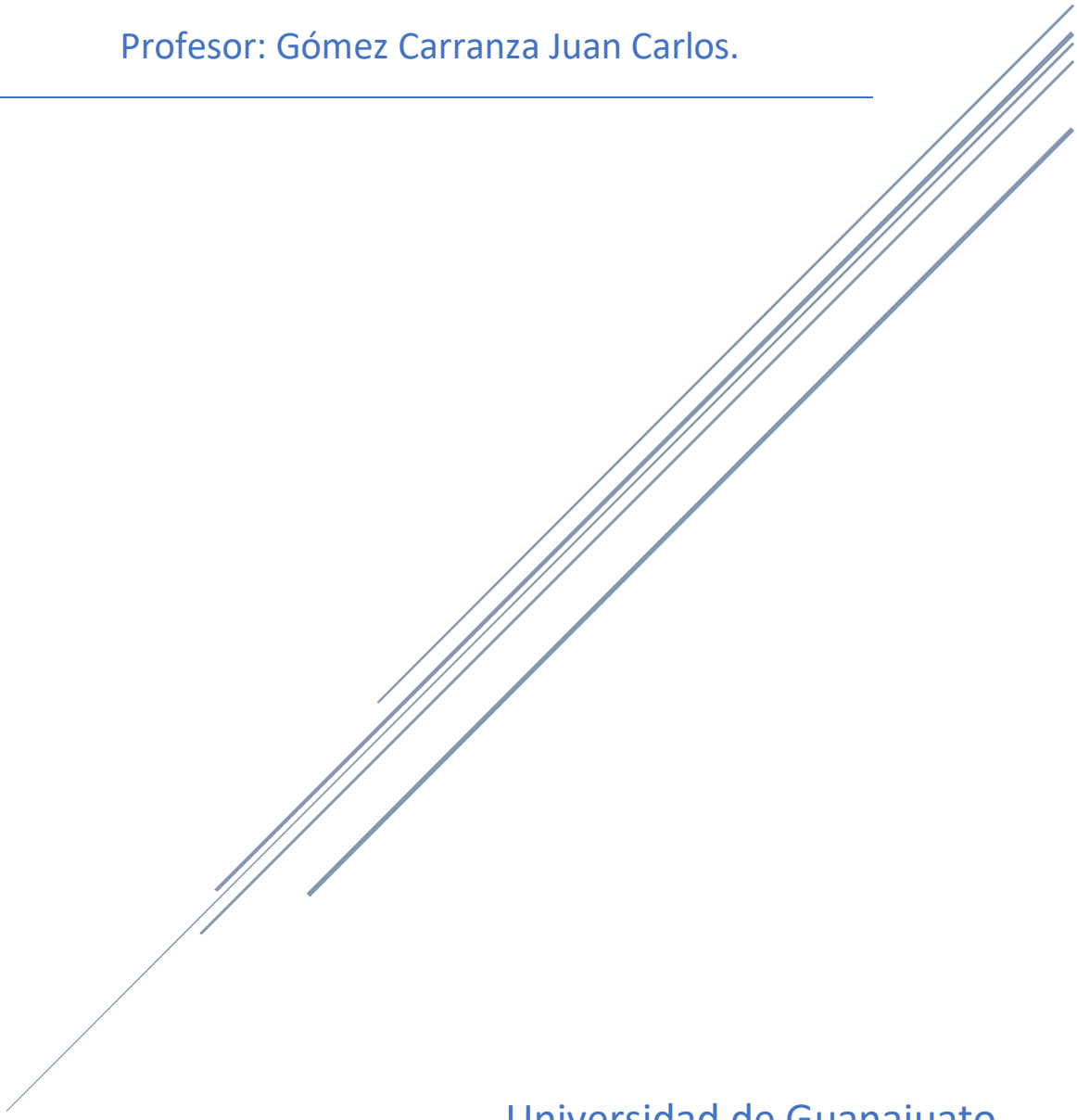
PRÁCTICA DE SISTEMAS DE INFORMACIÓN

Alumno: González Corona Hiram.

Profesor: Gómez Carranza Juan Carlos.



Universidad de Guanajuato.

Introducción: La arquitectura de software modelo vista controlador separa los datos de una aplicación, la interfaz de usuario y la lógica en tres componentes distintos.

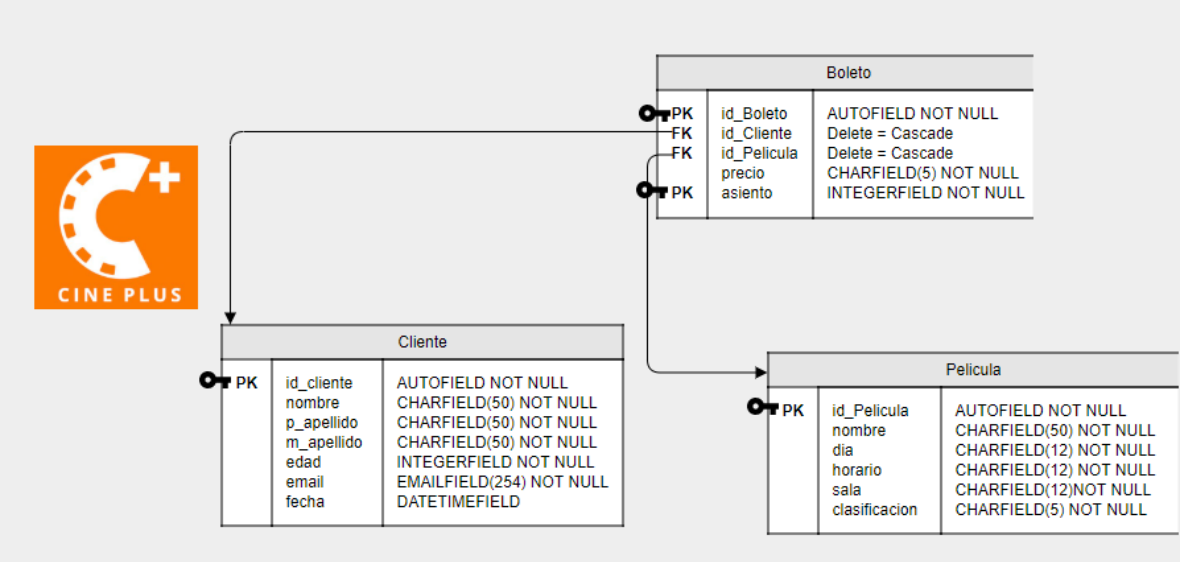
- El Modelo que contiene una representación de los datos que maneja el sistema, su lógica de negocio, y sus mecanismos de persistencia.
- La Vista, o interfaz de usuario, que compone la información que se envía al cliente y los mecanismos interacción con éste.
- El Controlador, que actúa como intermediario entre el Modelo y la Vista, gestionando el flujo de información entre ellos y las transformaciones para adaptar los datos a las necesidades de cada uno.

Objetivo: Desarrollar base de datos para realizar la compra de un boleto de cine (para solo una sucursal), para poder realizar la compra de un ticket el usuario debe de estar registrado, dicho usuario puede ver las películas que hay en cartelera.

Herramientas:

- SQLite.
- Django.

Diagrama relacional / explicación.



La base de datos cuenta con 3 tablas (Boleto, Cliente y Película), para realizar la compra debe de existir un usuario en la base de datos ya que esto nos permite saber que usuario compro el boleto, los id de los boletos serán únicos para evitar la existencia de un boleto con el mismo id en caso de que existan usuarios con nombres similares, la llave foránea id_Pelicula que existe en boletos nos permite unir la películas con el boleto ya que nos interesa comprar un boleto para una película.

```python
class Cliente(models.Model):
    id_Cliente = models.AutoField(primary_key=True, null=False)
    nombre = models.CharField(max_length=30, null=False, default=False
    p_apellido = models.CharField(max_length=50, null=False, default=F
    m_apellido = models.CharField(max_length=50, null=False, default=F
    edad = models.IntegerField(null=False, default=False)
    email = models.EmailField(max_length=254, null=False)
    fecha = models.DateTimeField('Fecha de registro', auto_now = True,

    class Meta:
        verbose_name='Cliente'
    def __str__(self):
        return self.nombre

class Pelicula(models.Model):
    id_Pelicula = models.AutoField(primary_key=True, null=False)
    nombre = models.CharField(max_length=50, null=False)
    dia = models.CharField(max_length=10,null=False, default=False)
    horario = models.CharField(max_length=15, null=False, default=Fals
    sala = models.CharField(max_length=20, null=False, default=False)
    clasificacion = models.CharField(max_length=5,choices=CLASIFICACIO

    class Meta:
        verbose_name='Pelicula'
    def __str__(self):
        return self.nombre
```

```python
class Boleto(models.Model):
    class Meta:
        unique_together = (('id_Boleto', 'asiento'),)
    id_Boleto = models.AutoField(primary_key=True, null=False)
    cliente = models.ForeignKey(Cliente, on_delete=models.CASCADE)
    pelicula = models.ForeignKey(Pelicula, on_delete=models.CASCADE)
    precio = models.IntegerField(null=False, default=False)
    asiento = models.IntegerField(null=False)
```

Ilustración 1: Modelos.

Conclusión: La importancia de la arquitectura MVC (modelo – vista – controlador) es que nos permite tener control sobre nuestro sistema, ya que cada componente de la arquitectura tiene ciertas responsabilidades, por ejemplo, la vista es responsable de recibir datos y mostrarlos al usuario.

Es interesante este tipo de arquitectura ya que nos permite desarrollar código con más orden. Se decidió utilizar Django ya que es un framework que sigue el principio DRY y nos permite mantener orden con nuestras plantillas, las vistas y el modelo.