Ingeniería en Sistemas Computacionales

Sistemas de información

Sistema de venta de boletos de cine

Asesor

Juan Carlos Gómez

Alumno

Manuel Pedro Matadamas Marín

INDICE

Introducción

Esta práctica consiste en desarrollar un sistema del tipo MVC que permita la compra de boletos para ver una película, está basado en el funcionamiento de un cine real (Cinepolis, Cinemex, etc) en donde permitirá al usuario interactuar con del de forma independiente o a través de un administrador.

MVC o Modelo-Vista-Controlador es un patrón de arquitectura de software que, utilizando 3 componentes (Vistas, Modelos y Controladores) separa la lógica de la aplicación de la lógica de la vista en una aplicación. Es una arquitectura importante puesto que se utiliza tanto en componentes gráficos básicos hasta sistemas empresariales; la mayoría de los frameworks modernos utilizan MVC (o alguna adaptación del MVC) para la arquitectura, entre ellos podemos mencionar a Python, Django, AngularJS y muchos otros más.

MODELO

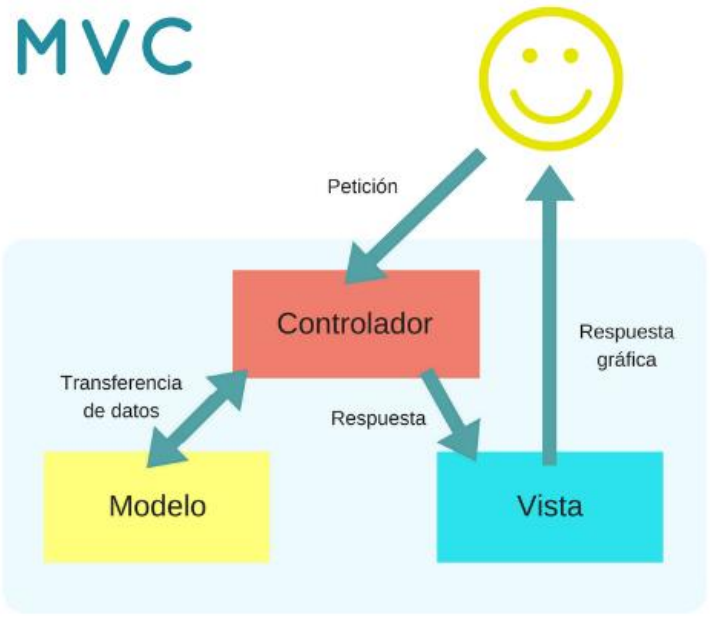
Se encarga de los datos, generalmente (pero no obligatoriamente) consultando la base de datos. Actualizaciones, consultas, búsquedas, etc. todo eso va aquí, en el modelo.

CONTROLADOR

Se encarga de... controlar, recibe las órdenes del usuario y se encarga de solicitar los datos al modelo y de comunicárselos a la vista.

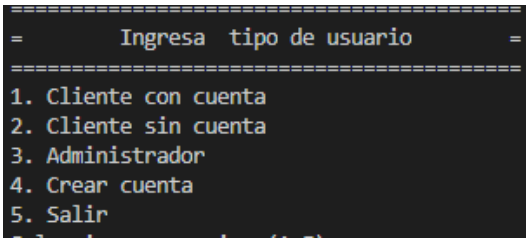
VISTAS

Son la representación visual de los datos, todo lo que tenga que ver con la interfaz gráfica va aquí. Ni el modelo ni el controlador se preocupan de cómo se verán los datos, esa responsabilidad es únicamente de la vista.



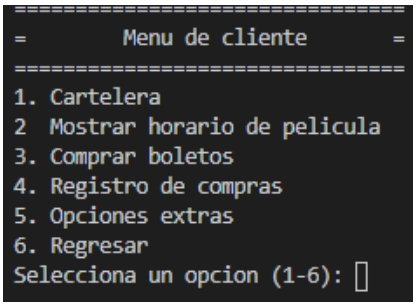
Desarrollo

Para la creación de administradores es muy importante saber que para crear un usuario del tipo administrador es necesario ingresar la clave: UserGeneral. Para la creación de clientes no es necesario ingresar una clave de administrador



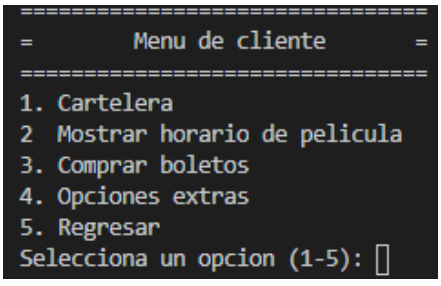
Pantalla de inicio de sistema

La forma en que los clientes podrán realizar sus compras será de dos maneras posibles, la primera será por medio de una cuenta de usuario en donde el iniciara sesión y todas las compras que el realice quedaran almacenadas en el sistema y además de contar con la ventaja de tener un control de sus boletos ya que existe la posibilidad de reimpresión en el caso del que usuario los haya extraviado.



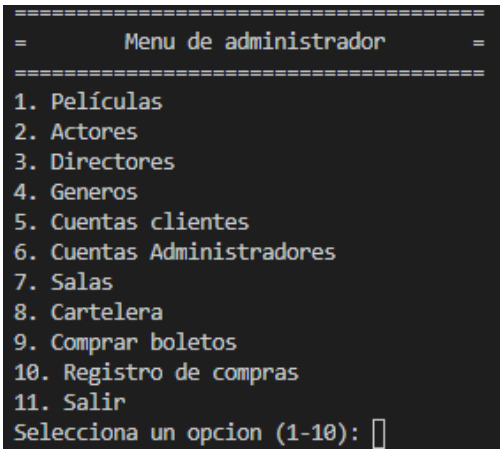
menú de usuario con cuenta

El cliente sin cuenta podrá realizar básicamente las mismas acciones que un cliente con cuenta, ver la cartelera, ver el horario de una película en específico, buscar la sinopsis de una película, etc. La diferencia que existirá con los usuarios registrados será el control de compras ya que este al no contar con una cuenta no será posible tener un registro de sus movimientos realizados.



menú de usuario sin cuenta

El administrador es el que se encargara de realizar todos los procesos necesarios para el correcto funcionamiento del sistema realizando actualizaciones y todas las operaciones CRUD necesarias para su adecuado manejo.



menú de usuario administrador

Dentro del modelo se realizarán todas las consultas necesarias para realizar los procesos adecuadamente en donde se solicitará toda la información de una tabla, una parte, o la unión de la información de dos o mas tablas para ofrecer los registros correspondientes a la función.

```
def read_a_peliculagen(self,gp_id_pelicula):
    try:
        sql = 'SELECT genero.id_genero, genero.genero FROM genpelicula JOIN genero ON genero.id_genero = genpelicula.gp_id_genero WHERE gp_id_pelicula = %s'
        vals = (gp_id_pelicula,)
        self.cursor.execute(sql, vals)
        record = self.cursor.fetchall()
        return record
    except connector.Error as err:
        return err
```

Realización de consulta de los géneros que contiene la tabla géneros películas en donde se almacenan todos los géneros de una película

También se recibirá la información que ingresara el usuario y será insertada en su tabla correspondientes siempre y cuando haya pasado los controles del controlador ya que no es posible solo insertar la información directamente como la ingreso el usuario en algunos casos, puede ser que necesite ser transformada en otro tipo de dato u ordenada adecuadamente.

```
def create_gen(self,genero):
    try:
        print(genero)
        sql = 'INSERT INTO genero (`genero`) VALUES (%s)'
        vals = (genero,)
        self.cursor.execute(sql,vals)
        self.cnx.commit()
        return True
    except connector.Error as err:
        print(err)
        self.cnx.rollback()
        return err
```

Agregar los géneros a una tabla de la base de datos encargada de almacenarlos

La actualización de datos en apariencia la actualización de datos será de las opciones más sencillas que contendrá, pero si no es manejada bien la información que requiere puede fallar el sistema y será un poco tedioso encontrar la falla.

```
def update_genpelicula(self, fields, vals):
    try:
        sql = 'UPDATE genpelicula SET '+','.join(fields)+'WHERE gp_id_genero = %s AND gp_id_pelicula = %s'
        self.cursor.execute(sql,vals)
        self.cnx.commit()
        return True
    except connector.Error as err:
        self.cnx.rollback()
        return err
```

Actualización de la tabla de géneros de las películas.

Por último, tenemos el borrado de contenido de las tablas en donde se deben de asegurar bien las condiciones de borrado ya que puede ocasionar errores en el funcionamiento desde que no se borre o que termines borrando todo el contenido de esa tabla.

```
def delete_client(self, id_cliente):
    try:
        sql = 'DELETE FROM cliente WHERE WHERE correo = %s'
        vals = (id_cliente,)
        self.cursor.execute(sql,vals)
        self.cnx.commit()
        count = self.cursor.rowcount
        return count
    except connector.Error as err:
        self.cnx.rollback()
        return err
```

Borrado de contenido de la tabla clientes.

Hasta aquí se vio un poco de funcionamiento del modelo ahora veremos un poco del funcionamiento de otra sección del MVC

La parte de vista se encargará de entablar la comunicación con el usuario ya que mostrara todos los mensajes en pantalla desde un cuestionario de inicio de sesión hasta una tabla de contenido que recibirá todos los valores que el controlador de entregue.

```
def user_menu(self):
    print('=========================================')
    print('=        Ingresa  tipo de usuario       =')
    print('=========================================')
    print('1. Cliente con cuenta')
    print('2. Cliente sin cuenta')
    print('3. Administrador')
    print('4. Crear cuenta')
    print('5. Salir')
```

menú de inicio de programa.

```
def ok(self, id, op):
    print('+'*(len(str(id))+len(op)+24))
    print('+'+str(id)+' se '+op+' correctamente! +')
    print('+'*(len(str(id))+len(op)+24))

def error(self, err):
    print(' Error! '.center(len(err)+4,'-'))
    print('-'+err+'-')
    print('-'*(len(err)+4))
```

manejo de mensajes de error o notificación de aceptación que el usuario recibirá

```
def show_a_Registro(self, record):
    print(f'{record[4]:<4}|{record[2]:<20}|{record[5]}  |{record[1]:<8}|{record[6]}')

def show_Registro_header(self, header):
    print(header.center(68,'*'))
    print('Sala'.ljust(3)+'|'+'Pelicula'.ljust(20)+'|'+'Hora'.ljust(10)+'|'+'Asiento'.ljust(8)+'|'+'Fecha'.ljust(6))
    print('-'*68)

def show_Registro_midder(self):
    print('-'*68)

def show_Registro_footer(self):
    print('-'*68)
```

Manejo de contenido para mostrar registro de compra de boletos.

```
********** Todos las compras **********
Sala|Pelicula           |Hora      |Asiento |Fecha
--------------------------------------------------------------------
1   |Cloud Atlas        |13:30:00  |G9      |2020-05-21
1   |Cloud Atlas        |13:30:00  |G10     |2020-05-21
1   |Cloud Atlas        |13:30:00  |G11     |2020-05-21
1   |Cloud Atlas        |13:30:00  |G12     |2020-05-21
1   |Cloud Atlas        |15:30:00  |E9      |2020-05-21
1   |Cloud Atlas        |15:30:00  |E10     |2020-05-21
1   |Cloud Atlas        |17:30:00  |E10     |2020-05-21
1   |Cloud Atlas        |17:30:00  |E11     |2020-05-21
1   |Cloud Atlas        |13:30:00  |D8      |2020-05-22
1   |Cloud Atlas        |13:30:00  |D9      |2020-05-22
2   |Batman             |17:30:00  |D10     |2020-05-22
2   |Batman             |17:30:00  |D11     |2020-05-22
--------------------------------------------------------------------
```

Registro de compra de boletos de un usuario

```
********** Cartelera **********
Sala|Pelicula           |Hoario
--------------------------------------------------------------------
1   |Cloud Atlas        |13:30:00|15:30:00|17:30:00
2   |Batman             |15:30:00|17:30:00|18:30:00
3   |Avatar             |15:30:00|17:30:00|18:30:00
--------------------------------------------------------------------
--------------------------------------------------------------------
```

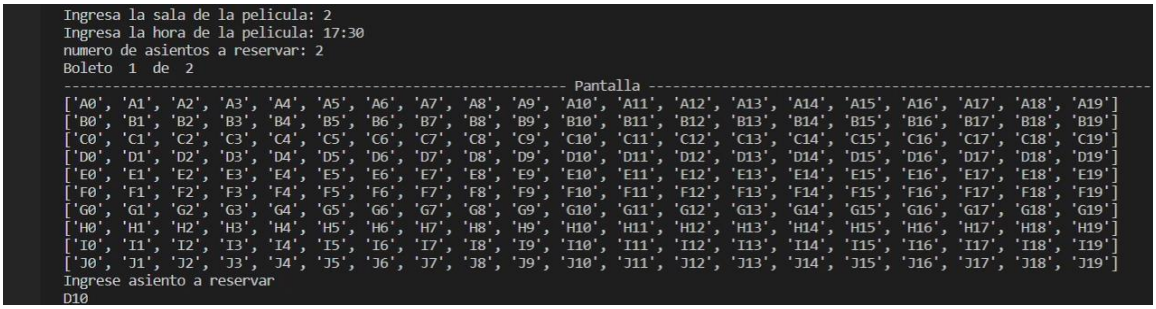
Muestra de cartelera en pantalla

La parte del controlador es la que se encargara enviar y recibir información tanto de la vista como del modelo, este la procesara, ordenara y decidirá como proceder dependiendo el análisis de los datos que obtenga, puede ser que muestre mensajes en pantalla con lo solicitado o muestre mensajes de error dependiendo lo que le sea suministrado tanto por el usuario, así como por la base de datos.

Los errores mas comunes son de sintaxis en donde el usuario puede ingresar erróneamente algún dato y es necesario que le controlador lo compruebe para informar que no se puede trabajar con ese dato o si es necesario volver a iniciar el proceso.

A continuación, se mostrará la parte que considero mas importante dentro del sistema que es la compra de boletos de una sala de cine y constara de los siguientes pasos:

1. primero se procederá a solicitar al cliente si este tiene una cuenta
2. en caso de que ingresara una cuenta y no exista se le notificara
3. se solicitará una fecha para reservar la función
4. se mostrará en pantalla una vista de la cartelera de la fecha solicitada
5. en caso de no existir se mostrará un mensaje en pantalla y regresará al menú anterior
6. se le solicitara al usuario ingrese la sala, hora y número de asientos a reservar
7. si los datos son ingresados vacíos mostrará un menaje de error y volverá al menú anterior
8. se obtendrá el nombre de la película que el usuario desea ver
9. en caso de que no exista una película en el horario deseado despliega un mensaje de error
10. se obtendrá una vista de todos los asientos de la sala
11. en caso de que no exista una sala nos desplegara una alerta
12. obtendremos todos los boletos que están reservados en la sala
13. se agregará a la vista de la sala todos los boletos que ya han sido reservados
14. se solicitará al usuario ingresar el numero del asiento que desea reservar repitiendo el ciclo hasta cumplir con el numero de asientos que ingreso previamente
15. muestra en pantalla el número de reservación que se está haciendo
16. muestra los asientos de la sala ocupados y disponibles
17. solicitará el asiento a reservar
18. comprueba que el numero de asiento ingresado sea correcto y no falten o sobren caracteres
19. transforma el número de asiento en dos números para ingresarlos a la matriz que controla los asientos de la sala
20. comprueba que los asientos ingresados estén dentro del rango de asientos de la sala
21. una ves comprobado que este en el rango y no este reservado se crea la reservación
22. se inserta el numero de asiento reservado dentro de una tabla de control
23. se llama a una función que mostrara todos los boletos que fueron comprados en ese momento

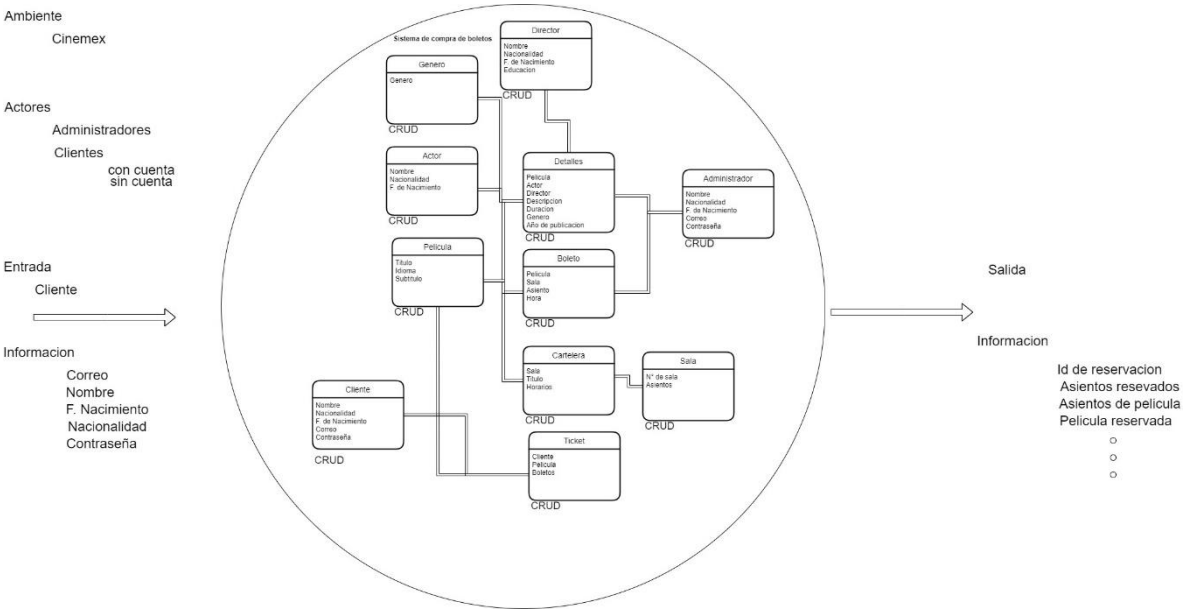


Interfaz con la que el usuario trabajara

```
def comprar_boleto(self):
    usuario = self.aks_usuario() # preguntamos al usuario si tiene cuenta
    if usuario == 0: # si el usuario ingreso una cuenta inexistente el programa regresara al menu anterior
        return
    fecha = self.ask_boleto_fechas()# preguntar fecha de reservacion
    car = self.read_carteleras(fecha,1) # llamada a vista de carteleras
    if car == False: #en caso de solicitar la vista de una cartelera que no existe el programa retornara al menu anterior
        return
    sala,hora,numero = self.ask_boleto() #peticion de informacion como sala de la funcion , hora de la funcion y el numero de asientos a reservar

    if sala == '' or numero == '' or hora == '': # si los datos son ingresados vacios mostrara un menaje de error y volvera al menu anterior
        self.view.error('Error no al ingresar la reserva')
        return

    pelicula = self.model.read_cartelera(sala,hora,fecha) # en esta seccion de obtendra la pelicula que el usuario desea ver
    if( pelicula == None): # en caso de que no exista una pelicula en el horario deseado despliega un mensaje de error
        self.view.error('No existe pelicula en ese horario intenta de nuevo')
        self.comprar_boleto() # nos solicitara una ves mas todos los datos

    asientos = self.vie_of_sala(sala) # aqui optendremos la vista de la sala en la que veremos la funcion
    # mostrara las numeraciones de los asientos A1 A2 .... A30

    if asientos == False: # en caso de que no exista una sala nos desplegara una alerte
        self.view.error('No existe la sala intenta de nuevo')
        self.comprar_boleto() # nos solicitara una ves mas todos los datos

    comprados=[] # almanecara todos los numeros de asientos que compre el usuario

    reserv = self.model.read_all_boletos(sala,hora,fecha)  # obtendremos todos los boletos que estan reservados en la sala
    if type(reserv) != list:
        self.view.error('Error en la consulta')
        return


    # esta seccion agregara a la vista de la sala todos los boletos que ya han sido reservados
    for j in range(len(reserv)):
        if(len(reserv[j][0])) >2:
            num=[reserv[j][0][1],reserv[j][0][2]]
            StrA = "".join(num)
            StrA = int(StrA)
            asientos[int(ord(reserv[j][0][0]))-65][StrA] =' X '
        else:
            # marcara con una X el asiento reservado
            asientos[int(ord(reserv[j][0][0]))-65][int(reserv[j][0][1])] =' X '

    contador =0 # llevara el control de cuntos boletos han sido comprados
contador =0 # llevara el control de cuntos boletos han sido comprados

while contador != int(numero): # repetira la operacion de la reservacion de asientos hasta que el controlador lo detenga
    self.view.contador_boleto(contador+1,numero) #muestra en pantalla el numero de reservacion que se esta haciendo
    self.view.show_asientos(asientos) # muetra los asientos de la sala
    self.view.numero_asientos()  # solicitara el asiento a reservar

    bande = 1
    while bande == 1:   #comprueba que el asiento no sea mayo a 3 digitos por que no existe o menor a 1 por que no existe por ejemplo A, A222
        reserva = input()  # introduce el numero del asiento el usuario
        if reserva == '0':  # de esta manera en caso de que el usuario ingrese 0 se cancelara la compra de boleto
            return
        if len(reserva) <= 1: # comprueba que el asiento consita minimo en dos caracterer para ser un asiento valido
            self.view.asiento_no_valido()
        if len(reserva) >= 2 and len(reserva) <= 3:
            bande = 2
    StrA = 0

    if len(reserva) > 2: # conviere en un numero el asiento ingresado
        num=[reserva[1],reserva[2]]
        StrA = "".join(num)
        StrA = int(StrA)
    else:
        StrA = int(reserva[1])

    asientos_existentes = self.model.read_sala(sala) #recibe el numero de asiento de una sala filas y columnas

    # comprueba que el asiento ingresado este dentro del rango de los asientos de la sala
    if int(ord(reserva[0]))-65 > asientos_existentes[1] or  StrA > asientos_existentes[2]:
        self.view.error(' No existe ese asiento intenta de nuevo ')
        self.comprar_boleto()
    # comprueba que el asiento de la sala no este reservado en caso de no estarlo este sera serevado
    if asientos[int(ord(reserva[0]))-65][StrA] != ' X ':
        asientos[int(ord(reserva[0]))-65][StrA] =' X '
        contador +=1
        self.model.create_asiento(reserva,sala,hora,fecha)
        self.model.create_boleto(reserva,pelicula[0],usuario,sala,hora,fecha)
        # se insertara el boleto en el registro de compra para el control del usuario y de la sala
        comprados.append(reserva)
    else:
        self.view.error_asiento()

#se encarga de la impresion de boletos echos en el momento de la compra
for k in range(len(comprados)):
    self.boleto_impreso(sala,pelicula[0],hora,comprados[k],fecha)
return
```

Código encargado de la reservación de boletos

Diagrama Modular



El diagrama modular muestra el ambiente en el que se trabajara en este caso Cinemex así como los actores que interactúan con ella, la información que será ingresada en este caso por un cliente con cuenta y la salida de información que el cliente con cuenta obtendrá.

Diagrama de casos de uso

Nombre: Compra de boleto

Suposiciones:
+ El cliente tiene cuenta
+ Inicio sesion
+ Tecleado la opcion de comprar boleto

Precondiciones:
~ La pelicula no haya comenzado
~ Asientos disponibles

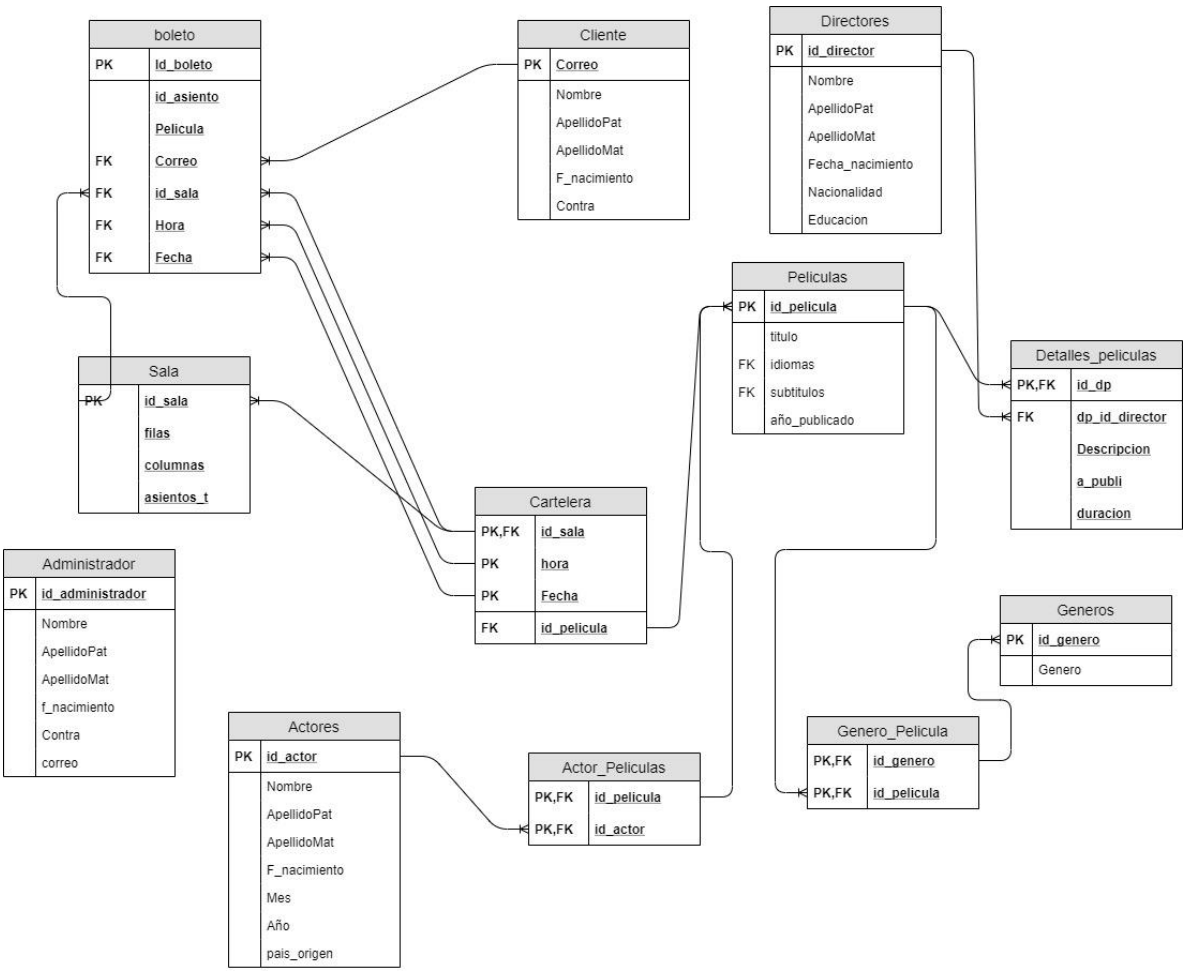
Inicio:
+ Teclear opcion de compra de boleto

Flujo de eventos:
s Solicita al usuario ingrese su correo electronico
s solicita al usuario ingrese la fecha de la reservacion
u ingresa la fecha en caso de ser otro dia de lo contrario solo dara enter
s Despliega en pantalla la tabla de cartelera
s Solicita al usuario ingrese el id de la pelicula que desea ver
u Teclea el id de la pelicula que desea ver
s Solicita el horario en el que vera la pelicula
u Ingresa tecleando el horario de la pelicula
s Solicita el numero de boletos que comprara
u Ingresa el numero de boletos deseados
s Despliega en pantalla los asientos disponibles
s Solicita que ingrese los asientos
u Ingresa los asientos requeridos
s Muestra en pantalla los boletos adquridos
s actualiza la base de datos de cine con nuevos valores
s Muestra el menu de usuario
u cierrra el programa

Poscondicion:
+ La BD cambio/actualizo

Se representa la acción de la compra de un boleto de un usuario con cuenta con el flujo de eventos que ocurrirán al momento de realizar la compra y el resultado que generara en el sistema

Diagrama Relacional



El sistema consistirá en 12 tablas que llevaran el control de toda la información necesaria para el funcionamiento adecuado del sistema de venta de boletos de cine

La tabla cartelera la considero principal ya que con ella se accederá al control de las demás tablas teles como el almacenamiento de un boleto ya que este contiene la película, horario y sala de la función que actuaran como llaves primarias dentro de ella

La tabla boleto tiene doble propósito en ella podremos registrar todos los asientos que fueron vendidos para al momento de querer visualizar una sala con ella serán marcados los asientos con una ‘x’ y el usuario no lo podrá elegir, el segundo propósito es llevar un registro de compras de un cliente ya que podremos extraer todas las películas que un cliente adquirió sus entradas.

Para la sala primero se creo un ID de sala y un numero de asientos dentro de ella, pero de esta manera no se podría visualizar la sala como lo hace la forma mas adecuada fue el solicitar que se ingresaran filas y columnas de una sala para de esta manera generar una matriz que nos facilitaría el control de la venta de boletos.

La tabla administradora podrá realizar las funciones de CRUD en todas las tablas y todas las funciones que tiene un cliente sin cuenta, pero esta no contara con un registro de compras por lo cual no fue relacionada con ninguna de las otras tablas ya que un administrador se puede registrar también como cliente y generaría un conflicto en la selección de compras al elegir ver todas sus compras así que solo la sección de cliente con cuenta contara con este registro.

Normalización

1NF

Las tablas cliente, administrador, actor y director la columna de nombre fue dividida en nombre y apellidos para hacer valores atómicos.

Se decidió dejar la columna de fecha de nacimiento como un valor DATE por lo que no fue necesario hacerlo atómico.

2NF

No fue necesario realizar esta normalización ya que lo cumplía previamente

3NF

No fue necesario realizar esta normalización ya que lo cumplía previamente

Conclusión

Previamente a la elaboración de esta practica tenía conocimientos de manejo de las bases de datos sin embargo fueron complementados de gran manera al usar algunas funciones extras como la unión de tablas para mostrar información basada en los id de dos tablas que tuvieran en común una sola así mismo como implementarlas dentro de funciones (model) ya que se vieron nuevos métodos o manejo como estaba acostumbrado tal ejemplo es la sección de update ya que con la realización de actualización de datos en controlador se vio otra manera de manejarlo.

Al momento de realizar la practica surgieron cambios importantes dentro de la base de datos debido a que se tenían contempladas de una manera previamente diseñada por medio de los diagramas, pero al momento de implementarla se fue adaptando a las funciones que fueron necesarias y se logro un mejor planteamiento de ellas eliminando tablas que contenían información similar y agrupadas en una tabla que englobaba todos estos valores necesarios cumpliendo la normalización.

El desarrollo del controlado fue un poco desafiante al inicio ya que requiere manejar la información de cierta manera ya que en diversas ocasiones de debieron hacer varias comprobaciones por que si el usuario ingresaba un dato que no correspondía este tronaba el programa teniendo que reiniciar y en base a prueba y error se fueron resolviendo estos problemas mostrando mensajes de error y regresando a pasos previos para que el usuario fuera capaz de corregir esta información y continuar con el proceso en el que estaba trabajando.