

Diseño de Sistema y BBDD



SHOWPASS

Fecha: 07/10/2025

BASE DE DATOS

La sección del diseño de la base de datos es fundamental para garantizar la **integridad, consistencia y eficiencia** del sistema desarrollado en Spring Boot. Se optó por un modelo de **base de datos relacional** robusto, implementado mediante **Java Persistence API (JPA)** e **Hibernate**.

El diseño se basa en **cinco entidades principales** (Usuario, TarjetaBancaria, Evento, Ticket, Carrito), junto con una clase componente (Invitado). Se ha priorizado la implementación de **relaciones unidireccionales** para simplificar la navegación y el mantenimiento del código, asegurando que cada entidad tenga una responsabilidad única y bien definida.

El objetivo principal es modelar las interacciones clave desde la **identificación segura** del usuario y la **gestión financiera**, hasta el **proceso de compra** y la **gestión de colecciones**. Esta estructura facilita las operaciones CRUD y también establece las bases para una lógica de negocio clara y escalable en las capas de servicio de la aplicación.

Entidad: Usuario

Propósito: Representar a los usuarios de la aplicación (Admin, Cliente, Vendedor)

Atributo	Tipo de Dato	Anotaciones/Restricciones	Relación
id	Long	@Id, @GeneratedValue	Clave Primaria (PK)
nombre	String		
email	String	@Unique	
password	String		
fechaNacimiento	LocalDate		
foto	String		URL/Ruta de la imagen de perfil
rol	Rol (Enum)	@Enumerated(EnumType.STRING)	Enum: ADMIN, CLIENTE, VENDEDOR
reportado	boolean		
tarjeta	TarjetaBancaria	@OneToOne(cascade=ALL)	1:1 con TarjetaBancaria

Rol(Enum): ADMIN, CLIENTE, VENDEDOR

Entidad: TarjetaBancaria

Propósito: Almacenar la información de pago de un usuario

Atributo	Tipo de Dato	Anotaciones/Restricciones	Notas
id	Long	@Id, @GeneratedValue	Clave Primaria (PK)
nombreTitular	String		
nTarjeta	String	@Column(length=16), @Unique	Número de tarjeta
fechaCaducidad	LocalDate		
cvv	String	@Column(length=4)	
saldo	BigDecimal		

Entidad: Evento

Propósito: Detalle de los eventos disponibles para compra.

Atributo	Tipo de Dato	Anotaciones/Restricciones	Relación
id	Long	@Id, @GeneratedValue	Clave Primaria (PK)
nombre	String	@Column(length=100)	
localizacion	String		
imagen	String		Imagen principal del evento
inicioEvento	LocalDateTime		
finEvento	LocalDateTime		
descripcion	String		
precio	double		
categoria	Categoria (Enum)	@Enumerated(EnumType.STRING)	Enum: MUSICA, DEPORTES, etc.
invitados	List<Invitado>	@ElementCollection @CollectionTable	Genera una tabla auxiliar (evento_invitados) para almacenar la lista de componentes @Embeddable.
carrusels	List<String>		Strings (imágenes).

Categoria(Enum): MUSICA, DEPORTES, ARTE, VIDEOJUEGOS, OTROS

Clase Componente: **Invitado**

Esta clase es un componente reutilizable y se almacena en la tabla de su entidad dueña (Evento).

Atributo	Tipo de Dato	Anotaciones/Restricciones	Notas
Clase	N/A	@Embeddable	Indica que sus atributos se incrustan en la tabla dueña.
nombre	String		
apellidos	String		
fotoURL	String	@Lob	Para URLs o datos de imagen potencialmente largos.
descripcion	String	@Column(length=250)	Máximo 250 caracteres.

Entidad: **Ticket**

Propósito: Registro de una compra de entrada, vinculando a un usuario y un evento.

Atributo	Tipo de Dato	Anotaciones/Restricciones	Relación
id	Long	@Id, @GeneratedValue	Clave Primaria (PK)
codigoQR	String	@Unique	Se recomienda para evitar duplicados.
fechaCompra	LocalDateTime		
usuario	Usuario	@ManyToOne	N:1 con Usuario. (FK usuario_id).
evento	Evento	@ManyToOne	N:1 con Evento. (FK evento_id).

Entidad: **Carrito**

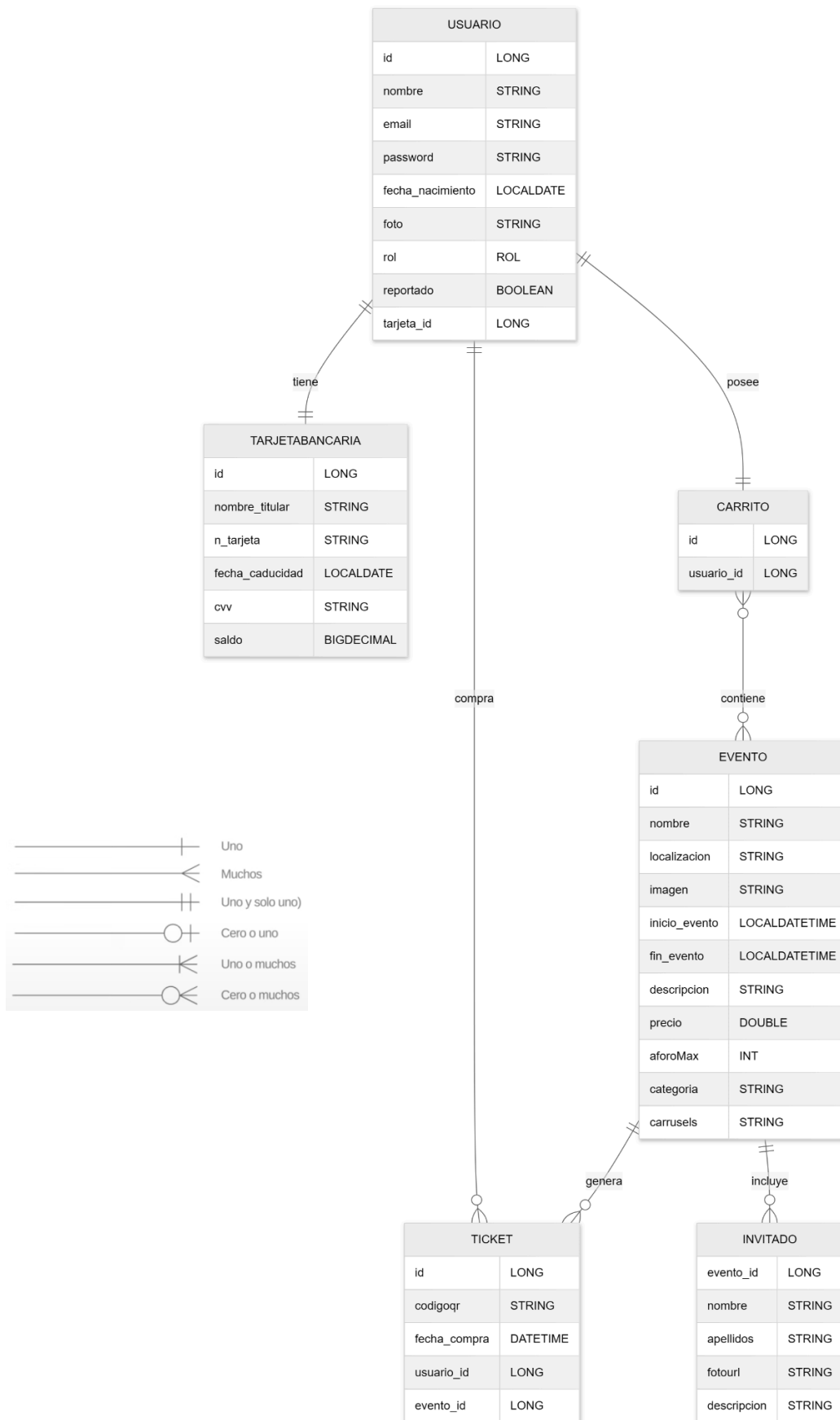
Propósito: Mantener la lista de eventos que un usuario planea comprar.

Atributo	Tipo de Dato	Anotaciones/Restricciones	Relación
id	Long	@Id, @GeneratedValue	Clave Primaria (PK)
usuario	Usuario	@OneToOne @JoinColumn	1:1 con Usuario (FK usuario_id).
eventos	List<Evento>	@ManyToMany @JoinTable(@JoinColumn)	N:M con Evento Genera tabla carrito_eventos.

Resumen de Relaciones

Entidad A	Relación JPA	Multiplicidad	Entidad B	Notas sobre Mapeo
Evento	@ElementCollection	1 : N	Invitado	Invitado es una clase @Embeddable (componente) que se almacena en una tabla auxiliar.
Usuario	@OneToOne	1 : 1	TarjetaBancaria	El usuario tiene una sola tarjeta.
Usuario	@OneToOne	1 : 1	Carrito	Un usuario tiene un solo carrito.
Usuario	@OneToMany	1 : N	Ticket	Un usuario puede tener muchos tickets.
Evento	@OneToMany	1 : N	Ticket	Un evento puede generar muchos tickets.
Carrito	@ManyToMany	N : M	Evento	Genera una tabla intermedia (carrito_eventos).

Diagrama Entidad-Relación



DISEÑO DE SISTEMA

Nuestra aplicación Spring Boot se dividirá en tres capas principales, cada una con una responsabilidad bien definida:

1. Capa de Presentación (Controladores)

- **Responsabilidad:** Manejar las peticiones HTTP entrantes (GET, POST, PUT, DELETE), validar los datos de entrada básicos, y devolver las respuestas HTTP (generalmente JSON). Actúa como la interfaz de la API REST.
- **Componentes Clave:**
 - **Controladores (@RestController):** Clases que definen los *endpoints* (rutas) de tu aplicación.
 - **Modelos de Transferencia de Datos (DTOs):** Clases simples usadas para enviar y recibir datos de la API. *No son las entidades JPA*. Esto aísla la capa de presentación de los modelos de la BBDD.

Módulo/Controlador	Responsabilidades	Ejemplos de Endpoints
ControlUsuario	Gestión de Usuarios, Login/Registro	POST /tfg/usuario/register GET /tfg/usuario/findAll
ControlEvento	Gestión y Consulta de Eventos	GET /tfg/evento/findByName, POST /tfg/evento/insert
ControlTickets	Compra y Consulta de Tickets	POST /tfg/tickets/insert, GET /tfg/tickets/validarQR
ControlCarrito	Gestión del Carrito	POST /tfg/carrito/EnviarPdfEmail GET /tfg/carrito/total{usuarioId}
ControlCuentaBancaria	Gestión de C.Bancarias	POST /tfg/cuentaBancaria/insert DELETE /tfg/cuentaBancaria/delete/{id}
CargarDatos	Cargar Datos por defecto	GET /tfg/utilidades/cargarDatos

2. Capa de Lógica de Negocio (Servicios)

- **Responsabilidad:** Contener todas las **reglas de negocio** complejas, realizar la coordinación de transacciones, aplicar la lógica de validación avanzada, y transformar los datos entre las capas.
- **Componentes Clave:**
 - **Servicios (@Service):** Clases que implementan la lógica de negocio. Un servicio puede utilizar múltiples repositorios.

- **Ejemplo de Lógica:** Verificar si un usuario está reportado antes de permitir una compra, generar el código QR, comprobar saldo del usuario

Módulo/Servicio	Responsabilidad Principal	Lógica de Negocio Asociada
ServicioUsuario	CRUD de Usuario, cambio de datos	Aplicar el hashing de contraseñas, verificar la unicidad del email.
ServicioEventos	CRUD de Eventos.	Validar fechas ($\text{inicioEvento} < \text{finEvento}$), verificar si la categoría existe.
ServicioTicket	Manejo de adiciones/eliminaciones.	Verificar si el evento existe y está disponible antes de agregarlo al carrito
ServicioCarrito	Proceso de Compra (Transacción).	Transaccionalidad: 1. Verificar saldo de la tarjeta. 2. Generar el código QR. 3. Guardar Ticket.
ServicioTarjetaBancaria	CRUD Tarjetas Bancarias, consulta saldo etc	Recargo de saldo, etc

3. Capa de Acceso a Datos (Persistencia)

- **Responsabilidad:** Abstraer el acceso a la base de datos. Se encarga de las operaciones básicas de Crear, Leer, Actualizar y Eliminar (CRUD) sobre una única entidad.
- **Componentes Clave:**
 - **Repositorios (@Repository / JpaRepository):** Interfaces que extienden JpaRepository de Spring Data. Spring se encarga de implementar los métodos básicos. Aquí es donde se definen las consultas personalizadas si son necesarias.

Módulo/Repositorio	Entidad JPA Asociada	Operaciones Típicas(Ejemplos)
RepositorioUsuario	Usuario	findById(id), findByEmail(email)
RepositorioEvento	Evento	findAllByCategoria(categoria), deleteById(id)
RepositorioTicket	Ticket	findByUsuarioId(userId), findByCodigoQR(qr)
RepositorioCarrito	Carrito	findByUsuarioId(userId)
RepositorioCuentaBancaria	TarjetaBancaria	save(), delete()

Componentes Adicionales y Herramientas

Componente	Tipo/Tecnología	Función en el Sistema
BBDD	H2 / MySQL	Persistencia de datos (definida por las entidades JPA).
Spring Security	Framework de Seguridad/JWT	Manejo de Autenticación (Login) y Autorización (Roles: ADMIN, CLIENTE, VENDEDOR).
Mapper	ModelMapper	Transformación de Entidades JPA a DTOs y viceversa (entre Service y Controller).
Validación	Bean Validation (JSR 303/349)	Validación de campos (ej. @NotNull, @Size, @Email) en los DTOs y Entidades.
Generador QR	Librería Externa	Generación del código QR al comprar un Ticket (Lógica implementada en TicketService).
MultipartFile	Base64	Parseo de imágenes a un formato en cadena, String