Proyecto 1 – BoletaMaster

Entrega 2: Diseño e implementación

Maria Valeria Plaza Aguirre, Gerónimo Rojas, Catalina González

19 de octubre de 2025

1. Contexto:

El sistema de Boleta Master es una plataforma para la creación de eventos, gestión d localidades y venta de tiquetes. Además, de que esta podrá tener acceso los usuarios que estén interesados en la compra de boletas, organizadores de eventos y el administrador (quién se encarga de todo el orden del evento).

2. Objetivos:

Crear una plataforma el cual los organizadores y administrador puedan acceder a la plataforma para la edición de este para el cambio de venue, además que los usuarios entren a la plataforma a comprar ticketes.

3. Restricciones:

a. Máximos tiquetes por transacción (simples múltiples números etc) toca definirlos
b. Ticket Deluxe no se puede transferir
c. No se puede transferir un paquete completo si ya se ha vendido o transferido alguno de sus tiquetes individuales
d. No es posible tener un evento sin organizador
e. Los organizadores pueden tener varios eventos activos al mismo tiempo
f. Un venue solo puede tener un evento en la misma fecha
g. Todo ticket en la misma localidad tiene el mismo precio
h. Es posible un evento con una única localidad y un único tipo de boleta
i. Todos los tiquetes de un evento deben tener el mismo costo de impresión

4. Organización del código

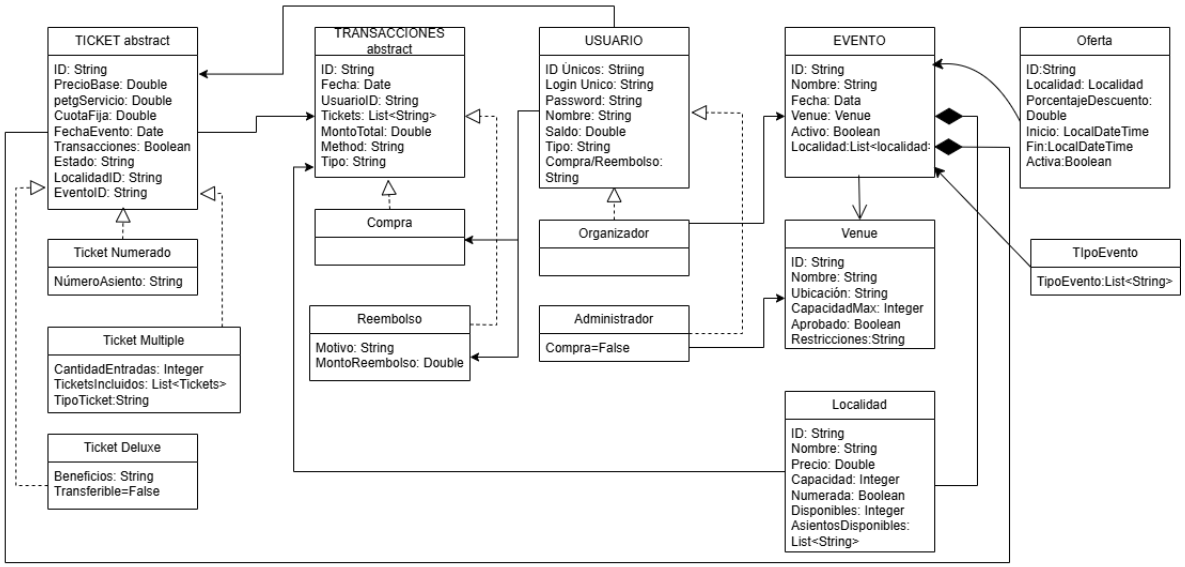
El proyecto está organizado en paquetes de la siguiente forma:

Seccion	Paquete	Función
Aplicación	boletamaster.app	Coordinación de operaciones y reglas del negocio
Modelo de dominio	boletamaster.usuarios, boletamaster.eventos, boletamaster.tiquetes	Representación del negocio (objetos y relaciones)
Persistencia	boletamaster.persistence	Gestión de almacenamiento en memoria

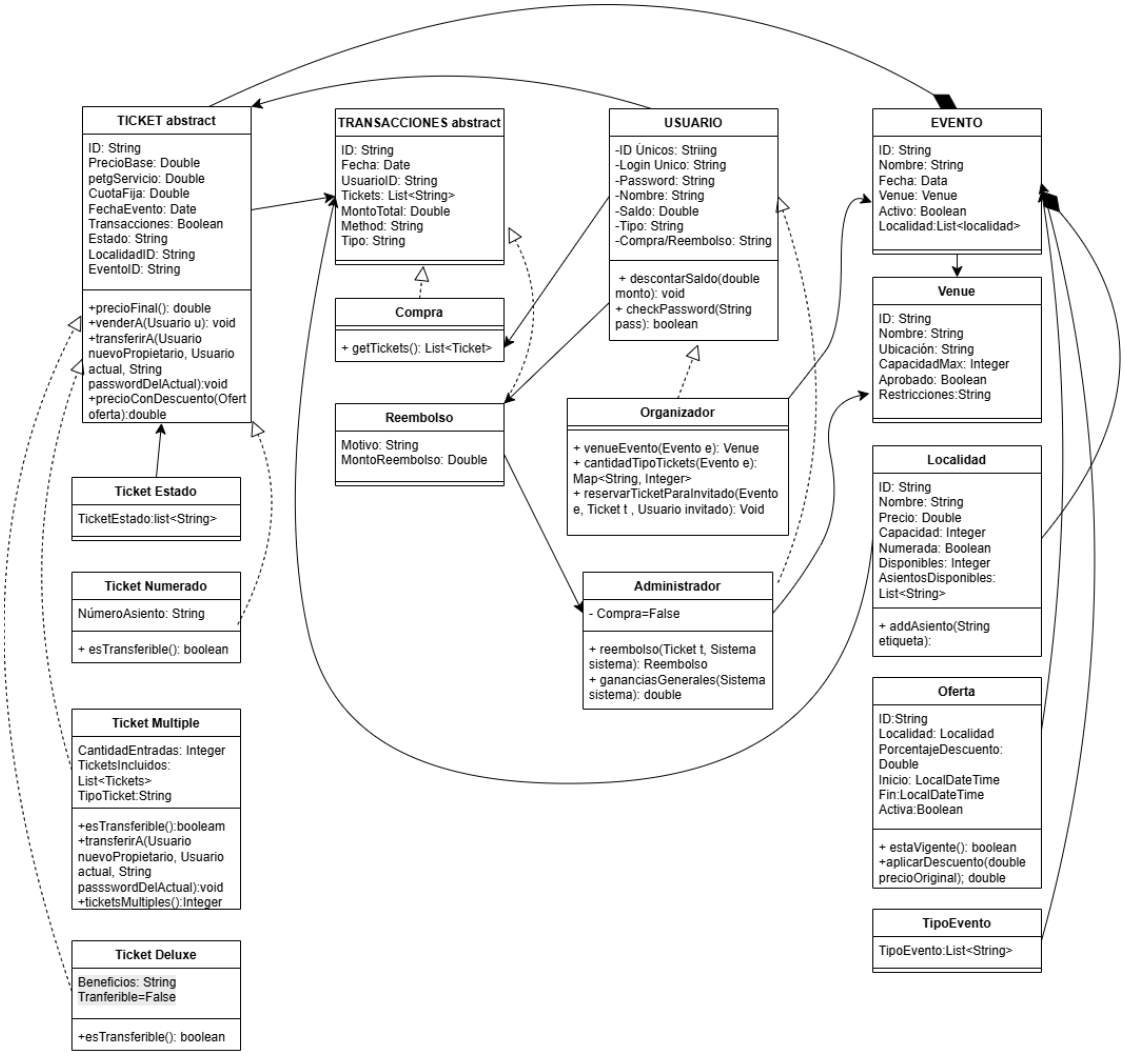
Seccion	Paquete	Función
Transacciones	boletamaster.transacciones	Gestión de compras y reembolsos

5. Diagrama de clase:

Entonces, a continuación, se podrá evidenciar el diagrama de clases del proyecto sin métodos, sino solo clases.



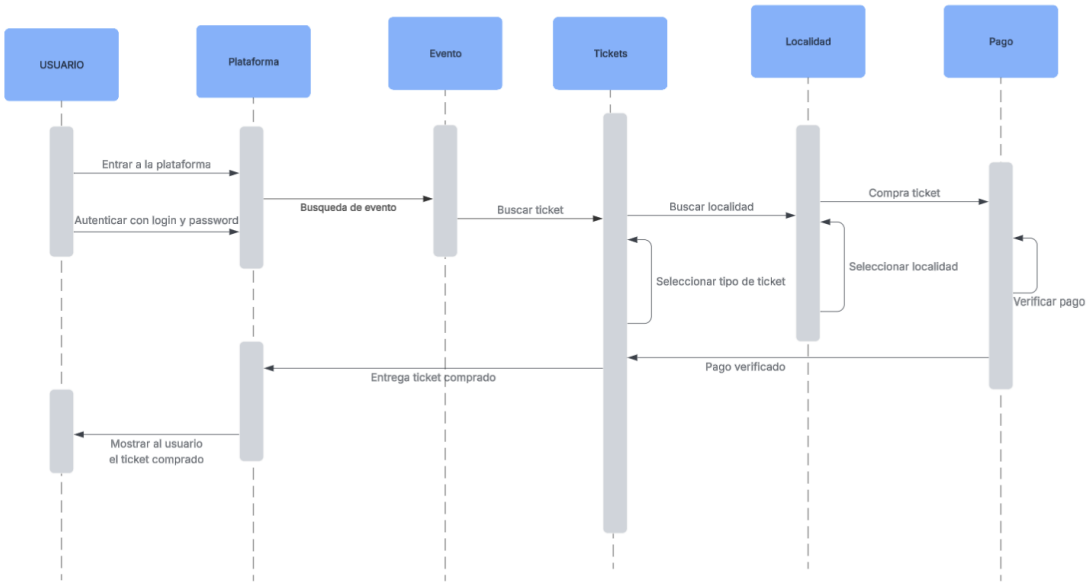
6. Diagrama de UML (Clases con atributos y métodos):



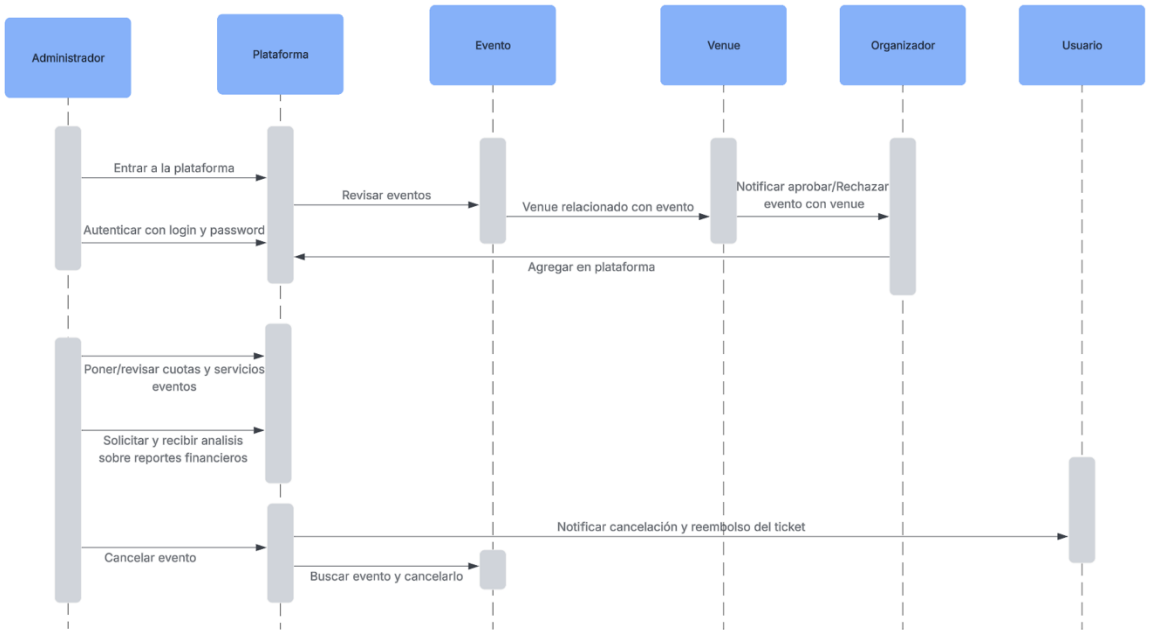
7. Diagrama de secuencia:

Para la demostración de los diagramas de secuencia se ha decidido hacerlo en 4 diagramas donde se estipulan: Compra por parte del usuario, gestión del administrador, gestión del organizador y transferencia de tickets.

- Compra de tickets por parte del usuario:
  En este parte se puede ver el proceso que se realiza cuando un usuario quiere ingresar a la plataforma para la compra de un ticket especifico dependiendo del evento, tipo de ticket y localidad que el usuario quiere, para luego pasarlo a la parte del pago y al final entregar el ticket comprado al usuario.

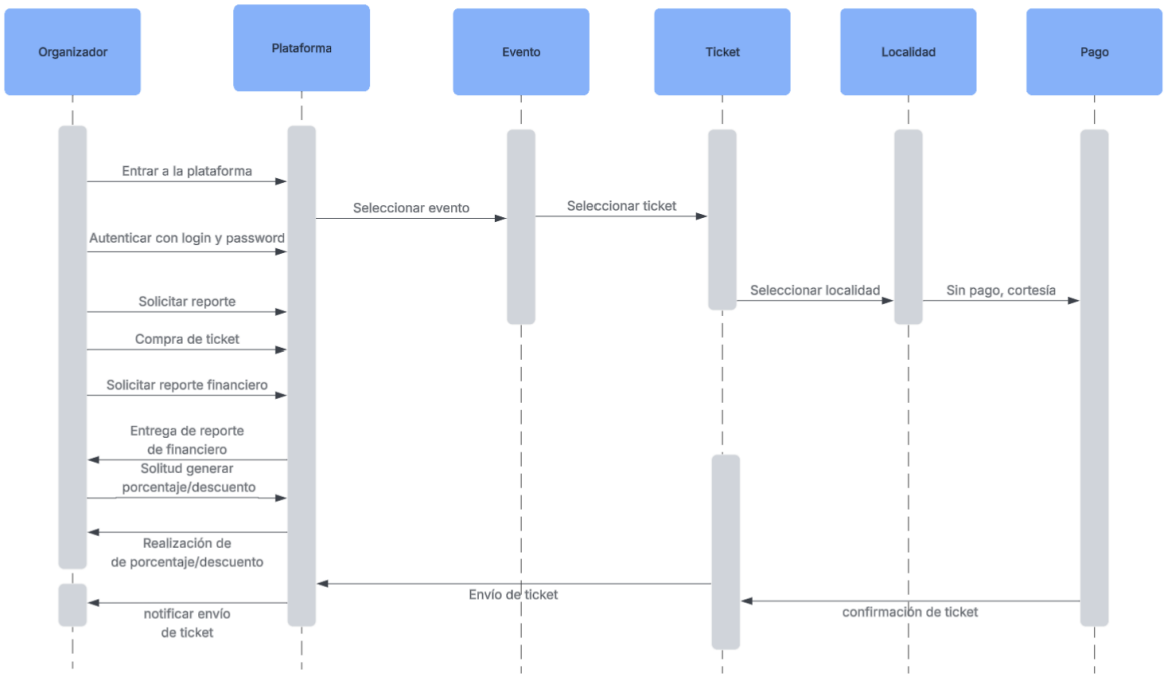


- Gestión del administrador:
  Este diagrama nos muestra como el administrador realiza la operación de revisar el evento realizado por el organizador para luego ser aceptada o rechazada, además de mostrar el proceso en caso de que un evento se cancele de manera espontánea lo que realizaría que exista un reembolso con el usuario, agregar el valor de las cuotas y por último la parte de solicitud de análisis de reportes financieros.

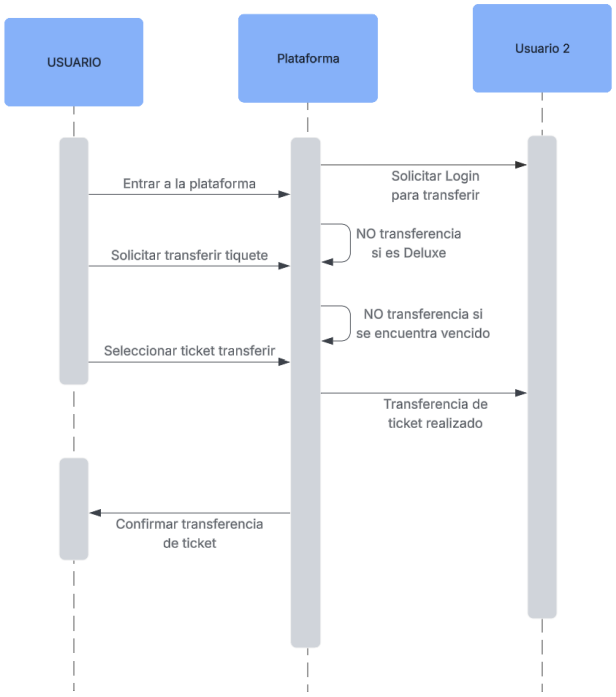


- Gestión por parte del organizador:
  En la gestión del organizador, se evidencia como por medio de la plataforma este pide el reporte financiero, la solicitud y realización del porcentaje y descuento de tickets, y por último en el caso que el organizador quiera “comprar” un ticket para que escoja en el evento, tipo de ticket y la localidad,

pero al ser organizador este no tendría que pagar nada ya que esto se consideraría algo de cortesía.



- Transferencia de tickets:
  Y el último diagrama demuestra la realización del proceso de transferencia de un usuario que quiere solicitar la transferencia de tiquetes a otro usuario, además de no dejar de lado tener en cuenta que no se puede transferir tiquetes Deluxe ni que estos estén vencidos.



8. Reglas de Dominio:

a. Un cliente puede comprar varios tiquetes, pero existe un límite máximo de compra de tiquetes por transacción.
b. Tiquetes Deluxe, no pueden ser transferidos.
c. Cada tiquete debe tener un identificador único para evitar las falsificaciones de estas.
d. Los usuarios deben tener un login y password para entrar a la plataforma y hacer sus compras.
e. Los organizadores y los administradores también deben tener login y password para la modificación de la plataforma.
f. Un venue no puede tener 2 eventos el mismo día en la misma hora.
g. La restricción de máxima cantidad de tiquetes para los ticketes múltiples se debe aplicar por sobre la agrupación, no sobre elementos individuales.
h. Los tickets múltiples se pueden transferir de manera individual o del paquete completo, pero no se puede transferir si el paquete completo se encuentra vencido.

9. Persistencia:

Aunque la Entrega 2 no exige persistencia permanente, el diseño incluye una implementación en memoria mediante SimpleRepository. Esta capa mantiene los datos durante la ejecución del programa y sirve de base para pruebas integrales del sistema.

Además, de ayudar a cumplir con el objetivo de seguridad y guardar la información din usar una base de datos externa.

10. Requerimientos funcionales y no funcionales:

10.1 Requerimientos funcionales:

- Gestión de usuarios:
  Permitir el registro y autenticación de usuarios, además de administración del sistema por parte de los organizadores y administrador.

- Compra de tiquetes:
  Usuarios pueda comprar y adquirir tiquetes para distintos eventos y según el tipo de tiquete deseen (Numerado, Deluxe, etc.)

- Gestión eventos:
  Creación de eventos por parte de los organizadores para poder ser asociados a Venue teniendo en cuenta las localidades, restricciones y precios según tipo.

- Gestión de venue y localidades:
  Sistema permita la creación y administración del venue al que corresponde el evento teniendo en cuenta la capacidad máxima y las restricciones.

- Transferencia de tiquetes:
  El sistema permita la transferencia de tiquetes de un usuario a otro, teniendo en cuenta el tipo de tiquete que es y respetando la restricción del tipo y que este tiquete no este vencido.

- Reembolso:
  Permitir solicitar y procesar reembolso de dinero en caso que el evento sea cancelado por el administrador por motivos justificados.

- Oferta y descuentos:
  El sistema debe permitir al administrador aplicar temporalmente ciertos descuento y ofertas para que el evento.

10.2 Requerimientos no funcionales :

1. Arquitectura y diseño

    - El sistema debe implementarse utilizando Programación Orientada a Objetos (POO).

2.Calidad del código

- Se debe evitar código duplicado mediante herencia y composición.
- Las clases deben tener una única responsabilidad funcional (SRP).
- El sistema debe manejar excepciones claras y controladas.

3.Persistencia

- La entrega debe gestionar los datos mediante un repositorio interno en memoria.
- No se permite el uso de bases de datos ni frameworks externos.
- La persistencia debe estar desacoplada de la lógica del negocio.

4.Mantenibilidad

- El sistema debe ser escalable para permitir agregar persistencia con archivos en futuras entregas.
- Debe seguir una estructura de paquetes clara y organizada.
- Los nombres de clases y métodos deben ser semánticos y autocontenidos.

5.Usabilidad

- Debe evitar acciones ambiguas en el flujo del sistema.
- Las operaciones críticas (como reembolso, cancelación o transferencia) deben estar claramente definidas.
- Deben mostrarse mensajes claros para estados inválidos (ticket vencido, saldo insuficiente, etc.).

6.Rendimiento

- El sistema debe permitir almacenar y consultar un número moderado de eventos y tickets sin pérdida de rendimiento.
- Las operaciones principales (compra, registro, transferencia, búsqueda) deben funcionar en tiempo aceptable dentro de la memoria.

7.Seguridad

- Las contraseñas deben manejarse con validación básica.
- Ningún usuario podrá realizar acciones fuera de su rol asignado.
- Las reglas de negocio deben proteger la integridad de los datos internos del sistema.

8.Portabilidad

- El sistema debe ser compatible con Java 8 o superior.
- Debe poder ejecutarse en cualquier entorno Java (Windows, Linux, Mac).

- No debe depender de herramientas externas distintas a la librería estándar de Java.

Descripción técnica de clases por paquete

3.1 Paquete **boletamaster.app**

Clase Sistema

Es la unidad central del dominio que coordina y valida las reglas del negocio. Proporciona métodos para:

- Registrar usuarios y eventos.
- Gestionar generación de tickets de diferentes tipos.
- Comprar tickets, garantizando que estén disponibles y que el evento no esté cancelado ni vencido.
- Transferir tickets entre usuarios autenticados.
- Cancelar eventos, generando automáticamente reembolsos de acuerdo con las políticas indicadas.
- Relacionar tickets con eventos y localidades para control de capacidad.
- Validar restricciones según el rol del usuario (comprador, organizador o administrador).

Esta clase utiliza el repositorio SimpleRepository como fuente de datos, logrando un desacoplamiento parcial entre la lógica y el almacenamiento.

3.2 Paquete **boletamaster.persistence**

Clase SimpleRepository

Es un repositorio en memoria que cumple la función de capa de persistencia básica. Almacena colecciones para:

- Usuarios
- Eventos
- Venues
- Tickets
- Transacciones (compras y reembolsos)

Esta capa actúa como sustituto temporal de una base de datos. En la Entrega 3, puede evolucionar hacia persistencia real mediante archivos o almacenamiento relacional. La ventaja del diseño actual es que abstrae la fuente de almacenamiento, permitiendo que Sistema opere independientemente de cómo se guarden los datos.

3.3 Paquete boletamaster.usuarios

Clase abstracta Usuario

Es la clase base para todos los usuarios del sistema. Contiene los atributos esenciales (login, password, nombre, saldo) y la responsabilidad de autenticación básica. Evalúa contraseñas y permite operaciones de manejo de saldo. Sirve como supertipo para roles específicos.

Subclase Comprador

Representa al usuario que adquiere tickets a través del sistema. Está restringido para evitar operaciones administrativas y tiene interacción controlada mediante los métodos de Sistema.

Subclase Organizador

Usuario encargado de crear y gestionar eventos. Contiene métodos adicionales necesarios para esta responsabilidad:

- eventosActivos(Sistema)
- cantidadTipoTickets(Evento)
- reservarTicketParaInvitado(...)
- venueEvento(...)

Restringe acceso para evitar que un organizador gestione eventos de otros.

Subclase Administrador

Usuario con privilegios elevados. Funciones:

- Autorizar y ejecutar reembolsos.
- Generar reportes financieros globales.
- Cancelar eventos y generar reembolsos masivos.

3.4 Paquete boletamaster.eventos

Clase Evento

Entidad principal que modela un evento real. Incluye:

- Nombre, fecha y hora
- Venue del evento
- Organizador
- Localidades
- Tickets emitidos
- Estado de cancelación

Permite asociar tickets y controlar capacidad a través del método limiteTickets(). Evita venta de tickets posterior a la cancelación del evento.

Clase Venue

Representa el lugar físico del evento. Contiene información como ubicación y capacidad.

Clase Localidad

Define zonas específicas dentro del Venue, cada una con capacidad y precio base. Cada ticket emitido tiene una localidad asociada.

3.5 Paquete boletamaster.tiquetes

Clase abstracta Ticket

Define estructura común de entrada:

- Precio base, porcentaje de servicio y cuota fija
- Propietario
- Estado (DISPONIBLE, VENDIDO, TRANSFERIDO, CANCELADO)
- Verificación de vencimiento usando fecha del evento (ticketVencido())
- Control de transferencia

Es el elemento central en la lógica transaccional del sistema.

Subclases:

Clase	Descripción
TicketSimple	Ticket básico
TicketNumerado	Incluye asiento
TicketDeluxe	Permite beneficios extra

Clase	Descripción
TicketMultiple	Agrupa varios tickets. Incluye ticketsMultiples()

3.6 Paquete **boletamaster.transacciones**

Clase Compra

Registra operaciones de adquisición de tickets. Relaciona usuario y monto total pagado.

Clase Reembolso

Modela devolución de dinero según políticas del sistema. Solo puede generarse a partir de autorización del administrador o cancelación de eventos.
