Documento de Diseño – Proyecto 1

1) Contexto y alcance

Contexto: Sistema para usuarios que simula venta de tiquetes, gestión de eventos/venues y finanzas, con autenticación básica

Problema: Se requiere una plataforma de venta de tiquetes para eventos con roles (Cliente, Organizador, Administrador), control de ofertas, paquetes especiales, transferencias, cancelaciones y estados financieros, con persistencia en archivos

Solución propuesta:

- Arquitectura por capas: Aplicación, Dominio, Infraestructura
    - Dominio centrado en Evento, Localidad, Tiquete, Transacción, Oferta y Saldo Virtual.
    - Persistencia en archivos planos (txt).
    - Los servicios de aplicación gestionan las reglas de negocio
    - La validación se realiza mediante pruebas automatizadas y ejecutables que ejercitan los casos de uso.

Stakeholders: Clientes, Organizadores, Administrador

Restricciones/condiciones:

- Persistencia: archivos planos en carpeta externa al código.
- Usuarios con login/password.
- No hay interfaz de usuario
- No se implementa pasarela de pago externa
- Existe Saldo Virtual para reembolsos y como medio de pago interno.

2) Objetivos y No-objetivos

Objetivos.

1. Implementar compra de tiquetes (simples, numerados, múltiples, multievento, Deluxe) respetando topes y precios (cargo servicio (%), cuota fija de la emisión, ofertas temporales).

2. Gestionar eventos/venues/localidades, incluyendo la aprobación de venues y unicidad de evento por venue-fecha, y generación de tiquetes.

3. Soportar transferencia (con restricciones), cancelaciones y reembolsos a Saldo Virtual.

4. Presentar consultas de finanzas del organizador y ganancias de la tiquetera.

No-objetivos.

- Interfaz de usuario completa
- Pasarelas de pago reales
- Reportes gráficos

4) Reglas del dominio

1. ID de Tiquete es único en toda la plataforma
2. Precio homogéneo por Localidad; si es numerada, el número de asiento es único dentro de su Localidad
3. Un Venue no aloja más de un Evento en la misma fecha
4. El tope por transacción se calcula sobre el ítem agregado no por tiquete individual
5. El PaqueteDeluxe es intransferible
6. Transferencia: Requiere login del receptor y password del emisor; restricción de paquetes completos si hay vencidos/transferidos
7. El Administrador fija CargoServicio por tipo de evento y CuotaEmision fija
8. Cancelacion y reembolsos: reglas diferenciadas, con acreditación de SaldoVirtual (1:1 con Usuario)
9. El Administrador no compra tiquetes
10. El organizador puede ser también Cliente
11. Si el organizador compra tiquetes para su propio evento estos no generarán un cobro que se refleje en los ingresos ni de la plataforma ni del mismo promotor.
12. No es posible tener eventos sin un promotor u organizador de eventos.
13. Para poder crear un evento se debe tener asociado un Venue.
14. El organizador es quien tiene la responsabilidad de asignar la cantidad y tipo de tiquetes disponibles al público.

5) Diseño de la solución

Vista de contexto

- **Actores externos:**
    - Cliente
    - Organizador
    - Administrador.

- **Sistema Boletamaster:**
    - Servicios de Autenticación
    - Eventos/Venues/Localidades
    - Ventas/Ofertas
    - Inventario de tiquetes
    - Transferencias

- Cancelaciones/Reembolsos
- Reportes.

Interacciones clave:

1. Organizador propone Venue → Admin aprueba
2. Organizador crea Evento+Localidades → sistema genera inventario de tiquetes
3. Cliente compra (aplica oferta, cargo %, cuota fija) → emite Tiquetes y Transacción
4. Cliente transfiere (validación)
5. Admin/Organizador cancelan → reglas de reembolso → acreditación en Saldo Virtual
6. Consultas financieras.

6) Elementos de la solución

Clases del dominio

Usuarios

Usuario (abstracta)

- **Atributos:**
  - login:String
  - password:String
  - nombre:String
  - email:String
  - rol:Rol

- **Métodos:**
  - boolean autenticar(String passPlano)
  - void cambiarPassword(String passActual, String passNuevo)
  - Rol getRol()

Cliente extends Usuario

- **Atributos:**
  - saldo:SaldoVirtual

- **Métodos:**
  - double consultarSaldo()
  - void acreditarSaldo(double monto)
  - void debitarSaldo(double monto)
  - SaldoVirtual getSaldoVirtual()

Organizador extends Usuario

- **Atributos:** ninguno adicional

- **Métodos:**
  - Evento crearEvento(CmdCrearEvento cmd)
  - Localidad configurarLocalidad(String idEvento, CmdLocalidad cmd)
  - Oferta crearOferta(String idLocalidad, CmdOferta cmd)
  - SolicitudVenue proponerVenue(CmdVenue cmd)
  - void solicitarCancelacionEvento(String idEvento)
  - EstadoFinanciero revisarFinanzas(FiltroFinanzas filtro)

Administrador extends Usuario

- **Atributos:** ninguno adicional

- **Métodos:**
  - void aprobarVenue(String idSolicitud)
  - void rechazarVenue(String idSolicitud, String motivo)
  - void fijarCargoServicio(TipoEvento tipo, double porcentaje)
  - void fijarCuotaEmision(double monto)
  - void cancelarEvento(String idEvento)
  - GananciasGenerales revisarGanancias(FiltroGanancias filtro)

SaldoVirtual

- **Atributos:**
  - usuario:Usuario
  - saldo:double
- **Métodos:**
  - void acreditar(double monto)
  - void debitar(double monto)
  - boolean puedePagar(double monto)

ESCENARIOS Y VENTAS

Venue

- **Atributos:**
  - id:String
  - nombre:String
  - ubicacion:String
  - capacidad:int
  - aprobado:boolean

    - restricciones:String

- **Métodos:**
    - void marcarAprobado()
    - void marcarRechazado()
    - boolean esDisponible(LocalDateTime fechaHora)

SolicitudVenue

- **Atributos:**
    - id:String
    - propuestoPor:Organizador
    - venuePropuesto: String
    - estado:EstadoSolicitud {PENDIENTE,APROBADA,RECHAZADA}
    - fecha:LocalDate
    - motivoRechazo:String
- **Métodos:**
    - void aprobar()
    - void rechazar(String motivo)

Evento

- **Atributos:**
    - id:String
    - nombre:String
    - fechaHora:LocalDateTime
    - tipo:TipoEvento
    - venue:Venue
    - organizador:Organizador
    - estado:EstadoEvento {PROGRAMADO,CANCELADO}
    - localidades:List<Localidad>

- **Métodos:**
    - Localidad agregarLocalidad(CmdLocalidad cmd)
    - int generarTiquetes(String idLocalidad)
    - double porcentajeOcupacion()
    - void cancelarPorAdmin()
    - void cancelarPorOrganizadorAutorizada()

Localidad (composición de Evento)

- **Atributos:**
    - id:String
    - evento:Evento

- nombre:String
- precioBase:double
- numerada:boolean
- capacidad:int
- asientos:Set<int>
- ofertas:List<Oferta>

- **Métodos:**
  - int generarInventario()
  - double precioVigenteCent(LocalDateTime ahora)
  - Tiquete emitirTiquete(Cliente duenio, String asientoOptional)
  - boolean asientoDisponible(String asiento)
  - void agregarOferta(Oferta oferta)

Oferta

- **Atributos:**
  - id:String
  - localidad:Localidad
  - porcentaje:double
  - inicio:LocalDateTime
  - fin:LocalDateTime

- **Métodos:**
  - boolean estaVigente(LocalDateTime ahora)
  - double aplicarOferta(double precioBase)

Tiquetes y paquetes

Tiquete (abstracta)

- **Atributos:**
  - id:String
  - evento:Evento
  - localidad:Localidad
  - dueño:Cliente
  - precioBase:double
  - precioPagado:double
  - coutoEmisionFija: double
  - estado:EstadoTiquete {EMITIDO,TRANSFERIDO,REEMBOLSADO,VENCIDO}
  - paquete: Paquete

- **Métodos:**
  - boolean esVencido(LocalDateTime ahora)

- void marcarTransferido(Cliente nuevoDueño)
- void marcarReembolsado()
- Paquete getPaquete()
- double montoReembolsoporAdmin() (=preciopagado-cuotaEmsionFija; minimo 0)
- double montoReembolsoPorOrganizador() (=precioBase)

<u>TiqueteSimple</u> extends Tiquete

- **Atributos/Métodos:** hereda

<u>TiqueteNumerado</u> extends Tiquete

- **Atributos:**
  - asiento:String

- **Métodos:** hereda

<u>TiqueteMultiple</u> (paquete, mismo evento/localidad)

- **Atributos:**
  - id:String
  - evento:Evento
  - localidad:Localidad
  - tiquetes:List<Tiquete>
  - precioPaquete:double
  - dueño:Cliente

- **Métodos:**
  - void transferirTiquete(String idTiquete, String loginEmisor, String passEmisor, String loginReceptor)
  - void transferirPaquete(String loginEmisor, String passEmisor, String loginReceptor)
  - boolean tieneTiqueteVencidoOTransferido()

<u>TiqueteMultiEvento</u> (pase con varios eventos)

- **Atributos:**
  - id:String
  - componentes:List<ItemEventoLocalidad>
  - precioPaquete:double

- dueño:Cliente

- **Métodos:**
  - void transferirTiquete(ItemEventoLocalidad tiquete, String loginEmisor, String passEmisor, String loginReceptor)
  - void transferirPaquete(String loginEmisor, String passEmisor, String loginReceptor)
  - boolean tieneTiqueteVencidoOTransferido()

PaqueteDeluxe ***(intransferible)***

- **Atributos:**
  - id:String
  - tiquetes:List<Tiquete>
  - beneficios:List<String>
  - precioPaquete:double
  - dueño:Cliente

- **Métodos:**
  - List<String> listarBeneficios()

***ItemEventoLocalidad**

- **Atributos:**
  - evento:Evento
  - localidad:Localidad
  - cantidad:int

- **Métodos:**
  - boolean agotado()

Transacciones, transferencias y reembolsos

TransaccionCompra

- **Atributos:**
  - id:String
  - fecha:LocalDate
  - cliente:Cliente
  - lineas:List<LineaTransaccion>
  - total:double

- **Métodos:**
  - void agregarLinea(LineaTransaccion l)
  - void validarTope() (tope a nivel de línea/paquete)

- double totalizar()
- Recibo aRecibo()

Transferencia

- **Atributos:**
  - id:String
  - emisor:Cliente
  - receptor:Cliente
  - tiquetes:List<Tiquete>
  - fecha:LocalDate

- **Métodos:**
  - void validarPoliticas()
  - void ejecutar(String loginEmisor, String passEmisor)

DTOS

Recibo (comprobante de compra)

- **Atributos:**

  - idTransaccion:String
  - fecha:LocalDate
  - loginCliente:String
  - items:List<ItemRecibo>
  - total:double
  - cargo:double
  - cuotaFija:double

ItemRecibo

- **Atributos:**

  - tipoItem:String
  - refItem:String
  - cantidad: int
  - precioBase: double
  - descuento: double
  - cargoServicio:double
  - cuotaEmisionFija:double
  - subtotal: double

LineaTransaccion

- **Atributos:**
  - TipoItem tipoItem;
  - String refItem;
  - int cantidad;
  - double precioBase;
  - double descuento;
  - double cargoServicio;
  - double cuotaEmisionFija;
  - double subtotal;

- **Métodos:**
  - double calcularSubtotal()

CmdCrearEvento

- **Atributos:**
  - idEvento: String
  - nombre: String
  - fechaHora: LocalDateTime
  - tipo: TipoEvento
  - venue: Venue

CmdLocalidad

- **Atributos:**
  - idLocalidad: String
  - nombre: String
  - precioBase: double
  - numerada: boolean
  - capacidad: int

CmdOferta

- **Atributos:**
  - idOferta: String
  - localidad: Localidad
  - porcentaje: int
  - inicio: LocalDateTime
  - fin: LocalDateTime

CmdVenue

- **Atributos:**
  - idSolicitud: String
  - nombrePropuesto: String
  - fecha: LocalDate

Reportes y filtros

FiltroFinanzas

- **Atributos:**
  - private LocalDate desde
  - private LocalDate hasta
  - private TipoEvento tipo

FiltroGanancias

- **Atributos:**
  - private LocalDate desde
  - private LocalDate hasta
  - private TipoEvento tipo

EstadoFinanciero

- **Atributos:**
  - private String loginOrganizador
  - private int tiquetesVendidos
  - private double ingresosBase
  - private Map<String, Double> ingresosPorEvento

- **Métodos:**
  - public void addIngresoEvento(String idEvento, double monto)

GananciasGenerales

- **Atributos:**
  - private int ticketsContados;
  - private double totalCargoServicio
  - private double totalCuotaEmision
  - private double totalPlataforma
  - private EnumMap<TipoEvento, Double> cargoPorTipo
- **Métodos:**
  - public void addCargoPorTipo(TipoEvento tipo, double cargo)

Run

AppModule

Atributos:

- private final TiqueteraContext ctx
- private final AuthService authService
- private final VenueService venueService
- private final EventoService eventoService
- private final CompraService compraService
- private final TransferenciaService transferenciaService
- private final ReporteService reporteService

Métodos:

- public TiqueteraContext getContext()
- public AuthService auth()
- public VenueService venues()
- public EventoService evento()
- public CompraService compras()
- public TransferenciaService transferencias()
- public ReporteService reportes()
- public AppModule setCuotaEmisionFija(double cuota)
- public AppModule setCargoServicio(enums.TipoEvento tipo, double porcentaje)

TiqueteraContext

Atributos:

- public final Map<String, Usuario> usuarios

- public final Map<String, Venue> venues

- public final Map<String, SolicitudVenue> solicitudes

- public final Map<String, Evento> eventos
- public final Map<String, Localidad> localidades
- public final Map<String, Tiquete> tiquetes

- public final List<TransaccionCompra> transacciones
- public final EnumMap<TipoEvento, Double> cargoPorTipo
- public Double cuotaEmisionFija

Métodos:

- public static String keyLoc(String idEvento, String idLocalidad)

- public Optional<Usuario> findUsuario(String login)

- public Optional<Venue> findVenue(String id)

- public Optional<SolicitudVenue> findSolicitud(String id)

- public Optional<Evento> findEvento(String id)

- public Optional<Localidad> findLocalidad(String idEvento, String idLoc)

- public List<Tiquete> tiquetesPorEvento(String idEvento)

APP

AuthService

- Usuario login(String login, String pass)

VenueService

- SolicitudVenue proponerVenue(CmdVenue cmd)
- void aprobarSolicitud(String idSolicitud)
- void rechazarSolicitud(String idSolicitud, String motivo)
- List<Venue> listarVenues()

EventoServicie

- Evento crearEvento(CmdCrearEvento cmd)
- Localidad agregarLocalidad(String idEvento, CmdLocalidad cmd)
- int generarTiquetes(String idEvento,String idLocalidad)

CompraService

- Cotización cotizar(CmdCompra cmd)
- Recibo comprar(CmdCompra cmd)
- double cargoServicio(TipoEvento tipo)

TransferenciaService

- void reembolsarPorAdmin(String idEvento) (usa Tiquete.montoReembolsoPorAdmin() y acredita SaldoVirtual)
- void reembolsarPorOrganizadorAutorizada(String idEvento) (usa Tiquete.montoReembolsoPorOrganizador())

ReporteService

- EstadoFinaciero finanzasOrganizador(String loginOrg, FiltroFinanzas f)
- GananciasGenerales gananciasAdministrador(FiltroGanacias f)

CmdCompra

- **Atributos:**
  - private String loginCliente, idEvento, idLocalidad, asiento
  - private int cantidad
  - private TipoItem tipoItem

Cotización

- **Atributos:**
  - private final double precioBase
  - private final double cargoServicio
  - private final double cuotaEmisionFija
  - private final double total

CONSOLA

public static void main(String[] args)

ENUMS

EstadoEvento{}

EstadoSolicitud{}

EstadoTiquete{}

Rol{}

TipoTiquete{}

TipoItem{}

IMPLEMENTACIÓN

AuthServiceSimple

Atributos:

- private final TiqueteraContext ctx

Métodos:

- public AuthServiceSimple(TiqueteraContext ctx)
- public Usuario login(String login, String pass)

CompraServiceSimple

Atributos:

- private final TiqueteraContext ctx

Métodos:

- public CompraServiceSimple(TiqueteraContext ctx)
- public Cotizacion cotizar(CmdCompra cmd)
- public Recibo comprar(CmdCompra cmd)
- public double cargoServicio(TipoEvento tipo)

EventosServiceSimple

- **Atributos:**
  - private final TiqueteraContext ctx

- **Métodos:**
  - public EventoServiceSimple(TiqueteraContext ctx)
  - public Evento crearEvento(CmdCrearEvento cmd)
  - public Localidad agregarLocalidad(String idEvento, CmdLocalidad cmd)
  - public int generarTiquetes(String idEvento, String idLocalidad)

ReporteServiceSimple

- **Atributos:**
  - private final TiqueteraContext ctx

- **Métodos:**
  - public ReporteServiceSimple(TiqueteraContext ctx)
  - public EstadoFinanciero finanzasOrganizador(String loginOrg, FiltroFinanzas f)

    - public GananciasGenerales gananciasAdministrador(FiltroGanancias f)
    - private boolean pasaRangoFecha(Evento e, LocalDate desde, LocalDate hasta)

TransferenciaServiceSimple

- **Atributos:**
    - private TiqueteraContext ctx

- **Métodos:**
    - public TransferenciaServiceSimple(TiqueteraContext ctx)
    - public void reembolsarPorAdmin(String idEvento)
    - public void reembolsarPorOrganizadorAutorizado(String idEvento)

VenueServiceSimple

- **Atributos:**
    - private final TiqueteraContext ctx

- **Métodos:**
    - public VenueServiceSimple(TiqueteraContext ctx)
    - public SolicitudVenue proponerVenue(CmdVenue cmd)
    - public void aprobarSolicitud(String idSolicitud)
    - public void rechazarSolicitud(String idSolicitud, String motivo)
    - public List<Venue> listarVenues()

RELACIONES

Herencia

- Usuario ⟶ Cliente, Organizador, Administrador

- Tiquete → TiqueteSimple, TiqueteNumerado

Composición

- Evento ■— Localidad
- TiqueteMultiple ■— Tiquete
- PaqueteDeluxe ■— Tiquete
- TiqueteMultiEvento ■— ItemEventoLocalidad

Agregación

- Localidad ◇— Oferta (1-1)
- EstadoFinanciero ◇— ResumenFinanzas (si se usa en reportes)
- GananciasGenerales ◇— (mapeos de cálculo)
- TiqueteMultiple ◇— Tiquete
- PaqueteDeluxe ◇— Tiquete

Asociación (simple/bidireccional)

- Evento — Venue (relación 1-1)
- Evento — Organizador (relación 1-1)
- Tiquete — Cliente (dueño) (relación *-1)
- TransaccionCompra — Cliente (relación *-1)

Asociación dirigida

- Localidad → Evento (apunta a su contenedor) (1-*)
- Oferta → Localidad (aplica sobre) (1-1)
- Transferencia → Cliente (emisor/receptor) (*-1)
- Tiquete → Localidad y Evento (validaciones y reembolsos) (*-1)
- SaldoVirtual → Usuario (1:1, pertenencia)
- CompraService → ParametrosRepo (para cargoServicio y cuota por defecto)
- ReembolsoService → TiqueteRepo & SaldoRepo (orquestación de reembolso)
- LineaTransaccion XOR → Tiquete, TiqueteMultiple, TiqueteMultiEvento, PaqueteDeluxe
- SolicitudVenue → Organizador (*-1)

7) Persistencia

Carpeta de datos: configurable fuera del proyecto. Por ejemplo: ../data-boletamaster

Archivos propuesto:

- archivo.txt

Esquemas mínimos:

- [PARAMETROS]
- cuotaEmisionFija=
- cargoPorTipo:

- [USUARIOS] total=
- - | | | saldo=

- [VENUES] total=
- - | | aprobado=

- [EVENTOS] total=
- - | | | estado= | aaaa-mm-dd hh:mm:ss | venue=

- [LOCALIDADES] total=
- - | | precioBase= | numerada= | cap=

- [TIQUETES] total=1
- - | ev= | loc= | | asiento= | estado= | base= | pagado= | cuota= | dueño=

- [TRANSACCIONES] total=
- - tx= | fecha=aaaa-mm-dd | cliente= | total=
- * TIQUETE ref= | cant= | base= | desc= | cargoPct= | cuotaFija= |subtotal=

- [REPORTES]
- GananciasGenerales
- cargos= | cuotas= | totalPlataforma= | tickets=
- EstadoFinanciero
- - org | base= | neto= | vendidos=

Transaccionalidad:

- El sistema garantiza el funcionamiento de las operaciones críticas. En caso de que surja un error o impedimento para el funcionamiento de dichas operaciones, el sistema evita estos casos al revisar si los parámetros dados son los correctos para el tipo de operación deseada, cuando esto sucede se notifica el motivo por el que el proceso se detiene y no puede continuar.

8) Algoritmos críticos

Estos algoritmos son esenciales para que la aplicación pueda funcionar. No se incluyen otros algoritmos que si bien no son importantes no son la base del funcionamiento del proyecto.

- transferirTiquete: Revisa los parámetros del método para revisar si son correctos. Con los datos verificados procede a la transferencia. Este algoritmo permite transferir los tiquetes entre los usuarios.
- registrarPago: Registra el pago si el monto es mayor a 0. Este algoritmo permite registrar el pago en el sistema, permitiendo entonces poder llevar la cuenta de otros métodos importantes como ganancias y posesión de tiquetes.
- transferirPaquete: Después de revisar los parámetros del método, se realiza el proceso de transferencia a otro cliente. Este algoritmo es esencial para permitir que solo ciertos paquetes sean transferibles.
- crearEvento: Mediante los datos correctos y la verificación de que no exista un evento en el venue deseado a la misma fecha, se genera el evento. Este algoritmo es la base que permite la creación de cualquier tipo de evento e impide el conflicto de tener más de un evento por venue a la misma fecha.
- aprobarVenue: Después de ser examinada la solicitud de un venue se da visto bueno a el venue sugerido. Este algoritmo es esencial para poder añadir posibles venues al sistema y permitir eventos en estos.
- generarTiquetes: Dependiendo de la capacidad del evento y del tipo de tiquetes disponibles, este algoritmo se encarga de crear los tiquetes de cualquier tipo deseados para un evento siempre y cuando cumpla con las restricciones del lugar. Este algoritmo es esencial ya que es el generador del producto que vende la aplicación y asimismo se encarga de distinguir y numerar los tiquetes.
- TiqueteraContext
- AppModule
- PlainTextWriter

9) Alternativas consideradas

- Se consideró como alternativa el uso de atributos en los clientes para almacenar sus tiquetes y paquetes, similar a la implementación usada en el taller 3.
- Se consideró como alternativa eliminar el tiquete simple y dejar únicamente tiquete como reemplazo al tiquete simple y al mismo tiempo como clase abstracta.

10) Preocupaciones transversales

- **Seguridad:**
    - Almacenamiento de password con Hash o posible cifrado de archivos.

- **Observabilidad:** log a archivo app.log por operación y IDs correlacionados.

- **Rendimiento:**
    - Carga lenta de inventario

- **Resiliencia:**
    - Escrituras con archivo temporal
    - Recuperaciones simples por re-lectura

- **Validaciones:**
    - Unicidad de ID
    - Ventana temporal de ofertas
    - Un evento por venue-fecha
    - Asiento único por localidad numerada