Proyecto 3 Boleta Master

Laura Sánchez, Santiago Ramírez, Pablo Rubio

ETAPA 1: Análisis del proyecto

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 tiquetes 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 tiquetera como para los organizadores.

Restricciones de dominio

- **Restricciones por *Venue***

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

- **Restricciones de Tiquetes**

Para los tiquetes 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 tiquetes 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 tiquete menos el costo de emisión, por otro lado si cancela el organizador por insolvencia el reembolso sólo incluirá el precio base del tiquete. 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 tiquete 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 tiquetera 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.

ETAPA 2: Implementación y pruebas

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 tiquetes 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 Tiquetes**

En prueba muestra el funcionamiento de la venta de tiquetes a un cliente dependiendo del tipo de tiquete. El sistema calculará el precio del tiquete 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 tiquetes adquiridos al cliente.

- **Transferencia de Tiquetes**

Se probará que un cliente puede transferir un tiquete 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 tiquete. En el caso de que se transfieran tiquetes 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.

ETAPA 3: Implementación interfaces gráficas

El diseño general del sistema Boleta Master es una aplicación desarrollada en Java que integra tanto la interfaz gráfica (GUI) hecha en Windowbuilder, como toda la lógica de dominio necesaria para la gestión de eventos, tiquetes, finanzas, cancelaciones y usuarios. El objetivo principal es mostrar cómo está organizado el sistema, qué clases lo componen, cómo se relacionan entre sí y por qué se tomaron ciertas decisiones de diseño basadas en las reglas de dominio del proyecto.

Nuestro diseño parte de una ventana inicial de Login, que es el punto de entrada para cualquier usuario. Dependiendo de las credenciales ingresadas, el sistema redirige al usuario a uno de los tres menús principales:

- Menú Administrador
- Menú Organizador
- Menú Cliente

Cada menú ofrece funcionalidades específicas según el rol. Por ejemplo, el administrador puede aprobar venues, revisar cancelaciones o gestionar cargos adicionales; el organizador puede crear eventos, asignar localidades o consultar ingresos; y el cliente puede comprar tiquetes, recargar saldo, pedir devoluciones e imprimir sus tiquetes.

De esta manera, la interfaz no solo guía el flujo del sistema, sino que organiza claramente las capacidades de cada tipo de usuario.

Diagrama de clases actualizado:
https://lucid.app/lucidchart/450e5ca6-94ad-47f5-9a68-daf118c877f5/edit?invitationId=inv_57f8b1d5-65b5-4633-9dbd-6e3d0559a799&page=0_0#

Diagrama clases interfaces:
https://lucid.app/lucidchart/8efce1c5-5c88-4125-826e-701d710a58b8/edit?invitationId=inv_6f3c7556-667e-44ab-a728-177bc0b4571f&page=0_0#

Diagrama clases de alto nivel:
https://lucid.app/lucidchart/586500c4-cc5e-4db6-bb2b-28f37c51d93c/edit?viewport_loc=-5503%2C-5008%2C11673%2C7257%2C0_0&invitationId=inv_1170f1de-baf1-44b5-b671-3a4e5133c715

Decisiones de diseño

- **Administrador único con credenciales fijas**

De acuerdo con las reglas de dominio, el sistema únicamente permite la existencia de un Administrador. Por ello se implementó un único perfil con login predeterminado. Usuario: ADMIN, Clave: DPOO. No se permite registrar nuevos administradores en la interfaz de Registro. De esta manera se garantiza el cumplimiento estricto del dominio y evita estados inconsistentes o problemas de seguridad derivados de múltiples cuentas administrativas.

- **Perfiles dinámicos para Organizadores y Clientes**

A diferencia del administrador, los perfiles de Organizador y Cliente si pueden ser creados vía Registro. Esto permite escalar el uso del sistema e incorporar múltiples actores que interactúan con eventos y compras de tiquetes. Representa la dinámica real del negocio, donde distintos clientes y organizadores interactúan constantemente con la plataforma.

- **Aprobación obligatoria de Venues por parte del Administrador**

Cuando un Organizador propone un Venue, este no se activa automáticamente sino que pasa a un estado de *pendiente* donde debe ser aprobado explícitamente por el Administrador. De esta manera se controla la disponibilidad y calidad de los escenarios, y garantiza la verificación de capacidad, costos, fechas y cumplimiento normativo.

- **Acceso del Organizador al menú de Cliente**

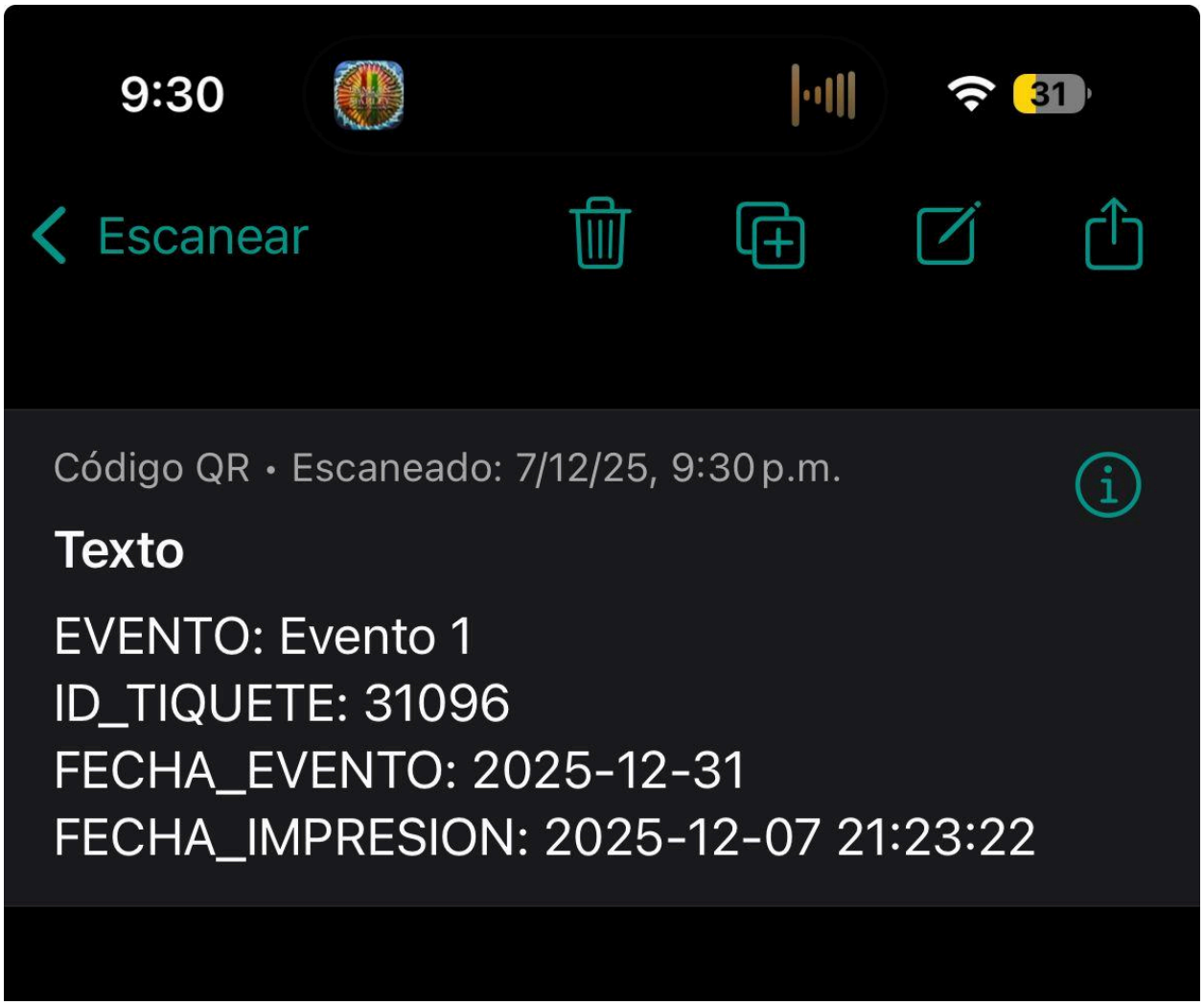
Los Organizadores pueden entrar al Menú de Cliente desde su propio menú, ya que ellos también pueden comprar tiquetes y de hecho muchos organizadores compran tiquetes para pruebas técnicas, invitados especiales o cortesías. Se modela así una interacción flexible sin romper las reglas de dominio.

Así mismo, se aplicaron las restricciones de dominio a la interfaz gráfica de la siguiente manera:

- La clase Venue contiene atributos de capacidad y calendario interno.
- La clase Evento valida disponibilidad antes de ser creado.
- Cada tiquete tiene un identificador único.
- Las clases Tiquete y PaqueteTiquetes manejan unicidad y transferencias.
- La clase Cliente valida límites por compra.
- Si admin cancela un evento, hay un reembolso que es el precio base – costo de emisión.
- Si organizador cancela evento por insolvencia, el reembolso es igual al precio base.
- Si es por algún asunto personal, el administrador define el monto.
- El Administrador es quien decide reembolsos y cancelaciones según su propio criterio

***Anotación sobre los QR**

Hay ciertas aplicaciones que no detectan la información de los qr de los tiquetes impresos pero estos son totalmente funcionales como aparece en el screenshot de la parte inferior


9:30
31
Escanear
Código QR • Escaneado: 7/12/25, 9:30 p.m.
Texto
EVENTO: Evento 1
ID_TIQUETE: 31096
FECHA_EVENTO: 2025-12-31
FECHA_IMPRESION: 2025-12-07 21:23:22