

Fecha de entrega: 28 de septiembre, 2025

*Vista general del UML, vista detallada adjunta en el repositorio.

Restricciones

1. Identidad y Unicidad
 - a. Identificador de tiquete único → No pueden existir dos instancias de Tiquete (ni cualquier subclase) con el mismo identificador.
 - b. Login único por usuario → El atributo login de Usuario (y sus subclases Cliente, OrganizadorDeEventos, Administrador) es único en todo el sistema.
 - c. Asientos únicos en localidades numeradas → En localidades con numerada = true, numeroAsiento no se repite dentro de la misma localidad para eventos con la misma fecha y hora.
2. Aprobaciones y Cancelaciones
 - a. Venue aprobado para programar eventos → Un Evento solo referencia un Venue con aprobado = true. Si es aprobado, debe existir aprobadoPor.
 - b. Eventos cancelados no se venden → Si cancelado en la clase Evento = true, no se pueden realizar nuevas compras de Tiquete asociadas al evento en cuestión.
3. Consistencia entre Evento, Venue, Localidades, Tiquetes
 - a. Capacidad del Venue no superada → La cantidad de Tiquete publicados no puede ser mayor a Venue.capacidadMaxima. Esto es lo mismo que decir que las sumas de las capacidades de las localidades no puede ser superada ($\text{Venue.capacidadMaxima} = \sum \text{Localidad.capacidad}$).
 - b. Localidad consistente → Todo Tiquete (incluyendo TiqueteBasico y TiqueteNumerado) debe referenciar una localidad válida para ese Evento.
 - c. Numeración coherente → Si el tiquete es TiqueteNumerado, localidad.numerada = true y numeroAsiento no es vacío.
4. Precios, cargos y no-negatividad
 - a. Precio homogéneo por localidad (adición extra definida por el grupo) → Para una misma Localidad de un Evento, todos los Tiquete mantienen el mismo precioBase.
 - b. No-negatividad → precioBase, cargoPorcentual, cuotaAdicionalEmision ≥ 0 en Tiquete; precio ≥ 0 en Localidad; porcentajeAdicional, cuotaAdicional ≥ 0 en Administrador; descuento ≥ 0 en Oferta.
5. Ofertas
 - a. Temporalidad correcta → inicio \leq final en Oferta.
 - b. Oferta aplica a su objetivo únicamente → Una Oferta sólo aplica a la localidad y/o evento referenciados en la misma instancia.
 - c. Oferta no genera precios negativos → El descuento no puede producir un cobro negativo.
6. Paquetes múltiples y deluxe
 - a. Integridad del paquete múltiple → tiquetesIncluidos no puede estar vacío, precioPaquete ≥ 0 y paquetesMaximos ≥ 1 .
 - b. Mismo evento vs. distintos → tiquetesIncluidos pertenecen al mismo Evento en TiqueteMultipleMismoEvento, mientras que pertenecen a distintos Evento en TiqueteMultipleDistintosEvento.
 - c. Paquete Deluxe intransferible → Los tiquetesIncluidos en un PaqueteDeluxe no pueden ser agregados a tiquetesTransferidos de Cliente (no son transferibles individualmente ni como paquete).
7. Transferencias y Estado

- a. Transferencias válidas → Un Cliente solo puede mover un ticket de ticketsComprados a ticketsTransferidos si el ticket no está vencido ni asociado a un Evento cancelado (el estado que combina ambas se define con estado en Ticket).
 - b. Recepción consistente → Si un ticket aparece en ticketsTransferidos de un cliente emisor, también debe aparecer en ticketsRecibidos del cliente receptor (misma instancia de Ticket).
- 8. Roles y Operaciones
 - a. Administrador no compra → Instancias de Administrador no deben tener relaciones de compra de Ticket.
 - b. Oferta creada por Organizador → En Oferta, organizador debe ser el mismo OrganizadorDeEventos responsable del Evento en cuestión.

Descripción para cada Programa de Prueba

1. **Creación y gestión de usuarios:** El objetivo aquí es verificar que se pueden crear Cliente, OrganizadorDeEventos y Administrador con login, password y saldo. En el programa de prueba podemos demostrar que la jerarquía de Usuario funciona y que cada rol tiene sus atributos y restricciones (ej. el administrador no puede comprar tickets).
2. **Gestión de Venues:** Queremos probar que un OrganizadorDeEventos puede sugerir un Venue y que el Administrador puede aprobarlo o rechazarlo. Aquí podemos revisar la conexión que encontramos entre el OrganizadorDeEventos, Administrador y Venue, además de la regla de dominio: un evento sólo puede crearse en un venue aprobado.
3. **Creación y asignación de Localidades:** En esta parte queremos que un organizador pueda crear un Evento en un Venue aprobado y definir sus Localidad, es decir, que cada Evento está asociado a un OrganizadorDeEventos y un Venue, y que se cumplen restricciones como no tener dos eventos el mismo día en el mismo lugar.
4. **Venta de tickets básicos y numerados:** Aquí revisamos que un Cliente puede comprar TicketBasico (ej: gramilla) y TicketNumerado (ej: silla con asiento específico), demostrando que las asociaciones entre Cliente, Evento, Localidad y Ticket funcionan correctamente, y que no se asignan dos veces los mismos asientos.
5. **Venta de tickets múltiples:** Revisamos que el sistema maneja correctamente TicketMultipleMismoEvento y TicketMultipleDistintosEventos, haciendo el caso que se respete el tope máximo de tickets por transacción.
6. **Transferencia de paquetes:** Permitir que un Cliente transfiera un ticket a otro cliente, respetando restricciones, como no transferir Deluxe y no transferir paquetes expirados.
7. **Ofertas y descuentos:** Queremos asegurarnos que un organizador pueda crear una Oferta sobre localidades específicas en un rango de fechas, y que efectivamente los tickets afectados por la oferta aplican el descuento correcto y que se respeta la vigencia de la promoción.
8. **Organizador y administrador:** Que un organizador pueda revisar ventas, ganancias etc. y que los valores concuerden: valores netos, los sobrecargos por servicio y las cuotas de emisión.
9. **Reembolsos:** Validar que el administrador pueda cancelar un evento o aprobar reembolsos por solicitud de clientes, y que al hacer un reembolso se acredite al cliente en su saldo.