FitFlow: Sistema de Gestión Integral para Gimnasios

Trabajo Práctico Anual Integrador - 2025

Contexto General

Contexto

La gestión eficiente de un gimnasio moderno depende de la correcta organización de múltiples flujos de información: horarios de clases, disponibilidad de entrenadores, inscripciones de socios y gestión de pagos. La centralización de esta información permite optimizar el uso de los recursos y mejorar significativamente la experiencia del socio.

Para que esta gestión sea exitosa, es crucial ordenar, clasificar y disponibilizar la información en tiempo real. Esto permite, por ejemplo, conocer la ocupación de una clase al instante, prevenir sobrecupos, visibilizar los horarios más demandados, identificar oportunidades para nuevas clases o gestionar eficientemente las membresías y sus vencimientos.

Para estos fines, contar con un sistema de información robusto, que organice la información de manera centralizada y accesible, es fundamental.

Nuestro Sistema

FitFlow será un sistema de código abierto encargado por una cadena de gimnasios para la gestión integral de sus operaciones. FitFlow estará diseñado para que otras franquicias o gimnasios independientes puedan instalarlo en sus servidores, gestionarlo y ofrecerlo a sus comunidades, en forma de instancias particulares.

El sistema debe ser confiable, maximizando la disponibilidad y veracidad de la información, y protegiendo los datos personales de socios y personal. Por eso, FitFlow, más allá de estadísticas básicas de uso, no recopilará datos sensibles de visitantes no registrados.

Con el objetivo de garantizar la disponibilidad de la información y mitigar fallas de infraestructura, esta se almacenará de forma centralizada pero con capacidad de integración descentralizada: cada instancia de FitFlow podrá interactuar con una o más fuentes de datos, servidas desde diferentes nodos. Un ejemplo es la integración con sistemas de pago externos

o servicios de clases especiales provistos por terceros.

Parte de la información será estática, cargada inicialmente desde lotes de datos (datasets), como la lista inicial de socios. Sin embargo, FitFlow también permitirá que los socios y administradores gestionen información de forma dinámica, como la inscripción a clases, la actualización de datos personales o el registro de pagos.

En este marco, diseñaremos e implementaremos:

- Una **fuente estática**: un servicio de solo lectura para la carga inicial de información, utilizando datasets de socios o clases provistos en archivos.
- Una **fuente dinámica**: el servicio principal para la gestión diaria de socios, inscripciones, pagos y horarios.
- Una **fuente proxy**: integraciones con servicios externos, como pasarelas de pago o sistemas de reserva de terceros.
- Un **servicio de agregación**, que se encargará de consultar las distintas fuentes de datos y consolidar la información (ej. unificar el calendario de clases internas con eventos especiales externos).
- Un **servicio de visualizaciones y estadísticas**, que podrán ser consultadas tanto online a través de una interfaz Web como consumidas por la administración.
- Un **módulo para la gestión de solicitudes** sobre cambios en membresías y protección de datos personales.

Diagrama de Despliegue Inicial

A modo orientativo, se tiene el siguiente diagrama de despliegue (Fig. 1) como contexto general del sistema. El mismo no es definitivo y sufrirá modificaciones conforme avance el desarrollo.

(El diagrama sería una adaptación del original, reemplazando "Hechos y Solicitudes" por "Inscripciones y Pagos", "Dataset" por "Dataset de Socios", "Fuente de Datos" por "Módulo de Gestión", "Agregador" por "Agregador de Horarios", etc.)

Glosario

- **Clase:** Actividad grupal dictada por un entrenador en un horario y lugar específicos, con un cupo limitado.
- **Plan de Membresía:** Conjunto de clases y servicios a los que un socio puede acceder.

Son creados y gestionados por administradores.
- **Socio:** Persona que se inscribe en el gimnasio. Puede tener un rol anónimo (consultando horarios) o registrado (con acceso a su perfil, inscripciones y pagos).
- **Dataset:** Lote de datos bien conocidos, usualmente en formato de archivo CSV, desde el cual la plataforma extrae información inicial (ej. lista de socios a migrar).
- **Datos Dinámicos:** Información proveniente de la operatoria diaria, como nuevas inscripciones a clases, pagos, o altas de socios, que crece en cantidad a lo largo del tiempo.
- **Datos Estáticos:** Información proveniente de archivos o bases de datos conocidas, que usualmente no cambian a lo largo del tiempo (ej. tipos de membresías, lista de salones).
- **Reserva:** Registro de la inscripción de un socio a una clase específica.
- **Fuentes de Datos:** Orígenes de la información en el sistema, pudiendo ser estáticas (datasets), dinámicas (aportadas por la operatoria diaria) o intermediarias (proxy, integraciones con pasarelas de pago u otros sistemas).
- **Instancia:** Un caso particular de despliegue de los servicios de FitFlow en los servidores de un gimnasio o franquicia.
- **Servicio de Agregación de Horarios:** Servicio encargado de combinar horarios de distintas fuentes (clases regulares, eventos especiales de terceros) para generar una vista unificada del calendario.

ENTREGA 1: Arquitectura y Modelado en Objetos - Parte I

Objetivos

- Entrar en contacto con el dominio y sus principales abstracciones (Socio, Clase, Plan).
- Incorporar conceptos y principios de Diseño Orientado a Objetos.
- Familiarizarse con el entorno de desarrollo y la arquitectura del sistema.

Alcance

- **Fuentes estáticas:** Carga masiva de socios.
- **Planes de Membresía.**
- **Clases:** Listado y filtrado por plan y horario.
- **Roles:** Administrador y Socio (en modo visualizador).
- **Solicitudes de baja de membresía.**

Dominio

- **Fuentes Estáticas:** Se debe implementar un componente que permita la lectura de un archivo .csv para la carga masiva inicial de socios al sistema.
- **Planes de Membresía:** Los planes representan los distintos tipos de abonos (ej. "Plan Musculación", "Plan Full Clases"). Tienen un título, descripción y precio. Los administradores pueden crear tantos planes como deseen.
- **Clases:** Cada clase tiene un título, descripción, entrenador, horario y cupo máximo. Las clases se asocian a uno o más Planes de Membresía. Un socio solo puede ver las clases incluidas en su plan.
- **Carga de Socios por CSV:** Se pide dar soporte a la carga de socios mediante archivos .csv. Se considerará que un socio está repetido si su DNI es el mismo; en tal caso, se actualizarán los datos del existente. El formato será: "Nombre", "Apellido", "DNI", "Email", "ID Plan Membresía"
- **Roles:** Se considera **visualizador** a toda persona que ingresa para ver los horarios, sin registrarse. El **administrador** gestiona planes y clases.
- **Solicitudes de Baja:** Cualquier socio debe poder solicitar la baja de su membresía. La solicitud requiere un texto justificado de al menos 150 caracteres. La solicitud quedará pendiente de aprobación por un administrador.

Requerimientos

1. Como **administrador**, deseo crear un Plan de Membresía.
2. Como **administrador**, deseo importar socios desde un archivo CSV.
3. Como **socio visualizador**, deseo navegar todas las clases disponibles de un Plan.
4. Como **socio visualizador**, deseo filtrar las clases por horario o día.
5. Como **socio registrado**, deseo poder solicitar la baja de mi membresía.
6. Como **administrador**, deseo poder aceptar o rechazar una solicitud de baja.

ENTREGA 2: Arquitectura y Modelado en Objetos - Parte II

Objetivos

- Diseñar e implementar nuevas funcionalidades de forma incremental.
- Incorporar nociones de ejecuciones de tareas asincrónicas.
- Familiarizarse con la integración de servicios externos a través de APIs.

Alcance

- **Fuentes dinámicas:** Gestión de reservas a clases.
- **Fuentes proxy:** Integración con sistemas externos (pasarelas de pago, sistemas de terceros).
- **Rechazo automático de solicitudes de baja incompletas.**

Dominio

- **Fuentes Dinámicas (Gestión de Reservas):** Se implementará la funcionalidad que permitirá a los socios registrados realizar reservas en las clases disponibles de su plan. Si un socio no está registrado, solo podrá visualizar. Un socio registrado podrá cancelar su reserva hasta 24 horas antes del inicio de la clase.
- **Fuentes Proxy (Intermediarias):**
  - **Fuente Pasarela de Pagos:** Existen sistemas externos para procesar pagos de membresías. Deberemos diseñar un componente Fuente Proxy que se integre con una pasarela de pagos ficticia, para la cual se provee una biblioteca cliente. El objetivo es que, una vez por día, el sistema verifique el estado de los pagos de las membresías.
  - **Fuente de Clases Externas:** El gimnasio tiene un convenio con una empresa que dicta talleres especiales. Esta empresa expone sus horarios vía una API REST. Se debe implementar una fuente que consuma esta API en tiempo real y muestre dichos talleres en el calendario general.
- **Rechazo Automático de Solicitudes:** Se cuenta con un componente pre-desarrollado que valida la completitud de un texto. Si el texto de la solicitud de baja no cumple con los criterios mínimos (ej. longitud, palabras clave), la solicitud deberá ser rechazada automáticamente.
  Java

```
interface ValidadorDeSolicitudes {
   boolean esValida(String texto);
}
```

Requerimientos

1. Como **socio registrado**, deseo poder reservar un cupo en una clase de mi plan.
2. Como **administrador**, quiero que el sistema actualice el estado de pago de los socios diariamente, consultando una pasarela de pagos externa.
3. Como **socio**, quiero poder ver en el calendario los talleres especiales ofrecidos por un proveedor externo, en tiempo real.
4. El sistema debe rechazar automáticamente las solicitudes de baja cuya justificación no sea considerada válida.

ENTREGA 3: Integración y Modelado en Objetos - Parte III

Objetivos

- Exponer un servicio propio a través de un protocolo de red (API REST).
- Incorporar flujos de trabajo asincrónicos para tareas pesadas.
- Modelar un servicio agregador.

Alcance

- **Servicio Agregador de Horarios.**
- **Gestión de Cupos y Listas de Espera.**
- **Exposición de una API REST propia.**

Dominio

- **Servicio de Agregación de Horarios:** Se requiere modelar el servicio agregador que consolide en una única vista los horarios de las clases propias (fuente dinámica) y los talleres de proveedores externos (fuentes proxy). Se pide que, una vez por hora, el servicio de agregación actualice su calendario consolidado.
- **Modos de Visualización del Calendario:**
  - **Modo Normal:** Se muestran todas las clases y talleres con cupo disponible.
  - **Modo Ocupado:** Se muestran también las clases que ya no tienen cupo.
- **Gestión de Cupos y Lista de Espera:** Para optimizar la ocupación, al crear una clase, se podrá configurar una **lista de espera**. Si un socio reserva en una clase con cupo lleno, quedará en lista de espera. Si se libera un lugar, el primer socio en la lista será notificado y tendrá un tiempo limitado para confirmar su lugar. La asignación de lugares desde la lista de espera puede ser un proceso costoso, por lo que debe ejecutarse en horarios de baja carga del sistema (ej. durante la noche).
- **Exposición REST:** Se deberán exponer las siguientes operaciones en una API REST:
  - **API administrativa:**
    - Operaciones CRUD sobre Clases y Planes.
    - Agregar o quitar clases de un plan.
    - Aprobar o denegar una solicitud de baja.
  - **API pública para Socios:**
    - Consulta de clases dentro de un plan.
    - Generar una reserva para una clase.
    - Navegación filtrada sobre el calendario.

Requerimientos

1. Como **socio**, deseo poder elegir si quiero ver todas las clases o solo las que tienen cupo.
2. Como **administrador**, quiero asociar una lista de espera a una clase.
3. Como **administrador**, quiero modificar las clases incluidas en un plan.
4. El Sistema debe permitir la realización de operaciones a través de los endpoints REST solicitados.

ENTREGA 4: Persistencia

Objetivos

- Incorporar persistencia de datos en un motor de base de datos relacional usando un ORM.
- Incorporar nociones de desnormalización para optimizar consultas.
- Desarrollar un servicio de estadísticas.

Alcance

- **Persistencia del modelo de objetos.**
- **Soporte para contenido multimedia en clases.**
- **Exportación de datos en formato CSV.**
- **Servicio de Estadísticas.**

Dominio

- **Servicio de Estadísticas:** Se desea desarrollar un servicio que genere periódicamente estadísticas sobre el uso del gimnasio. Cada cierto intervalo de tiempo, se recalcularán las estadísticas y se mostrarán en un dashboard. Se piden datos para responder:
  - ¿En qué franja horaria se concentra la mayor cantidad de reservas?
  - ¿Cuál es la clase con mayor cantidad de inscriptos históricamente?
  - ¿Qué plan de membresía genera más ingresos?
  - ¿Cuál es la tasa de presentismo promedio en las clases?
- **Contenido Multimedia:** Se debe permitir que los administradores o entrenadores puedan asociar un video o imagen a la descripción de una clase (ej. un video demostrativo de los ejercicios).
- **Requerimientos Detallados:**
  1. Se deberán persistir las entidades del modelo (Socio, Clase, Plan, Reserva, etc.) utilizando un ORM.
  2. Implementar el servicio de estadísticas.
  3. Como **administrador**, deseo poder agregar una imagen o video a una clase.
  4. Como **administrador**, deseo exportar un reporte de asistencia de una clase en formato CSV.

ENTREGA 5: Arquitectura Web MVC y Maquetado de Interfaz de Usuario

Objetivos

- Incorporar nociones de Diseño UI/UX y maquetado Web (HTML5/CSS).
- Implementar un Cliente Liviano (Server-Side Rendering).

Alcance

- **Diseño y maquetado de interfaces de usuario.**
- **Implementación de un Cliente Liviano.**

Requerimientos

1. Como **socio**, deseo poder visualizar el calendario de clases en una vista semanal/mensual.
2. Como **administrador**, deseo poder iniciar sesión en un panel de control.
3. Como **administrador**, deseo poder configurar desde mi panel de control los planes y las clases.
4. Como **administrador**, deseo poder aprobar o rechazar las solicitudes de baja desde una interfaz.

ENTREGA 6: Despliegue, Observabilidad y Seguridad

Objetivos

- Familiarizarse con técnicas de despliegue en la nube.
- Incorporar herramientas de monitoreo, observabilidad y seguridad.

Alcance

- **Sistema desplegado en la nube.**
- **Incorporar herramientas de observabilidad y monitoreo.**
- **Implementar medidas de seguridad.**

Requerimientos

1. Se deberá desplegar el sistema en la nube para que pueda ser accedido públicamente.
2. Incorporar herramientas que permitan la **observabilidad** del sistema (ej. logs centralizados, métricas de rendimiento).
3. Incorporar herramientas que permitan el **monitoreo y supervisión** del sistema (ej. reinicio automático ante una caída).
4. Incorporar un sistema de **Rate Limiting** para proteger la API de reservas de abusos.
5. Incorporar un sistema de **Bloqueo de IPs**.
6. Implementar **WebSockets** en el calendario para que los socios vean actualizaciones de cupos en tiempo real sin necesidad de recargar la página.