**
 - Unidad atómica del entrenamiento.
 - Campos: id, category_id (FK), name, instruction (text).
4. **routines:**
 - Definición de la rutina.
 - Campos: id, name (string, obligatorio), description (opcional).
5. **exercise_routine (Tabla Pivote 1):**
 - Define qué ejercicios componen una rutina.

- FKs: exercise_id, routine_id.
- Atributos extra (Pivot): sequence (orden), target_sets, target_reps, rest_seconds.

6. **routine_user (Tabla Pivot 2):**

- Implementa la relación **N:M** entre usuarios y rutinas (un usuario tiene muchas rutinas, una rutina puede ser usada por muchos usuarios).
- FKs: user_id, routine_id.

2.3. Modelado (Eloquent ORM)

Implementar los Modelos con sus relaciones correctas:

- **User:** Relación belongsToMany hacia Routine.
- **Routine:**
 - Relación belongsToMany hacia User.
 - Relación belongsToMany hacia Exercise (con withPivot para los datos de series/reps).
- **Category:** RelaciónhasMany hacia Exercise.
- **Exercise:** Relación belongsTo hacia Category.

2.4. Controladores y Rutas

Definir endpoints en routes/api.php para la gestión de datos.

- GET /api/routines: Debe devolver las rutinas asociadas al usuario autenticado (filtrando por la tabla pivot routine_user).
- POST /api/routines:
 1. Crea la rutina en la tabla routines con su nombre.
 2. Asocia (attach) la rutina al usuario actual en routine_user.
 3. Asocia (attach) los ejercicios indicados en exercise_routine.

3. Requisitos

Para superar la evaluación básica, el alumno debe implementar **al menos dos** de los siguientes componentes:

A. API Resources (Transformación de Datos)

Uso de **Eloquent Resources** para estructurar la respuesta JSON.

- **Reto:** El endpoint de detalle de rutina debe devolver un objeto que incluya el nombre de la rutina, y una lista de ejercicios donde cada ejercicio muestre sus datos básicos y los datos de la pivot (sets, reps) al mismo nivel de jerarquía para facilitar el consumo en el frontend.

B. Validación Robusta (Form Requests)

Implementar StoreRoutineRequest.

- Validar que el campo name de la rutina es obligatorio y string.
- Validar que el array de ejercicios no está vacío.
- **Validación de integridad:** Asegurar que los IDs de los ejercicios enviados existen realmente en la base de datos (exists:exercises,id).

C. Seeding y Factories Complejos

Generar un escenario de prueba realista en DatabaseSeeder:

- Crear usuarios de prueba.
- Crear categorías musculares (Pecho, Espalda, Pierna).
- **Lógica de Seed:** Al crear una rutina mediante factory, esta debe:
 1. Asignarse automáticamente a varios usuarios aleatorios (llenar routine_user).
 2. Tener ejercicios asignados con series y repeticiones aleatorias (llenar exercise_routine).

4. Especificación de la API (Resumen)

(Realiza al menos las que están el verde, las otras son opcionales pero necesarias si se quiere hacer la parte del cliente completa)

Verbo	Ruta	Parámetros (URL)	Cuerpo (JSON)	Acceso	Descripción
POST	/login	-	email, password	Público	Autentica al usuario y devuelve el token.
POST	/register	-	name, email, password, ...	Público	Crea una cuenta de usuario nueva.
POST	/logout	-	-	Token	Invalida el token actual del usuario.
GET	/categories	-	-	Público	Lista todas las categorías (ej: Cardio, Fuerza).
GET	/categories/{id}	id_category	-	Público	Detalle de una categoría específica.
POST	/categories	-	name	Token	Crea una nueva categoría.
PUT	/categories/{id}	id_category	name	Token	Edita una categoría existente.
DELETE	/categories/{id}	id_category	-	Token	Borra una categoría.
GET	/categories/{id}/exer	id_category	-	Público	Lista ejercicios de esa

	cises				categoría.
GET	/exercises	-	-	Público	Lista todos los ejercicios del sistema.
GET	/exercises/{id}	id_exercises	-	Público	Detalle de un ejercicio concreto.
POST	/exercises	-	name, desc, cat_id	Token	Crea un ejercicio nuevo.
PUT	/exercises/{id}	id_exercises	name, ...	Token	Modifica un ejercicio.
DELETE	/exercises/{id}	id_exercises	-	Token	Elimina un ejercicio.
GET	/routines	-	-	Público	Lista todas las rutinas públicas.
GET	/routines/{id}	id_routine	-	Público	Detalle completo de una rutina.
POST	/routines	-	name, descriptio n	Token	Crea una rutina
PUT	/routines/{id}	id_routine	name, ...	Token	Edita una rutina
DELETE	/routines/{id}	id_routine	-	Token	Borra una rutina
GET	/routines/{id}/exerci ses	id_routine	-	Público	Muestra los ejercicios que componen una rutina.
POST	/routines/{id}/exerci ses	id_routine	exercise_i d, reps, sets	Token	Añade un ejercicio a una rutina con los datos adicionales como repeticiones, número de series, descanso, etc.
DELETE	/routines/{id}/exerci ses/{e_id}	id_routine, id_exercise	-	Token	Quita un ejercicio de una rutina específica.
GET	/my-routines	-	-	Token	Lista las rutinas creadas por el usuario logueado.
POST	/my-routines	-	routine_id	Token	Suscribe al usuario a una rutina existente.
DELETE	/my-routines{id}	id_routine		Token	Desuscribe al usuario de una rutina.

5. Entregables

1. Repositorio de código.
2. Defensa del proyecto.