Proyecto 1

Documento de Diseño

Santiago Hernández Vélez – 202320909

David Varela Trujillo – 202321469

Bastien Quentin Clement Thirion - 202525085

Historias de usuario

Cliente:

1. Registro

Como cliente, quiero poder registrarme con un login y una contraseña para tener mi cuenta personal en la plataforma.

- Entrada: login, contraseña
- Salida: confirmación de registro exitoso o error si el login ya existe.
- Criterio de aceptación: El login es único, la contraseña se guarda correctamente, y los datos se almacenan en persistencia (clientes.ser).

2. Consulta de eventos

Como cliente, quiero consultar los eventos disponibles para elegir a cuál asistir.

- Entrada: ninguna (consulta general).
- Salida: lista de eventos con nombre, fecha, tipo, lugar y precio de las localidades.
- Criterio de aceptación: El sistema muestra todos los eventos activos cargados desde persistencia.

3. Compra de tiquetes

Como cliente, quiero comprar una o varias boletas para asistir a un evento.

- Entrada: evento seleccionado, cantidad, tipo de localidad y método de pago (saldo virtual o tarjeta).
- Salida: confirmación de compra y listado de tiquetes asignados.
- Criterio de aceptación: Si hay saldo suficiente, se descuentan los fondos y los tiquetes se asocian al usuario y al evento.

4. Visualización de tiquetes

Como cliente, quiero ver mis tiquetes comprados y sus detalles.

- Entrada: ninguna.
- Salida: lista con el tipo de tiquete, evento, fecha, hora, precio y estado (usado, transferido, disponible).

5. Transferencia de tiquete

Como cliente, quiero transferir un tiquete a otro usuario registrado.

- Entrada: ID del tiquete, login del destinatario.
- Salida: confirmación de transferencia exitosa o mensaje de error.
- Criterio de aceptación: No se permite la transferencia si el tiquete pertenece a un Paquete Deluxe o si está vencido.

6. Solicitud de reembolso

Como cliente, quiero solicitar un reembolso si el evento fue cancelado o si ya no puedo asistir.

- Entrada: ID del tiquete, motivo de solicitud.
- Salida: solicitud registrada en la lista del administrador.
- Criterio de aceptación: El administrador puede aceptar o rechazar la solicitud; si se aprueba, el dinero se devuelve al saldo virtual del cliente.

7. Reventa de tiquetes (nuevo módulo MarketPlace)

Como cliente, quiero publicar un tiquete transferible en el marketplace para revenderlo.

- Entrada: ID del tiquete, descripción, precio de venta.
- Salida: confirmación de publicación y registro en el LogReventa.
- Criterio de aceptación: Solo se pueden publicar tiquetes transferibles; el tiquete se bloquea hasta su venta o cancelación de la oferta.

8. Contraoferta en reventa

Como cliente, quiero poder hacer una contraoferta sobre una boleta publicada por otro usuario.

- Entrada: oferta seleccionada, nuevo precio, opción de usar saldo virtual.
- Salida: confirmación de contraoferta registrada.
- Criterio de aceptación: No se puede ofertar sobre una boleta propia. Si el vendedor acepta, el sistema transfiere automáticamente el tiquete y el dinero

Organizador:

1. Registro y aprobación

Como organizador, quiero registrarme con un login y contraseña para acceder a la plataforma, y que mi cuenta deba ser aprobada por el administrador antes de poder crear eventos.

- Entrada: login, contraseña
- Salida: confirmación de registro pendiente de aprobación.
- Criterio de aceptación: El administrador recibe la solicitud y debe aprobar al organizador para que pueda operar.

2. Creación de eventos

Como organizador, quiero crear un nuevo evento con nombre, descripción, tipo, fecha, hora y venue.

- Entrada: nombre, descripción, tipo de evento, fecha, hora, venue aprobado.
- Salida: evento creado y almacenado en persistencia (eventos.ser).
- Criterio de aceptación: El venue debe estar aprobado y disponible para esa fecha.

3. Creación de localidades

Como organizador, quiero definir las localidades de mi evento, incluyendo capacidad, tipo de tiquete y precio.

- Entrada: nombre, capacidad, tipo (“BASICO”, “ENUMERADO”), precio, descuento opcional.
- Salida: confirmación y actualización de la lista de localidades del evento.
- Criterio de aceptación: Se generan automáticamente los tiquetes asociados a la localidad.

4. Consulta de ganancias

Como organizador, quiero consultar las ganancias generadas por mis eventos.

- Entrada: selección de evento o consulta general.

- Salida: listado con ingresos por evento o total acumulado.
- Criterio de aceptación: El sistema muestra las ganancias netas sin incluir la tarifa de servicio.

5. Propuesta de nuevos venues

Como organizador, quiero proponer nuevos venues para mis eventos futuros.

- Entrada: nombre del venue, ubicación, capacidad.
- Salida: solicitud enviada al administrador.
- Criterio de aceptación: El administrador recibe la solicitud y puede aprobarla o rechazarla.

6. Cancelación de eventos

Como organizador, quiero solicitar la cancelación de un evento si es necesario.

- Entrada: evento y motivo de cancelación.
- Salida: solicitud enviada al administrador.
- Criterio de aceptación: Solo el administrador puede confirmar la cancelación; los clientes reciben reembolsos automáticos.

Administrador:

1. Aprobación de organizadores

Como administrador, quiero aprobar o rechazar los nuevos organizadores registrados.

- Entrada: lista de solicitudes pendientes.
- Salida: organizadores aprobados o rechazados con notificación.

2. Aprobación de venues

Como administrador, quiero aprobar o rechazar los lugares propuestos.

- Entrada: lista de solicitudes de venues.
- Salida: venue aprobado o rechazado.

3. Gestión de solicitudes de reembolso o cancelación

Como administrador, quiero revisar las solicitudes de reembolso o cancelación de eventos.

- Entrada: lista de solicitudes con su tipo y motivo.
- Salida: aceptación o rechazo con las acciones correspondientes (reembolso, cambio de estado del evento).

4. Control financiero

Como administrador, quiero consultar las ganancias por organizador, evento o fecha.

- Entrada: filtro seleccionado (por fecha, organizador o evento).
- Salida: informe de ganancias totales.

5. Gestión de usuarios

Como administrador, quiero poder crear, modificar o eliminar usuarios.

- Entrada: información del usuario.
- Salida: base de datos actualizada (persistencia).

7. Consulta del log del marketplace

Como administrador, quiero visualizar el historial de transacciones del marketplace para monitorear las operaciones entre clientes.

- Entrada: ninguna.
- Salida: lista de eventos registrados (publicaciones, ventas, rechazos).
- Criterio de aceptación: Solo el administrador puede acceder a este log.

Definición del dominio.

Una historia de usuario se considera terminada cuando todas las condiciones funcionales son respetadas, las reglas del dominio se aplican correctamente, las excepciones son

manejadas y los objetos persistentes (clientes, organizadores, eventos, tiquetes) son guardados y leídos correctamente desde los archivos serializados.

Clases Tiquetes

Tiquete

Clase abstracta que genera el objeto Tiquete. Esta clase guarda la localidad a la que pertenece, el id del usuario al que pertenece, un atributo estático del precio de la impresión, de tal forma que emitir cualquier tiquete tiene el mismo costo, una fecha LocalDate, una hora LocalTime, un id final que depende de la localidad y el evento, el usuario al que pertenece la boleta, su precio base, un String que determina el tipo de tiquete, el precio real y un atributo estático final tiquetesMaximosPorTransaccion que es el límite de tiquetes individuales que se pueden comprar al tiempo y un boolean comprado y otro transferible, que nos permiten saber si aquel tiquete ya fue comprado o si se puede transferir.

Decisiones relevantes.

Se decide dejar fijo un valor máximo de tiquetes que se pueden comprar de forma simultánea, pues consideramos que es un atributo ligado a la forma de operar de la tiquetera y no de un evento o localidad específica.

TiqueteBasico

Clase que hereda de tiquete, cuenta con un atributo final estático “BASICO” que se asigna a su tipo cuando es creado.

TiqueteEnumerado

Clase que hereda de tiquete, cuenta con un atributo estático “ENUMERADO” que se asigna a su tipo cuando es creado. Cuenta con un atributo entero idSilla, que es el número de la silla que se compra.

TiqueteMultiple

Clase abstracta que hereda de Tiquete, genera un objeto TiqueteMultiple, como atributos tiene una lista con los Tiquetes que contiene y un static int tiquetesMaximosPorTransaccionMultiples, que determina el número de tiquetes múltiples que se pueden comprar a la vez.

Decisiones relevantes.

Un tiquete multiple representa la compra de un paquete que incluye diferentes tiquetes. La gracia detrás de comprar tiquetes múltiples consiste en los precios asociados a su compra, los tiquetes que contiene dentro son entidades Tiquete descritas anteriormente. Todos los tiquetes que se generan a partir de un tiquete multiple son TiquetesBasicos.

TiqueteMultiEntrada

Esta clase hereda de TiqueteMultiple, crea la cantidad de tiquetes individuales que se le indica, y adapta su precio a lo que indique el organizador que crea una localidad de tiquetes múltiples, de tal forma que sean consistentes los valores.

Decisiones relevantes.

Esta clase es para comprar una cierta cantidad de tiquetes para un evento específico, todos los tiquetes comprados han de pertenecer a la misma localidad.

TiqueteMultiEvento

Clase que hereda de TiqueteMultiple, cuenta con un HashMap static donde, dependiendo de la cantidad de tiquetes, se le asocia un valor de descuento por comprar en cantidad. Esta clase busca tiquetes disponibles de los eventos solicitados y los asocia a sí mismo, modificando su precio según su cantidad para mantener una consistencia en ganancias.

Decisiones relevantes.

Se asume que, cuando se quiere comprar un tiquete con acceso a varios eventos, todos los tiquetes asociados son básicos, y todos los eventos a los que se quiere ir tienen un único Tiquete, como comprar un abono de un equipo de fútbol, que da acceso de 1 tiquete a diferentes partidos. Además, independientemente de los eventos, el valor de el tiquete múltiple depende exclusivamente de la cantidad de tiquetes que se compren, por eso se encuentra como estático en la clase.

Clases Cliente

Cliente

Clase que crea un objeto cliente, un cliente se crea a partir de un usuario y una contraseña, su saldo virtual inicia en 0 y tiene un mapa donde guarda cada tiquete por su id.

Organizador

Clase que hereda de Cliente. Además de los atributos y métodos de un usuario común, tiene un ArrayList con los eventos creados.

Decisiones relevantes.

El organizador debe ser capaz de revisar sus estados financieros, para eso almacena los eventos que él ha organizado. Los eventos cuentan con sus localidades, y cada localidad con sus tiquetes, por lo cual puede consultar sus estados financieros globales, por evento y por localidad específica de un evento. De la misma forma puede calcular los porcentajes de venta en las mismas instancias.

Otras Clases

Administrador

El administrador cuenta con un login y una contraseña, pero recordemos que no es un usuario del sistema. Además cuenta con una lista de solicitudes que debe atender.

Evento

Un evento cuenta con los siguientes atributos, un nombre, una fecha y hora, un Venue, un organizador, un ArrayList con las localidades del evento, y atributos finales estáticos que permiten ver los diferentes tipos de evento que puede tener: musical deportivo, religioso y cultural. Además cuenta con un String que es el estado del evento, un string que es el tipo del evento, y un HashMap estático de comisión por eventos, donde por cada tipo de evento, tiene asociado un porcentaje adicional de cobro específico, estos porcentajes los puede fijar el administrados.

Decisiones relevantes.

Se menciona que los sobrecargos por servicios son porcentajes que dependen del tipo de evento, por eso se guarda como estático para poder acceder siempre al sobrecargo dependiendo de los posibles tipos de evento: musical, cultural, deportivo, religioso. Además, al momento de crear una localidad, que puede ser o no numerada, cuando se añade al evento se le asigna un identificador para ese evento concreto, y se guarda en el HashMap mencionado.

Localidad

Cuenta con un nombre propio, el precio de los tiquetes que tienen acceso a la localidad, la capacidad, un boolean que determina si contiene tiquetes numerados, el venue del que hace parte, y el evento al evento. Tiene un ArrayList con todos los clientes asociados. Cuando se crea una localidad, aumtomaticamente se crean todos los tiquetes que pertenecen a ella, y se inician con un boolean comprado = false.

Decisiones relevantes.

Se decide almacenar todos los tiquetes en las localidades pues, dado que todas las localidades están ligadas a su evento, a través del evento se pueden consultar toda la información de los tiquetes por cada localidad, que para los estados financieros será muy útil.

Venue

Contiene la información de un venue, un HashMap con eventos que tiene reservados por fecha, una capacidad, un nombre y una ubicación.

Decisiones relevantes.

Se decide crear el HashMap de cada Venue de evento por fecha, para así al momento de crear un evento, no se vaya a crear en una misma fecha que otro, cosa que no se puede hacer.

Clase abstracta Solicitud

Sirve como entidad de organización para las solicitudes que se presentan entre entidades.

Decisiones relevantes.

Funciona como transporte de referencias a las entidades relacionadas en la solicitud, cuando la solicitud llegue al administrador, este ya tiene la referencia al objeto de la persona que hizo la solicitud (o sobre el evento que se hizo), no hay necesidad de busquedas adicionales.

Volver esta clase abstracta nos permite tener un comportamiento aislado para el caso de aceptación o rechazado de una solicitud, por ejemplo, para cancelar un evento el organizador envía la solicitud con el evento, y el administrador al recibirla puede invocar el método de aceptar la solicitud que automáticamente cancelara el evento y se encargara de todo lo que esto conlleva. Note que este proceso es mucho menos dispendioso debido a que la clase contiene dentro de ella referencias a todas las clases implicadas.

Funciona como un object value interno que se transfiere entre entidades.

Funciona como unidad de almacenamiento, el administrador puede consultar sus solicitudes que se guardan en un arraylist en su clase (se guardan polimorficamente)

Clase BoletasMaster

Esta es la clase centralizada del negocio, se encarga de controlar las interacciones entre entidades y más importante es la unica clase que se conecta con la persistencia y la consola (object value).

Decisiones relevantes.

Con la ayuda de los atributos esAdministrador, esOrganizador, esCliente es posible saber que usuario se ha logeado en cualquier momento, por ende, sabemos que métodos desplegar para que el usuario interactúe con ellos.

La clase almacena los objetos para ser procesados en persistencia garantizando la unicidad de la informacion. Esto es importante porque se puede almacenar la información de varias sesiones haciendo que, por ejemplo, las peticiones del organizador las pueda responder el administrador independientemente de que se hayan realizado en ejecuciones diferentes del programa

La unicidad de la informacion no solo se garantiza para el almacenamiento, sino para la escritura. En todo momento la clase logra que las interacciones tengan efecto sobre todas las clases pertinentes, por ejemplo, hacer una compra supone generar el nuevo Tiquete y añadirlo al Usuario y al Evento.

Se han almacenado los eventos en un arreglo secundario que los sorte por fecha, almacenar la información así mejora exponencialmente la complejidad temporal para acceder a eventos de una fecha.

Finalmente, se indican métodos de visualización para los casos donde es primordial para la entidad consultar información estructurada especifica como los estados financieros.

Clase ArchivoSerializable

La clase ArchivoSerializable pertenece al paquete Persistencia, se encarga de gestionar la persistencia de datos en el sistema mediante la serialización de objectos Java.

Es el componente que permite guardar y recuperar la información de los diferentes del dominio (usuarios, administrador, venues, eventos y tiquetes) gracias a archivos .ser de la carpeta Datos.

Decisiones relevantes.

Si el archivo de datos no existe, se crea automáticamente

Usamos el formato serializable que es de Java porqué es más fácil guardar y leer objetos complejos mientras que necesita más código con archivos planos

Las funciones de ArchivoSerialible se usan en la clase BoletasMaster para leer y escribir cada tipo de datos directamente desde el sistema

Clases de MarketPlace:

MarketPlace

Clase principal del módulo de reventa. Administra las ofertas, contraofertas y el registro de actividades (log).
Cuenta con una lista de Ofertas activas y un objeto de tipo LogReventa que guarda la trazabilidad de todas las acciones realizadas (publicaciones, eliminaciones, aceptaciones, rechazos).

Decisiones relevantes:
Se centraliza toda la lógica del intercambio en una única clase para mantener el control del flujo de las operaciones y facilitar la persistencia.
El uso del LogReventa como bitácora permite registrar toda la actividad sin afectar las clases principales, separando la lógica de negocio del seguimiento de acciones.

Oferta

Representa una publicación de venta en el marketplace.
Contiene un Tiquete en venta, el Cliente vendedor, una descripción, el precio, un booleano vendida y una lista de ContraOfertas recibidas.

El constructor verifica que el tiquete sea transferible antes de crear la oferta.
Permite agregar y eliminar contraofertas, modificar el precio o la descripción, y consultar el estado de venta.

Decisiones relevantes:
Se validó que sólo se puedan ofrecer tiquetes transferibles, previniendo errores de reventa indebida.
El manejo de contraofertas dentro de cada oferta permite mantener un historial claro de negociaciones por tiquete.

ContraOferta

Clase que representa la propuesta de un comprador sobre una Oferta existente.
Almacena el Cliente comprador, la Oferta original, el nuevo precio ofrecido, un booleano aceptada y otro usarSaldo para indicar si el pago se hace con saldo virtual.

Cuando una contraoferta es aceptada, se transfieren los fondos y el tiquete al comprador, actualizando el saldo de ambas partes.
Si es rechazada, se elimina del listado de contraofertas de la oferta original.

Decisiones relevantes:
El flujo de compra en la reventa mantiene las mismas reglas del sistema principal: si el comprador usa saldo virtual, se valida que tenga fondos suficientes.
La aceptación de la contraoferta ejecuta automáticamente la transferencia de tiquete, garantizando coherencia en las transacciones.

LogReventa

Clase auxiliar que registra todas las acciones realizadas dentro del marketplace.
Guarda una lista de cadenas con la descripción de cada evento y la fecha/hora en que ocurrió.

Decisiones relevantes:
El log permite tener trazabilidad completa del marketplace sin depender de una base de datos externa.

Se implementa como clase Serializable, lo que permite su persistencia junto con el resto del sistema, garantizando que los eventos se conserven entre sesiones.

Responsabilidades

Usuario

Un usuario tiene la capacidad de comprar tiquetes, ya sean básicos, enumerados, deluxe múltiples de un solo evento y múltiples de varios eventos. Además, tiene la capacidad de transferir dichos tiquetes y tiquetes singulares de los tiquetes múltiples. Puede usar los tiquetes, que al hacerlo quedan marcados como usados y puede solicitar un reembolso por calamidad, cuando es aceptado, se ejecuta el procedimiento para realizar el reembolso.

Cliente

No cuenta con responsabilidades o capacidades diferentes al usuario.

Organizador

Además de las capacidades que hereda de usuario, tiene la responsabilidad de crear los eventos, solicitar la creación de un venue, asignar la cantidad y tipo de tiquetes disponibles. Además tiene la capacidad de revisar sus estados financieros, globales, es decir de todos los eventos, por evento, y por localidad de un evento.

Administrador

Tiene la responsabilidad de fijar el sobrecargo por servicio a cada tipo de evento que se ofrecen, también fija el cobro por emisión de los tiquetes, el cual es fijo e igual para todos los tiquetes. La creación y aprobación de nuevos venues también depende del administrador. Además, puede cancelar un evento, en dicho caso se hace un reembolso por el precio base de los tiquetes. Finalmente, el administrador tiene la capacidad de ver los estados financieros de la tiquetera, evaluando ganancias totales, por fecha, evento u organizador.

Reglas y restricciones

Estados financieros

Hay múltiples restricciones ligadas a los estados financieros. En primer lugar. Todos los tiquetes que se generan reciben su precio de la siguiente forma. A partir de un precio base, que depende de la localidad a la cual pertenezca el tiquete, se calcula el precio real sobre este precio base, el costo de impresión del tiquete y un porcentaje de cobro por servicio que depende del tipo de evento. Cuando se crea el tiquete, se verifica que el usuario que lo este

comprando no sea el organizador del evento, en dicho caso, se asigna 0 a los precios base y real del tiquete.

Por otro lado, cuando se trata de un tiquete múltiple entendemos que la gracia de estos tiquetes es que resulten más económicos para el cliente, por lo cuál estos no dependen de las localidades si no de la asignación previa de un organizador o administrador. Para los tiquetes múltiples de único evento, existe un precio fijo para la totalidad de comprar cierta cantidad de tiquetes en cierta localidad. En estos casos, este precio se divide en el número de tiquetes que posee el tiquete múltiple y calcula su precio real sumando su precio de emisión y su cobro por servicio, se espera que esto genere normalmente un precio base más económico a los clientes. Por otro lado, para los tiquetes múltiples con varios eventos sucede algo similar, y es que para comprar cierta cantidad de tiquetes un precio asociado a esta totalidad. Sobre esta totalidad se calcula un nuevo precio base y sobre este se calcula el nuevo precio real.

Ahora, para obtener tanto los estados financieros del organizador como los del administrador, solo es necesario recorrer eventos, las localidades de sus eventos, y el valor de estos tiquetes, ya sea el base, el real o la diferencia entre los dos (corresponde a la ganancia por servicio de la tiquetera). Pues, cuando se creen tiquetes para el organizador del evento, estos quedan registrados como 0 y no se tienen en cuenta, cuando sean tiquetes múltiples, su valor ya no depende de la localidad si no del precio generado por comprar en un paquete, entonces toma un precio real de ganancia, y por último, los tiquetes normales generan su valor de forma convencional, por lo cual al momento de calcular estados financieros, la generación misma de los tiquetes se asegura de que no existan problemas.

Autenticación y operaciones

Todo usuario que interactúe con la plataforma debe autenticarse mediante un login y contraseña válidos. Si las credenciales son incorrectas, el sistema debe impedir el acceso y notificar el error. De igual forma, cualquier operación que no esté permitida para el rol del usuario será rechazada, garantizando que cada actor del sistema solo pueda ejecutar las acciones que le corresponden.

Compra de tiquetes

La compra de tiquetes está sujeta a un límite máximo por transacción. Si un cliente intenta adquirir más tiquetes de los permitidos, la operación se cancela. Además, el sistema valida que el usuario cuente con saldo suficiente para completar la compra; en caso contrario, la transacción no se ejecuta. Estas restricciones aseguran un control sobre la disponibilidad y la capacidad de pago.

Transferencia y uso de tiquetes

Los tiquetes pueden transferirse entre usuarios, pero existen condiciones que lo impiden. No se permite transferir tiquetes que pertenezcan a paquetes Deluxe, ni aquellos que estén vencidos o ya utilizados. Si el tiquete solicitado no existe o no pertenece al usuario que

intenta transferirlo, la operación se rechaza. Estas reglas garantizan la integridad del sistema y evitan fraudes.

Eventos y venues

Cada evento debe estar asociado a un único venue, y un venue no puede albergar más de un evento en la misma fecha. Si el venue solicitado no está disponible, el sistema impide la creación del evento. Además, las localidades tienen una capacidad máxima; si se intenta asignar más tiquetes de los permitidos, la operación se cancela. Estas reglas garantizan la correcta organización de los eventos y la seguridad de los asistentes.

Reembolsos

Los reembolsos solo se permiten en casos específicos: cancelación del evento por parte del administrador o solicitud por calamidad, sujeta a aprobación. Si el reembolso no está autorizado, el sistema lo rechaza. Esta restricción asegura que las devoluciones se realicen bajo condiciones controladas.