Guía Técnica de Construcción: Sistema Pilates Fit

Este documento constituye la guía técnica definitiva para el diseño, desarrollo y despliegue del sistema **Pilates Fit**. Está dirigido al equipo de ingeniería y proporciona una justificación arquitectónica, especificaciones de la pila tecnológica, flujos de implementación detallados y mejores prácticas operativas. El objetivo es establecer un plan de trabajo claro y coherente que garantice la construcción de una plataforma robusta, escalable y segura.

A continuación, se presenta un resumen de la pila tecnológica seleccionada para el proyecto, que servirá como referencia rápida a lo largo de este documento.

Tabla de Resumen de la Pila Tecnológica

Dominio	Tecnología/Framework	Librerías/Herramientas Clave
Backend	Django 4.2	Django REST Framework (DRF), drf-spectacular, dj-rest-auth
Base de Datos	MySQL 8.0	-
Frontend (Web)	React 18	TanStack Table, Recharts, React Hook Form, Zustand
Frontend (Móvil)	Flutter 3.10	Syncfusion Flutter Charts, Firebase Messaging Plugin
Tareas Asíncronas	Celery	Redis (Broker), django-celery-beat
Comunicación en Tiempo Real	Django Channels	Daphne (Servidor ASGI), Redis (Channel Layer)
IA / Machine Learning	Python	Hugging Face Transformers (T5), TensorFlow Lite, Rasa, OpenCV

Despliegue	Docker, Kubernetes	Nginx Ingress Controller, Helm
CI/CD	GitLab CI	-
Seguridad y Calidad	DevSecOps	SonarQube, OWASP ZAP

I. Plan Arquitectónico del Sistema: Una Perspectiva del Modelo C4

Esta sección establece la visión arquitectónica de alto nivel para el sistema Pilates Fit utilizando el modelo C4. Este enfoque garantiza la claridad y un entendimiento compartido entre todas las partes interesadas, desde los analistas de negocio hasta los desarrolladores, al proporcionar una vista jerárquica y contextualizada del software.

1.1. Filosofía Arquitectónica: Diagramas como Código

Para visualizar la arquitectura del software, se adoptará el modelo C4, que descompone el sistema en niveles de Contexto, Contenedores, Componentes y Código.[1] Este no es un esfuerzo de documentación único, sino una práctica continua integrada en el ciclo de vida del desarrollo.

Todos los diagramas se definirán utilizando el Lenguaje Específico de Dominio (DSL) de Structurizr.[3] Los archivos de definición (

workspace.dsl) se almacenarán en el repositorio principal de código fuente. Este enfoque, conocido como "Diagramas como Código", asegura que la documentación arquitectónica esté versionada, sea revisable a través de *pull requests* y se renderice automáticamente, evitando así la deriva arquitectónica.[2]

La elección de Structurizr y C4 es una decisión estratégica que promueve una arquitectura evolutiva. Al integrar las definiciones arquitectónicas en el repositorio Git, la estructura del sistema se convierte en un ciudadano de primera clase del código base. Cualquier modificación en los contenedores del sistema o en sus interacciones

debe reflejarse en el DSL, lo que convierte las revisiones de arquitectura en una parte obligatoria del proceso de revisión de código. Esta práctica evita el problema común de que la documentación se vuelva obsoleta e irrelevante, imponiendo así una coherencia arquitectónica a lo largo de todo el ciclo de desarrollo.

1.2. Nivel 1: Diagrama de Contexto del Sistema

Este diagrama proporciona la vista de "panorama general", mostrando cómo el sistema Pilates Fit se integra en su entorno operativo. Define los límites del sistema, sus usuarios y las interacciones con sistemas externos.

- **Actores:**
  - **Miembro:** El usuario final principal, que interactúa con el sistema a través de la Aplicación Móvil para reservar clases, seguir su progreso y gestionar su cuenta.
  - **Personal (Instructor/Admin):** El personal del estudio que gestiona clases, miembros, horarios y finanzas a través de la Aplicación Web.
  - **Administrador del Sistema:** Un rol técnico responsable de la configuración del sistema, la gestión de roles de usuario y el mantenimiento general.
- **Sistemas de Software:**
  - **Sistema Pilates Fit:** El sistema central que se está construyendo, que engloba toda la lógica de negocio.
  - **Servicio de Correo Electrónico (Externo):** Un servicio de terceros (ej. SendGrid, Mailgun) utilizado para enviar correos electrónicos transaccionales, como confirmaciones de reserva y recordatorios de contraseña.
  - **Pasarela de Pago (Externa):** Un procesador de pagos (ej. Stripe, Braintree) para gestionar las transacciones con tarjeta de crédito de forma segura, cumpliendo con el estándar PCI.
  - **API de Nutrición (Externa, Opcional):** Para una futura funcionalidad, se prevé la integración con la API FoodData Central del USDA para proporcionar información nutricional a los miembros.[5]
  - **Servicio de Notificaciones Push de Google/Apple (Externo):** El sistema interactuará con estos servicios a través de Firebase Cloud Messaging (FCM) para enviar notificaciones a los dispositivos móviles.

1.3. Nivel 2: Diagrama de Contenedores

Este diagrama "hace zoom" en el sistema Pilates Fit, descomponiéndolo en sus unidades desplegables o ejecutables de forma independiente (contenedores). Esta vista es fundamental para los equipos de desarrollo y DevOps, ya que define los límites tecnológicos y de despliegue.

- **Contenedores dentro del límite del Sistema Pilates Fit:**
  - **Aplicación Web (Portal Administrativo):** Una Aplicación de Página Única (SPA) construida con React 18, que sirve como portal para el personal y los administradores. Se comunica exclusivamente con el Backend API a través de HTTPS.[8]
  - **Aplicación Móvil (Portal de Miembros):** Una aplicación multiplataforma (iOS y Android) desarrollada con Flutter 3.10 para los miembros. Se comunica con el Backend API.[10]
  - **Servidor de API Backend:** Una aplicación Django 4.2 que expone una API RESTful a través de HTTPS. Este es el motor de lógica de negocio central del sistema y el único contenedor con acceso directo de escritura a la base de datos.[12]
  - **Trabajador Asíncrono (Asynchronous Worker):** Un proceso Celery que se ejecuta de forma independiente para manejar tareas en segundo plano (ej. envío de notificaciones, generación de informes). Consume tareas del Message Broker.[14]
  - **Pasarela en Tiempo Real (Real-Time Gateway):** Un servidor ASGI Daphne que ejecuta Django Channels, gestionando conexiones WebSocket persistentes para funcionalidades en tiempo real.[16]
  - **Base de Datos:** Una base de datos relacional MySQL 8.0 que almacena todos los datos de negocio principales. Es el único almacén de estado persistente para los datos operativos.[18]
  - **Message Broker:** Una instancia de Redis utilizada como intermediario de mensajes para Celery y como capa de canal (channel layer) para Django Channels.[16]

1.4. Implementación con Structurizr DSL

A continuación se presenta un archivo workspace.dsl completo que define los

actores, sistemas, contenedores y sus relaciones descritas anteriormente. Este archivo servirá como la única fuente de verdad para los diagramas C4 y deberá mantenerse actualizado junto con el código.

Fragmento de código

```
workspace "Sistema Pilates Fit" "Guía técnica para el sistema de gestión de estudios Pilates Fit" {

  !identifiers hierarchical

  model {
      # --- Actores (Personas) ---
      miembro = person "Miembro" "Cliente final del estudio. Usa la App Móvil."
      personal = person "Personal (Admin/Instructor)" "Gestiona el estudio. Usa la App Web."
      adminSistema = person "Administrador del Sistema" "Gestiona la configuración técnica del sistema."

      # --- Sistemas Externos ---
      emailService = softwareSystem "Servicio de Correo Electrónico" "Envía notificaciones por email (ej. SendGrid)." "External"
      paymentGateway = softwareSystem "Pasarela de Pago" "Procesa pagos con tarjeta de crédito (ej. Stripe)." "External"
      pushService = softwareSystem "Servicios de Notificación Push" "Plataformas de Apple (APNS) y Google (FCM)." "External"
      nutritionAPI = softwareSystem "API de Nutrición" "API externa para datos nutricionales (ej. USDA FoodData Central)." "External, Optional"

      # --- Sistema Pilates Fit y sus Contenedores ---
      pilatesFitSystem = softwareSystem "Sistema Pilates Fit" "Plataforma central para la gestión del estudio." {
          tags "Main System"

          # --- Frontends ---
          mobileApp = container "Aplicación Móvil" "App multiplataforma para
```

```
miembros." "Flutter 3.10" {
            tags "Mobile App"
        }
        webApp = container "Aplicación Web" "Portal SPA para personal y
administradores." "React 18" {
            tags "Web App"
        }

        # --- Backend y Servicios ---
        apiServer = container "Servidor de API Backend" "Gestiona la lógica de
negocio, autenticación y datos." "Django 4.2 / DRF" {
            tags "API Server"
        }
        asyncWorker = container "Trabajador Asíncrono" "Ejecuta tareas en segundo
plano." "Celery" {
            tags "Worker"
        }
        realtimeGateway = container "Pasarela en Tiempo Real" "Gestiona conexiones
WebSocket." "Django Channels / Daphne" {
            tags "WebSocket Server"
        }

        # --- Almacenamiento de Datos ---
        database = container "Base de Datos" "Almacén de datos relacional." "MySQL
8.0" {
            tags "Database"
        }
        messageBroker = container "Message Broker" "Cola de tareas y capa de
canal." "Redis" {
            tags "Broker"
        }
    }

    # --- Relaciones ---
    miembro -> pilatesFitSystem.mobileApp "Usa"
    personal -> pilatesFitSystem.webApp "Usa"
    adminSistema -> pilatesFitSystem.webApp "Configura y gestiona"

    pilatesFitSystem.mobileApp -> pilatesFitSystem.apiServer "Realiza llamadas API"
```

```
"HTTPS"
        pilatesFitSystem.webApp -> pilatesFitSystem.apiServer "Realiza llamadas API"
"HTTPS"

        pilatesFitSystem.mobileApp -> pilatesFitSystem.realtimeGateway "Establece
conexión" "WebSocket"
        pilatesFitSystem.webApp -> pilatesFitSystem.realtimeGateway "Establece
conexión" "WebSocket"

        pilatesFitSystem.apiServer -> pilatesFitSystem.database "Lee y escribe en"
"JDBC/TCP"
        pilatesFitSystem.apiServer -> pilatesFitSystem.messageBroker "Encola tareas
para"
        pilatesFitSystem.apiServer -> paymentGateway "Procesa pagos a través de"
"HTTPS"
        pilatesFitSystem.apiServer -> emailService "Envía correos a través de"
"HTTPS/SMTP"
        pilatesFitSystem.apiServer -> nutritionAPI "Consulta datos de" "HTTPS"

        pilatesFitSystem.asyncWorker -> pilatesFitSystem.database "Lee datos para
tareas" "JDBC/TCP"
        pilatesFitSystem.asyncWorker -> pilatesFitSystem.messageBroker "Consume
tareas de"
        pilatesFitSystem.asyncWorker -> emailService "Envía correos masivos"
"HTTPS/SMTP"
        pilatesFitSystem.asyncWorker -> pushService "Envía notificaciones push a través
de" "HTTPS"

        pilatesFitSystem.realtimeGateway -> pilatesFitSystem.messageBroker "Usa para
la capa de canal"
    }

    views {
        # --- Diagrama de Contexto ---
        systemContext pilatesFitSystem "Contexto" {
            include *
            autoLayout lr
            description "Diagrama de Contexto del Sistema Pilates Fit"
        }
```

```
    # --- Diagrama de Contenedores ---
    container pilatesFitSystem "Contenedores" {
        include *
        autoLayout lr
        description "Diagrama de Contenedores del Sistema Pilates Fit"
    }

    # --- Estilos ---
    styles {
        element "Person" { background #08427b; color #ffffff; shape Person }
        element "Software System" { background #1168bd; color #ffffff; }
        element "External" { background #666666; color #ffffff; }
        element "Container" { background #438dd5; color #ffffff; }
        element "Database" { shape Cylinder }
        element "Broker" { shape Pipe }
        element "Web App" { shape WebBrowser }
    }
  }
}
```

II. La Fundación de Datos: Esquema MySQL e Integridad de Datos

La base de datos es el pilar fundamental de la aplicación. El esquema proporcionado [18] es robusto, bien normalizado y está diseñado no solo para las necesidades operativas actuales, sino también para futuras funcionalidades basadas en datos.

2.1. Análisis del Modelo Lógico de Datos

El esquema se divide lógicamente en cinco áreas funcionales, como se detalla en los materiales de origen.[18] Estas áreas trabajan en conjunto para modelar el negocio de manera integral.

- **CRM Central y Suscripciones:** Las tablas Miembros, PlanesMembresia y SuscripcionesMiembro forman el núcleo de la relación con el cliente y la generación de ingresos. Definen quiénes son los clientes y qué servicios han contratado.
- **Operaciones:** Personal, RolesPersonal, Clases, ClasesProgramadas y Asistencia. Este clúster gestiona el día a día del estudio: quién imparte las clases, cuándo se programan y quién asiste. La tabla Asistencia es una tabla de unión central que conecta a los miembros con las clases programadas.
- **Finanzas:** Pagos, Gastos y CategoriasGasto. Un sistema de contabilidad efectivo, aunque simple, para rastrear todo el flujo de dinero, tanto ingresos (pagos de miembros) como egresos (gastos operativos).
- **IA y Personalización:** MetasMiembro, LogsEntrenamiento y FeedbackMiembro. Este es el motor de recopilación de datos para futuras características de inteligencia. No son simplemente registros, sino un conjunto de datos estructurado y preparado para el análisis.
- **Configuración:** PreferenciasDefault y PreferenciasUsuarioOverride. Un sistema flexible que permite configuraciones globales de la aplicación y personalizaciones a nivel de usuario.

La inclusión de tablas como MetasMiembro, LogsEntrenamiento y FeedbackMiembro no es meramente para el mantenimiento de registros. Constituyen un *data lake* estructurado dentro de la base de datos operativa. Un miembro (Miembros) establece un objetivo (MetasMiembro), asiste a clases (Asistencia), realiza ejercicios (LogsEntrenamiento) y proporciona retroalimentación (FeedbackMiembro). Esta cadena causal completa y conectada está perfectamente estructurada para entrenar modelos de aprendizaje automático que podrían, por ejemplo, recomendar clases basadas en objetivos y rendimiento pasado, o predecir la deserción de miembros en función de los patrones de asistencia y las puntuaciones de retroalimentación. Este diseño proactivo evita costosos esfuerzos de ingeniería de datos en el futuro.

2.2. Implementación Física: Scripts MySQL Validados

Esta sección presenta los scripts CREATE TABLE completos, validados y corregidos sintácticamente, basados en el material proporcionado.[18]

- **Motor y Codificación:** Todas las tablas utilizarán explícitamente ENGINE=InnoDB para garantizar el soporte de claves foráneas y transacciones, una mejor práctica

fundamental para la integridad de los datos.[20] Se utilizará DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci para soportar una gama completa de caracteres, incluidos los emojis en los campos de texto como comentario o notas_personales.

- **Políticas de Claves Foráneas:** Las políticas ON DELETE (CASCADE, SET NULL, RESTRICT) son reglas de negocio críticas.
  - ON DELETE CASCADE: Se utiliza en tablas como SuscripcionesMiembro y Asistencia. Si un miembro es eliminado del sistema, todas sus suscripciones y registros de asistencia se eliminarán automáticamente, manteniendo la base de datos limpia.
  - ON DELETE SET NULL: Se utiliza en Personal (instructor_id) y Pagos. Si un instructor es eliminado, las clases programadas que impartía no se eliminan, sino que su instructor_id se establece en NULL, indicando que se necesita un reemplazo. De manera similar, si un miembro es eliminado, sus pagos se conservan pero se desvinculan del miembro.
  - ON DELETE RESTRICT: Se utiliza en SuscripcionesMiembro (plan_id). Impide que se elimine un PlanesMembresia si hay alguna suscripción activa que lo utilice, protegiendo así los datos de ingresos históricos y activos.

A continuación, se presenta el script SQL completo para la creación de la base de datos:

```sql
-- PARTE I & II: ESTRUCTURAS FUNDAMENTALES - MIEMBROS Y SUSCRIPCIONES

CREATE TABLE PlanesMembresia (
    plan_id INT AUTO_INCREMENT PRIMARY KEY,
    nombre_plan VARCHAR(150) NOT NULL UNIQUE,
    descripcion TEXT,
    precio DECIMAL(10, 2) NOT NULL,
    duracion_dias INT COMMENT 'Para planes con duración definida (ej. 30, 365)',
    numero_clases INT COMMENT 'Para planes tipo bono de clases',
    tipo_plan ENUM('Solo Pilates', 'Solo Gimnasio', 'Completo', 'Bono') NOT NULL,
    activo BOOLEAN NOT NULL DEFAULT TRUE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

```sql
CREATE TABLE Miembros (
    miembro_id INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    apellido VARCHAR(100) NOT NULL,
    email VARCHAR(255) NOT NULL UNIQUE,
    telefono VARCHAR(20),
    fecha_nacimiento DATE,
    fecha_registro TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    estado ENUM('activo', 'inactivo', 'congelado', 'potencial') NOT NULL DEFAULT 'potencial',
    direccion VARCHAR(255),
    ciudad VARCHAR(100),
    codigo_postal VARCHAR(20),
    notas_administrativas TEXT
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

CREATE TABLE SuscripcionesMiembro (
    suscripcion_id INT AUTO_INCREMENT PRIMARY KEY,
    miembro_id INT NOT NULL,
    plan_id INT NOT NULL,
    fecha_inicio DATE NOT NULL,
    fecha_fin DATE NOT NULL,
    estado ENUM('activa', 'vencida', 'cancelada', 'futura') NOT NULL,
    precio_pagado DECIMAL(10, 2) NOT NULL,
    clases_restantes INT,
    FOREIGN KEY (miembro_id) REFERENCES Miembros(miembro_id) ON DELETE CASCADE,
    FOREIGN KEY (plan_id) REFERENCES PlanesMembresia(plan_id) ON DELETE RESTRICT
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;


-- PARTE III: EXCELENCIA OPERATIVA - PERSONAL, CLASES Y ASISTENCIA

CREATE TABLE RolesPersonal (
    rol_id INT AUTO_INCREMENT PRIMARY KEY,
    nombre_rol VARCHAR(100) NOT NULL UNIQUE,
    descripcion_rol TEXT
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

CREATE TABLE Personal (
    personal_id INT AUTO_INCREMENT PRIMARY KEY,
```

```
    nombre VARCHAR(100) NOT NULL,
    apellido VARCHAR(100) NOT NULL,
    email VARCHAR(255) NOT NULL UNIQUE,
    telefono VARCHAR(20),
    fecha_contratacion DATE NOT NULL,
    rol_id INT,
    especialidades TEXT,
    activo BOOLEAN NOT NULL DEFAULT TRUE,
    FOREIGN KEY (rol_id) REFERENCES RolesPersonal(rol_id) ON DELETE SET NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

CREATE TABLE HorariosPersonal (
    horario_id INT AUTO_INCREMENT PRIMARY KEY,
    personal_id INT NOT NULL,
    inicio_turno DATETIME NOT NULL,
    fin_turno DATETIME NOT NULL,
    notas TEXT,
    FOREIGN KEY (personal_id) REFERENCES Personal(personal_id) ON DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

CREATE TABLE Clases (
    clase_id INT AUTO_INCREMENT PRIMARY KEY,
    nombre_clase VARCHAR(150) NOT NULL UNIQUE,
    descripcion TEXT,
    duracion_minutos_default INT,
    capacidad_maxima_default INT
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

CREATE TABLE ClasesProgramadas (
    clase_programada_id INT AUTO_INCREMENT PRIMARY KEY,
    clase_id INT NOT NULL,
    instructor_id INT,
    fecha_hora_inicio DATETIME NOT NULL,
    fecha_hora_fin DATETIME NOT NULL,
    sala_o_ubicacion VARCHAR(100),
    capacidad_actual INT NOT NULL DEFAULT 0,
    estado ENUM('Programada', 'Completada', 'Cancelada') NOT NULL DEFAULT 'Programada',
    FOREIGN KEY (clase_id) REFERENCES Clases(clase_id) ON DELETE RESTRICT,
    FOREIGN KEY (instructor_id) REFERENCES Personal(personal_id) ON DELETE SET NULL
```

```
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

CREATE TABLE Asistencia (
    asistencia_id INT AUTO_INCREMENT PRIMARY KEY,
    clase_programada_id INT NOT NULL,
    miembro_id INT NOT NULL,
    fecha_reserva TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    fecha_check_in DATETIME,
    estado ENUM('Reservado', 'Asistió', 'Canceló', 'No se presentó') NOT NULL DEFAULT 'Reservado',
    FOREIGN KEY (clase_programada_id) REFERENCES
ClasesProgramadas(clase_programada_id) ON DELETE CASCADE,
    FOREIGN KEY (miembro_id) REFERENCES Miembros(miembro_id) ON DELETE CASCADE,
    UNIQUE (clase_programada_id, miembro_id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;


-- PARTE IV: MOTOR FINANCIERO - INGRESOS Y GASTOS

CREATE TABLE Pagos (
    pago_id INT AUTO_INCREMENT PRIMARY KEY,
    miembro_id INT,
    suscripcion_id INT,
    monto DECIMAL(10, 2) NOT NULL,
    fecha_pago TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    metodo_pago ENUM('Tarjeta de Crédito', 'Transferencia', 'Efectivo', 'Otro') NOT NULL,
    concepto VARCHAR(255) NOT NULL,
    id_transaccion_externa VARCHAR(255),
    FOREIGN KEY (miembro_id) REFERENCES Miembros(miembro_id) ON DELETE SET NULL,
    FOREIGN KEY (suscripcion_id) REFERENCES SuscripcionesMiembro(suscripcion_id) ON
DELETE SET NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

CREATE TABLE CategoriasGasto (
    categoria_gasto_id INT AUTO_INCREMENT PRIMARY KEY,
    nombre_categoria VARCHAR(100) NOT NULL UNIQUE,
    tipo_gasto ENUM('Fijo', 'Variable') NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

CREATE TABLE Gastos (
```

```
    gasto_id INT AUTO_INCREMENT PRIMARY KEY,
    categoria_gasto_id INT NOT NULL,
    monto DECIMAL(10, 2) NOT NULL,
    fecha_gasto DATE NOT NULL,
    proveedor VARCHAR(150),
    descripcion TEXT,
    personal_id_relacionado INT,
    FOREIGN KEY (categoria_gasto_id) REFERENCES CategoriasGasto(categoria_gasto_id) ON DELETE RESTRICT,
    FOREIGN KEY (personal_id_relacionado) REFERENCES Personal(personal_id) ON DELETE SET NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;


-- PARTE V: A PRUEBA DE FUTURO - DATOS PARA IA

CREATE TABLE MetasMiembro (
    meta_id INT AUTO_INCREMENT PRIMARY KEY,
    miembro_id INT NOT NULL,
    tipo_meta ENUM('Pérdida de peso', 'Ganancia muscular', 'Mejora de flexibilidad', 'Aumento de resistencia', 'Bienestar general') NOT NULL,
    valor_objetivo VARCHAR(50),
    fecha_inicio DATE NOT NULL,
    fecha_objetivo DATE,
    estado_meta ENUM('Activa', 'Alcanzada', 'Abandonada') NOT NULL DEFAULT 'Activa',
    FOREIGN KEY (miembro_id) REFERENCES Miembros(miembro_id) ON DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

CREATE TABLE LogsEntrenamiento (
    log_id INT AUTO_INCREMENT PRIMARY KEY,
    miembro_id INT NOT NULL,
    asistencia_id INT,
    fecha_ejercicio DATE NOT NULL,
    nombre_ejercicio VARCHAR(100) NOT NULL,
    series INT,
    repeticiones INT,
    peso_kg DECIMAL(6, 2),
    duracion_segundos INT,
    distancia_km DECIMAL(6, 2),
```

```
    notas_personales TEXT,
    FOREIGN KEY (miembro_id) REFERENCES Miembros(miembro_id) ON DELETE CASCADE,
    FOREIGN KEY (asistencia_id) REFERENCES Asistencia(asistencia_id) ON DELETE SET NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

CREATE TABLE FeedbackMiembro (
    feedback_id INT AUTO_INCREMENT PRIMARY KEY,
    miembro_id INT NOT NULL,
    clase_programada_id INT,
    instructor_id INT,
    calificacion INT NOT NULL CHECK (calificacion BETWEEN 1 AND 5),
    comentario TEXT,
    fecha_feedback TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (miembro_id) REFERENCES Miembros(miembro_id) ON DELETE CASCADE,
    FOREIGN KEY (clase_programada_id) REFERENCES
ClasesProgramadas(clase_programada_id) ON DELETE SET NULL,
    FOREIGN KEY (instructor_id) REFERENCES Personal(personal_id) ON DELETE SET NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

CREATE TABLE PreferenciasDefault (
    nombre_preferencia VARCHAR(100) PRIMARY KEY,
    valor_default TEXT NOT NULL,
    tipo_dato VARCHAR(50) NOT NULL,
    descripcion TEXT
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

CREATE TABLE PreferenciasUsuarioOverride (
    miembro_id INT NOT NULL,
    nombre_preferencia VARCHAR(100) NOT NULL,
    valor_personalizado TEXT NOT NULL,
    PRIMARY KEY (miembro_id, nombre_preferencia),
    FOREIGN KEY (miembro_id) REFERENCES Miembros(miembro_id) ON DELETE CASCADE,
    FOREIGN KEY (nombre_preferencia) REFERENCES
PreferenciasDefault(nombre_preferencia) ON DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

2.3. Integridad y Validación de Datos

La integridad de los datos se garantizará mediante una estrategia de defensa en profundidad, combinando restricciones a nivel de base de datos con validación a nivel de aplicación.

- **Nivel de Base de Datos:** El uso de NOT NULL, UNIQUE, ENUM y CHECK (como en calificacion BETWEEN 1 AND 5 en FeedbackMiembro) impone reglas de datos en la capa más baja y segura. Esto previene la inserción de datos corruptos o inconsistentes, independientemente de la fuente de la aplicación.
- **Nivel de Aplicación:** Los validadores del ORM de Django y los serializadores de DRF se utilizarán para proporcionar mensajes de error claros y amigables al usuario antes de que una operación inválida llegue a la base de datos. Esto mejora la experiencia del usuario y reduce la carga en la base de datos.
- **Comprobaciones de Coherencia:** Inspirado en las prácticas de bases de datos de nivel empresarial [21], se puede programar una tarea administrativa periódica (usando Celery Beat) que ejecute comandos como
  CHECK TABLE en tablas críticas. Esto permite detectar y alertar sobre corrupción física o lógica de datos de forma proactiva.

III. El Motor Central: Servicios Backend con Django 4.2

El backend es el sistema nervioso central de la plataforma, orquestando la lógica de negocio, la seguridad y el acceso a los datos. Se ha seleccionado Django 4.2 por su madurez, su filosofía de "baterías incluidas" y su robusto ecosistema, que acelera el desarrollo sin sacrificar la calidad.[12]

3.1. Arquitectura de API con DRF

Se utilizará Django REST Framework (DRF) para construir la API, siguiendo un diseño orientado a recursos que es estándar en la industria y fácil de consumir por los clientes frontend.

- **Versionado:** La API se versionará mediante el versionado de la ruta URL (p. ej., /api/v1/...). Esto es crucial para garantizar la compatibilidad hacia atrás con las

aplicaciones móviles y web a medida que la API evoluciona.

- **Autenticación:** Se implementará un sistema de autenticación basado en tokens utilizando dj-rest-auth en conjunto con SimpleJWT. Este enfoque proporciona un flujo de autenticación seguro y estándar (tokens de acceso y de refresco) para las SPAs y aplicaciones móviles.
- **Permisos:** Las clases de permisos de DRF se utilizarán de forma extensiva para controlar el acceso a los recursos. Se definirá una jerarquía de permisos:
  - IsAuthenticated: Para la mayoría de los endpoints accesibles por usuarios logueados.
  - IsAdminUser: Para acciones administrativas (ej. creación de planes de membresía, gestión de personal).
  - Permisos Personalizados: Se crearán permisos específicos como IsOwnerOrAdmin para permitir que un usuario edite sus propios datos o que un administrador edite los datos de cualquier usuario.

3.2. Generación Automatizada de Esquemas con drf-spectacular

La documentación de la API y la generación de clientes se automatizarán utilizando drf-spectacular. Esta es una elección estratégica sobre el generador de esquemas incorporado en DRF, que ha sido obsoleto en favor de paquetes de terceros más potentes.[24]

drf-spectacular ofrece una extracción de esquemas más precisa y una mayor flexibilidad.[13]

El esquema OpenAPI 3.0 se generará automáticamente a través de un comando de gestión (./manage.py spectacular --file schema.yml --validate) dentro de la pipeline de CI/CD. Este esquema validado se convierte en un artefacto del build.

- **Configuración clave en settings.py:**

```
Python
SPECTACULAR_SETTINGS = {
    'TITLE': 'Pilates Fit API',
    'DESCRIPTION': 'API completa para el sistema de gestión del estudio Pilates Fit.',
    'VERSION': '1.0.0',
    'SERVE_INCLUDE_SCHEMA': False,  # El esquema se servirá estáticamente desde el artefacto de CI
    'COMPONENT_SPLIT_REQUEST': True, # Crucial para una generación de cliente limpia [26]
```

```
}
```

El uso de drf-spectacular es un pilar fundamental para el desacoplamiento entre el frontend y el backend. El esquema OpenAPI 3.0 generado por esta herramienta es legible por máquina, lo que permite alimentar herramientas como OpenAPI Generator para crear automáticamente bibliotecas de cliente API tipadas para los frontends de React (TypeScript) y Flutter (Dart). Este proceso automatiza la creación de Objetos de Transferencia de Datos (DTOs) y las llamadas de servicio de la API en el lado del cliente, eliminando la codificación manual de clientes API, reduciendo errores por tipos de datos no coincidentes y permitiendo que los equipos de frontend y backend trabajen en paralelo contra un contrato compartido y reforzado por máquina. La configuración 'COMPONENT_SPLIT_REQUEST': True [26] es especialmente importante, ya que crea componentes distintos para los modelos de solicitud (escritura) y respuesta (lectura), lo que es esencial para generar un código de cliente correcto y tipado.

3.3. Implementación de Servicios Centrales (Ejemplos de Flujo de Código)

Para ilustrar la implementación, se detalla el flujo de una operación de negocio clave: la reserva de una clase por un miembro.

- **Flujo: Miembro Reserva una Clase**
  1. **Cliente (App Móvil):** El usuario pulsa "Reservar". La aplicación envía una petición POST a /api/v1/asistencia/ con el cuerpo { "clase_programada_id": 123 }. El token de autenticación JWT se incluye en la cabecera Authorization.
  2. **Vista DRF (AsistenciaViewSet):**
     - El middleware de autenticación valida el token JWT y asocia el request.user con el Miembro correspondiente.
     - La clase de permisos (IsAuthenticated) verifica que el usuario esté logueado.
  3. **Serializador (AsistenciaCreateSerializer):**
     - Valida los datos de entrada. Comprueba que el campo clase_programada_id se ha proporcionado.
     - Realiza una validación a nivel de objeto (validate method) para asegurar que la ClaseProgramada con el ID 123 existe y su estado es 'Programada'.
  4. **Lógica de Servicio (dentro del método perform_create de la ViewSet):**

- Se inicia una transacción de base de datos con @transaction.atomic para garantizar la atomicidad de la operación.
- Se recupera la suscripción activa del miembro (SuscripcionesMiembro.objects.get(miembro=request.user, estado='activa')). Si no existe, se lanza una ValidationError.
- Se comprueba si el plan es de tipo 'Bono'. Si es así, se verifica que suscripcion.clases_restantes > 0.
- Se bloquea la fila de ClasesProgramadas para escritura (select_for_update()) para evitar condiciones de carrera. Se verifica que clase.capacidad_actual < clase.clase.capacidad_maxima_default.
- Si todas las comprobaciones pasan:
  - Se crea el nuevo registro de Asistencia.
  - Se decrementa suscripcion.clases_restantes si corresponde.
  - Se incrementa clase.capacidad_actual.
  - Se guardan los objetos actualizados.

5. **Respuesta:** La API devuelve un código de estado 201 CREATED con los datos del nuevo objeto Asistencia serializado.

A continuación, se presenta una tabla resumen de los endpoints clave de la API.

Tabla de Resumen de Endpoints de la API

Endpoint	Descripción	ViewSet/View	Permisos	Cuerpo de Solicitud (Serializer)	Respuesta Exitosa
POST /api/v1/asistencia/	Reserva a un miembro en una clase programada.	AsistenciaViewSet	IsAuthenticated	AsistenciaCreateSerializer	201 CREATED, AsistenciaDetailSerializer
GET /api/v1/miembros/me/	Obtiene los detalles del miembro autenticado.	MiembroDetailView	IsAuthenticated	N/A	200 OK, MiembroDetailSerializer
PATCH /api/v1/miembros/{id}/	Actualiza los datos de un miembro.	MiembroViewSet	IsOwnerOrAdmin	MiembroUpdateSerializer	200 OK, MiembroDetailSerializer
GET	Lista las	ClasesProgr	IsAuthenticat	N/A (Query	200 OK, List

/api/v1/clases-programadas/	clases programadas con filtros.	amadasViewSet	ed	Params)	
POST /auth/login/	Autentica a un usuario y devuelve tokens.	LoginView (dj-rest-auth)	AllowAny	LoginSerializer	200 OK, {access_token, refresh_token}

IV. Capacidades Asíncronas y en Tiempo Real

Para garantizar una experiencia de usuario fluida y descargar el procesamiento pesado del ciclo de solicitud-respuesta, el sistema empleará colas de tareas y mensajería en tiempo real.

4.1. Procesamiento de Tareas en Segundo Plano con Celery

- **Casos de Uso:** Envío de correos electrónicos de bienvenida, procesamiento de informes financieros al final del día, envío de notificaciones push programadas, desactivación de suscripciones caducadas, y cualquier otra operación que pueda tardar más de unos pocos cientos de milisegundos.
- **Arquitectura:**
  - **Broker:** Se utilizará Redis como intermediario de mensajes por su velocidad y simplicidad, lo cual es suficiente para las necesidades de este sistema.[14] Aunque RabbitMQ ofrece mayores garantías de durabilidad, Redis a menudo ya está presente en la pila tecnológica para el almacenamiento en caché, lo que reduce la sobrecarga operativa.[14]
  - **Celery Worker:** Un contenedor separado ejecutará el proceso de trabajo de Celery, que escuchará las tareas en la cola de Redis y las ejecutará. Se pueden escalar horizontalmente añadiendo más contenedores de workers.
  - **Celery Beat:** Un proceso planificador, Celery Beat, se utilizará para tareas periódicas. Se gestionará a través de django-celery-beat, que permite definir horarios desde la base de datos y el panel de administración de Django.[27]
- **Ejemplo: Verificación Programada del Estado de las Suscripciones**

1. **Configuración:** En el panel de administración de Django, se crea una PeriodicTask utilizando django-celery-beat.
2. **Horario:** Se asocia con un CrontabSchedule configurado para ejecutarse diariamente a una hora de baja actividad (p. ej., a las 3:00 AM).[28] La expresión cron sería
   minute='0', hour='3'.
3. **Tarea:** La tarea apunta a una función en tasks.py, por ejemplo tasks.check_expired_subscriptions.
4. **Lógica de la Tarea:**

```python
from celery import shared_task
from django.utils import timezone
from.models import SuscripcionesMiembro

@shared_task
def check_expired_subscriptions():
    today = timezone.now().date()
    expired_subs = SuscripcionesMiembro.objects.filter(
        fecha_fin__lt=today,
        estado='activa'
    )
    count = expired_subs.update(estado='vencida')
    return f"{count} suscripciones actualizadas a 'vencida'."
```

4.2. Comunicación en Tiempo Real con Django Channels

- **Casos de Uso:** Actualización instantánea de la disponibilidad de clases en la pantalla de reservas para todos los usuarios conectados, una función de chat simple entre el personal y los miembros, notificaciones en tiempo real en el panel de administración (p. ej., "Nuevo miembro registrado").
- **Arquitectura:**
  - Django Channels extiende Django para manejar protocolos asíncronos como WebSockets, superando la naturaleza síncrona de WSGI.[16]
  - **Servidor ASGI:** Daphne actuará como el servidor ASGI de cara al cliente, gestionando las conexiones WebSocket entrantes.[17]
  - **Capa de Canal (Channel Layer):** Se utilizará Redis como backend de la capa

de canal. Esto permite que diferentes instancias de la aplicación (conocidas como *consumers*) se comuniquen entre sí, lo cual es esencial para la difusión de mensajes a grupos de usuarios.[16]

- **Ejemplo: Actualización en Vivo de la Capacidad de una Clase**
  1. **Disparador:** Cuando un miembro reserva una clase con éxito (como se describe en la sección 3.3), después de que la transacción de la base de datos se confirme, se emite una señal de Django (post_save en el modelo Asistencia).
  2. **Manejador de Señal:** El manejador de la señal recupera la clase_programada_id y su nueva capacidad_actual.
  3. **Envío al Grupo:** El manejador envía un mensaje a un grupo de canal específico de esa clase. El nombre del grupo podría ser, por ejemplo, clase_123.

     Python
     ```
     from channels.layers import get_channel_layer
     from asgiref.sync import async_to_sync

     channel_layer = get_channel_layer()
     async_to_sync(channel_layer.group_send)(
        "clase_123",
        {
            "type": "capacity.update",
            "new_capacity": 15,
            "max_capacity": 20
        }
     )
     ```

  4. **Consumidor WebSocket (WebsocketConsumer):** Todas las instancias de WebsocketConsumer que se hayan suscrito al grupo clase_123 recibirán este evento.[29]
  5. **Recepción y Envío al Cliente:** El consumidor tiene un método que coincide con el type del mensaje (capacity_update). Este método formatea los datos y los envía a través de la conexión WebSocket a los clientes conectados.

     Python
     ```
     # En el WebsocketConsumer
     async def capacity_update(self, event):
        await self.send(text_data=json.dumps({
            'type': 'CAPACITY_UPDATE',
            'payload': {
     ```

```
            'current': event['new_capacity'],
            'max': event['max_capacity']
        }
    }))
```

6. **Cliente Frontend:** El código JavaScript o Dart en el cliente escucha los mensajes entrantes en el WebSocket y actualiza la interfaz de usuario en tiempo real sin necesidad de recargar la página.

V. Arquitectura de la Interfaz de Usuario: Frontends Web y Móvil

El sistema requiere dos aplicaciones de cara al usuario distintas, cada una adaptada a su audiencia y aprovechando una pila tecnológica moderna y de alto rendimiento. Esta separación deliberada de las interfaces de usuario es una decisión estratégica. El portal de administración necesita tablas de datos complejas, formularios y paneles, un dominio en el que el ecosistema de React (con librerías como TanStack Table y Recharts) sobresale. La experiencia del usuario se centra en la densidad de datos y la funcionalidad. Por otro lado, la aplicación móvil para miembros prioriza una sensación nativa y fluida, animaciones rápidas y posibles capacidades sin conexión, que son los puntos fuertes de Flutter.[11] Aunque una PWA o una única aplicación de React Native podrían haber servido para ambos, este enfoque de doble pila optimiza la experiencia para cada grupo de usuarios sin compromisos, reconociendo que las necesidades de un administrador de back-office son fundamentalmente diferentes de las de un miembro del gimnasio en movimiento.

5.1. El Portal Web Administrativo (React 18)

- **Propósito:** Una completa Aplicación de Página Única (SPA) para que el personal y los administradores gestionen todos los aspectos del negocio.
- **Tecnología Principal:** React 18.[8] Se aprovecharán las nuevas características como el
  Automatic Batching para agrupar actualizaciones de estado y las *Transitions* para mantener la interfaz de usuario receptiva durante las renderizaciones de datos.
- **Librerías Clave:**

    - **Tablas de Datos:** Se utilizará TanStack Table (anteriormente React Table) para todas las visualizaciones de datos tabulares (miembros, clases, pagos). Su naturaleza *headless* (sin UI predefinida) proporciona la máxima flexibilidad para un estilo personalizado que se ajuste a la marca. Se implementará la ordenación, el filtrado y la paginación del lado del servidor, haciendo que el componente de la tabla active llamadas a la API de DRF para obtener los datos necesarios.[31]
    - **Visualización de Datos:** Se utilizará Recharts para construir los *widgets* del panel de control, como los ingresos a lo largo del tiempo, las tendencias de asistencia a las clases y el crecimiento de las membresías. Su API de componentes componibles es ideal para crear gráficos personalizados.[33]
    - **Gestión de Formularios:** Se utilizará React Hook Form para todos los formularios (creación de miembros, programación de clases, etc.). Ofrece un rendimiento superior y una mejor experiencia para el desarrollador al minimizar las re-renderizaciones y simplificar la validación.[36]
- **Gestión del Estado:** Para el estado local y simple, bastará con los hooks incorporados de React (useState, useReducer). Para el estado global complejo (p. ej., información del usuario autenticado, configuraciones de todo el sistema), se preferirá una librería ligera como Zustand o Jotai en lugar de Redux para reducir el código repetitivo y la complejidad.

5.2. La Aplicación Móvil para Miembros (Flutter 3.10)

- **Propósito:** Una aplicación de alto rendimiento y multiplataforma (iOS y Android) para que los miembros reserven clases, sigan su progreso y gestionen su cuenta.
- **Tecnología Principal:** Flutter 3.10, aprovechando las actualizaciones de diseño de Material 3, las mejoras de rendimiento del motor de renderizado Impeller (especialmente en iOS) y la seguridad de nulos completa en Dart 3.[10]
- **Librerías Clave:**
    - **Visualización de Datos:** Se utilizará syncfusion_flutter_charts para mostrar el progreso específico del miembro a partir de las tablas LogsEntrenamiento y MetasMiembro, como la pérdida de peso a lo largo del tiempo o las ganancias de fuerza. Ofrece un rico conjunto de gráficos interactivos adecuados para dispositivos móviles.[40]
    - **Notificaciones Push:** Se integrará el plugin oficial de Firebase Messaging para Flutter para recibir las notificaciones enviadas desde el backend a través de fcm-django.[42]

    - **Cliente de API:** La especificación OpenAPI generada por drf-spectacular se utilizará para generar un cliente de Dart tipado, agilizando las solicitudes de red y garantizando la coherencia con el backend.
- **Gestión del Estado:** Se elegirá una solución de gestión de estado robusta como Riverpod o Bloc para gestionar el estado de la aplicación, separando la lógica de negocio de la interfaz de usuario y facilitando las pruebas.

VI. Capa de Inteligencia: Integración de IA y Machine Learning

Esta sección detalla la implementación de características avanzadas de IA que diferenciarán a Pilates Fit del software de gestión estándar, proporcionando un valor añadido significativo tanto para los administradores como para los miembros.

6.1. Consulta de Base de Datos en Lenguaje Natural (Text-to-SQL)

- **Objetivo:** Proporcionar a los administradores una potente herramienta de análisis donde puedan hacer preguntas en lenguaje natural (p. ej., "Muéstrame los ingresos totales de los planes 'Completo' del mes pasado") y recibir una tabla de datos como resultado, democratizando el acceso a los datos.
- **Implementación:**
    1. **Selección del Modelo:** Se utilizará un modelo T5 pre-entrenado (p. ej., t5-base o t5-small para un equilibrio entre rendimiento y coste computacional) de Hugging Face, que ha demostrado ser eficaz para tareas de texto a texto.[44]
    2. ***Fine-tuning* (Ajuste Fino):** El modelo se ajustará con un conjunto de datos creado específicamente a partir de nuestro esquema MySQL. Esto implica generar pares de pregunta-SQL. El contexto para el prompt será el conjunto de sentencias CREATE TABLE de las tablas relevantes.[18] Este proceso seguirá la metodología descrita en la investigación, que ha demostrado alcanzar una alta precisión.[46]
    3. **Ingeniería de Prompts:** El prompt se estructurará de forma clara para guiar al modelo: Tablas: <esquema> \n\nPregunta: <pregunta en lenguaje natural> \n\nRespuesta:. Esta estructura proporciona al modelo toda la información necesaria para generar una consulta SQL válida.

4. **Despliegue:** El modelo ajustado se alojará como un microservicio independiente (o en una función sin servidor como AWS Lambda o Google Cloud Functions) al que el backend de Django podrá llamar. Esto aísla la carga de trabajo de la IA de la aplicación principal.
5. **Ejecución y Seguridad:** El SQL generado **no se ejecutará directamente**. Este es un paso de seguridad crítico. La consulta SQL será primero analizada y validada utilizando una librería como sqlparse [48] para asegurar que es únicamente una sentencia
   SELECT, que no contiene comandos maliciosos (como DROP, UPDATE, DELETE) y que solo consulta las tablas permitidas. Una vez validada, la consulta se ejecutará contra una réplica de la base de datos de solo lectura para aislar las cargas de trabajo de análisis de la base de datos de producción.

6.2. Visión por Computadora en el Dispositivo (Hoja de Ruta Futura)

- **Objetivo:** Ofrecer a los miembros retroalimentación en tiempo real sobre la forma de sus ejercicios o permitirles registrar su progreso a través de imágenes, creando una experiencia de entrenamiento más interactiva y basada en datos.
- **Propuesta de Implementación:**
  1. **Entrenamiento del Modelo:** Se entrenará un modelo personalizado de detección de objetos o estimación de pose (p. ej., basado en EfficientNet o MobileNet) para reconocer ejercicios clave de Pilates y posturas corporales.
  2. **Conversión:** El modelo entrenado de TensorFlow/Keras se convertirá al formato .tflite utilizando el TFLiteConverter.[49] Este formato está optimizado para la inferencia en dispositivos móviles.
  3. **Despliegue:** El modelo .tflite se empaquetará directamente dentro de la aplicación móvil Flutter, permitiendo la inferencia sin conexión a internet y garantizando una baja latencia y la privacidad de los datos del usuario.
  4. **Preprocesamiento:** La aplicación utilizará la cámara del dispositivo y una librería como OpenCV (o un equivalente en Dart) para preprocesar los fotogramas de vídeo. Esto incluye redimensionar los fotogramas al tamaño de entrada esperado por el modelo (p. ej., 224x224) y normalizar los valores de los píxeles antes de la inferencia.[51]

6.3. Asistente de IA Conversacional (Rasa)

- **Objetivo:** Proporcionar un chatbot automatizado dentro de las aplicaciones móvil y web para responder a preguntas frecuentes ("¿A qué hora es la próxima clase de Reformer?") y realizar acciones ("Resérvame en la clase de las 7 PM de mañana").
- **Arquitectura:**
  1. **Rasa NLU:** Se entrenará para entender las intenciones del usuario (p. ej., find_class, book_class) y extraer entidades (p. ej., class_name="Reformer", time="7 PM"). El archivo data/nlu.yml definirá estos datos de entrenamiento.[55]
  2. **Rasa Core:** Gestionará el flujo del diálogo basándose en la salida de NLU, decidiendo la siguiente acción a tomar.
  3. **Servidor de Acciones Personalizadas (Custom Action Server):** Este es el punto de integración crítico. Se ejecutará un servidor de acciones independiente. Cuando Rasa prediga una acción personalizada (p. ej., action_check_availability), hará una llamada API a este servidor.[56] Este servidor será una aplicación ligera de Python (p. ej., Flask o Sanic) que configurará los ajustes de Django para poder utilizar su ORM. Esto le permitirá consultar la base de datos de Pilates Fit directamente, garantizando el acceso a datos en tiempo real (como la capacidad actual de una clase) sin duplicar la lógica de negocio y proporcionando una interfaz segura a los datos de la aplicación principal.[58]

VII. Estrategia de Despliegue y Orquestación

Una estrategia de despliegue moderna y basada en contenedores es esencial para la fiabilidad, escalabilidad y mantenibilidad del sistema Pilates Fit.

7.1. Contenerización con Docker

Cada servicio definido en el Diagrama de Contenedores (Sección 1.3) se empaquetará en su propia imagen Docker. Esto garantiza la coherencia del entorno desde el

desarrollo local hasta la producción.

- Se crearán Dockerfiles individuales para cada servicio:
  - backend.Dockerfile: Para la aplicación Django/DRF/Channels.
  - worker.Dockerfile: Para el trabajador de Celery.
  - frontend.Dockerfile: Un build de varias etapas para la aplicación React, que dará como resultado un servidor estático Nginx que sirve los archivos compilados.
- Se creará un archivo docker-compose.yml para el desarrollo local, permitiendo levantar toda la pila tecnológica (backend, frontend, base de datos, Redis) con un solo comando.

7.2. Orquestación: Kubernetes para Producción

- **Justificación:** Aunque Docker Swarm es más simple [60], la naturaleza de microservicios de Pilates Fit (API, workers, pasarela en tiempo real, múltiples frontends, futuros servicios de IA) hace que Kubernetes sea la opción superior para producción. K8s ofrece una red avanzada, auto-reparación, auto-escalado y gestión de secretos, que son críticos para un sistema de esta complejidad y ambición.[61] Su vasto ecosistema y el hecho de ser el estándar de la industria garantizan un soporte a largo plazo.
- **Manifiestos de Kubernetes:** Se definirán archivos YAML para describir el estado deseado del sistema en el clúster:
  - **Deployments:** Para los servicios sin estado como el backend, el worker y el frontend. Kubernetes se encargará de mantener el número deseado de réplicas en ejecución.
  - **Services:** Para exponer los deployments dentro del clúster (ClusterIP) o al mundo exterior (LoadBalancer o NodePort).
  - **Ingress:** Un controlador de Ingress (p. ej., Nginx Ingress Controller) gestionará el enrutamiento del tráfico externo a los servicios correctos basándose en el host y la ruta (p. ej., las peticiones a pilatesfit.com/api/* se dirigirán al servicio del backend, mientras que pilatesfit.com/ se dirigirá al servicio del frontend).
  - **ConfigMaps y Secrets:** Para gestionar la configuración no sensible y las credenciales sensibles (claves de API, contraseñas de la base de datos) de forma segura, separándolas de las imágenes de los contenedores.
  - **StatefulSet:** Para la instancia de Redis, para garantizar una identidad de red

estable y un almacenamiento persistente si es necesario.
- **Base de Datos Externa:** La base de datos MySQL se gestionará a través de un servicio de base de datos administrado por un proveedor de la nube (como AWS RDS o Google Cloud SQL). Esto descarga la complejidad de la gestión, las copias de seguridad y la alta disponibilidad de la base de datos al proveedor de la nube, lo que permite al equipo de DevOps centrarse en la aplicación.

VIII. Marco Integral de Seguridad y Garantía de Calidad

La seguridad y la calidad no son consideraciones posteriores, sino que están integradas en el proceso de desarrollo desde el principio, siguiendo los principios de DevSecOps.

8.1. Pipeline de CI/CD con GitLab CI

Se definirá un archivo .gitlab-ci.yml que orquestará la pipeline de Integración Continua y Despliegue Continuo, activándose en cada *push* y *merge request*. La pipeline se estructurará en etapas para proporcionar retroalimentación rápida y eficiente.

Una pipeline de varias etapas optimiza el tiempo de los desarrolladores y el uso de recursos. Las comprobaciones rápidas como el *linting* y las pruebas unitarias se ejecutan primero. Las pruebas más lentas y exhaustivas, como los escaneos de seguridad, se ejecutan más tarde en el proceso. Este enfoque escalonado asegura que los errores básicos se detecten de inmediato.

Tabla de Etapas de la Pipeline de CI/CD

Etapa	Herramientas	Propósito y Descripción
Lint	flake8, prettier, eslint	**Análisis de Estilo de Código:** Se asegura de que todo el código (Python y

		JavaScript/TypeScript) se adhiera a las guías de estilo del proyecto. Falla rápidamente si hay errores de sintaxis o estilo, proporcionando retroalimentación inmediata.
Test	pytest (con pytest-cov), jest	**Pruebas Unitarias y de Integración:** Ejecuta el conjunto de pruebas automatizadas para el backend y el frontend. Genera informes de cobertura de código. La pipeline falla si alguna prueba no pasa o si la cobertura cae por debajo de un umbral definido.
Security-SAST	SonarQube	**Pruebas de Seguridad de Aplicaciones Estáticas:** Escanea el código fuente en busca de vulnerabilidades de seguridad conocidas, errores y "code smells". La pipeline se detiene si la puerta de calidad definida en SonarQube no se cumple (p. ej., no se permiten nuevas vulnerabilidades críticas).[63]
Build	docker build	**Construcción de Artefactos:** Construye las imágenes Docker para cada servicio y las sube al registro de contenedores de GitLab, etiquetándolas con el hash del commit para su trazabilidad.
Deploy-Staging	kubectl apply	**Despliegue en Entorno de Pruebas:** Despliega las nuevas imágenes Docker en un entorno de *staging* (pre-producción) que es una

		réplica del entorno de producción.
Security-DAST	OWASP ZAP	**Pruebas de Seguridad de Aplicaciones Dinámicas:** Ejecuta un escaneo de seguridad automatizado contra la aplicación en vivo en el entorno de *staging*. La pipeline falla si se descubren vulnerabilidades de alto riesgo.[64]
Deploy-Prod	kubectl apply (Disparo Manual)	**Despliegue en Producción:** Una vez que todas las etapas anteriores han pasado, un miembro autorizado del equipo puede disparar manualmente el despliegue a producción, promoviendo las imágenes validadas.

8.2. Escaneo de Seguridad Automatizado

- SonarQube (SAST - Static Application Security Testing):
  Integrado en la pipeline de CI, SonarQube analizará las bases de código de Django y React en cada merge request. Proporcionará un análisis detallado de la calidad del código, identificando vulnerabilidades de seguridad (como inyección de SQL, XSS), bugs y "code smells". Se configurará una "Puerta de Calidad" que definirá las condiciones para que una build pase (p. ej., 0 nuevas vulnerabilidades de tipo "Blocker", cobertura de código > 80%). Si no se cumple, la pipeline fallará, impidiendo que el código de baja calidad se fusione.63
- OWASP ZAP (DAST - Dynamic Application Security Testing):
  La pipeline orquestará un escaneo de línea base de ZAP (Baseline Scan) contra el entorno de staging después de cada despliegue exitoso en dicho entorno.64 El script
  zap-baseline.py se ejecutará dentro de su contenedor Docker oficial. La configuración del .gitlab-ci.yml especificará la URL de destino (la aplicación en *staging*) y las reglas de fallo. Por ejemplo, la pipeline puede configurarse para analizar el informe HTML o Markdown generado por ZAP y fallar si se encuentra

alguna alerta de riesgo "Alto".[65] Esto crea una red de seguridad automatizada que captura vulnerabilidades de la aplicación en ejecución antes de que lleguen a producción.

Ejemplo de un *job* de GitLab CI para ZAP:

```yaml
zap_baseline_scan:
  stage: Security-DAST
  image:
    name: owasp/zap2docker-stable
    entrypoint: [""]
  script:
    - mkdir -p /zap/wrk/
    - /zap/zap-baseline.py -t $STAGING_URL -g gen_rules.conf -r report.html
    # Script adicional para analizar report.html y fallar si hay alertas de alto riesgo
  artifacts:
    paths:
      - /zap/wrk/report.html
    when: always
```

Conclusiones

La guía técnica presentada detalla un plan de construcción para el sistema **Pilates Fit** que va más allá de un simple software de gestión. La arquitectura propuesta está diseñada para ser **escalable, segura y preparada para el futuro**, sentando las bases para una plataforma que puede evolucionar con las necesidades del negocio y las expectativas de los usuarios.

Las decisiones clave, como la adopción de una **arquitectura de microservicios contenerizada y orquestada con Kubernetes**, el uso de **pilas tecnológicas separadas y optimizadas para los frontends web y móvil**, y la **integración proactiva de una capa de inteligencia artificial**, no son meramente elecciones técnicas, sino inversiones estratégicas. Estas decisiones garantizan que el sistema pueda manejar el crecimiento, ofrecer experiencias de usuario superiores y proporcionar ventajas competitivas a través de funcionalidades basadas en datos.

La implementación de un **marco integral de DevSecOps**, con "Diagramas como

Código", CI/CD automatizado y escaneos de seguridad SAST y DAST, asegura que la calidad y la seguridad sean componentes intrínsecos del proceso de desarrollo, no una ocurrencia tardía.

Se recomienda encarecidamente seguir este plan de manera rigurosa. La coherencia en la implementación de las arquitecturas, tecnologías y flujos de trabajo aquí definidos será fundamental para lograr los objetivos del proyecto y construir un producto de software de primer nivel que impulse el éxito del negocio de Pilates Fit.

Obras citadas

1. C4 model - Wikipedia, fecha de acceso: julio 27, 2025, https://en.wikipedia.org/wiki/C4_model
2. How to Create Software Architecture Diagrams Using the C4 Model - freeCodeCamp, fecha de acceso: julio 27, 2025, https://www.freecodecamp.org/news/how-to-create-software-architecture-diagrams-using-the-c4-model/
3. How to master your architecture with the C4 model and Structurizr ..., fecha de acceso: julio 27, 2025, https://www.schibsted.pl/blog/how-to-master-your-architecture-with-the-c4-model-and-structurizr/
4. Software Architecture as Code With Structurizr - mydeveloperplanet.com, fecha de acceso: julio 27, 2025, https://mydeveloperplanet.com/2024/03/20/software-architecture-as-code-with-structurizr/
5. USDA Nutrients - Public APIs, fecha de acceso: julio 27, 2025, https://publicapis.io/usda-nutrients-api
6. FoodData Central API Guide, fecha de acceso: julio 27, 2025, https://fdc.nal.usda.gov/api-guide
7. API Guide | USDA FoodData Central, fecha de acceso: julio 27, 2025, https://fdc.nal.usda.gov/api-guide.html
8. React 18: The Features and The Future | Medium, fecha de acceso: julio 27, 2025, https://medium.com/@ojebiyifulness/react-18-the-features-and-the-future-db3158c4a24a
9. What's new in React 18? - Webkul, fecha de acceso: julio 27, 2025, https://webkul.com/blog/whats-new-in-react-18/
10. What's New in Flutter 3.10: Know About the Latest Updates - DhiWise, fecha de acceso: julio 27, 2025, https://www.dhiwise.com/post/whats-new-in-flutter-3-10-the-latest-updates-and-features
11. Flutter 3.10 and Dart 3: The Ultimate Upgrade for Speed, Security, and UI Enhancements, fecha de acceso: julio 27, 2025, https://www.techspian.com/blog/googles-flutter-3-10-and-dart-3-a-dive-into-the-latest-update/

12. Exploring Django 4.2 LTS: New Features and Enhancements | by Preet Sharma - Medium, fecha de acceso: julio 27, 2025, https://medium.com/@ITservices_expert/exploring-django-4-2-lts-new-features-and-enhancements-87eba0f75569
13. tfranzel/drf-spectacular: Sane and flexible OpenAPI 3 schema generation for Django REST framework. - GitHub, fecha de acceso: julio 27, 2025, https://github.com/tfranzel/drf-spectacular
14. Choosing The Right Python Task Queue - Judoscale, fecha de acceso: julio 27, 2025, https://judoscale.com/blog/choose-python-task-queue
15. Pros and cons to use Celery vs. RQ [closed] - python - Stack Overflow, fecha de acceso: julio 27, 2025, https://stackoverflow.com/questions/13440875/pros-and-cons-to-use-celery-vs-rq
16. Django Channels vs WebSockets: What's the difference? - Ably, fecha de acceso: julio 27, 2025, https://ably.com/topic/django-channels-vs-websockets-whats-the-difference
17. Django Channels — Channels 4.2.0 documentation, fecha de acceso: julio 27, 2025, https://channels.readthedocs.io/
18. CREATE TABLES ~~ mysql.pdf
19. Redis as a Message Broker: Deep Dive - DEV Community, fecha de acceso: julio 27, 2025, https://dev.to/nileshprasad137/redis-as-a-message-broker-deep-dive-3oek
20. Create ER diagram from sql file in mysql Workbench - Stack Overflow, fecha de acceso: julio 27, 2025, https://stackoverflow.com/questions/19480976/create-er-diagram-from-sql-file-in-mysql-workbench
21. Database Consistency Checker (DBCC) for Analysis Services - Learn Microsoft, fecha de acceso: julio 27, 2025, https://learn.microsoft.com/en-us/analysis-services/instances/database-consistency-checker-dbcc-for-analysis-services?view=asallproducts-allversions
22. How can I check database consistency to avoid schema corruption? - IBM, fecha de acceso: julio 27, 2025, https://www.ibm.com/docs/en/workload-automation/10.2.2?topic=fdc-how-can-i-check-database-consistency-avoid-schema-corruption
23. Django 4.2 documentation - DevDocs, fecha de acceso: julio 27, 2025, https://devdocs.io/django~4.2/
24. Schemas - Django REST framework, fecha de acceso: julio 27, 2025, https://www.django-rest-framework.org/api-guide/schemas/
25. drf-spectacular documentation, fecha de acceso: julio 27, 2025, https://drf-spectacular.readthedocs.io/
26. Client generation - drf-spectacular documentation, fecha de acceso: julio 27, 2025, https://drf-spectacular.readthedocs.io/en/latest/client_generation.html
27. Periodic Tasks — Celery 5.5.3 documentation, fecha de acceso: julio 27, 2025, https://docs.celeryq.dev/en/stable/userguide/periodic-tasks.html
28. django-celery-beat - Database-backed Periodic Tasks ..., fecha de acceso: julio

27, 2025, https://django-celery-beat.readthedocs.io/
29. Introduction to Django Channels and WebSockets | by Ordinary Programmer - Medium, fecha de acceso: julio 27, 2025, https://medium.com/@adabur/introduction-to-django-channels-and-websockets-cb38cd015e29
30. React 18: New Features and Improvements - Simplilearn.com, fecha de acceso: julio 27, 2025, https://www.simplilearn.com/tutorials/reactjs-tutorial/react-eighteen-new-features
31. A complete guide to TanStack Table (formerly React Table ..., fecha de acceso: julio 27, 2025, https://www.contentful.com/blog/tanstack-table-react-table/
32. A complete guide to TanStack Table (formerly React Table) - LogRocket Blog, fecha de acceso: julio 27, 2025, https://blog.logrocket.com/tanstack-table-formerly-react-table/
33. recharts examples - CodeSandbox, fecha de acceso: julio 27, 2025, https://codesandbox.io/examples/package/recharts
34. Top 10 Examples of "recharts in functional component" in JavaScript - CloudDefense.AI, fecha de acceso: julio 27, 2025, https://www.clouddefense.ai/code/javascript/example/recharts
35. Recharts Example - StackBlitz, fecha de acceso: julio 27, 2025, https://stackblitz.com/edit/recharts-example
36. Get Started - React Hook Form, fecha de acceso: julio 27, 2025, https://react-hook-form.com/get-started
37. React Hook Form Complete Tutorial - YouTube, fecha de acceso: julio 27, 2025, https://www.youtube.com/watch?v=lp8xXoG0xQY
38. Form React Hook Basics for Beginners - Daily.dev, fecha de acceso: julio 27, 2025, https://daily.dev/blog/form-react-hook-basics-for-beginners
39. Cross-Platform Development Framework Flutter Gets A Major Update With 3.10 Version, fecha de acceso: julio 27, 2025, https://www.prismetric.com/latest-flutter-version/
40. Flutter Charts | Beautiful & Interactive Live Charts - Syncfusion, fecha de acceso: julio 27, 2025, https://www.syncfusion.com/flutter-widgets/flutter-charts
41. Overview of Syncfusion Flutter Charts, fecha de acceso: julio 27, 2025, https://help.syncfusion.com/flutter/backup/chart/overview
42. Firebase Cloud Messaging, fecha de acceso: julio 27, 2025, https://firebase.google.com/docs/cloud-messaging
43. fcm-django — fcm-django latest documentation, fecha de acceso: julio 27, 2025, https://fcm-django.readthedocs.io/en/latest/
44. Priyanshu05/text-to-sql-t5 - Hugging Face, fecha de acceso: julio 27, 2025, https://huggingface.co/Priyanshu05/text-to-sql-t5
45. From Natural Language to SQL: How to Build a Text-to-SQL Application Using LangChain and Hugging Face | by Elif Besli | Jun, 2025 | Medium, fecha de acceso: julio 27, 2025, https://medium.com/@elif.besli/from-natural-language-to-sql-how-to-build-a-text-to-sql-application-using-langchain-and-hugging-ea88d17c28e1

46. AnMol12499/Text-to-SQL-using-T5-Model: Natural Language to SQL - GitHub, fecha de acceso: julio 27, 2025, https://github.com/AnMol12499/Text-to-SQL-using-T5-Model
47. Fine tuning a T5 small model to generate SQL from natural ... - Medium, fecha de acceso: julio 27, 2025, https://medium.com/@martinkeywood/fine-tuning-a-t5-small-model-to-generate-sql-from-natural-language-with-92-3-accuracy-fb29e062c638
48. SQL Parsing with Python, Pt. II | Andi Albrecht, fecha de acceso: julio 27, 2025, https://andialbrecht.wordpress.com/2009/03/29/sql-parsing-with-python-pt-ii/
49. tf.lite.TFLiteConverter | TensorFlow v2.16.1, fecha de acceso: julio 27, 2025, https://www.tensorflow.org/api_docs/python/tf/lite/TFLiteConverter
50. TensorFlow Lite Model Maker | Google AI Edge - Gemini API, fecha de acceso: julio 27, 2025, https://ai.google.dev/edge/litert/libraries/modify
51. Build and deploy a custom object detection model with TensorFlow Lite (Android), fecha de acceso: julio 27, 2025, https://developers.google.com/codelabs/tflite-object-detection-android
52. Resizing and Rescaling Images with OpenCV, fecha de acceso: julio 27, 2025, https://opencv.org/blog/resizing-and-rescaling-images-with-opencv/
53. Normalize an Image in OpenCV Python - GeeksforGeeks, fecha de acceso: julio 27, 2025, https://www.geeksforgeeks.org/computer-vision/normalize-an-image-in-opencv-python/
54. Why should we always resize input images to 224x224 for VGG classification? I have tried building the network with other larger size input, but out of memory while out convolution and come into fully connected layer, is that the reason? - Quora, fecha de acceso: julio 27, 2025, https://www.quora.com/Why-should-we-always-resize-input-images-to-224x224-for-VGG-classification-I-have-tried-building-the-network-with-other-larger-size-input-but-out-of-memory-while-out-convolution-and-come-into-fully-connected-layer
55. Chatbots Using Python and Rasa - GeeksforGeeks, fecha de acceso: julio 27, 2025, https://www.geeksforgeeks.org/machine-learning/chatbots-using-python-and-rasa/
56. Writing Custom Actions | Rasa Documentation, fecha de acceso: julio 27, 2025, https://rasa.com/docs/pro/build/custom-actions/
57. Introduction to Rasa Action Server | Rasa Documentation, fecha de acceso: julio 27, 2025, https://rasa.com/docs/action-server/
58. Smart Marketing with Django, Rasa NLU, and Vonage, fecha de acceso: julio 27, 2025, https://developer.vonage.com/en/blog/smart-marketing-with-django-rasa-nlu-and-vonage
59. RASA - Custom forms - DEV Community, fecha de acceso: julio 27, 2025, https://dev.to/petr7555/rasa-custom-forms-3d20
60. Docker Swarm vs Kubernetes | Blog - Northflank, fecha de acceso: julio 27, 2025,

https://northflank.com/blog/docker-swarm-vs-kubernetes
61. Kubernetes vs. Docker Swarm: Choosing the Right Orchestration Tool for Your Team, fecha de acceso: julio 27, 2025, https://cloudfleet.ai/blog/cloud-native-how-to/2024-09-docker-swarm-vs-kubernetes/
62. Docker Swarm vs. Kubernetes - Key Differences Explained - Spacelift, fecha de acceso: julio 27, 2025, https://spacelift.io/blog/docker-swarm-vs-kubernetes
63. ridwanray/django-sonarqube-pytest-coverage: Measure ... - GitHub, fecha de acceso: julio 27, 2025, https://github.com/ridwanray/django-sonarqube-pytest-coverage
64. Setting up ZAP Scan in CI/CD pipeline - Cloufish's Blog, fecha de acceso: julio 27, 2025, https://cloufish.github.io/blog/posts/Setting-up-zap-scan-in-cicd-pipeline/
65. How to integrate OWASP ZAP in Gitlab CI/CD pipeline. - Codific, fecha de acceso: julio 27, 2025, https://codific.com/integrate-owasp-zap-and-gitlab/
66. zap-baseline - GitLab.org, fecha de acceso: julio 27, 2025, https://gitlab.com/gitlab-org/zap-baseline/-/tree/master
67. How to integrate ZAP in GitLab CI/CD pipeline. - Codific, fecha de acceso: julio 27, 2025, https://codific.com/how-to-integrate-zap-in-gitlab/