

ETAPA 1: Análisis del proyecto

Laura Sánchez, Santiago Ramírez, Pablo Rubio

Modelo de Dominio

https://lucid.app/lucidchart/450e5ca6-94ad-47f5-9a68-daf118c877f5/edit?invitationId=inv_57f8b1d5-65b5-4633-9dbd-6e3d0559a799

Problema a resolver

La aplicación busca resolver la necesidad de digitalizar la venta de tickets para conciertos y otro tipo de eventos. Esto de manera segura y ordenada contemplando los diferentes roles que pueden tener los diferentes usuarios del sistema, los diferentes tipos de entradas que existen y un control financiero transparente tanto para la ticketera como para los organizadores.

Restricciones de dominio

- **Restricciones por *Venue***

El *Venue* tiene una capacidad definida y no se pueden vender más tickets que la capacidad del *Venue*, por otro lado los *Venue* solo pueden alojar un evento por fecha.

- **Restricciones de Tickets**

Para los tickets cada uno debe tener un identificador único, estos se pueden vender individualmente o en paquetes y en mayoría son transferibles entre usuarios, pero si el paquete es de categoría *Deluxe* este no es transferible. Por último debe existir un máximo de tickets por transacción.

- **Restricciones de Cancelaciones y Reembolsos**

Los reembolsos dependen de qué clase de usuario cancela el evento, si cancela el administrador el reembolso debe incluir el precio base del ticket menos el costo de emisión, por otro lado si cancela el organizador por insolvencia el reembolso sólo incluirá el precio base del ticket. La única situación que exceptúa estos otros dos casos es por calamidad personal, en este caso dependerá totalmente del administrador el valor del reembolso.

- **Restricciones de Finanzas**

El organizador genera ganancias solo del precio base del ticket mientras que el administrador se queda con los cargos adicionales. El organizador tiene acceso a el total de ingresos, los porcentajes de venta y reportes dependiendo del evento o localidad, el administrador solo tiene acceso a las ganancias que genera la ticketera por recargos y reportes generales.

- **Restricciones de Persistencia**

La aplicación debe tener un sistema de persistencia implementado que use archivos planos o binarios, la carpeta donde estos archivos deben ser almacenados debe ser independiente a la carpeta donde se encuentra el código.

- **Restricciones de Implementación**

La aplicación debe ser implementada en su totalidad en Java.

Descripción de los programas de pruebas

- **Gestión de Usuarios**

Esta prueba busca demostrar la capacidad de la aplicación de crear diferentes tipos de usuarios (clientes, organizadores, administradores) a través de un inicio de sesión con login y contraseña. Se validará que el sistema reconoce correctamente el tipo de usuario y las capacidades que tiene este para usar el sistema.

- **Creación de Eventos**

Esta prueba demostrará el correcto funcionamiento de la programación de eventos. Un organizador inicia sesión y tendrá la facultad de programar un evento nuevo, para ello seleccionara un *Venue* aprobado por el administrador, asigna la fecha y la hora, define las localidades disponibles y el número de tickets que se habilitarán para cada una. La prueba demostrara que el sistema respeta las restricciones de capacidad del venue y que solo se permite un evento por fecha en el mismo lugar

- **Venta de Tickets**

En prueba muestra el funcionamiento de la venta de tickets a un cliente dependiendo del tipo de ticket. El sistema calculará el precio del ticket incluyendo el precio base, el cargo porcentual por servicio y la cuota fija de emisión. El resultado final refleja el cobro correcto y la asignación de los tickets adquiridos al cliente.

- **Transferencia de Tickets**

Se probará que un cliente puede transferir un ticket a otro usuario de la plataforma. Para completar la operación, se solicita el login del receptor y la contraseña del emisor, garantizando seguridad en el proceso y respetando las restricciones por tipo de ticket. En el caso de que se transfieran tickets múltiples se deben respetar las reglas de transferencias parciales o completas, demostrando el correcto funcionamiento de las ventas.

- **Aplicación de Oferta**

Esta prueba demostrará el correcto funcionamiento de la capacidad que tiene un organizador de aplicar descuentos temporales en una localidad específica de un evento. El sistema muestra cómo se reduce el precio de las entradas en el período definido y, al expirar la oferta, los valores vuelven automáticamente a la tarifa normal.

- **Cancelación de Eventos**

En esta prueba se demostrará la capacidad que tiene el administrador de cancelar un evento. Dependiendo de la causa, el sistema aplica reglas distintas de reembolso: Si lo cancela el administrador, se devuelve al cliente el valor total menos el costo de impresión. Si lo cancela el organizador, el cliente recibe únicamente el precio base, quedando los cargos adicionales en poder de la tiquetera. Esta prueba también demuestra el buen funcionamiento del cálculo del reembolso.

- **Consulta de Finanzas**

Esta prueba demuestra que tanto organizadores como administradores pueden consultar sus reportes financieros. El organizador accede a sus ingresos netos, con posibilidad de verlos globalmente, por evento o por localidad. El administrador, en cambio, puede observar las ganancias de la tiquetera, derivadas de los cargos por servicio y emisión, con reportes por fecha, evento o promotor.

Desplazamiento de responsabilidades

- **Usuario**

Esta clase incluye datos comunes de autenticación (login/password). Sirve como superclase para los distintos roles en la empresa como Administrador, Cliente y Organizador, compartiendo comportamientos y atributos generales.

- **Cliente**

Representa al usuario comprador. Puede explorar eventos/localidades, comprar tiquetes (generando pagos), solicitar devoluciones y transferir tiquetes, manteniendo la propiedad y el estado de sus entradas.

- **Administrador**

Esta clase es la encargada de supervisar las operaciones y políticas del sistema. Puede fijar cargos (servicio/impresión), aprobar o rechazar devoluciones, gestionar aprobación/cancelación de eventos y consultar ganancias a través de EstadosFinancieros.

- **Organizador**

Es el dueño de uno o varios eventos. Crea eventos (quedando asociado como su organizador), gestiona localidades y promociones (ofertas) para sus localidades, y puede consultar sus ingresos y porcentajes de venta por evento/localidad.

- **EstadosFinancieros**

Lleva la contabilidad del sistema, a través de un registro de transacciones. Utiliza los métodos de Pago para sacar las ganancias totales de la empresa por rango de fechas, por evento o por organizador.

- **Evento**

Esta clase es la encargada de representar e instanciar el evento que es programado con un organizador, con su respectivo venue y sus respectivas localidades.

- **Localidad**

Esta clase representa una localidad dentro de un evento, esta define dónde se sentará el dueño de un tiquete, también define el precio base para el tiquete y tiene una cantidad fija de puestos limitando su disponibilidad.

- **Venue**

Esta clase representa el sitio donde se va a llevar a acabo el evento, basado en la capacidad del venue es que se decide la capacidad del evento y la disponibilidad de este.

- **Oferta**

Esta clase se encarga de instanciar una oferta particular generada por un organizador para una localidad específica, esta cuenta con una fecha de expedición y vencimiento que determina el plazo de su vigencia.

- **Pago**

Esta clase tiene la responsabilidad de generar los pagos para los clientes. Al instanciar un pago no solo se calcula el monto total de la compra del cliente sino que también se registran datos esenciales para el registro financiero como lo son la fecha de expedición del pago, el método de pago usado por el cliente, un estado del pago y un identificador único para la instancia del pago

- **estadoPago y metodoPago**

Ambas clases son enumerators que contienen los posibles estados que pueden tener un pago y los posibles métodos de pago que tiene disponible el cliente.

- **Tiquete (abstracta)**

Esta clase representa el modelo principal de cualquier tiquete dentro del sistema. Guarda la información básica como el id, el dueño actual, si se puede transferir o no, el tipo y el estado del tiquete. Además, define métodos que las demás clases deben implementar, como el cálculo del precio total o la forma en la que se transfiere un tiquete.

- **TiqueteSimple**

Esta clase representa un tiquete individual que pertenece a un evento y a una localidad. Calcula el precio total sumando el valor base de la localidad con los cargos adicionales. Si la localidad tiene asientos numerados, esta clase también se encarga de asignar el asiento correspondiente al comprador.

- **TiqueteMultiple**

Esta clase representa un paquete que agrupa varios tiquetes simples. Puede tener un precio total definido o calcularse sumando el precio de cada entrada. Permite hacer transferencias completas o parciales, dependiendo del estado del paquete y de las reglas del sistema.

- **PaqueteDeluxe**

Esta clase hereda de TiqueteMultiple, pero a diferencia de este, no se puede transferir. Todos sus métodos de transferencia están bloqueados, ya que los paquetes Deluxe son exclusivos del comprador original.

- **estadoTiquete (enum)**

Esta clase define los diferentes estados que puede tener un tiquete, como disponible, comprado, transferido o expirado. Con esto el sistema puede saber en qué momento está cada tiquete y qué acciones se pueden hacer con él.