CINE

Justificación.

Este sistema es necesario para optimizar la administración de un cine, permitiendo la gestión de personas, funciones, reservas, promociones y espacios físicos. Las clases permiten automatizar tareas como la asignación de asientos, la modificación de funciones, y el manejo de promociones, contribuyendo a una mejor experiencia tanto para los empleados como para los usuarios.

Persona:

La clase Persona centraliza los atributos comunes a todos los tipos de personas que usen el sistema, tales como su nombre, correo y edad. Al tener una lista de todas las personas registradas, se facilita la administración y control de los usuarios y empleados.

Métodos y Atributos:

- **__init__**: Inicializa una persona con su nombre, correo y edad.
- **registrar**: Muestra un mensaje de confirmación al registrar a la persona.
- **actualizar_datos**: Permite actualizar los datos personales de una persona.
- **eliminar_persona**: Elimina una persona de la lista global.
- **personas_registradas**: Muestra un listado de todas las personas registradas en el sistema.

Usuario:

El Usuario es la entidad que interactúa directamente con el cine, haciendo reservas, consultando horarios y accediendo a promociones. La clase Usuario extiende las funcionalidades de Persona, añadiendo métodos que permiten realizar actividades específicas de los usuarios en el sistema de cine.

Métodos y Atributos:

- **reserva**: Realiza una reserva de un asiento en una función específica.
- **cancelar_reserva**: Permite al usuario cancelar una reserva previamente realizada.
- **acceder_promo**: Facilita al usuario acceder a promociones especiales.

Empleado:

Los empleados tienen una serie de responsabilidades operativas, como la gestión de funciones, películas, horarios, salas y promociones. La clase Empleado

extiende de Persona, pero ofrece más métodos específicos que permiten la gestión administrativa dentro del sistema de cine.

Métodos y Atributos:

- **acceder_sistema**: Permite a los empleados acceder al sistema.
- **agregar_funcion**: Añade una nueva función o proyección al sistema.
- **eliminar_funcion**: Elimina una función o proyección existente.
- **agregar_pelicula**: Añade una película a la base de datos.
- **eliminar_pelicula**: Elimina una película de la base de datos.
- **agregar_promo**: Agrega una nueva promoción.
- **modificar_promo**: Modifica una promoción existente.
- **eliminar_promo**: Elimina una promoción.
- **agregar_sala**: Agrega una nueva sala de cine.
- **eliminar_sala**: Elimina una sala de cine.
- **agregar_horario**: Añade un horario para una función de cine.
- **eliminar_horario**: Elimina un horario existente.

Espacio:

Toda función o servicio dentro del cine se lleva a cabo en un espacio físico, como una sala de cine o una zona de comida. La clase Espacio es una base para describir estos espacios, permitiendo detallar atributos comunes como el tamaño y el identificador de cada uno.

Métodos y Atributos:

- **__init__**: Inicializa un espacio con un tamaño y un identificador.
- **descripcion**: Muestra una descripción básica del espacio.

Sala:

La Sala es donde se realizan las proyecciones de las películas. Gestionar los asientos, la disponibilidad de la sala y la asignación de asientos es crucial para la operación del cine. Esta clase extiende de Espacio y permite asignar asientos, consultar disponibilidad y mostrar el estado de los asientos.

Métodos y Atributos:

- **__init__**: Inicializa una sala con un tamaño, tipo, identificador y un diccionario de asientos disponibles o ocupados.
- **descripcion**: Muestra una descripción de la sala.
- **asignar_asientos**: Asigna un asiento específico si está disponible.

- **disponibilidad_asiento**: Consulta la disponibilidad de un asiento específico.
- **mostrar_asientos**: Muestra el estado de todos los asientos de la sala.
- **consultar_disponibilidad**: Indica si la sala está disponible para una función.

ZonaComida:

Las zonas de comida en un cine son esenciales para mejorar la experiencia del cliente. Esta clase permite gestionar el menú, los precios de los productos y el inventario de insumos.

Métodos y Atributos:

- **__init__**: Inicializa una zona de comida con tamaño, tipo, menú, precios e insumos.
- **descripcion**: Muestra la descripción de la zona de comida.
- **consultar_disponibilidad**: Consulta si la zona de comida está disponible.
- **mostrar_menu**: Muestra el menú de la zona de comida.
- **mostrar_precios**: Muestra los precios de los productos.
- **mostrar_insumos**: Muestra la cantidad de insumos disponibles.

Pelicula:

Las películas son el núcleo de cualquier cine. Esta clase proporciona los detalles de cada película que se proyecta en el cine, como el título, el género y la duración.

Métodos y Atributos:

- **__init__**: Inicializa una película con título, género y duración.
- **descripcion**: Muestra una descripción de la película.

Funcion:

Las funciones son los horarios específicos en los que se proyectan las películas. Cada función está asociada a una película, una sala y un horario.

Métodos y Atributos:

- **__init__**: Inicializa una función con una película, sala y horario.
- **descripcion**: Muestra una descripción de la función, indicando la película, sala y horario.

Promocion:

Las promociones son herramientas que pueden incentivar a los clientes a asistir al cine. Esta clase gestiona los descuentos y las condiciones bajo las cuales se aplican.

Métodos y Atributos:

- **__init__**: Inicializa una promoción con nombre, descuento y condiciones.
- **descripcion**: Muestra la descripción de la promoción.
- **aplicar_promocion**: Aplica el descuento de la promoción a un precio.
- **cancelar_promocion**: Elimina el descuento de la promoción en un precio.
- **modificar_promocion**: Modifica los detalles de la promoción.
- **eliminar_promocion**: Elimina la promoción del sistema.

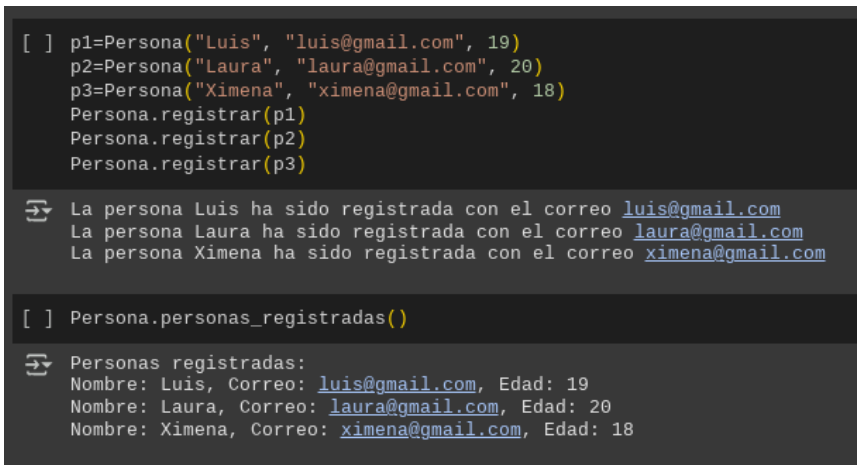
Reserva:

Las reservas son fundamentales para gestionar el acceso a las funciones. Esta clase registra las reservas de los usuarios, permitiendo realizar un seguimiento de los asientos ocupados y las funciones seleccionadas.

Métodos y Atributos:

- **__init__**: Inicializa una reserva con un ID, usuario, función y asiento.
- **descripcion**: Muestra los detalles de la reserva.
- **confirmar_reserva**: Confirma que una reserva ha sido realizada correctamente.
- **cancelar_reserva**: Cancela una reserva realizada anteriormente.

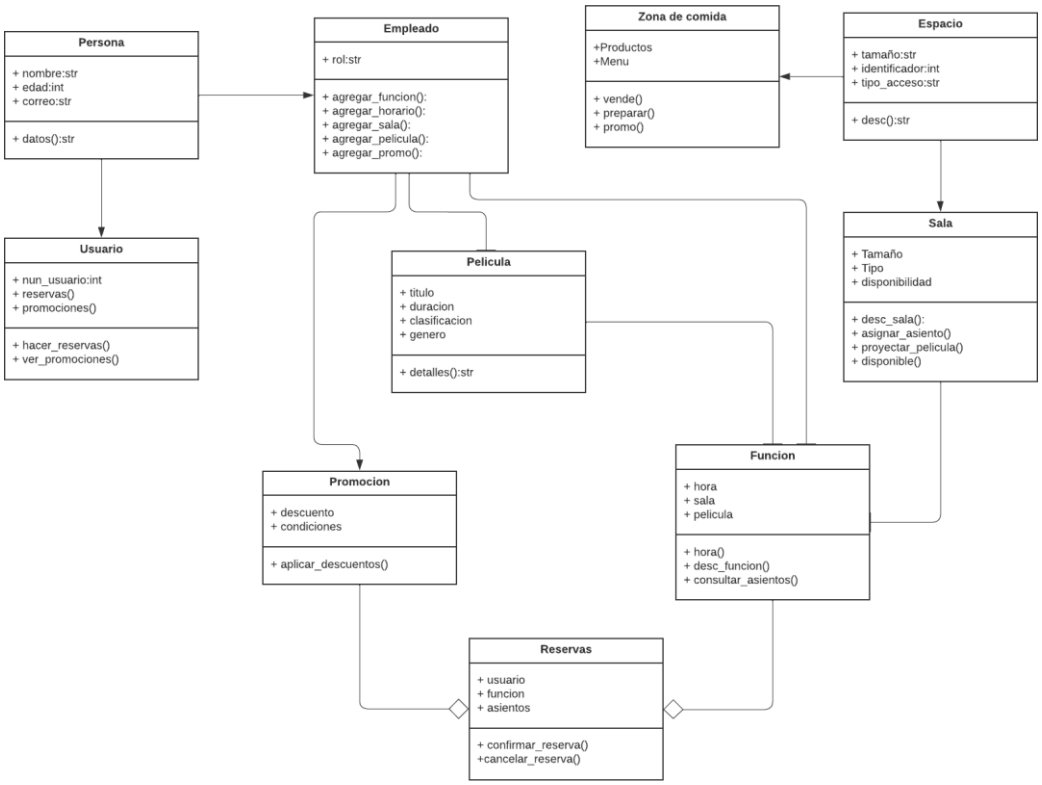
Capturas.



```
[ ] sala1=Sala(50, "Sala cine", "3D", {i: False for i in range(1, 50)})
    sala1.descripcion()
    sala1.asignar_asientos(1)
    sala1.consultar_disponibilidad()

    El lugar tiene un tamaño de 50 y su id es Sala cine
    La sala es de tipo 3D
    Asiento 1 asignado con éxito.
    La sala esta disponible

[ ] sala1.mostrar_asientos()

    Asiento 1: Ocupado
    Asiento 2: Disponible
    Asiento 3: Disponible
    Asiento 4: Disponible
    Asiento 5: Disponible
    Asiento 6: Disponible
    Asiento 7: Disponible
    Asiento 8: Disponible
    Asiento 9: Disponible
    Asiento 10: Disponible
    Asiento 11: Disponible
    Asiento 12: Disponible
    Asiento 13: Disponible
    Asiento 14: Disponible
    Asiento 15: Disponible
    Asiento 16: Disponible
    Asiento 17: Disponible
    Asiento 18: Disponible
    Asiento 19: Disponible
    Asiento 20: Disponible
    Asiento 21: Disponible
    Asiento 22: Disponible
    Asiento 23: Disponible
    Asiento 24: Disponible
    Asiento 25: Disponible
    Asiento 26: Disponible
    Asiento 27: Disponible
    Asiento 28: Disponible
    Asiento 29: Disponible
    Asiento 30: Disponible
    Asiento 31: Disponible
    Asiento 32: Disponible
    Asiento 33: Disponible
    Asiento 34: Disponible
    Asiento 35: Disponible
    Asiento 36: Disponible
    Asiento 37: Disponible
    Asiento 38: Disponible
    Asiento 39: Disponible
    Asiento 40: Disponible
    Asiento 41: Disponible
    Asiento 42: Disponible
    Asiento 43: Disponible
    Asiento 44: Disponible
    Asiento 45: Disponible
    Asiento 46: Disponible
    Asiento 47: Disponible
    Asiento 48: Disponible
    Asiento 49: Disponible
```

```
[ ] pelicula1 = Pelicula("Matrix", "Ciencia Ficción", 136)
    pelicula2 = Pelicula("Titanic", "Drama/Romance", 195)

    sala1 = Sala(48,"Sala 1","3DX", {i: False for i in range(1, 48)})
    sala2 = Sala(50,"Sala 2","Tradicional", {i: False for i in range(1, 50)})

    funcion1 = Funcion(pelicula1, sala1, "18:00")
    funcion2 = Funcion(pelicula2, sala2, "20:00")

    usuario1 = Usuario("Ana Pérez", "ana.perez@email.com", 25)
    empleado1 = Empleado("Luis Martínez", "luis.martinez@email.com", 38, "Gerente")
    usuario1.registrar()
    empleado1.registrar()

    usuario1.reserva(funcion1, 3)
    usuario1.cancelar_reserva(funcion1)

    promocion1 = Promocion(20, 25, "Valido de lunes a jueves")
    promocion1.descripcion()
    promocion1.modificar_promocion(21, 30, "Válido todos los días antes de las 5 PM.")

    Persona.personas_registradas()

    La persona Ana Pérez ha sido registrada con el correo ana.perez@email.com
    La persona Luis Martínez ha sido registrada con el correo luis.martinez@email.com
    Has realizado una reserva
    Has cancelado tu reserva
    La promocion 20 tiene un descuento del 25% y se aplica a las siguientes condiciones: Valido de lunes a jueves
    La promocion ha sido modificada
    Personas registradas:
    Nombre: Ana Pérez, Correo: ana.perez@email.com, Edad: 25
    Nombre: Luis Martínez, Correo: luis.martinez@email.com, Edad: 38
```

Diagrama UML



CAFETERIA

Justificación.

Este sistema es necesario para optimizar la administración de una cafetería, facilitando la gestión de productos, empleados, clientes, inventario, pedidos y promociones. La automatización de procesos como la toma de pedidos, la actualización de inventarios, la asignación de roles y el manejo de promociones contribuye a una operación más ágil, reduciendo los errores humanos y mejorando tanto la eficiencia del personal como la experiencia de los clientes.

Persona:

La clase Persona centraliza los atributos comunes de todas las personas que utilizan el sistema, ya sean empleados o clientes. Al tener una lista de todas las personas registradas, se facilita la administración de datos y el control de las personas involucradas en el proceso.

Métodos y Atributos:

- **__init__(self, nombre, id_P):** Inicializa una persona con su nombre e ID único.
- **datos(self):** Muestra los datos básicos de la persona (nombre e ID).
- **__str__(self):** Representación en cadena de la persona, mostrando su nombre y ID.

Empleado:

La clase Empleado extiende de Persona y permite gestionar las funciones específicas de los empleados, como la actualización del inventario, la gestión de pedidos y la administración del sistema.

Métodos y Atributos:

- **__init__(self, nombre, id_P, rol):** Inicializa al empleado con su nombre, ID y rol.
- **datos(self):** Muestra los datos del empleado y su rol.
- **abrir_inventario(self):** Permite abrir el inventario para actualizar productos.
- **actualizar_inventario(self, producto_nombre, cantidad_cambio):** Actualiza el inventario con una cantidad determinada de productos.
- **cerrar_inventario(self):** Cierra el inventario una vez se han realizado los cambios.
- **__str__(self):** Representación en cadena de la persona y su rol.

Cliente:

La clase Cliente también extiende de Persona y está orientada a la gestión de la experiencia del cliente en la cafetería, como la creación de pedidos, el historial de compras y la acumulación de puntos.

Métodos y Atributos:

- **__init__(self, nombre, id_P):** Inicializa al cliente con su nombre e ID.
- **hacer_pedido(self, pedido):** Permite realizar un pedido.
- **actualizar_pedido(self):** Permite modificar un pedido previamente realizado.
- **mostrar_historial_pedidos(self):** Muestra el historial de pedidos del cliente.
- **confirmar_pedido(self):** Confirma el pedido realizado.
- **acumular_puntos(self, puntos):** Acumula puntos por fidelidad.

- **canjear_puntos(self, puntos_a_canjear):** Permite al cliente canjear puntos por descuentos.
- **__str__(self):** Representación en cadena del cliente con su nombre y puntos acumulados.

ProductoBase:

La clase ProductoBase sirve como base para definir los productos disponibles en la cafetería, como bebidas o postres. Gestiona atributos comunes como nombre y precio.

Métodos y Atributos:

- **__init__(self, nombre, precio):** Inicializa un producto con su nombre y precio.
- **datos(self):** Muestra la información del producto (nombre y precio).

Bebida:

La clase Bebida extiende de ProductoBase y permite gestionar atributos específicos de las bebidas, como el tamaño, tipo y extras (por ejemplo, leche o azúcar).

Métodos y Atributos:

- **__init__(self, nombre, precio, tamaño, tipo, extras):** Inicializa una bebida con su tamaño, tipo y extras.
- **aplicar_extra(self, extra, precio_extra):** Aplica un extra a la bebida.
- **agregar_extra(self, extra):** Agrega un extra a la bebida.
- **eliminar_extra(self, extra):** Elimina un extra de la bebida.
- **calcular_total(self):** Calcula el total de la bebida, considerando el precio base y los extras.
- **__str__(self):** Representación en cadena de la bebida.

Postre:

La clase Postre extiende de ProductoBase y maneja los productos de postres disponibles, incluyendo características como si son aptos para veganos o sin gluten.

Métodos y Atributos:

- **__init__(self, nombre, precio, tipo, vegano=False, sin_gluten=False):** Inicializa un postre con su tipo, y las opciones vegano y sin gluten.
- **calcular_total(self):** Calcula el precio total del postre basado en sus características.

Inventario:

La clase Inventario gestiona los productos disponibles en la cafetería, permitiendo agregar y actualizar el stock de ingredientes y productos.

Métodos y Atributos:

- **agregar_producto(self, producto, cantidad):** Agrega un producto al inventario.
- **actualizar_stock(self, producto, cantidad_cambio):** Actualiza el stock de un producto determinado.
- **verificar_stock(self, ingredientes_necesarios)**: Verifica si los productos e ingredientes necesarios están disponibles en el inventario.
- **cerrar_inventario(self):** Cierra el inventario y muestra su estado.

Pedido:

La clase Pedido permite gestionar los pedidos realizados por los clientes, con la capacidad de agregar productos, eliminar productos y calcular el total del pedido.

Métodos y Atributos:

- **agregar_producto(self, producto):** Agrega un producto al pedido.
- **eliminar_producto(self, producto):** Elimina un producto del pedido.
- **calcular_total(self):** Calcula el total del pedido, sumando los precios de los productos.
- **confirmar_pedido(self):** Confirma el pedido realizado por el cliente.
- **actualizar_estado(self, estado_nuevo):** Actualiza el estado del pedido.
- **datos(self):** Muestra los detalles del pedido.

Promocion:

La clase Promocion gestiona las promociones aplicables en la cafetería, como descuentos en productos o combinaciones de productos a precios especiales.

Métodos y Atributos:

- **__init__(self, nombre, descripcion, condiciones, tipo_descuento, valor_descuento, producto_que_aplican=None, es_frecuente=False, puntos_necesarios=0):** Inicializa una promoción con su nombre, descripción, condiciones y tipo de descuento.
- **aplicar(self, pedido, cliente=None):** Aplica la promoción al pedido si el cliente cumple las condiciones.
- **aplicar_promo(self, pedido, cliente=None):** Aplica el descuento de la promoción.

- **total_promocion(self, pedido):** Muestra el total del descuento que se aplica a un pedido.

Capturas:

```
[ ] p1 = Persona("Juan", 3001)
    p2 = Persona("Maria", 3002)
    p1.datos()
    p2.datos()

Nombre: Juan
ID: 3001
Nombre: Maria
ID: 3002
```

```
[ ] e1 = Empleado("Juan", 2005, "Gerente")
    e1.datos()
    e1.abrir_invetario()
    e1.actualizar_inventario(i2, "Pastel zanahoria", 5)
    e1.cerrar_inventario()

Nombre: Juan
ID: 2005
Rol: gerente
Abriendo inventario...
Inventario de Pastel zanahoria actualizado. Cambio: 5. Stock actual: 30
Inventario actualizado por Juan, (gerente): Pastel zanahoria - Cambio: 5
Cerrando inventario...
```

```
[ ] c1 = Cliente("Maria", 3002)
    c1.datos()
    c1.hacer_pedido(pedido= "Cafe americano")
    c1.confirmar_pedido()
    c1.mostrar_historial_pedidos()
    c1.canjear_puntos(10)

Nombre: Maria
ID: 3002
Nombre: Maria ID: 3002
Pedido realizado por Maria: Cafe americano
¿Desea confirmar su pedido?:si
Pedido confirmado.
Historial de Pedidos:
Cafe americano
Cliente Maria no tiene suficientes puntos para canjear.
0
```

```
[ ] pb1 = ProductoBase("Cafe americano", 50)
    pb1.datos()

Nombre: Cafe americano
Precio: 50 pesos
```

```
[ ] b1 = Bebida("Cafe americano", 50, "Grande", "Frio", {"Extra leche": 5})
    b1.datos()
    b1.aplicar_extra("Extra leche", 5)
    b1.aplicar_extra("Sin azucar", 0)
    b1.calcular_total()

Nombre: Cafe americano
Precio: 50 pesos
Tamaño: Grande
Tipo: Frio
Extras: {}
Se ha agregado el extra 'Extra leche' al pedido.
Se ha agregado el extra 'Sin azucar' al pedido.
70
```

```
[ ] pos1 = Postre("Pastel fresas", 100, "Dulce")
    pos1.vegano = True
    pos1.sin_gluten = False
    pos1.datos()

Nombre: Pastel fresas
Precio: 100 pesos
Tipo: Dulce
Vegano: True
Sin gluten: False
```

```
[ ] i1 = Inventario()
    i1.agregar_producto("Cafe americano", 10)
    i1.agregar_producto("Pastel fresas", 5)
    i1.agregar_producto("Pastel fresas", 5)
```

```
El producto Cafe americano ya existe en el inventario.
El producto Pastel fresas ya existe en el inventario.
Se agregaron 5 de Pastel fresas al inventario.
```

```
[ ] i2 = Inventario()
    i2.agregar_producto("Pastel zanahoria", 20)
    i2.actualizar_stock("Pastel zanahoria", 5)
    i2.cerrar_inventario()
```

```
El producto Pastel zanahoria ya existe en el inventario.
Inventario de Pastel zanahoria actualizado. Cambio: 5. Stock actual: 25
Cerrando inventario...
Productos en el inventario:
Pastel zanahoria: 25
```

```
[ ] ped1 = Pedido(c1, e1)
    ped1.agregar_producto(b1)
    ped1.agregar_producto(pos1)
    ped1.calcular_total()
    ped1.confirmar_pedido()
    ped1.datos()
```

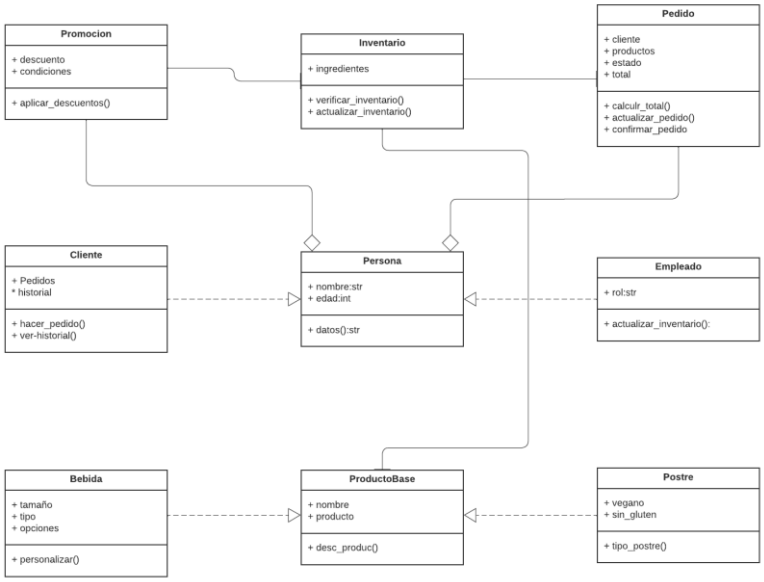
```
Producto Cafe americano agregado al pedido.
Producto Pastel fresas agregado al pedido.
¿Desea confirmar su pedido?:si
Pedido confirmado.
ID del pedido: 134724579017232
Cliente: Maria
Toma de pedido: Empleado: Nombre: Juan, ID: 2005 - Rol: Gerente
Estado: Pendiente
Fecha del pedido: 2025-02-12 00:10:35
Productos:
- Cafe americano
- Pastel fresas
Total: 180
```

```
[ ] pro1 = Promocion("Promo 1", "50% de descuento en postres veganos", "Prom
    pro1.datos()
    pro1.aplicar(ped1)
    pro1.aplicar_promo(ped1)
    pro1.total_promocion(ped1)
```

```
Nombre: Promo 1
Descripcion: 50% de descuento en postres veganos
Condiciones: Promo valida solo los Miercoles
Tipo de descuento: Porcentaje
Valor de descuento: 0.05
Cliente frecuente: True
Puntos necesarios: 100
Total de la promocion: 0.09
```

```
[ ] El producto cafe_molido ya existe en el inventario.
    El producto leche_entera ya existe en el inventario.
    El producto leche_almendra ya existe en el inventario.
    El producto azucar ya existe en el inventario.
    El producto jarabe_vainilla ya existe en el inventario.
    El producto tarta_manzana ya existe en el inventario.
    El producto brownie ya existe en el inventario.
    El extra 'leche_almendra' no esta disponible.
    El extra 'sin_azucar' no esta disponible.
    Producto Café con Leche agregado al pedido.
    Producto Tarta de Manzana agregado al pedido.
    ¿Desea confirmar su pedido?:si
    Pedido confirmado.

    Total del pedido (sin promo): 85.00
    Descuento aplicado (fidelidad): 0.00
    Total del pedido (con descuento): 85.00
    Inventario de cafe_molido actualizado. Cambio: -20. Stock actual: 980
    Inventario de leche_almendra actualizado. Cambio: -0.3. Stock actual: 299.7
    Inventario de azucar actualizado. Cambio: -5. Stock actual: 1995
    Inventario de tarta_manzana actualizado. Cambio: -1. Stock actual: 49
    No hay historial de pedidos para este cliente.
    None
    Inventario:
    cafe_molido: 980
    leche_entera: 500
    leche_almendra: 299.7
    azucar: 1995
    jarabe_vainilla: 100
    tarta_manzana: 49
    brownie: 60
    Total de productos en el inventario: 4010
    Empleado: Nombre: Ana, ID: 1 - Rol: Gerente
    Cliente: Nombre: Sofía, ID: 101 - Puntos: 0
     - Descuento: 0.1% - Cliente frecuente: Si - Puntos necesarios: 40
     - Descuento: 0.5% - Cliente frecuente: No
```

Diagrama UML.



BIBLIOTECA

Justificación.

Este sistema es necesario para optimizar la administración de una biblioteca, gestionando materiales como libros, revistas y materiales digitales, así como usuarios, préstamos y penalizaciones. Las clases permiten automatizar tareas como el registro de materiales, la gestión de préstamos, y la actualización de estados, lo que contribuye a una mejor experiencia tanto para los empleados como para los usuarios.

Persona:

La clase Persona centraliza los atributos comunes a todos los tipos de personas que usen el sistema, tales como su nombre y su ID. Al tener una lista de todas las personas registradas, se facilita la administración y control de los usuarios y empleados.

Métodos y Atributos:

- **__init__:** Inicializa una persona con su nombre y ID.
- **datos:** Muestra los datos de la persona.

Usuario:

El Usuario es la entidad que interactúa directamente con la biblioteca, realizando consultas y préstamos de materiales. La clase Usuario extiende las

funcionalidades de Persona, añadiendo métodos que permiten realizar actividades específicas de los usuarios en el sistema.

Métodos y Atributos:

- **consultar_catalogo:** Permite al usuario consultar el catálogo de la biblioteca.
- **datos_us:** Muestra los detalles del usuario, incluyendo los materiales que ha prestado.

Bibliotecario:

Los bibliotecarios gestionan los materiales y los préstamos dentro de la biblioteca. La clase Bibliotecario extiende de Persona y ofrece métodos para añadir materiales, gestionar préstamos y devoluciones, y consultar los préstamos.

Métodos y Atributos:

- **agregar_material:** Añade un material al catálogo de la biblioteca.
- **gestionar_prestamo:** Permite al bibliotecario gestionar un préstamo de material a un usuario.
- **gestionar_devolucion:** Permite al bibliotecario gestionar la devolución de un material prestado.
- **consultar_prestamos:** Consulta los materiales prestados a un usuario.

Material:

La clase Material es la clase base para diferentes tipos de materiales, como libros, revistas y materiales digitales. Los materiales se identifican por su título y su ISBN.

Métodos y Atributos:

- **__init__:** Inicializa un material con un título y un ISBN.
- **datos:** Muestra los detalles del material (título y ISBN).

Libro:

El Libro es un tipo de material que extiende de la clase Material. Añade atributos específicos, como autor, género y estado, que indican si el libro está disponible o prestado.

Métodos y Atributos:

- **__init__:** Inicializa un libro con título, ISBN, autor, género y estado.

- **datos_lib:** Muestra los detalles del libro, incluyendo el autor, género y estado.

Revista:

La Revista es otro tipo de material que extiende de Material. Añade información adicional como la edición y la periodicidad de la publicación.

Métodos y Atributos:

- **__init__:** Inicializa una revista con título, ISBN, edición, periodicidad y estado.
- **datos_rev:** Muestra los detalles de la revista, incluyendo la edición y la periodicidad.

MaterialDigital:

El MaterialDigital es un tipo de material que extiende de la clase Material, pero su formato es digital. Se agregan atributos como tipo de archivo y enlace de descarga.

Métodos y Atributos:

- **__init__:** Inicializa un material digital con título, ISBN, tipo de archivo y enlace de descarga.
- **datos_md:** Muestra los detalles del material digital, incluyendo el tipo de archivo y el enlace de descarga.

Sucursal:

La clase Sucursal gestiona el catálogo de materiales y realiza las operaciones de préstamo y devolución de materiales. Permite agregar materiales y consultar el estado de la sucursal.

Métodos y Atributos:

- **agregar_material:** Agrega un material al catálogo de la sucursal.
- **prestar_material:** Permite prestar un material a un usuario.
- **devolver_material:** Permite devolver un material prestado.
- **detalles_sucursal:** Muestra los detalles de la sucursal y los materiales disponibles.

Prestamo:

La clase Prestamo gestiona los detalles de un préstamo de un material a un usuario. Mantiene información sobre el material, el usuario y las fechas de préstamo y devolución.

Métodos y Atributos:

- **__init__:** Inicializa un préstamo con el usuario, material, fecha de préstamo y fecha de devolución.
- **datos_pres:** Muestra los detalles del préstamo.

Penalizacion:

La clase Penalizacion maneja las multas o penalizaciones impuestas a los usuarios, por ejemplo, por no devolver un material en el tiempo acordado.

Métodos y Atributos:

- **__init__:** Inicializa una penalización con el usuario, monto y descripción.
- **datos_pen:** Muestra los detalles de la penalización.

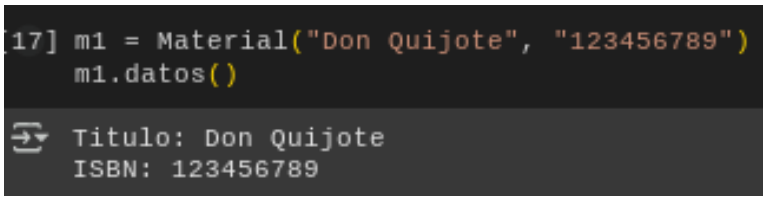
Catalogo:

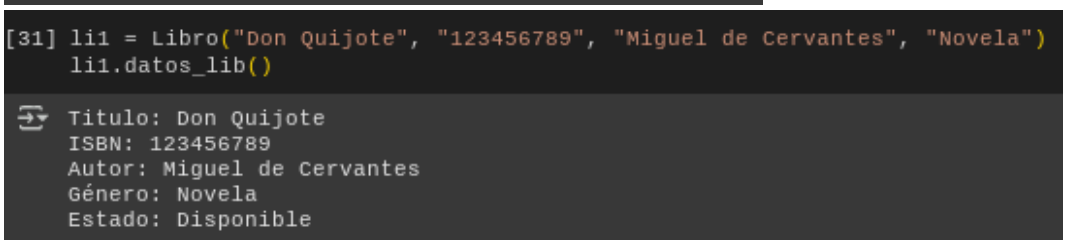
La clase Catalogo es responsable de gestionar los materiales de la biblioteca. Permite agregar materiales y mostrar todos los materiales disponibles en el catálogo.

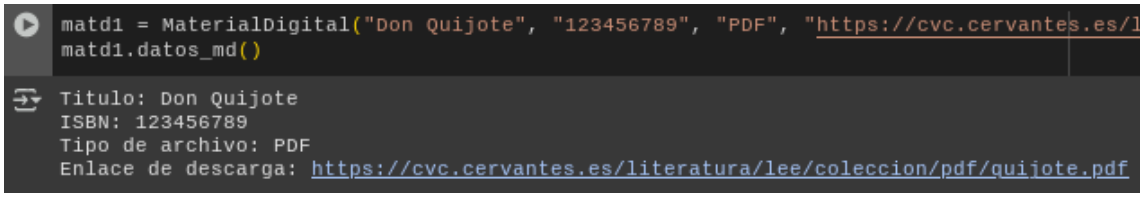
Métodos y Atributos:

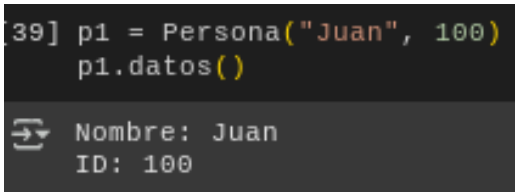
- **agregar_material:** Agrega un material al catálogo.
- **mostrar_materiales:** Muestra todos los materiales en el catálogo.

Capturas.









```
[78] us1 = Usuario("Juan", 100)
     us1.datos_us()
     us1.consultar_catalogo(sucursal1.catalogo)
```

```
Nombre: Juan
ID: 100
Materiales prestados: []
- Don Quijote (Libro)
- El Mundo (Revista)
- Don Quijote (MaterialDigital)
- Cien años de soledad (Libro)
- National Geographic (Revista)
- El Quijote (MaterialDigital)
```

```
[46] bi1 = Bibliotecario("Juan", 100, "Sucursal 1")
     bi1.datos_bi()
```

```
Nombre: Juan
ID: 100
Sucursal: Sucursal 1
```

```
[69] su1 = Sucursal("Sucursal 1", Catalogo)
     s1 = Libro("Don Quijote", "123456789", "Miguel de Cervantes", "Novela")
     s1.datos_lib()
     su1.agregar_material(s1)
     su1.detalles_sucursal()
```

```
Titulo: Don Quijote
ISBN: 123456789
Autor: Miguel de Cervantes
Género: Novela
Estado: Disponible
Nombre de la sucursal: Sucursal 1
- Don Quijote (Libro)
```

```
[63] pres1 = Prestamo(us1, s1, "01/01/2023", "01/02/2023")
     pres1.datos_pres()
```

```
Usuario: Juan
Material: Don Quijote
Fecha de préstamo: 01/01/2023
```

```
[70] pen1 = Penalizacion(us1, 100, "Multa por retraso en la devolución")
     pen1.datos_pen()
```

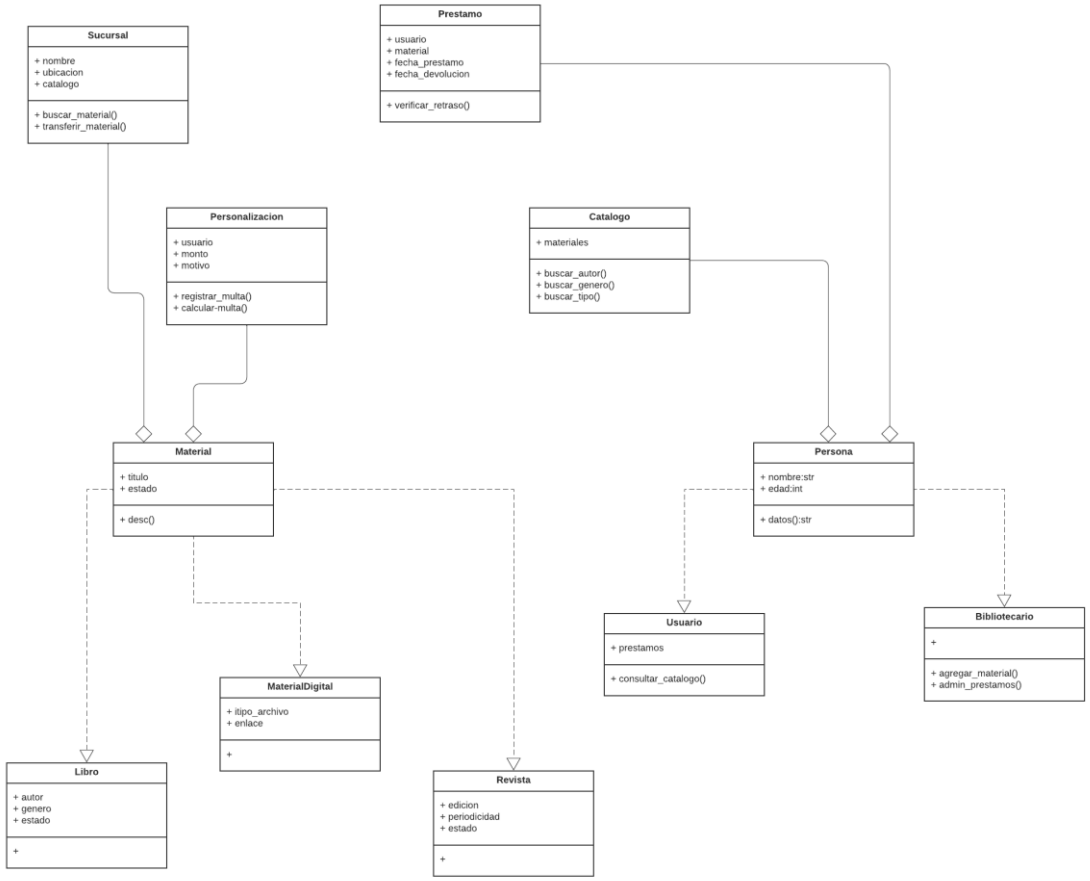
```
Usuario: Juan
Monto: 100
Descripción: Multa por retraso en la devolución
```

```
[61] s1 = Libro("Don Quijote", "123456789", "Miguel de Cervantes", "Novela")
     s2 = Revista("El Mundo", "123456780", "Primera", "Mensual")
     s3 = MaterialDigital("Don Quijote", "123456789", "PDF", "https://cvc.cerv
```

```
[79] sucursal1 = Sucursal("Sucursal Centro", Catalogo)
     sucursal1.agregar_material(s1)
     sucursal1.agregar_material(s2)
     sucursal1.agregar_material(s3)
     bibliotecario1 = Bibliotecario("Ana", "12345", sucursal1)

     libro1 = Libro("Cien años de soledad", "978-0307262545", "Gabriel García
     revista1 = Revista("National Geographic", "1234-5678", "Enero 2024", "Mer
     material_digital1 = MaterialDigital("El Quijote", "978-0060932552", "PDF"

     bibliotecario1.agregar_material(libro1)
     bibliotecario1.agregar_material(revista1)
     bibliotecario1.agregar_material(material_digital1)

     usuario1 = Usuario("Juan", "67890")
     usuario1.consultar_catalogo(sucursal1.catalogo)

     bibliotecario1.gestionar_prestamo(usuario1, libro1)
```

```
- Don Quijote (Libro)
- El Mundo (Revista)
- Don Quijote (MaterialDigital)
- Cien años de soledad (Libro)
- National Geographic (Revista)
- El Quijote (MaterialDigital)
Material 'Cien años de soledad' no disponible
```

Diagrama UML.



Abril Aragón Téllez

https://github.com/abrilaragon97/prograara