Proyecto: App de Acondicionamiento Físico Politécnico Salesiano “Arquides Calderón “

Implementado por: Magister Adaury Paulino

Coordinación: Magister Bernardo Hernández (Informática) y el Lic. Humberto Cruz (Acondicionamiento Físico)

Equipos (colores): Rojo (Auth & Perfiles), Azul (Rutinas & Ejercicios), Amarillo (Gimnasios & Entrenadores), Verde (Seguimiento & Analitico), **Negro** (Integración, DevOps & QA).

1. DOCUMENTO DE PROYECTO (RESUMEN, ALCANCE Y ENTREGABLES)

1.1 Objetivo

Desarrollar una aplicación móvil multiplataforma (Android/iOS) para usuarios de gimnasios en la ciudad de Moca que permita registro biométrico, creación y seguimiento de rutinas, gestión de gimnasios y entrenadores, y análisis de progreso.

1.2 Alcance inicial

- Registro y autenticación de usuarios.
- Perfil con datos: edad, sexo, peso, talla, % grasa, nivel, objetivo.
- Biblioteca de ejercicios y creación de rutinas.
- CRUD de gimnasios y entrenadores (foco Moca).
- Registro de sesiones y métricas (peso/% grasa/reps).
- Panel admin web para gestionar datos.

1.3 Entregables por equipo

- **Rojo:** Módulo Auth + pantalla perfil + documentación de endpoints de usuario.

- **Azul:** Biblioteca de ejercicios, motor de rutinas, UI para rutinas.
- **Amarillo:** CRUD de gimnasios/entrenadores, lista/mapa Moca.
- **Verde:** Registro de sesiones, métricas, gráficas.
- **Negro:** OpenAPI, Docker Compose, CI/CD, despliegue staging y pruebas.

2. OPENAPI INICIAL (YAML) — ENDPOINTS PRINCIPALES

```
openapi: 3.0.1
info:
 title: FitMoca API
 version: 0.1.0
servers:
 - url: https://api.fitmoca.edu
paths:
 /auth/register:
  post:
   summary: Registrar usuario
   requestBody:
    required: true
    content:
     application/json:
      schema:
       $ref: '#/components/schemas/UserRegister'
   responses:
    '201':
     description: Usuario creado
 /auth/login:
```

```
  post:
    summary: Login
    requestBody:
      required: true
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/UserLogin'
    responses:
      '200':
        description: Token JWT
  /users/{id}:
    get:
      summary: Obtener perfil de usuario
      parameters:
        - in: path
          name: id
          schema:
            type: string
          required: true
      responses:
        '200':
          description: Perfil
    put:
      summary: Actualizar perfil
      parameters:
```

```
    - in: path
      name: id
      schema:
        type: string
      required: true
  requestBody:
    required: true
    content:
      application/json:
        schema:
          $ref: '#/components/schemas/UserProfile'
  responses:
    '200':
      description: Actualizado
/gyms:
  get:
    summary: Listar gimnasios (filter: city)
    parameters:
      - in: query
        name: city
        schema:
          type: string
    responses:
      '200':
        description: Lista de gimnasios
  post:
```

```
    summary: Crear gym
    requestBody:
      required: true
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/Gym'
    responses:
      '201':
        description: Gym creado
  /trainers:
    get:
      summary: Lista de entrenadores (filter gym_id)
      parameters:
        - in: query
          name: gym_id
          schema:
            type: string
      responses:
        '200':
          description: Lista
  /exercises:
    get:
      summary: Lista ejercicios
      responses:
        '200':
```

```
      description: OK
  post:
    summary: Crear ejercicio
    requestBody:
      required: true
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/Exercise'
    responses:
      '201':
        description: Creado
/routines:
  post:
    summary: Crear rutina
    requestBody:
      required: true
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/Routine'
    responses:
      '201':
        description: Creada
/sessions:
  post:
```

```
    summary: Registrar sesión
    requestBody:
      required: true
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/Session'
    responses:
      '201':
        description: Sesión creada
  /metrics/user/{userId}:
    get:
      summary: Obtener métricas de usuario
      parameters:
        - in: path
          name: userId
          schema:
            type: string
      responses:
        '200':
          description: Métricas
components:
  schemas:
    UserRegister:
      type: object
      required: [email,password,name,age]
```

```
  properties:
    email:
      type: string
    password:
      type: string
    name:
      type: string
    age:
      type: integer
  UserLogin:
    type: object
    required: [email,password]
    properties:
      email:
        type: string
      password:
        type: string
  UserProfile:
    type: object
    properties:
      name: { type: string }
      age: { type: integer }
      sex: { type: string }
      weight: { type: number }
      height: { type: number }
      body_fat_pct: { type: number }
```

```
    level: { type: string }
    objectives: { type: string }
  Gym:
   type: object
   required: [name,city,address]
   properties:
    name: { type: string }
    address: { type: string }
    city: { type: string }
    lat: { type: number }
    lng: { type: number }
    phone: { type: string }
    schedule: { type: string }
  Trainer:
   type: object
   properties:
    name: { type: string }
    certifications: { type: string }
    gym_id: { type: string }
  Exercise:
   type: object
   required: [name,muscle_group]
   properties:
    name: { type: string }
    muscle_group: { type: string }
    description: { type: string }
```

```
    video_url: { type: string }
Routine:
  type: object
  properties:
    name: { type: string }
    user_id: { type: string }
    level: { type: string }
    items:
      type: array
      items:
        type: object
        properties:
          exercise_id: { type: string }
          sets: { type: integer }
          reps: { type: integer }
Session:
  type: object
  properties:
    user_id: { type: string }
    routine_id: { type: string }
    date: { type: string, format: date }
    duration_min: { type: integer }
    items:
      type: array
      items:
        type: object
```

```
      properties:
        exercise_id: { type: string }
        sets_done: { type: integer }
        reps_done: { type: integer }
```

3. DIAGRAMA ER (TEXTO) Y SQL MIGRATIONS INICIALES

3.1 Diagrama ER (simplificado)

```
USERS --< SESSIONS >-- ROUTINES
 |              |
 |              > ROUTINE_ITEMS -- EXERCISES
 |              |
 > METRICS

GYMS --< TRAINERS

USERS --< PAYMENTS (opcional)
```

3.2 SQL migrations (PostgreSQL)

```
-- 001_create_users.sql
CREATE TABLE users (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  email VARCHAR(255) UNIQUE NOT NULL,
  password_hash VARCHAR(255) NOT NULL,
  name VARCHAR(255),
  age INT,
  sex VARCHAR(20),
  weight NUMERIC(5,2),
```

```
  height NUMERIC(5,2),
  body_fat_pct NUMERIC(5,2),
  level VARCHAR(50),
  objectives TEXT,
  created_at TIMESTAMP WITH TIME ZONE DEFAULT now(),
  updated_at TIMESTAMP WITH TIME ZONE DEFAULT now()
);

-- 002_create_gyms.sql
CREATE TABLE gyms (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  name VARCHAR(255) NOT NULL,
  address TEXT,
  city VARCHAR(100),
  lat NUMERIC(9,6),
  lng NUMERIC(9,6),
  phone VARCHAR(50),
  schedule TEXT,
  created_by UUID,
  created_at TIMESTAMP WITH TIME ZONE DEFAULT now()
);

-- 003_create_trainers.sql
CREATE TABLE trainers (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  name VARCHAR(255) NOT NULL,
```

```
  certifications TEXT,
  bio TEXT,
  phone VARCHAR(50),
  email VARCHAR(255),
  gym_id UUID REFERENCES gyms(id) ON DELETE SET NULL,
  created_at TIMESTAMP WITH TIME ZONE DEFAULT now()
);

-- 004_create_exercises.sql
CREATE TABLE exercises (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  name VARCHAR(255) NOT NULL,
  muscle_group VARCHAR(100),
  description TEXT,
  video_url TEXT,
  difficulty VARCHAR(50),
  created_at TIMESTAMP WITH TIME ZONE DEFAULT now()
);

-- 005_create_routines.sql
CREATE TABLE routines (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  name VARCHAR(255) NOT NULL,
  user_id UUID REFERENCES users(id) ON DELETE SET NULL,
  level VARCHAR(50),
  objective VARCHAR(255),
```

```
  created_at TIMESTAMP WITH TIME ZONE DEFAULT now()
);

-- 006_create_routine_items.sql
CREATE TABLE routine_items (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  routine_id UUID REFERENCES routines(id) ON DELETE CASCADE,
  exercise_id UUID REFERENCES exercises(id) ON DELETE SET NULL,
  sets INT,
  reps INT,
  weight_suggested NUMERIC(6,2),
  rest_seconds INT,
  sort_order INT
);

-- 007_create_sessions.sql
CREATE TABLE sessions (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  user_id UUID REFERENCES users(id) ON DELETE CASCADE,
  routine_id UUID REFERENCES routines(id) ON DELETE SET NULL,
  date TIMESTAMP WITH TIME ZONE DEFAULT now(),
  duration_min INT,
  completed BOOLEAN DEFAULT false
);

-- 008_create_session_items.sql
```

```
CREATE TABLE session_items (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  session_id UUID REFERENCES sessions(id) ON DELETE CASCADE,
  exercise_id UUID REFERENCES exercises(id),
  sets_done INT,
  reps_done INT,
  weight_used NUMERIC(6,2)
);

-- 009_create_metrics.sql
CREATE TABLE metrics (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  user_id UUID REFERENCES users(id) ON DELETE CASCADE,
  date DATE NOT NULL,
  weight NUMERIC(6,2),
  body_fat_pct NUMERIC(5,2),
  note TEXT
);

-- 010_create_payments.sql (opcional)
CREATE TABLE payments (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  user_id UUID REFERENCES users(id) ON DELETE CASCADE,
  plan VARCHAR(100),
  amount NUMERIC(8,2),
  status VARCHAR(50),
```

```
  date TIMESTAMP WITH TIME ZONE DEFAULT now()
);
```

Nota: Las funciones gen_random_uuid() requieren la extensión pgcrypto o pguuid; alternativamente use uuid_generate_v4() con la extensión uuid-ossp.

4. SPRINT PLAN: 12 SEMANAS, DÍA A DÍA (84 DÍAS) — TAREAS PARA CADA EQUIPO

Premisa: Semana laboral de 5 días (lunes–viernes). Los días estimados son 12 semanas x 5 días = 60 días útiles. Si realmente quieres incluir sábados, lo puedo extender; aquí se entrega un plan *día a día* para 12 semanas de trabajo académico suponiendo 5 días/semana (60 días laborales). Cada día indica tareas por equipo (Rojo/Azul/Amarillo/Verde/Negro). Ajusta número de alumnos por equipo según tu realidad.

Semana 0 — Preparación (pre-semestre)

Día 0 (Kickoff - reunión inicial)

- Todos: Kickoff(Pantalla inicial) con Magister Adaury Paulino, Bernardo Hernández y Humberto Cruz. Aprobación alcance y asignación estudiantes por equipos. Definición herramientas (GitHub, Slack/Discord, Figma, Postman).

Semana 1 (Días 1–5)

Día 1 (Lunes)

- Rojo: Crear repositorio backend/auth y esqueleto proyecto Node+TS. Implementar modelos básicos de User.
- Azul: Repo backend/core o monorepo; diseñar esquema de Exercise y Routine en markdown.
- Amarillo: Investigación y recogida de gimnasios en Moca; plantilla de datos (nombre, dirección, horario).
- Verde: Definir métricas a recolectar y diseño inicial de tablas metrics.
- Negro: Crear monorepo, pipeline CI inicial (lint), definir branching strategy y plantilla README.
-

Día 2 (Martes)

- Rojo: Endpoint POST /auth/register (mock), validaciones básicas.
- Azul: CRUD básico de exercises (mock endpoints).
- Amarillo: Crear modelo Gym y endpoints mock para /gyms.
- Verde: Script de migration inicial (users, metrics). Añadir ejemplos de métricas de muestra.
- Negro: Crear OpenAPI base (archivo YAML) y publicar en repo.

Día 3 (Miércoles)

- Rojo: Endpoint POST /auth/login (mock token). Implementar bcrypt hashing.
- Azul: Diseño UI de pantalla biblioteca ejercicios (Figma). Entregar prototipo básico.
- Amarillo: Agregar 5 gimnasios de Moca al dataset inicial.
- Verde: Endpoint GET /metrics/user/{id} (mock).
- Negro: Docker Compose con Postgres y backend mock.

Día 4 (Jueves)

- Rojo: Integrar JWT issuance en login; crear middleware auth.
- Azul: Crear tabla exercises en DB (migration) y endpoint GET /exercises real.
- Amarillo: Implementar POST /gyms y GET /gyms?city=Moca.
- Verde: Implementación inicial de metrics table y almacenamiento de métricas.
- Negro: Pipeline CI: ejecutar migrations y tests unitarios básicos.

Día 5 (Viernes)

- Rojo: Pantalla Flutter: formulario de registro y login (mock API).
- Azul: Conectar Flutter a endpoint /exercises y mostrar lista.
- Amarillo: Pantalla admin web simple para añadir gimnasios.
- Verde: Definir endpoints para subir métricas desde app.
- Negro: Revisión de sprint 1; demo interno.

Semana 2 (Días 6–10)

Día 6 (Lunes)

- Rojo: Validaciones avanzadas en registro (edad rango, email único).
- Azul: Definición y modelo de Routines y RoutineItems.
- Amarillo: Endpoint GET /trainers con filtro gym_id (mock) y diseño de Trainer model.
- Verde: Implementación de API POST /metrics para guardar nuevas mediciones.
- Negro: Setup staging environment en DigitalOcean (droplet pequeño).

Día 7 (Martes)

- Rojo: Endpoint GET /users/:id protegida por JWT.
- Azul: API para crear rutina (POST /routines) y agregar items.
- Amarillo: CRUD completo de trainers (POST/GET/PUT/DELETE).
- Verde: Crear vistas básicas de gráficas (prototipo en Flutter o web) con datos hardcoded.
- Negro: Asegurar HTTPS (Let's Encrypt) en staging.

Día 8 (Miércoles)

- Rojo: Implementar recuperación de contraseña (esquema básico).
- Azul: Algoritmo básico para sugerir rutina según nivel (función en backend).
- Amarillo: UI móvil: pantalla de gimnasio (lista y detalle).
- Verde: Endpoint GET /metrics/user/{id} devuelve datos reales de DB.
- Negro: Tests unitarios para endpoints críticos (auth, exercises).

Día 9 (Jueves)

- Rojo: Integración móvil: login + persistencia token (secure storage).
- Azul: UI móvil: ver rutina y marcar ejercicio como completado.
- Amarillo: Exportar lista de gimnasios a CSV para revisión.
- Verde: Implementar registro de sesiones (POST /sessions).
- Negro: Revisar seguridad: hashing correcto y protección de endpoints.

Día 10 (Viernes)

- Rojo: Demo de registro/login funcional.
- Azul: Demo de creación y ejecución de rutina en móvil.
- Amarillo: Demo de CRUD gimnasios/entrenadores.
- Verde: Demo de gráficas básicas y registro de sesión.
- Negro: Sprint review; corregir issues críticos.

Semana 3 (Días 11–15)

Día 11 (Lunes)

- Rojo: Perfil usuario: pantalla y endpoint PUT /users/:id.
- Azul: Implementar media (video_url) para ejercicios y mostrar en la app.
- Amarillo: Añadir geolocalización y lat/lng para gyms en DB.
- Verde: Calcular indicadores: peso promedio, cambio %.
- Negro: Configurar backups automáticos de Postgres en staging.

Día 12 (Martes)

- Rojo: Social/login via Google (opcional) — diseño de flujo.
- Azul: Optimizar consulta de rutinas (pagination).
- Amarillo: Añadir fotos a gimnasios (upload a storage S3-compatible).
- Verde: Integrar librería de gráficas en Flutter (ej: charts_flutter).
- Negro: Preparar entorno para pruebas de integración.

Día 13 (Miércoles)

- Rojo: Tests de seguridad para endpoints auth.
- Azul: Soporte para plantillas públicas de rutinas (templates).
- Amarillo: Panel admin: asignar entrenadores a gimnasios.
- Verde: Endpoint para exportar métricas (CSV) para análisis.
- Negro: Implementar monitoring básico (healthcheck endpoint).

Día 14 (Jueves)

- Rojo: UI: edición de perfil y subida de avatar.
- Azul: Guardar historial de rutinas (routines.templates + user_routines).
- Amarillo: Implementar búsqueda por nombre y filtros (por ciudad, horario).
- Verde: Validación de métricas y limpieza de datos (scripts).
- Negro: Revisión de PRs y merge policy.

Día 15 (Viernes)

- Rojo: Demo: perfil completo y flujo de edición.
- Azul: Demo: crear rutina desde plantilla.
- Amarillo: Demo: mapa de gimnasios en Moca.
- Verde: Demo: gráficas de progreso con datos reales.
- Negro: Revisión semanal y preparación de sprint 4.

Semana 4 (Días 16–20)

Día 16 (Lunes)

- Rojo: Polishing UX y validaciones en móvil.
- Azul: Implementar marcador de progreso por ejercicio (sets completados).
- Amarillo: Validación de registros de gimnasios (contacto con gimnasios locales si posible).
- Verde: Notificaciones/recordatorios (pila inicial, FCM).
- Negro: Preparar pipeline de despliegue automatizado a staging.

Día 17 (Martes)

- Rojo: Auditoría de logs de acceso y eventos importantes.
- Azul: Añadir imágenes/gifs demostrativos a ejercicios.
- Amarillo: Agregar campo tarifas o tipo de membresía por gym (opcional).
- Verde: Implementar calculadora sencilla de IMC y mostrar en perfil.

- Negro: Crear checklist de QA para integración.

Día 18 (Miércoles)

- Rojo: Revisión de políticas de privacidad y consentimiento.
- Azul: Ruta para compartir rutinas (link o código)
- Amarillo: Endpoint para reservas sesiones con entrenador (básico).
- Verde: Alertas automáticas si usuario no entrena X días.
- Negro: Pruebas E2E en staging (simular flujo completo usuario).

Día 19 (Jueves)

- Rojo: Implementar rate limiting en endpoints críticos.
- Azul: Mejora de rendimiento en queries de rutina.
- Amarillo: Reporte semanal de actividad por gym (número sesiones).
- Verde: Integrar exportación de gráficas como imagen (para compartir).
- Negro: Harden server (cabeceras seguridad, CORS bien configurado).

Día 20 (Viernes)

- Demo inter-equipo: flujo completo (registro → rutina → sesión → métricas).
- Negro: Corrección de fallas encontradas durante E2E.

Semana 5 (Días 21–25)

Día 21 (Lunes)

- Rojo: Implementar refresh tokens y expiración correcta.
- Azul: Añadir tags a ejercicios (equipamiento requerido).
- Amarillo: Mapa interactivo (integración básica con Google Maps / OpenStreetMap).
- Verde: Implemente storage histórico en DB y paginación de métricas.
- Negro: Documentar entorno de despliegue y cost breakdown.

Día 22 (Martes)

- Rojo: Revisión de UX de login y recuperación de contraseña.
- Azul: Implementar filtros por grupo muscular y dificultad.
- Amarillo: Envío de confirmación por email al crear gym o trainer (mock/SMTP).
- Verde: Pruebas de integridad de datos (backfill si necesario).
- Negro: Optimizar Docker images para tamaño y build cache.

Día 23 (Miércoles)

- Rojo: Monitoreo de seguridad (alertas por intentos fallidos).
- Azul: Export/import de rutinas (JSON shareable).
- Amarillo: Establecer roles: entrenador / admin / usuario.
- Verde: Reporte semanal de usuarios activos.
- Negro: Instrumentación para logs estructurados (json logs).

Día 24 (Jueves)

- Rojo: Integrar OAuth social si procede (Google Sign-in flow demo).
- Azul: Mejoras UI/UX en pantalla de rutinas.
- Amarillo: Integración con calendarios para reservar sesiones.
- Verde: Implementar test de carga ligero (100 usuarios simulados).
- Negro: Revisión final de seguridad antes de beta pública.

Día 25 (Viernes)

- Demo: features nuevas (refresh tokens, maps, export routines).
- Negro: Cierre de sprint y lista de tareas pendientes.

Semana 6 (Días 26–30)

Día 26 (Lunes)

- Rojo: Hardening de endpoints y revisión de dependencias (vulnerabilidades npm).
- Azul: Agregar descripciones paso a paso para cada ejercicio (ejecución correcta).

- Amarillo: Obtener feedback de 3 gimnasios locales (si posible).
- Verde: Ajustes al cálculo de progreso (normalizar datos).
- Negro: Preparar demo para stakeholders (Magister y coordinadores).

Día 27 (Martes)

- Rojo: Polishing UI y mensajes de error.
- Azul: Quick wins: autoscroll en listas largas y cache de imágenes.
- Amarillo: Añadir campo de experiencia al trainer (años, especialidad).
- Verde: Implementar badges/achievements básicos por frecuencia de entreno.
- Negro: Auditar costos y proponer optimizaciones.

Día 28 (Miércoles)

- Rojo: Tests de usabilidad en flujo de registro con 5 usuarios.
- Azul: Ajustes al algoritmo de generación de rutinas por feedback.
- Amarillo: Ajustes al panel admin por feedback.
- Verde: Preparar dataset de demostración para presentación.
- Negro: Preparar checklist de entrega parcial.

Día 29 (Jueves)

- Rojo: Backup y documentación de endpoints de usuario.
- Azul: Documentar la librería de ejercicios y cómo añadir nuevos.
- Amarillo: Preparar material explicativo para entrenadores.
- Verde: Finalizar gráficas y exportaciones.
- Negro: Deploy a staging listo para demo.

Día 30 (Viernes)

- Presentación intermedia al Magister Adaury Paulino y coordinadores. Recolección de feedback.

Semana 7 (Días 31–35)

Día 31 (Lunes)

- Rojo: Corregir feedback recibido (prioridad alta).
- Azul: Añadir tutorial in-app para crear y ejecutar rutinas.
- Amarillo: Registrar 10 entrenadores reales o ficticios con datos completos.
- Verde: Implementar segments por objetivos (pérdida de grasa, fuerza).
- Negro: Revisión de accesos y roles en staging.

Día 32 (Martes)

- Rojo: Implementar consentimientos y guardado de aceptación T&C.
- Azul: Soporte para descanso entre series configurable.
- Amarillo: Workflow para que un entrenador acepte/gestione reservas.
- Verde: Notificaciones push de recordatorio de sesión.
- Negro: Tests de compatibilidad móvil (Android & iOS).

Día 33 (Miércoles)

- Rojo: Dashboard pequeño de actividad del usuario (últimas sesiones).
- Azul: Soporte para variantes de ejercicios (sin equipo, con banda).
- Amarillo: Exportar listado de trainers por gym (para impresión).
- Verde: Tracking de consistencia (días seguidos entrenando).
- Negro: Revisión de performance de DB (indices si necesario).

Día 34 (Jueves)

- Rojo: Revisar flujos de seguridad y logout remoto.
- Azul: Mejoras en reproducibilidad de rutina (cronograma semanal).
- Amarillo: Integrar fotos y reseñas (opiniones) de usuarios por gym.
- Verde: Alerta automática a entrenador cuando un cliente reserva.
- Negro: Refinar backups y políticas de retención.

Día 35 (Viernes)

- Demo de mitad de proyecto con métricas y casos de uso.

Semana 8 (Días 36–40)

Día 36 (Lunes)

- Rojo: Preparar scripts para crear usuarios demo y data seed.
- Azul: Polishing en UI de rutinas y ayuda contextual.
- Amarillo: Preparar guía para que gyms suban su información.
- Verde: Crear reportes semanales automáticos para coordinadores.
- Negro: Revisar logs y errores producidos en staging.

Día 37 (Martes)

- Rojo: Implementación final de OAuth/social (si se aprobó).
- Azul: Integración de videos de ejercicios (CDN).
- Amarillo: Revisar y normalizar horarios de gyms.
- Verde: Añadir comparativas entre semanas (delta % de peso).
- Negro: Ajustes en escalabilidad para picos menores.

Día 38 (Miércoles)

- Rojo: Tests de integración auth + profile.
- Azul: Permitir duplicar una rutina como plantilla.
- Amarillo: Añadir búsqueda por entrenador (especialidad).
- Verde: Ajustes UI de gráficas para mayor claridad.
- Negro: Preparar snapshot para backup completo antes de beta.

Día 39 (Jueves)

- Rojo: Revisar y documentar endpoints de seguridad.
- Azul: Añadir instrucciones paso a paso para cada ejercicio (texto + video).
- Amarillo: Polishing panel admin y permisos.
- Verde: Validaciones finales de datos.
- Negro: Ejecutar prueba de carga adicional.

Día 40 (Viernes)

- Beta interna: invitar 20 usuarios (estudiantes y personal) a probar.

Semana 9 (Días 41–45)

Día 41 (Lunes)

- Rojo: Corregir bugs reportados en beta.
- Azul: Ajustes en algoritmo de recomendación por feedback.
- Amarillo: Contacto con gimnasios para pruebas de campo (Moca).
- Verde: Análisis de datos de beta y preparar mejoras.
- Negro: Parchear fallas críticas y desplegar hotfix.

Día 42 (Martes)

- Rojo: Mejorar mensajes de error y UX.
- Azul: Añadir opción de marcar ejercicios favoritos.
- Amarillo: Probar reservas reales con entrenadores (si procede).
- Verde: Ajustar badge/achievements según uso real.
- Negro: Revisar logs de errores y performance.

Día 43 (Miércoles)

- Rojo: Tests de seguridad post-beta.
- Azul: Export/import de rutinas para compartir entre usuarios.
- Amarillo: Recoger testimonios y casos de uso locales.
- Verde: Mejoras en gráficas basadas en datos reales.
- Negro: Revisión de costes de hosting según uso.

Día 44 (Jueves)

- Rojo: Cleanup y preparar versión para entrega parcial.
- Azul: Preparar documentación de cómo añadir ejercicios nuevos.
- Amarillo: Preparar instructivo para entrenadores usar la app.

- Verde: Preparar dataset para presentación final.
- Negro: Preparación de entorno de producción mínimo viable.

Día 45 (Viernes)

- Demo de mejoras tras beta; priorizar tareas para sprint final.

Semana 10 (Días 46–50)

Día 46 (Lunes)

- Rojo: Finalizar endpoints pendientes y documentar.
- Azul: Optimizar queries y reducir latencia en rutinas.
- Amarillo: Finalizar integración de pagos (si aplica) o marcar como futura fase.
- Verde: Testing final de gráficas y exports.
- Negro: Implementar dominio y certificado SSL para producción.

Día 47 (Martes)

- Rojo: Revisión y pruebas unitarias finales.
- Azul: Pulir UX y animaciones pequeñas para la app.
- Amarillo: Preparar material impreso (flyers) para gimnasios.
- Verde: Preparar métricas de evaluación para la presentación final.
- Negro: Backup y snapshot antes del despliegue a producción.

Día 48 (Miércoles)

- Rojo: Preparar script de migración para producción.
- Azul: Realizar pruebas E2E completas.
- Amarillo: Coordinar agenda con entrenadores para presentación.
- Verde: Compilar análisis de uso y recomendaciones.
- Negro: Deploy a producción (MVP) en entorno controlado.

Día 49 (Jueves)

- Rojo: Monitoreo post-deploy (errores de auth, latencia).

- Azul: Resolución de bugs menores detectados en producción.
- Amarillo: Probar flujos de reserva y contacto.
- Verde: Monitoreo de métricas de uso inicial.
- Negro: Asegurar backups y políticas de rollback.

Día 50 (Viernes)

- Revisión general y pruebas de humo en producción.

Semana 11 (Días 51–55)

Día 51 (Lunes)

- Rojo: Optimizar experiencia de onboarding.
- Azul: Añadir ayuda contextual y FAQs en la app.
- Amarillo: Reunión con coordinadores para feedback final.
- Verde: Preparar slides de métricas para la presentación.
- Negro: Revisión final de seguridad y logs.

Día 52 (Martes)

- Rojo: Preparar video corto demo (2–3 min) del flujo auth.
- Azul: Preparar video demo de creación y ejecución de rutina.
- Amarillo: Preparar video demo del panel de gyms/entrenadores.
- Verde: Preparar video demo de gráficas y seguimiento.
- Negro: Preparar checklist de entrega técnica.

Día 53 (Miércoles)

- Todos: Ensayo general de la demo final (role play: usuario, entrenador, admin).

Día 54 (Jueves)

- Todos: Ajustes finales por feedback del ensayo. Documentación final.

Día 55 (Viernes)

- Presentación final a Magister Adaury Paulino, Bernardo Hernández y Humberto Cruz. Entrega de repositorios, README, OpenAPI, scripts de DB, y vídeo-demostraciones.

Semana 12 (Días 56–60) — Cierre y mejoras post-entrega

Día 56 (Lunes)

- Rojo: Correcciones menores y pulido final.
- Azul: Añadir mejoras solicitadas en la presentación.
- Amarillo: Registrar feedback de gimnasios.
- Verde: Ajustes a métricas y envío de reportes finales.
- Negro: Generar informe técnico (deploy, costos, recomendaciones).

Día 57 (Martes)

- Todos: Preparar paquete de entrega (ZIP con documentación, SQL dumps, accesos de staging/producción).

Día 58 (Miércoles)

- Todos: Sesión de transferencia de conocimiento — cada equipo explica su módulo.

Día 59 (Jueves)

- Todos: Checklist de cierre: backups, permisos, accesos, retirar credenciales temporales.

Día 60 (Viernes)

- Celebración y retroalimentación — lecciones aprendidas y posibles fases futuras.

5. Costos y hosting (resumen)

- Prototipo académico: **USD 25–70 / mes** (droplet pequeño + managed Postgres + storage).
- MVP producción inicial: **USD 100–400+/mes** dependiendo de usuarios, almacenamiento y CDN.

6. Archivos incluidos y formatos entregables

- Documento de proyecto (este archivo).
- OpenAPI YAML (incluido arriba).
- SQL migrations (scripts incluidos arriba).
- Sprint plan día a día (este documento).
- Repositorios: sugerencia monorepo con carpetas mobile/, backend/, infra/, docs/.

7. SIGUIENTES PASOS RECOMENDADOS

1. Revisar y aprobar este documento con Magister Adaury Paulino y coordinadores.
2. Crear repositorios y asignar estudiantes a equipos.
3. Equipo Negro publica OpenAPI en formato alojado (SwaggerHub o raw YAML en repo).
4. Kickoff oficial (día 0) y comenzar Semana 1.

Contacto y notas finales

Si deseas, puedo ahora:

- Generar el **PDF listo para impresión** de este documento.
- Crear el **OpenAPI.json** (convertir YAML a JSON).