Club Deportivo DAMA Sports

Gestión Integral de Socios y Reservas con JPA y JavaFX

Álvaro Llorente

Tomás Perez

Álvaro Guy

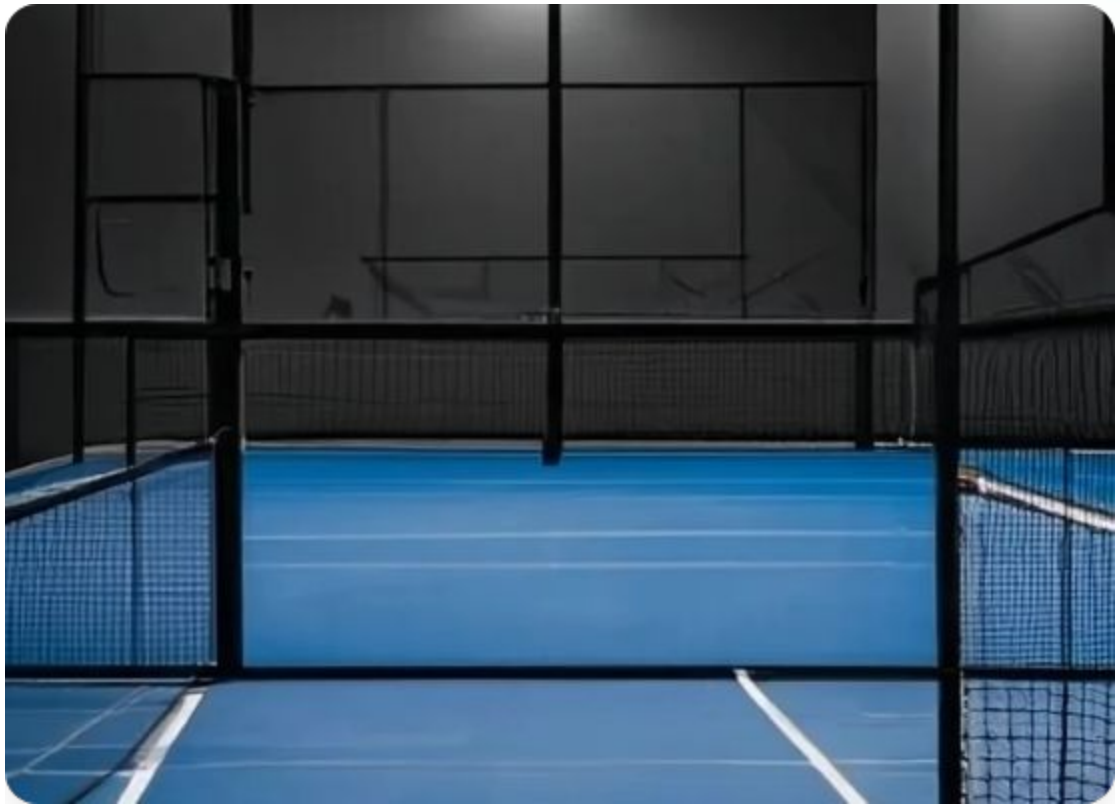
Alejandro Sandoval

Contexto y Arquitectura

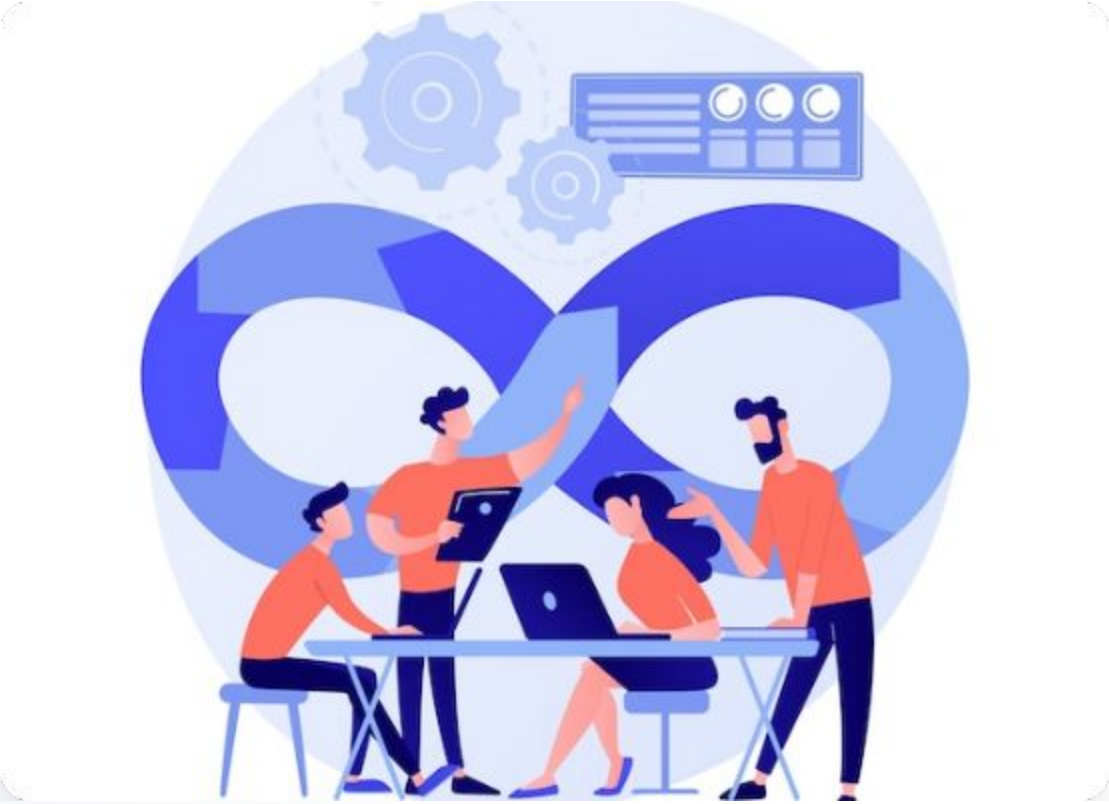
El objetivo es desarrollar una aplicación de escritorio robusta para la gestión de socios y pistas deportivas (Tenis, Pádel, Fútbol Sala).

Stack Tecnológico:

- **Lenguaje:** Java
- **Interfaz:** JavaFX
- **Persistencia:** JPA (Hibernate)
- **Base de Datos:** MariaDB



Metodología y Git



Trabajo en Equipo

Repositorio privado en github con estructura organizada.

Roles Definidos:

- **Líder Técnico:** Arquitectura y Revisiones.
- **Persistencia:** JPA y Consultas.
- **Dominio:** Reglas de Negocio.
- **Integración:** Conexión con UI.

Uso de ramas por desarrollador para evitar conflictos en main.

Modelo de Datos (Entidades)

Socio

Entidad principal anotada con @Entity. Mantiene una relación @OneToMany con Reservas usando CascadeType.ALL.

Pista

Define las instalaciones. Incluye atributos de control como disponible (Boolean) para gestionar el estado de la pista.

Reserva

Entidad débil que une Socio y Pista. Usa @ManyToOne y @JoinColumn para garantizar la integridad referencial.

Estrategia de Persistencia

La capa de datos se gestiona mediante **Hibernate** implementando el estándar JPA.

Gestión de EntityManager

Se utiliza el patrón "*EntityManager per operation*". Cada método (alta, baja, listado) crea su propia instancia y la cierra en un bloque finally.

Optimización de Consultas

Uso de **JPQL** con JOIN FETCH para cargar relaciones (Socio y Pista) en una sola consulta, evitando el problema N+1.

Código de Ejemplo

```java
public void altaPista(Pista pista) throws Exception {  1 usage  Alvaro Guy +1
    EntityManager em = emf.createEntityManager();

    try {
        em.getTransaction().begin();

        Pista existente = em.find(Pista.class, pista.getIdPista());

        if (existente != null)
            throw new Exception("Ya existe una pista con ese ID");

        em.persist(pista);

        em.getTransaction().commit();

    } catch (Exception e) {

        if (em.getTransaction().isActive())
            em.getTransaction().rollback();

        throw e;

    } finally {

        em.close();

    }
}
```

Retos Técnicos y Soluciones

1. LazyInitializationException

Al cerrar el EntityManager, las relaciones cargadas de forma perezosa (Lazy) fallaban al pintarse en la tabla JavaFX.

Solución: *Implementar consultas `JOIN FETCH` para traer la entidad principal y sus relaciones en una sola transacción.*

2. Consistencia de Datos

Riesgo de borrar un socio que tuviera reservas activas, dejando huérfanos en la base de datos.

Solución: *Validación lógica previa a la llamada `remove()` lanzando excepción controlada.*

Lógica de Negocio (Service)

Validaciones

Reglas estrictas implementadas en ClubDeportivo:

- **Disponibilidad:** No se puede reservar una pista no disponible: !pista.getDisponible().
- **Integridad de Borrado:** Prohibido eliminar un Socio si tiene reservas activas (!socio.getReservas().isEmpty()).

Transacciones

Gestión segura de operaciones ACID:

- Inicio explícito: em.getTransaction().begin().
- Confirmación: commit() tras éxito.
- Rollback: En bloque catch para prevenir inconsistencias en caso de error.

Integración con JavaFX

Controladores y Vistas

La clase MainApp orquesta la navegación mediante un BorderPane.

Cada opción del menú ("Alta Socio", "Reservar") carga dinámicamente una nueva vista en el centro del layout.

Se capturan las excepciones del Service Layer y se muestran al usuario mediante Alert (Dialogs).

Demostración y Preguntas

Pasamos a demostrar la correcta funcionalidad del sistema.