Alan Antonio Alvarez Sánchez

Práctica Intermedia

Sistemas de Información

Objetivo

Diseñar e implementar un sistema que administre un cinema. El proceso de diseño e implementación deberá contar con los siguientes puntos:

- Realizar el diseño modular para el sistema.
- Hacer diagramas de casos de uso. Describir solo un caso específico del sistema.
- Crear el modelo relacional para la base de datos, indicar la llave primaria y las llaves foráneas. Incluir las relaciones entre las tablas del diagrama. Deberá de estar normalizado.
- La implementación deberá estar escrita en *MySQL*, para generar el script de la base de datos, y *Python*, para implementar la arquitectura **Model-View-Controller** (MVC) para gestionar los diferentes casos y procesos con los que contará el sistema.

Consideraciones

1. Hacer el diseño de una base de datos que permita comprar un ticket para una película.
2. El sistema es solo para un único cine con muchas salas. Cada sala deberá contener un número distinto de asientos.
3. El sistema deberá controlar las películas, salas, asientos, funciones, boletos y usuarios.
4. Debe haber dos tipos de usuarios: administradores y usuarios generales.
5. El usuario general solo puede ver las funciones para las películas y comprar un boleto.
6. Permitir que el usuario general ver qué películas hay por día y las funciones para una película en específico.
7. El administrador puede manipular la información para las salas, asientos, películas, funciones y administradores.
8. Incluir cualquier otra función y proceso que se considere necesaria para hacer el sistema más robusto o fácil de usar.

Diseño modular del sistema

El sistema deberá cumplir con los módulos lógicos y básicos para funcionar en el ambiente de un cinema. Se deberá entender que no se está diseñando un sistema de administración para un cinema al nivel de gestionar empleados o dulcería. Se está diseñando un sistema cuya finalidad principal sea permitirle al usuario general realizar la compra de un boleto para una función en específico.

Los módulos que conformarán el sistema serán los siguientes:

- Películas: Las películas son una entidad individual que no depende de otras para tener un contexto dentro del sistema. Una película puede tener distintos datos e información relevante, pero por simplicidad y con la finalidad de realizar el desarrollo un poco más rápido, se optó por brindarle los atributos de nombre y descripción de la trama. Cabe aclarar que añadir más campos a este atributo no añade complejidad al sistema.

- Salas: Las salas, al igual que las películas, son entidades independientes. Los atributos de mas relevancia para el sistema son el número de sala, que pude discernir como él mismo id; el número de filas y el número de asientos por fila. Esos dos atributos permiten calcular el número total de asientos para una sala.
- Funciones: Las funciones no son un elemento independiente. Dependen de una película y una sala para poder tener un contexto dentro de nuestro sistema. Además de estas características se debe añadir una fecha y hora y un precio por entrada.
- Asientos: La idea clásica de un asiento se puede definir cómo dependiente de una sala. Y en efecto, está correcto. Si se ve desde el punto de vista físico, una sala está conformada de asientos. Sin embargo, no se está evaluando el asiento desde el punto de vista físico.

  Lo que yo propongo como asiento es al uso del espacio (asiento físico) en un periodo de tiempo en específico (función). Por lo tanto, el asiento no depende de una sala, depende de una función. Por ejemplo, dentro de la Sala 1 del cine, existe un asiento denominado A1. Dicho asiento tiene el atributo de estar vacío u ocupado. Si suponemos que el asiento es dependiente únicamente de la sala. Al momento de cambiar su estado de desocupado a ocupado no posee un valor temporal, es decir este continuará ocupado y no importará qué película se esté proyectando. Ahora, si el mismo asiento A1 depende únicamente de la función que se esté proyectando. Dicho asiento continua existiendo y es libre de estar ocupado o vacío en diferentes funciones.

  El asiento no existe en la base de datos hasta que una función se crea, podemos decir que los lugares reservados son libres de adquirirse hasta que se crean funciones nuevas. Por esa razón se almacena el número de filas y los asientos por filas que contiene una sala.
- Usuario: El usuario es un elemento independiente que puede existir sin ser dependiente de otros módulos. Dentro de sus atributos se encuentra él id, nombre, correo, contraseña y su rol.
- Compras: Las compras, también podemos referirnos a ellas como una colección de tickets, reflejan todas las adquisiciones de boletos que ha generado el usuario. Una compra está conformada por un id, él id del usuario que realizó la compra, fecha y el total. Los boletos no se almacenan aquí porque se genera una relación muchos a muchos entre las compras y los asientos, debido a que en una misma compra se pueden adquirir mas boletos.
- Boletos: Un boleto depende en su totalidad de tres características; id de la compra a la que pertenece, asiento comprado y función.

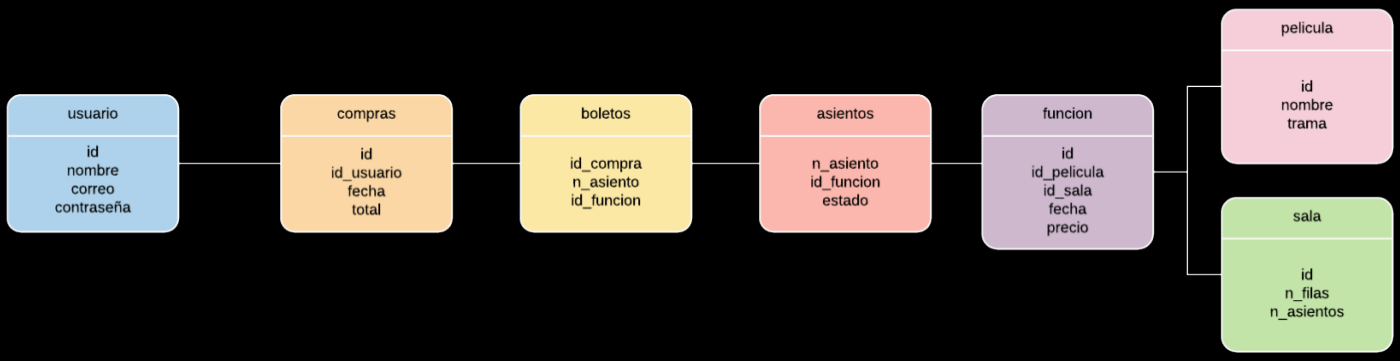


IMAGEN 1: DISEÑO MODULAR DEL SISTEMA

Ambiente
Cinema independiente. Complejo de cine. Salas de cine locales.

Actores
Clientes y personal administrativo.

Entradas
Nuevas entidades en las relaciones.
Editar información de las entidades dentro de las relaciones.
Eliminar entidades.
Solicitar información.
Entrada de texto y línea de comandos.

Salidas
Información del flujo de eventos.
Mensajes.
Información solicitada (Funciones a una hora específica, películas en el cinema, clientes, compras, etc.).
Salida de texto.

Plataforma
Escritorio (Línea de comandos)

Diagrama de casos de uso

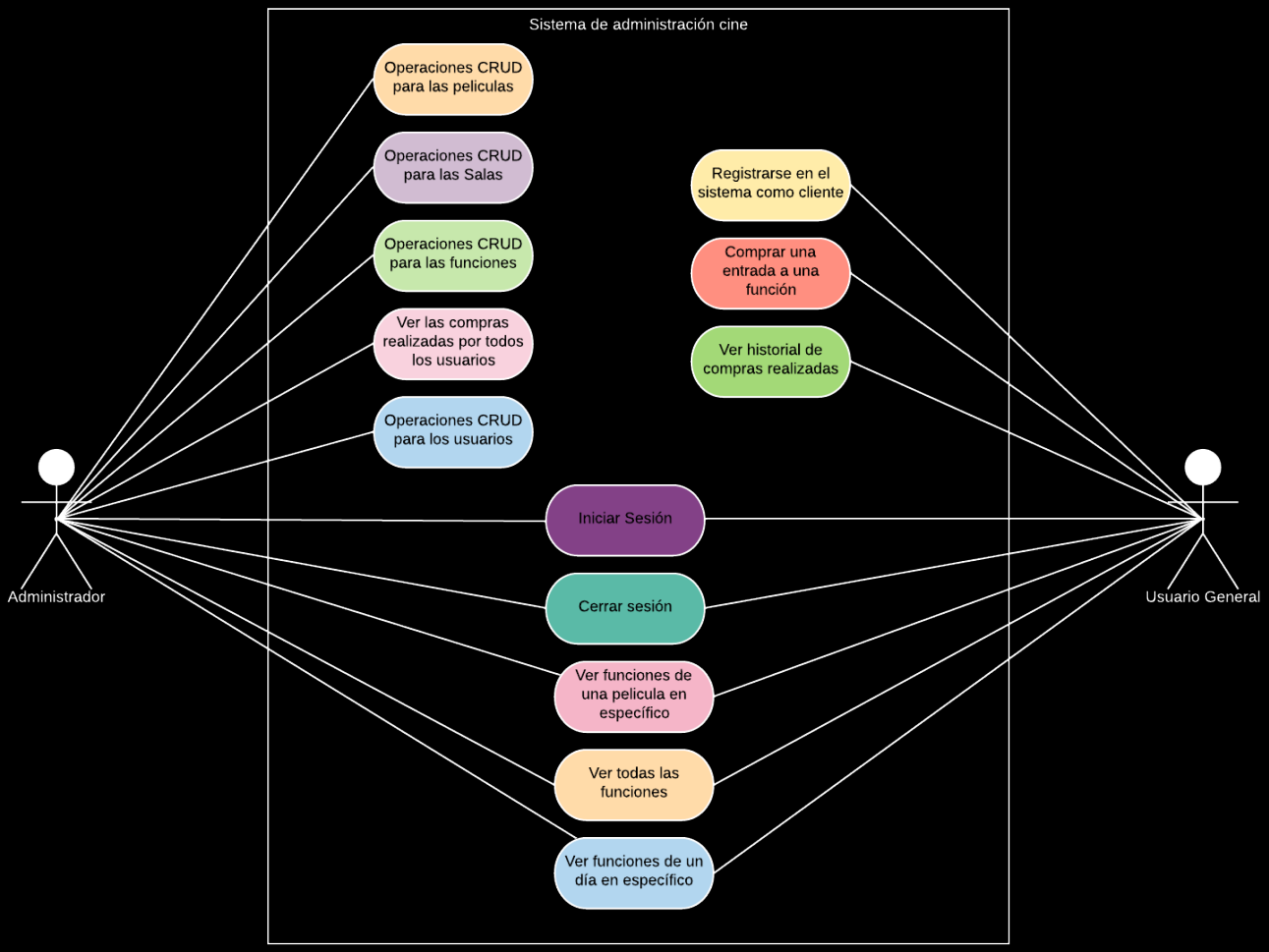


IMAGEN 2: DIAGRAMA DE CASOS DE USO

Descripción de un caso de uso

Nombre

Comprar una entrada a una función

Suposiciones

Salas registradas en el sistema, con información coherente.
Películas registradas en el sistema con información coherente.
Funciones registradas en el sistema con éxito.
Asientos generados de manera exitosa para cada función.

Id de usuario registrado en el sistema.
Sesión Iniciada por un usuario general.

Pre-Condiciones

Se ha desplegado el menú de opciones al usuario.

Inicio

El usuario ingresa la opción del menú válida para iniciar el proceso de compra.

Flujo de eventos

Sistema:
- Crea una nueva compra.
- Inicializa el total de la compra en 0.00
- Salida - Despliega menú: “1.- Añade más boletos. 2.- Terminar compra”
- En espera de la entrada de usuario.

Usuario:
- Entrada - “1”

Sistema:
- Salida - “Ingresa el ID de la función a la que quieres comprar una entrada”
- En espera de la entrada del usuario.

Usuario:
- Entrada - “1” (Cualquier número)

Sistema:
- Comprueba si él ID existe dentro de la base de datos.
- Consulta la lista de asientos para el ID de función.
- Salida - Despliega lista de asientos disponibles y ocupados para la función .
- Salida - “Ingresa el número de asiento que deseas comprar”.
- En espera de la entrada del usuario.

Usuario:
- Entrada - “A1”.

Sistema:
- Comprueba si el número de asiento es válido.
- Comprueba si el asiento está disponible.
- Actualiza el estado del asiento de disponible a ocupado.
- Actualiza con el costo del asiento al total de la compra.
- Crea el registro del boleto en la base de datos (compra del asiento).
- Salida - “Boleto añadido con éxito”.
- Salida - Despliega menú: “1.- Añade más boletos. 2.- Terminar compra”.
- En espera de la entrada del usuario.

Usuario:
- Entrada - “2”

Sistema:
- Termina el registro de la compra en la base de datos.
- Salida - Resumen de la compra

Post-Condiciones

Lista de asientos actualizada.
Registro de compras actualizado.
Base de datos actualizada.

Diagrama Relacional

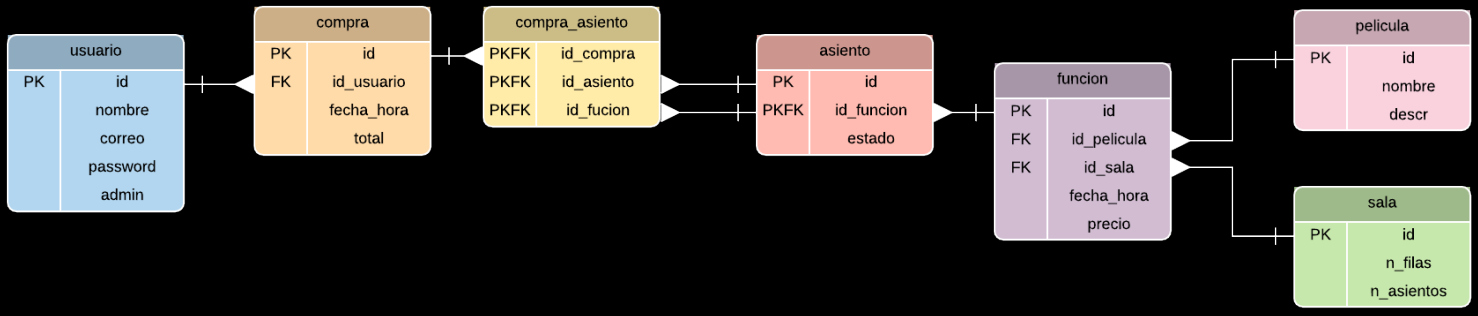


IMAGEN 3: DIAGRAMA RELACIONAL

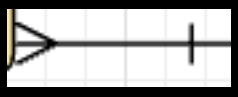
Hay siete relaciones en total:

1. Usuario
2. Compra
3. Compra_asiento (Cumple la función de Boleto)
4. Asiento
5. Función
6. Película
7. Sala

Se realizó una normalización 3N para la base de datos. Ejemplo, la relación compra_asiento soluciona una relación muchos a muchos entre las relaciones compra y asiento.

En el diagrama se indica qué atributo es la llave primaria con las siglas “PK”, llave foránea con las siglas “FK” y si un atributo es llave primaria y llave foránea al mismo tiempo, con las siglas “PKFK”.

De igual manera la unión entre tablas se indica con el siguiente objeto:



Donde la parte angosta ejemplifica “de uno” y la parte ancha “a muchos”. Se puede apreciar así el flujo de las tablas. Por ejemplo, hay una relación de uno a muchos entre usuario y compra.

Conclusiones

Es una parte fundamental realizar el diseño de un sistema, conocer los requerimientos y crear historias de usuario. Es una guía que permite plantear cómo resolver un problema antes de programarlo, haciendo más fácil su desarrollo en código.
Un buen diseño inicial de un sistema de software prevé problemas que de otra manera se captarían a mitad de la codificación, y significarían replantear situaciones. Representan una perdida de tiempo.
Igual de importante es un buen diseño de base de datos. Tener en cuenta las entidades que participarán en tu sistema y cómo interactúan entre sí, es fundamental para que el sistema tenga éxito. Sin un buen diseño de base de datos podemos caer en redundancia o en una implementación disfuncional.

La implementación en Python fue realizada con éxito. Se logró concretar cada módulo que se planteó. Y añadir las características de **login** y **signin**, que no estaban planteadas pero desde mi punto de vista eran necesarias.

En general el modelo MVC me resulta fácil de entender y de implementar, sin embargo no me termina de convencer del todo. Cuando estaba codificando me resultaba muy tedioso tener el Controlador separado de la Vista. Al menos esa impresión me dio al desarrollar esta app en ventana de comandos.

Fue una buena práctica que ejemplifica bien el diseño e implementación de un sistema.