Proyecto 1 – BoletaMaster

Entrega 1: Documento de Análisis

Valeria Plaza, Geronimo Rojas, Catalina Gonzalez

21 de septiembre de 2025

Proyecto 1 – BoletaMaster
Entrega 1: Documento de Análisis Valeria Plaza, Geronimo Rojas, Catalina Gonzalez 21 de septiembre de 2025

1. Resumen

El presente documento describe el modelo de dominio del sistema **BoletaMaster**, una plataforma para la creación de eventos, gestión de localidades y venta de tiquetes. El objetivo es ofrecer un sistema que permita a organizadores, compradores y administradores interactuar para la venta, transferencia y reembolso de tiquetes de manera segura y controlada.

2. Modelo de Dominio

En la Figura 1 se presenta el diagrama de clases del sistema, organizado en paquetes: *Usuarios*, *Eventos*, *Tiquetes* y *Transacciones*.

3. Restricciones e Invariantes

1. **Unicidad de Ticket:** cada `Ticket.id` debe ser único en el sistema.
2. **No Transferible:** `TicketDeluxe.transferible = false`.
3. **Transferencia de Paquetes:** `TicketMultiple` solo puede transferirse si todas sus entradas están sin usar y no han sido transferidas individualmente.
4. **Restricción de Venue:** un `Venue` no puede tener dos eventos en la misma fecha.
5. **Cálculo de Precio:** el precio final de un ticket se calcula como

$$precioFinal = precioBase + precioBase \times porcentajeServicio + cuotaFija.$$

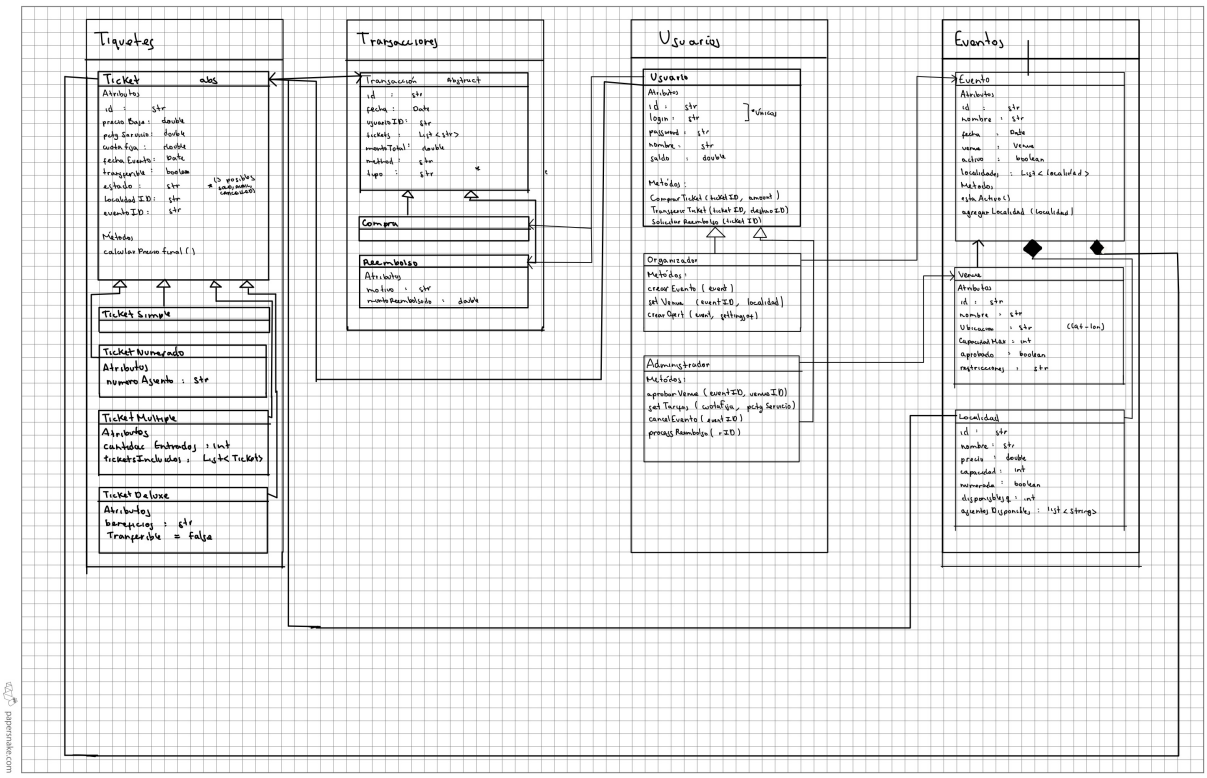

Figura 1: Diagrama de clases – Modelo de dominio del sistema.

6. **Reembolsos:** si el Administrador cancela un evento, se genera un reembolso completo (excepto la cuota de emisión).

4. Descripción de Entidades

4.1. Usuarios

- **Usuario**: `id:String`, `login:String`, `password:String`, `nombre:String`, `saldo:double`
- **Organizador**: hereda de Usuario, crea eventos y localidades.
- **Administrador**: hereda de Usuario, aprueba venues, define tarifas y procesa reembolsos.

4.2. Eventos y Localidades

- **Venue**: datos de ubicación, capacidad máxima y aprobación.
- **Evento**: nombre, fecha, relación con venue, lista de localidades.
- **Localidad**: nombre, precio, capacidad, numerada o no, lista de asientos.

4.3. Tiquetes

- **Ticket** (abstracta): precio base, porcentaje de servicio, cuota fija, estado.
- **TicketSimple, TicketNumerado, TicketMultiple, TicketDeluxe**: especializaciones con atributos propios.

4.4. Transacciones

- **Transaccion** (abstracta): id, fecha, usuario, monto total.
- **Compra, Reembolso**: subclases para diferenciar las operaciones.

5. Programas de Prueba Planeados

Aunque no se implementan en esta entrega, se describen los programas de prueba que se desarrollarán para validar el sistema:

1. **Carga de Datos**: leer archivos de usuarios, eventos y tickets para inicializar el sistema. Mostrará un resumen de los datos cargados.
2. **Simulación de Compra**: un comprador selecciona evento, localidad y cantidad de tickets. El sistema valida capacidad, calcula precio final y genera una transacción de compra.
3. **Transferencia de Ticket**: prueba transferencia exitosa entre usuarios, intento de transferencia de TicketDeluxe (debe fallar) y transferencia de TicketMultiple como paquete.
4. **Cancelación y Reembolso**: el administrador cancela un evento y se generan reembolsos automáticos a los saldos de los usuarios afectados.
5. **Reporte Financiero**: el organizador genera un informe con número de tickets vendidos, ingresos y distribución por localidad.

6. Decisiones de Diseño

- Uso de herencia para modelar roles de usuario y tipos de tickets.
- Uso de composición para localidades dentro de eventos.
- Separación de compras y reembolsos en subclases para permitir reglas de negocio diferentes.
- Definición de restricciones como invariantes para asegurar integridad del modelo.