UNIVERSIDAD DE GUANAJUATO

DIVISIÓN DE INGENIERÍAS CAMPUS IRAPUATO-SALAMANCA

DICIS

SISTEMAS DE INFORMACIÓN

Práctica Individual

Semestre Enero 2020 – Junio 2020

Profesor: Juan Carlos Gomes Carranza

Alumna: Frida Martínez Flores | NUA: 146015

29/05/2020

Introducción

El uso de sistemas de información es en nuestros días algo indispensable para cualquier organización o empresa, ya que automatizan procesos y almacenan valiosa información de modo que la organización puede acelerar sus ventas mientras permite que su usuario tenga un acceso practico y amigable.

Los sistemas de información de transacciones son unos de los más básicos e importantes en cualquier organización, estos entre otras cosas, son los encargados de llevar a cabo los procesos cotidianos incluyendo las transacciones económicas.

El propósito de esta practica individual es simular un sistema de transacciones que representa un cine, específicamente se debe cumplir con la realización de la compra de un boleto para alguna función por un cliente.

Como se menciono antes, esta practica es individual por lo que cuenta con detalles particulares en su sistema, a continuación, se especifican.

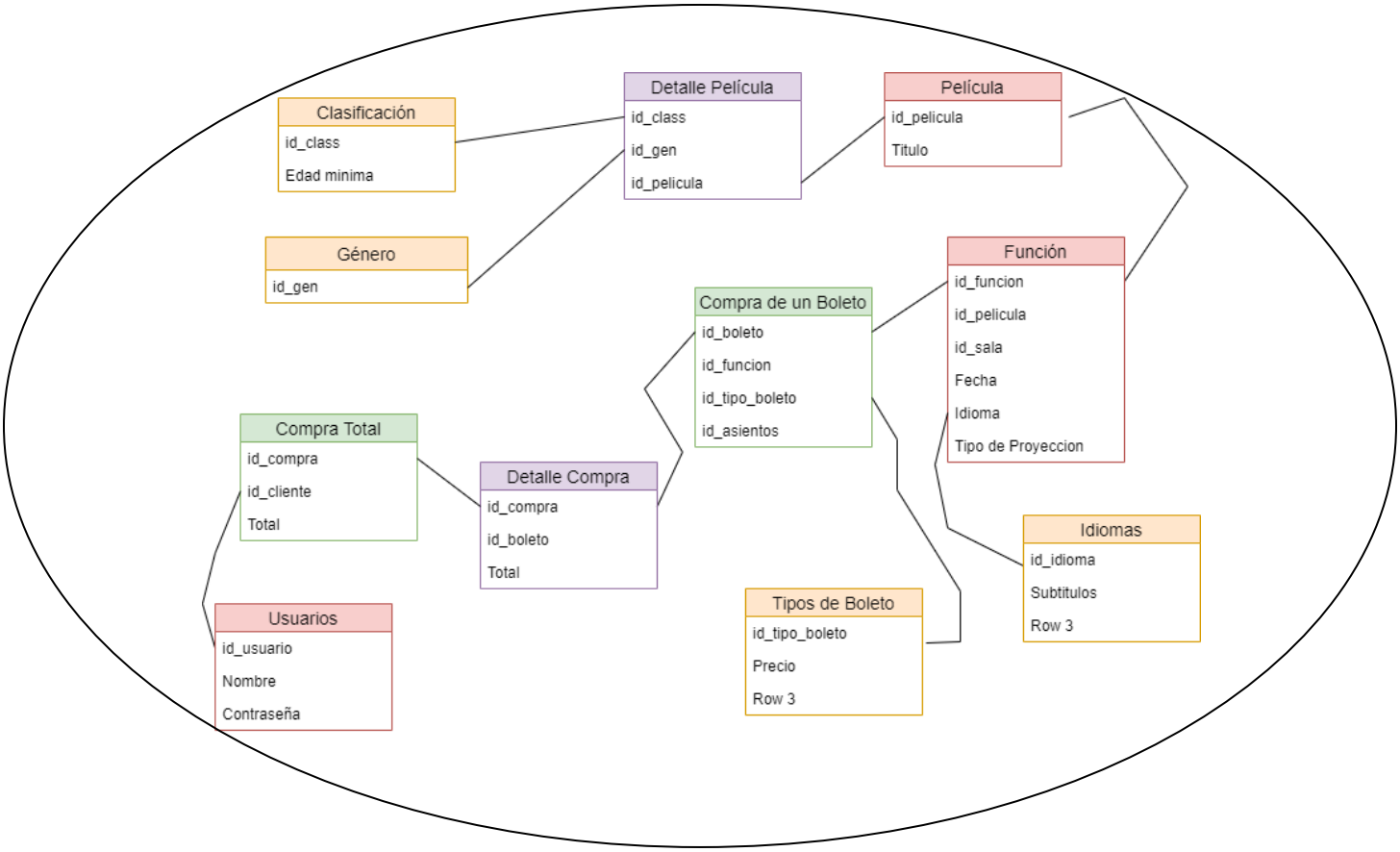
Primeramente, el sistema se construye y se administra por un usuario de tipo “Administrador”, solo un administrador puede crear otro y otorgarle responsabilidades del manejo de la información. El sistema puede ser usado obviamente por el admin pero también por usuarios de tipo “Clientes” que son el objetivo, el ingreso monetario a la empresa.

Este sistema no pide que el usuario cliente pague digitalmente su boleto de entrada, simplemente, guarda su entrada, es decir, su boleto con función y lugar deseado en la sala. Se asume que el boleto será pagado físicamente en la taquilla del cine y hasta ese momento el cliente recibirá sus boletos. El cliente podrá intercambiar su pago por el boleto mínimo 10 minutos antes de que comience su función, de lo contrario, su boleto será anulado y su lugar de asiento será puesto en venta nuevamente.

Teniendo esto en mente a continuación se desarrollan los diagramas para representar el modelo se compra de un boleto de cine y su implementación.

Metodología

El sistema que se desarrolló se basó en un diagrama modular y a su vez este representa sus métodos particulares en un diagrama de casos de uso.



Actores:
Administrador del sistema
Clientes, usuarios que desean comprar un boleto de cine.

El diagrama modular explica las relaciones que existen entre todos los elementos del sistema. Para comprar un boleto de cine debe existir un cine con funciones, cada una de estas funciones esta compuesta por un identificador de esa función, la película que se presentara, la sala donde se proyectara y su tipo de proyección, el idioma en el que se presentara la película y por su puesto la fecha y hora de función.

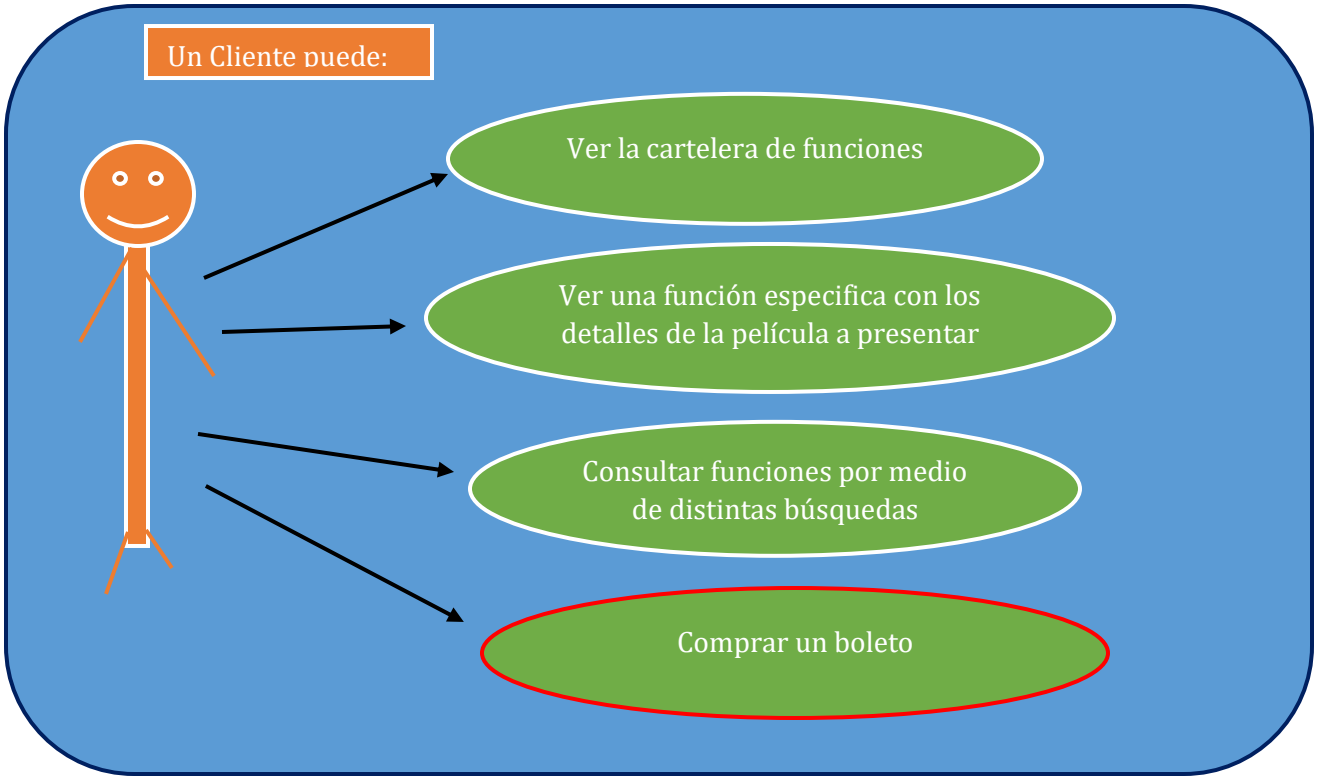
Cada película tiene como detalle el género de esta película y su clasificación los cuales componen el detalle de la película.

El sistema es usado por usuarios, los usuarios de tipo Administrador tendrán relación con cada elemento del sistema ya que el de este fue diseñado y desarrollado por el administrador principal. Los usuarios de tipo cliente solo tendrán acceso a los métodos relacionados con la compra de su boleto y sus especificaciones.

Cada compra será única y almacenará el identificador del cliente que la realizo. Cada compra incluirá un detalle de compra con el número de boletos que se desea comprar, el identificador de la función a la que se pretende entrar y el tipo de boleto que se pide.

El tipo de boleto dependerá de los futuros dueños de los boletos, es decir, su estado (adulto, tercera edad, niño, estudiante).
Cada vez que se cree una compra también se creara un detalle de la compra con el tipo de boleto, función deseada y lugar deseado en la sala.

Para explicar de manera más detallada como el sistema realizara la compra de un boleto de cine se demuestra en el siguiente diagrama de casos de uso de un Cliente.



Caso: ***Comprar un boleto***

Asunción: Haber visto la cartelera de películas y funciones. Saber para qué función desea comprar los boletos.

Precondición: Tener una cuenta de usuario en el sistema. Haber iniciado sesión.

Iniciación: Elegir la opción para comenzar la compra.

Flujo de Eventos:

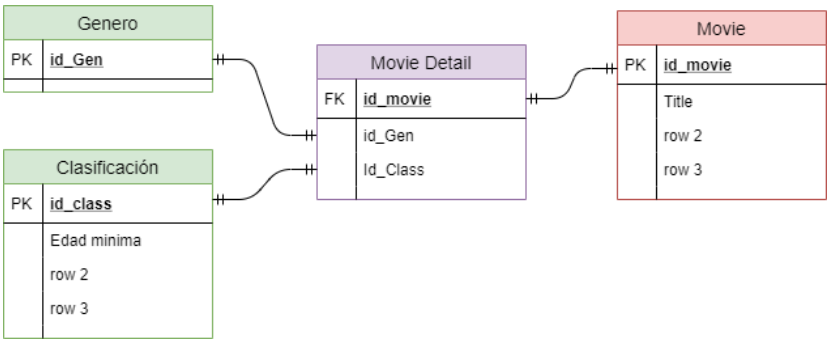
1. Mensaje en pantalla: “Agregar boletos a la compra”
2. El sistema crea un registro de compras realizado por el cliente activo y su respectivo detalle de compra con su total en blanco (se actualiza la base de datos)..
3. Mensaje en pantalla: “Elegir función”
4. Usuario introduce numero de función que desea ver
5. Se almacena la función deseada en la compra del boleto (se actualiza la base de datos).
6. Mensaje en pantalla: “Introducir cantidad de boletos a comprar”
7. Usuario introduce cantidad de boletos que desea comprar
8. Ciclo para elegir tipo de boletos y asientos:
    - ✓ Mensaje en pantalla: “Introducir tipo de boleto”.
    - ✓ Usuario introduce el tipo de usuario para ese boleto (Adulto, niño, preferencial).
    - ✓ Sistema almacena el precio del boleto individual en la compra del boleto (se actualiza la base de datos).
    - ✓ Mensaje en pantalla: “Fila de la sala deseada para asiento”
    - ✓ Usuario introduce fila de la sala deseada
    - ✓ Sistema genera el registro de una compra de boleto (se actualiza la base de datos).
    - ✓ Sistema relaciona el id del boleto(s) con el detalle de la compra.
9. El sistema suma el precio de todos los boletos individuales y registra el total en la compra (se actualiza la base de datos).

Post Condición: Mensaje en pantalla: “Compra realizada correctamente”.

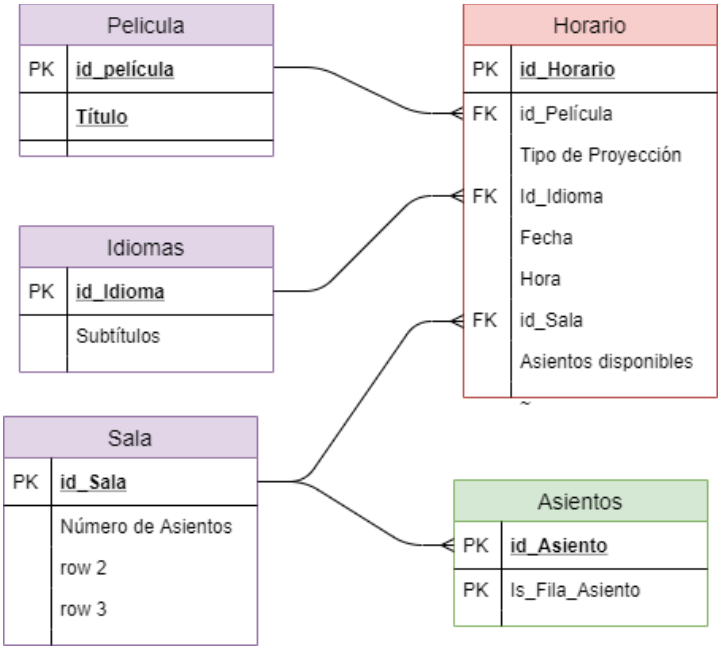
Resultados

El sistema se desarrolló básicamente en dos partes: la estructura del sistema que construyo siguiendo un modelo MVC: Model View Controller usando Python como lenguaje de programación y la base de datos donde se almaceno la información generada y solicitada usando SQL como lenguaje de consulta.

La base de datos fue desarrollada a partir del siguiente modelo relacional. El modelo se ha dividido en partes por fines prácticos y explicativos pero se relaciona de la misma manera que el diagrama modular presentado anteriormente.

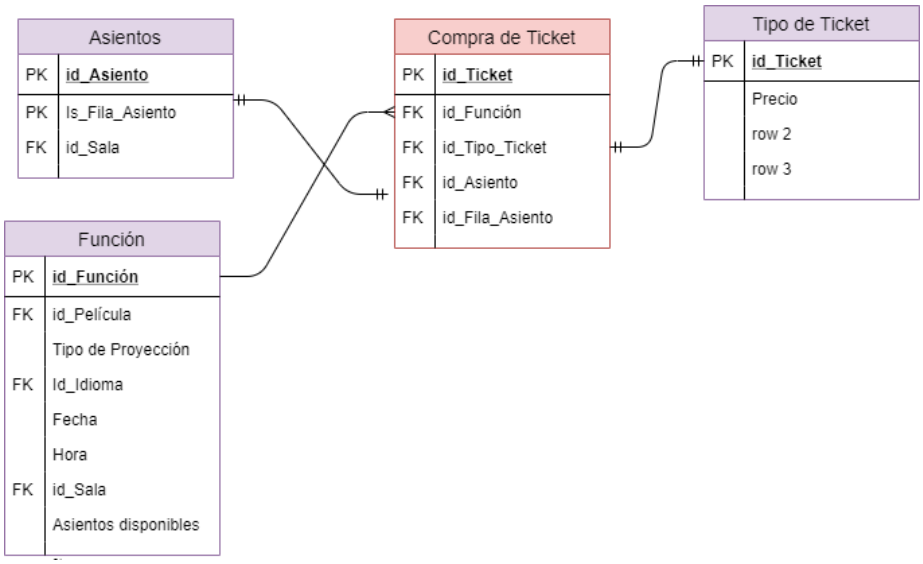


- Una película puede tener un detalle de película (relación: 1-1)
- Un detalle de película puede tener un género (relación: 1-1)
- Un detalle de película puede tener una película (relación: 1-1)

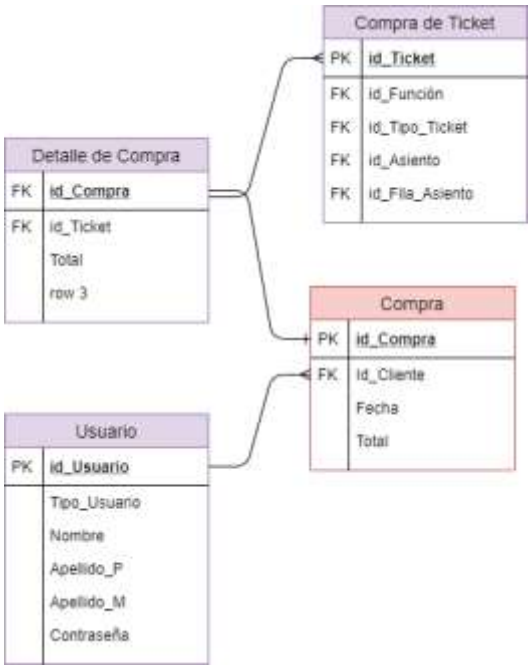


Nota: En este caso el uso del nombre de la tabla Horario es incorrecto, el nombre adecuado para ella es Función.

- Las funciones pueden presentar muchas películas (relación: 1-∞)
- Las funciones pueden presentarse en varios idiomas (relación: 1-∞)
- Las funciones pueden presentarse en varias salas (relación: 1-∞)
- Una sala puede tener muchos asientos (relación: 1-∞)

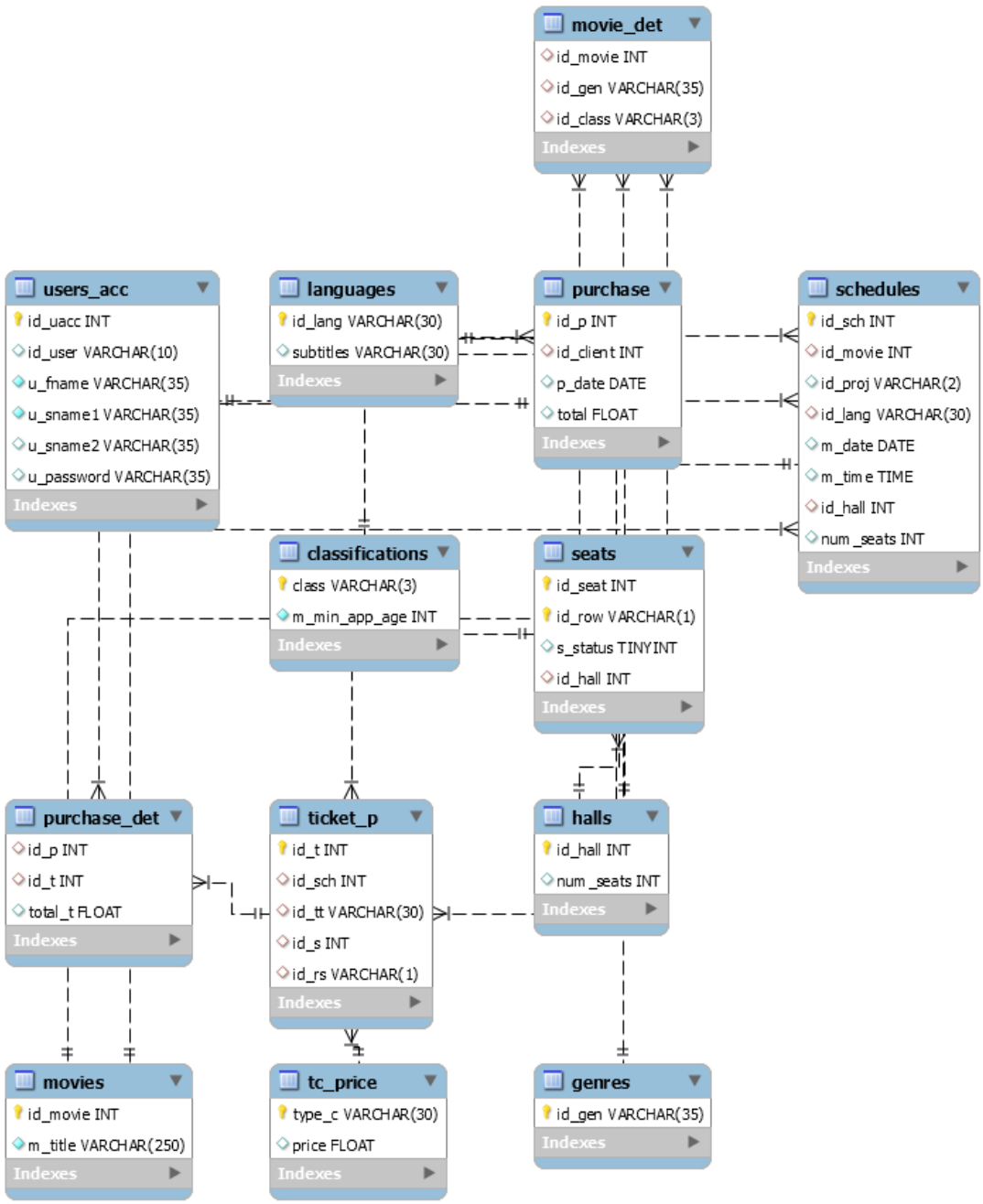


- La compra de un boleto tiene un precio individual (relación: 1-1)
- La compra de un boleto tiene asignado un lugar específico en una sala (relación: 1-1)
- Se pueden comprar muchos boletos para una función (relación: 1-∞)



- Muchos usuarios pueden generar una compra(relación: 1-∞)
- Una compra tiene un detalle de compra individual (relación: 1-1)
- Cada detalle de compra puede tener muchas compras de boletos (relación: 1-∞)

Después de haber escrito el scripts en MySQL este fue el diagrama relacional generado usando ingeniería inversa:

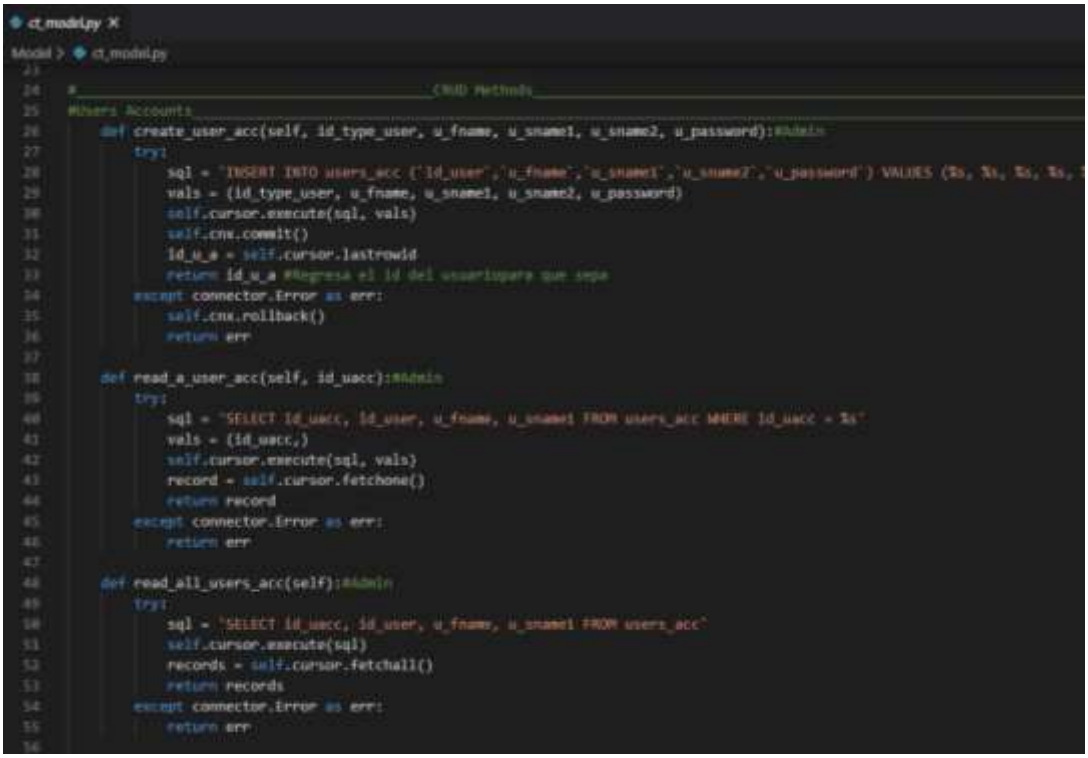


Se siguió el modelo genérico del diagrama MVC para el diseño del sistema

En el Model se hizo la conexión a la base de datos y se crearon los métodos básicos CRUD (crear, leer, actualizar y eliminar) para cada una de las tablas del modelo relacional.

```python
from mysql import connector

class Model:
    def __init__(self, config_db_file='ct_config.txt'):
        self.config_db_file = config_db_file
        self.config_db = self.read_config_db()
        self.connect_to_db()

    def read_config_db(self):
        d = {}
        with open(self.config_db_file) as f_r:
            for line in f_r:
                (key, val) = line.strip().split(':')
                d[key] = val
        return d

    def connect_to_db(self):
        self.cnx = connector.connect(**self.config_db)
        self.cursor = self.cnx.cursor()

    def close_db(self):
        self.cnx.close()
```

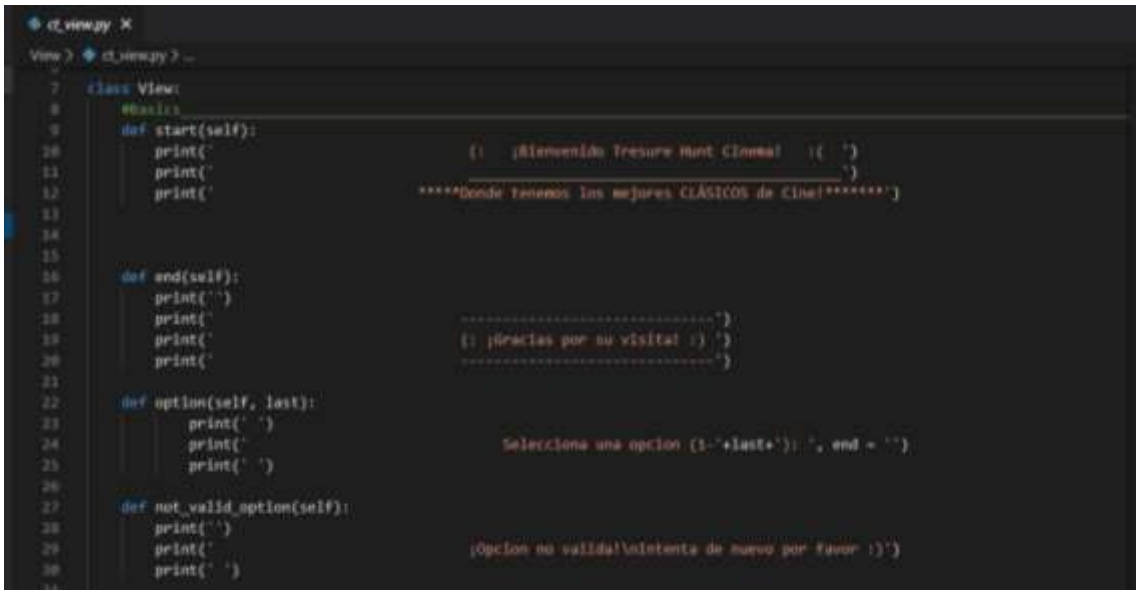
Conexión del modelo a la base de datos.



```python
    #_____CRUD Methods_____
    #Users Accounts
    def create_user_acc(self, id_type_user, u_fname, u_sname1, u_sname2, u_password):#Admin
        try:
            sql = "INSERT INTO users_acc (`id_user`,`u_fname`,`u_sname1`,`u_sname2`,`u_password`) VALUES (%s, %s, %s, %s, %
            vals = (id_type_user, u_fname, u_sname1, u_sname2, u_password)
            self.cursor.execute(sql, vals)
            self.cnx.commit()
            id_u_a = self.cursor.lastrowid
            return id_u_a #Regresa el id del usuariopara que sepa
        except connector.Error as err:
            self.cnx.rollback()
            return err

    def read_a_user_acc(self, id_uacc):#Admin
        try:
            sql = "SELECT id_uacc, id_user, u_fname, u_sname1 FROM users_acc WHERE id_uacc = %s"
            vals = (id_uacc,)
            self.cursor.execute(sql, vals)
            record = self.cursor.fetchone()
            return record
        except connector.Error as err:
            return err

    def read_all_users_acc(self):#Admin
        try:
            sql = "SELECT id_uacc, id_user, u_fname, u_sname1 FROM users_acc"
            self.cursor.execute(sql)
            records = self.cursor.fetchall()
            return records
        except connector.Error as err:
            return err
```

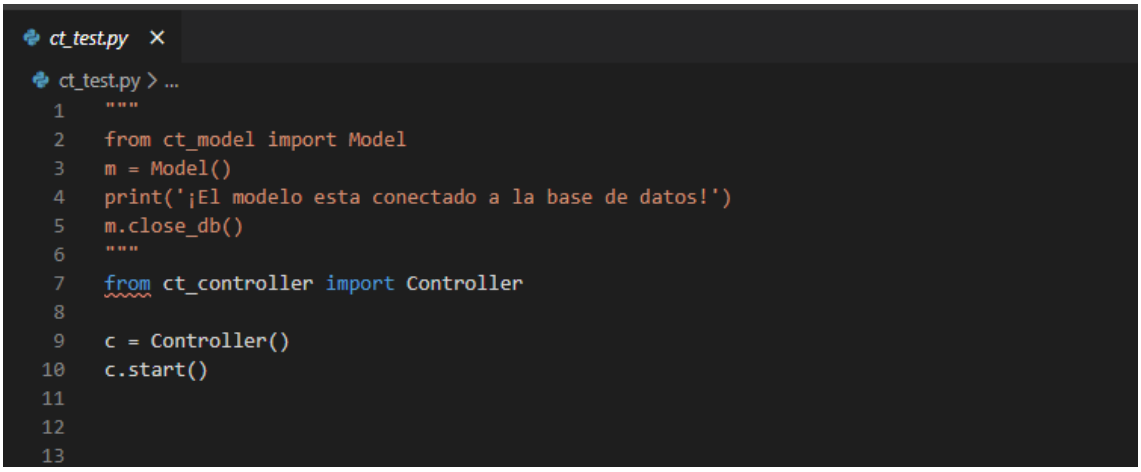
Creación de los métodos CRUD para cada tabla.

En el View se realizaron las diversas vistas, es decir, la interfaz de texto a partir de impresiones en pantalla.



El Controller manda llamar un menú de opciones escrito en el View para cada usuario, después de que el usuario ingresa la opción que desea realizar se llama al Model para realizar la consulta y regresar el valor requerido, después lo despliega en pantalla.

Se usa un script llamado Test para ejecutar el programa completo



Nota: Ese género ese error ya que mi tiene un problema con el pylint pero solo es cuestión de reordenar y organizar el directorio de mi sistema.

Conclusión

En general me agrado bastante llevar a cabo esta práctica, debo decir que es el programa mas largo que he escrito, se que esto debe ser malo porque para lo que llevo de la carrera ya debería haber desarrollado aplicaciones mas grandes o mas sin embargo no he tenido esa experiencia.

El realizar este proyecto de manera individual me encomio a trabajar mejor y mas rápido, a organizarme y ser paciente ya que de vez en cuando me surgían errores que no comprendía, sin embargo, terminaba siendo muy fáciles de resolver, solo fue cuestión de prestar más atención a lo que codificaba.

Me habría gustado no perder tanto tiempo diseñando mi diagrama modular ya que pensaba mucho en alcanzar la forma más practica de hacerlo, aunque debo admitir que no estoy segura de haberlo logrado.

De haber tenido más tiempo habría agregado una interfaz más linda y también quitaría algunas funciones que están de mas o pueden ahorrarse.

Me siento muy bien por el trabajo que realice, creo que no fue el mejor y tiene algunos errores en impresiones, faltas de ortografía y demás detalles, pero creo que fue un trabajo duro y honesto de casi un mes del que me siento orgullosa de haber realizado.