Práctica Unidad 1

Programación Avanzada

Nombre del Alumno: José Alexander Caballero Palma
Nombre del profesor: Jaime Alejandro Romero Sierra
Materia: Programación Avanzada
Práctica Unidad 1

Link de Github: https://github.com/Alexander117161/Pr-ctica-Unidad-1-Prog.-Avanzada

1. Sistema de Reservas para un Cine

JUSTIFICACIÓN DE LA NECESIDAD DEL PRIMER PROGRAMA:

El cine es un negocio que depende de una buena organización para ofrecer una experiencia cómoda y eficiente a los clientes. Sin un sistema automatizado, la gestión de reservas, la venta de boletos y la programación de funciones sería caótica, lo que podría generar confusión, errores y malestar en los espectadores. Este sistema es necesario para organizar eficientemente las reservas, evitar errores en la asignación de asientos y mejorar la experiencia del cliente. Sin él, habría riesgo de sobreventas, confusión en las funciones y dificultades para administrar promociones y horarios. Además, agiliza el trabajo del personal y permite que los espectadores reserven de forma rápida y segura, garantizando un servicio ordenado y sin inconvenientes.

CLASES A OCUPAR, ATRIBUTOS Y MÉTODOS NECESARIOS:

1. Clase: Persona

Atributos:

- nombre: (String) Nombre de la persona.
- correo: (String) Correo electrónico de la persona.

Métodos:

- __str__: Devuelve una cadena con el nombre y correo de la persona.

2. Clase: Usuario (Subclase de Persona)

Atributos:

- reservas: (Lista) Lista de reservas realizadas por el usuario.

Métodos:

- hacer_reserva: Añade una reserva a la lista de reservas del usuario.
- cancelar_reserva: Cancela una reserva de la lista de reservas del usuario y libera los asientos reservados.

3. Clase: Empleado (Subclase de Persona)

Atributos:

- rol: (String) Rol del empleado (taquillero, administrador, limpieza, etc.).

Métodos:

- agregar_funcion: Crea una nueva función de película en una sala con un horario específico.
- agregar_pelicula: Crea una nueva película con título, duración, clasificación y género.
- agregar_promocion: Crea una nueva promoción con una descripción, descuento y condiciones.
- modificar_promocion: Modifica una promoción existente.

4. Clase: Espacio (Clase base para espacios físicos en el cine)

Atributos:

- nombre: (String) Nombre del espacio físico (puede ser una sala o una zona de comida).

Métodos:

- Ningún método específico en esta clase base, solo para ser heredada por otras clases.

5. Clase: Sala (Subclase de Espacio)

Atributos:

- tipo: (String) Tipo de sala (2D, 3D, IMAX).
- capacidad: (Entero) Número máximo de asientos en la sala.
- asientos_ocupados: (Conjunto) Conjunto de asientos ocupados (números de asientos).

Métodos:

- verificar_disponibilidad: Verifica si hay suficiente capacidad para reservar los asientos solicitados.
- reservar_asientos: Reserva los asientos especificados si están disponibles.

6. Clase: ZonaComida (Subclase de Espacio)

Atributos:

- productos: (Diccionario) Diccionario donde la clave es el nombre del producto y el valor es una lista con el precio y el stock.

Métodos:

- agregar_producto: Agrega un producto a la zona de comida con precio y cantidad inicial.
- vender_producto: Vende una cantidad específica de un producto, actualizando el stock.
- mostrar_productos: Muestra todos los productos disponibles con su precio y stock.

7. Clase: Pelicula

Atributos:

- titulo: (String) Título de la película.
- duracion: (Entero) Duración de la película en minutos.
- clasificacion: (String) Clasificación de la película (PG, PG-13, R, etc.).
- genero: (String) Género de la película (comedia, drama, acción, etc.).

Métodos:

- Ningún método específico, solo se usa para almacenar información sobre la película.

8. Clase: Promocion

Atributos:

- descripcion: (String) Descripción de la promoción.
- descuento: (Entero) Descuento en porcentaje.
- condiciones: (String) Condiciones bajo las cuales la promoción es válida.

Métodos:

- __str__: Devuelve una cadena con la descripción de la promoción y sus condiciones.

9. Clase: Funcion

Atributos:

- pelicula: (Instancia de Pelicula) Película que se proyectará.
- sala: (Instancia de Sala) Sala donde se proyectará la película.
- horario: (String) Hora de la proyección.

Métodos:

- Ningún método específico, solo se usa para almacenar información sobre la función.

10. Clase: Reserva

Atributos:

- usuario: (Instancia de Usuario) Usuario que realizó la reserva.
- funcion: (Instancia de Funcion) Función a la que corresponde la reserva.
- asientos: (Lista) Lista de asientos reservados.
- promocion: (Instancia de Promocion, opcional) Promoción aplicada a la reserva.

Métodos:

- Ningún método específico, solo se usa para asociar un usuario con una función y los asientos reservados. Si se utiliza una promoción, se aplica a la reserva.

CAPTURA DE PANTALLA DEL CÓDIGO EJECUTADO

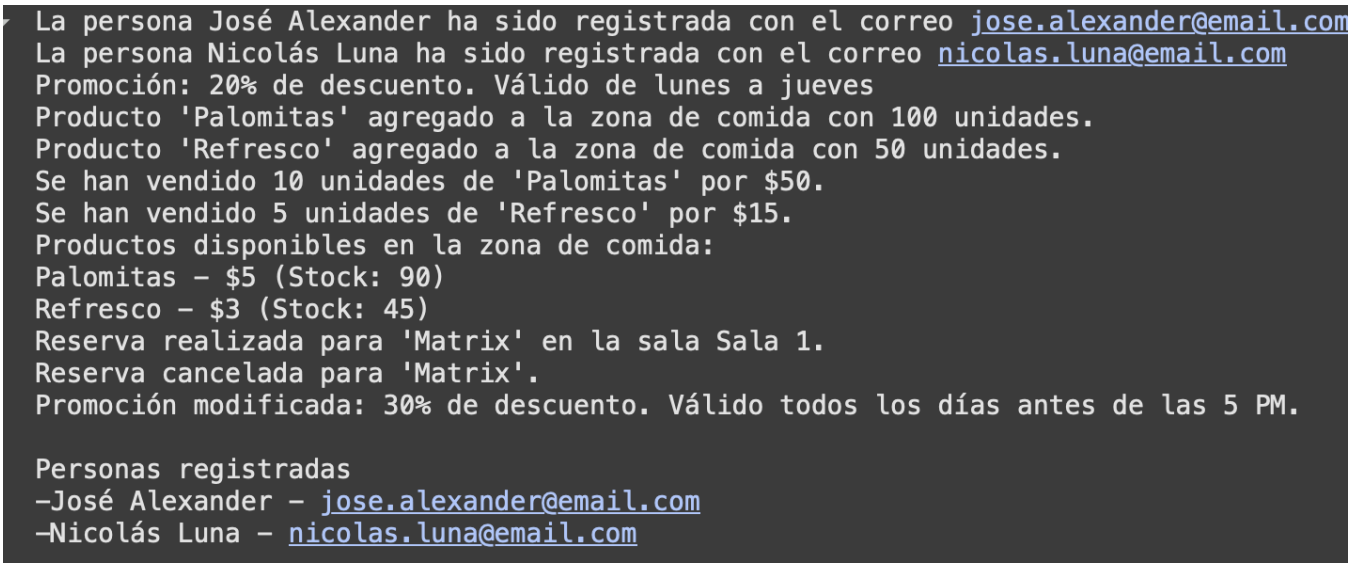
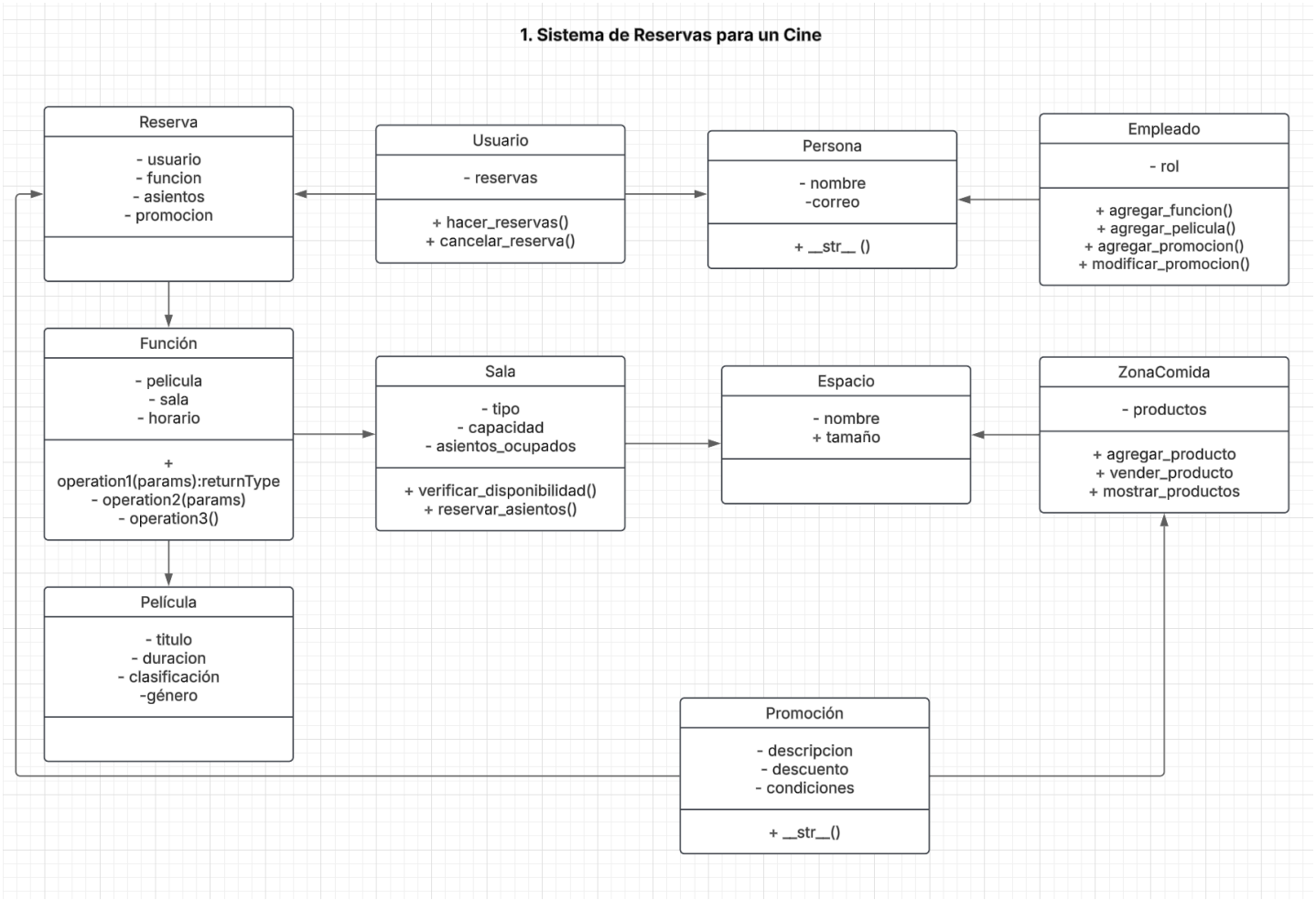


DIAGRAMA UML PRIMER SISTEMA



2. Sistema Gestión de Pedidos en una Cafetería

JUSTIFICACIÓN DE LA NECESIDAD DEL SEGUNDO PROGRAMA:

Un sistema de gestión de pedidos automatizado es esencial para garantizar un servicio ágil y organizado en la cafetería. Sin este sistema, la toma de pedidos, la personalización de productos y el control del inventario pueden volverse desordenados, lo que lleva a errores, demoras y falta de stock. Este sistema permite gestionar los pedidos de forma precisa, actualizar el inventario en tiempo real y aplicar promociones para clientes frecuentes. Facilita el trabajo del personal y asegura que los clientes reciban un servicio rápido, preciso y sin problemas, mejorando la eficiencia general del negocio y la satisfacción del cliente.

CLASES A OCUPAR, ATRIBUTOS Y MÉTODOS NECESARIOS:

1. Clase: Persona

Atributos:

- nombre: Nombre de la persona.
- correo: Correo electrónico de la persona.

Métodos:

- __init__(self, nombre, correo): Constructor para inicializar los atributos nombre y correo.

2. Clase: Cliente (Subclase de Persona)

Atributos:

- historial_pedidos: Lista de los pedidos realizados por el cliente.

Métodos:

- __init__(self, nombre, correo): Constructor que inicializa nombre, correo y historial_pedidos.
- realizar_pedido(self, pedido): Añade un pedido al historial del cliente.

3. Clase: Empleado (Subclase de Persona)

Atributos:

- rol: Rol del empleado (por ejemplo, barista, mesero, gerente).

Métodos:

- __init__(self, nombre, correo, rol): Constructor que inicializa nombre, correo y rol.
- actualizar_inventario(self, inventario, producto, cantidad): Actualiza el inventario con una cierta cantidad de producto.

4. Clase: ProductoBase (Clase base para productos)

Atributos:

- nombre: Nombre del producto.
- precio: Precio del producto.

Métodos:

- __init__(self, nombre, precio): Constructor para inicializar nombre y precio.

5. Clase: Bebida (Subclase de ProductoBase)

Atributos:

- tamano: Tamaño de la bebida (por ejemplo, grande, mediana).
- tipo: Tipo de bebida (por ejemplo, caliente, fría).
- opciones_personalizables: Lista de opciones adicionales personalizables para la bebida (por ejemplo, extra leche, sin azúcar).

Métodos:

- __init__(self, nombre, precio, tamano, tipo, opciones_personalizables=None): Constructor para inicializar los atributos.
- personalizar(self, opcion): Añade una opción personalizable a la bebida.

6. Clase: Postre (Subclase de ProductoBase)

Atributos:

- vegano: Indica si el postre es vegano (True/False).
- sin_gluten: Indica si el postre es sin gluten (True/False).

Métodos:

- __init__(self, nombre, precio, vegano, sin_gluten): Constructor para inicializar los atributos.

7. Clase: Inventario

Atributos:

- stock: Diccionario que guarda la cantidad de cada producto en el inventario.

Métodos:

- __init__(self): Constructor que inicializa el inventario vacío.
- agregar_producto(self, producto, cantidad): Añade un producto al inventario.
- verificar_stock(self, producto, cantidad): Verifica si hay suficiente stock para un producto.
- actualizar_stock(self, producto, cantidad): Actualiza la cantidad de un producto en el inventario.

8. Clase: Pedido

Atributos:

- cliente: El cliente que realizó el pedido.
- productos: Lista de productos en el pedido.
- estado: Estado del pedido (por ejemplo, pendiente, en preparación, entregado).
- total: Total del costo del pedido.

Métodos:

- __init__(self, cliente): Constructor para inicializar el pedido.
- agregar_producto(self, producto, inventario): Añade un producto al pedido si hay stock disponible.
- confirmar_pedido(self): Cambia el estado del pedido a "En preparación" y muestra el total.
- entregar_pedido(self): Cambia el estado del pedido a "Entregado".
- __str__(self): Representa el pedido de manera legible (con nombre de productos, total y estado).

9. Clase: Promoción

Atributos:

- tipo: Tipo de descuento ('porcentaje' o 'fijo').
- descuento: El valor del descuento (porcentaje o monto fijo).

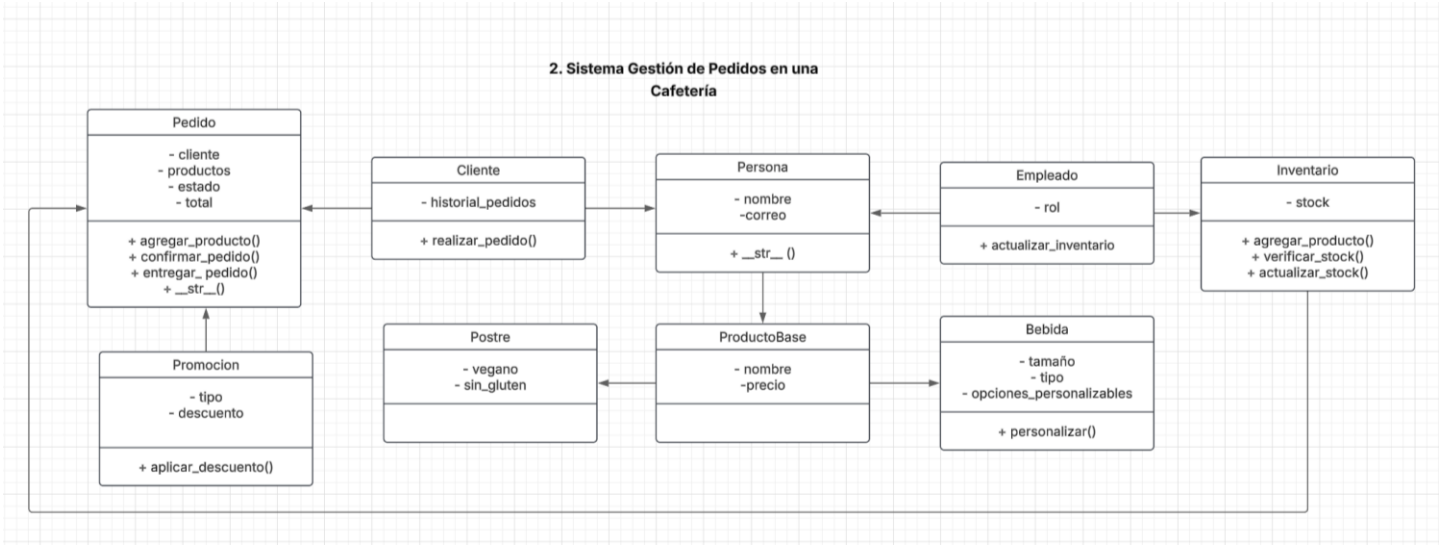
Métodos:

- __init__(self, tipo, descuento): Constructor para inicializar los atributos tipo y descuento.
- aplicar_descuento(self, pedido): Aplica un descuento al pedido basado en el tipo de promoción.

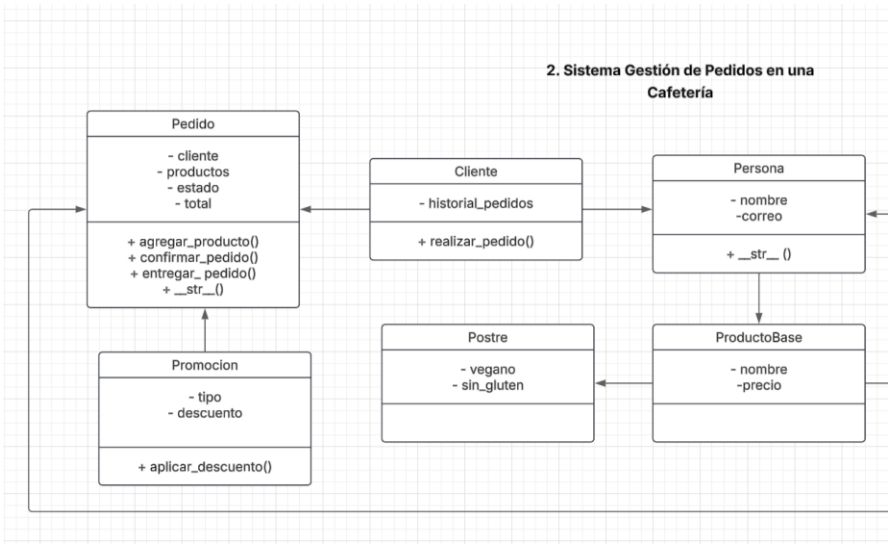
CAPTURA DE PANTALLA DEL CÓDIGO EJECUTADO

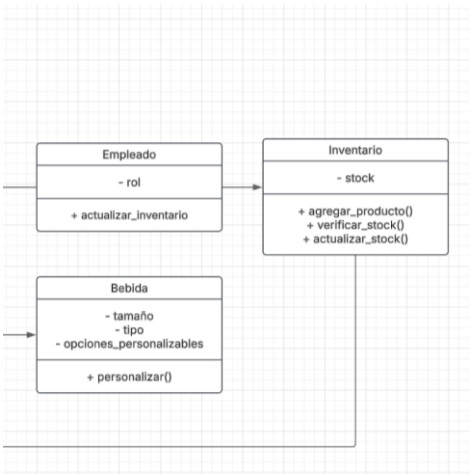
```
Producto Café con leche agregado al inventario: 10 unidades
Producto Tarta de manzana agregado al inventario: 5 unidades
Se ha actualizado el inventario, Café con leche: 9 unidades restantes.
Producto añadido: Café con leche
Producto Café helado no disponible debido a falta de stock.
Se ha actualizado el inventario, Tarta de manzana: 4 unidades restantes.
Producto añadido: Tarta de manzana
Pedido realizado por Alexander: Pedido de Alexander: Café con leche, Tarta de manzana - Total: 5.5 - Estado: Pendiente
Descuento aplicado: 0.55. Total final con descuento: 4.95.
Pedido confirmado para Alexander: En preparación
Total del pedido: 4.95
Pedido entregado a Alexander: Entregado
```

DIAGRAMA UML SEGUNDO SISTEMA



Mejor visualización:





3. Sistema Biblioteca Digital

JUSTIFICACIÓN DE LA NECESIDAD DEL TERCER PROGRAMA:

El programa es esencial para gestionar de manera eficiente los préstamos, devoluciones y penalizaciones en una biblioteca. Automatiza el proceso de préstamo y control de materiales, mejorando la organización y reduciendo errores. Además, permite aplicar penalizaciones por retrasos de forma automática, asegurando el cumplimiento de las normativas. También facilita la consulta del catálogo por parte de los usuarios y la gestión de materiales entre sucursales, optimizando la experiencia tanto para los usuarios como para los bibliotecarios.

CLASES A OCUPAR, ATRIBUTOS Y MÉTODOS NECESARIOS:

1. Clase: Material

Atributos:

- titulo: Título del material.
- estado: Estado del material (por defecto, "disponible").

Métodos:

- __init__(self, titulo, estado='disponible'): Constructor para inicializar título y estado.
- cambiar_estado(self, estado): Cambia el estado del material a "disponible" o "prestado".

2. Clase: Libro (Subclase de Material)

Atributos:

- autor: Autor del libro.
- genero: Género del libro.

Métodos:

- __init__(self, titulo, autor, genero, estado='disponible'): Constructor para inicializar título, autor, género y estado.

3. Clase: Revista (Subclase de Material)

Atributos:

- edicion: Edición de la revista.
- periodicidad: Periodicidad de la revista (mensual, semanal, etc.).

Métodos:

- __init__(self, titulo, edicion, periodicidad, estado='disponible'): Constructor para inicializar título, edición, periodicidad y estado.

4. Clase: MaterialDigital (Subclase de Material)

Atributos:

- tipo_archivo: Tipo de archivo digital (PDF, EPUB, etc.).
- enlace_descarga: Enlace de descarga del material digital.

Métodos:

- __init__(self, titulo, tipo_archivo, enlace_descarga, estado='disponible'): Constructor para inicializar título, tipo de archivo, enlace de descarga y estado.

5. Clase: Persona

Atributos:

- nombre: Nombre de la persona.
- correo: Correo electrónico de la persona.

Métodos:

- __init__(self, nombre, correo): Constructor para inicializar nombre y correo.

6. Clase: Usuario (Subclase de Persona)

Atributos:

- materiales_prestados: Lista de materiales prestados por el usuario.

Métodos:

- __init__(self, nombre, correo): Constructor que inicializa nombre, correo y materiales prestados.
- consultar_catalogo(self, catalogo): Permite consultar el catálogo de materiales.
- realizar_prestamo(self, prestamo): Realiza un préstamo de un material.
- devolver_material(self, material): Devuelve un material prestado y cambia su estado a "disponible".

7. Clase: Bibliotecario (Subclase de Persona)

Atributos:

- rol: Rol del bibliotecario (por ejemplo, "Bibliotecario").

Métodos:

- __init__(self, nombre, correo, rol): Constructor que inicializa nombre, correo y rol.
- agregar_material(self, material, sucursal): Agrega materiales al catálogo de una sucursal.
- gestionar_prestamos(self, prestamo): Gestiona los préstamos de materiales.

8. Clase: Sucursal

Atributos:

- nombre: Nombre de la sucursal.
- catalogo: Catálogo de materiales disponibles en la sucursal.

Métodos:

- __init__(self, nombre): Constructor que inicializa nombre y catálogo.
- agregar_material(self, material): Agrega un material al catálogo de la sucursal.
- buscar_material(self, criterio, valor): Busca un material por un criterio (por ejemplo, autor, género).
- mostrar_materiales(self): Muestra todos los materiales disponibles en el catálogo.

9. Clase: Prestamo
Atributos:

- usuario: Usuario que realizó el préstamo.
- material: Material prestado.
- fecha_prestamo: Fecha en que se realizó el préstamo.
- fecha_devolucion: Fecha en que debe devolverse el material.

Métodos:

- __init__(self, usuario, material, fecha_prestamo, fecha_devolucion): Constructor para inicializar usuario, material, fecha de préstamo y fecha de devolución.
- calcular_retraso(self, fecha_actual): Calcula el retraso en días si el material no se devuelve a tiempo.
- es_retrasado(self, fecha_actual): Verifica si el material ha sido devuelto con retraso.

10. Clase: Penalizacion
Atributos:

- usuario: Usuario al que se le aplica la penalización.
- monto: Monto de la penalización.
- estado: Estado de la penalización ("pendiente" o "pagada").

Métodos:

- __init__(self, usuario, monto): Constructor que inicializa usuario, monto y estado.
- aplicar_penalizacion(self, prestamo, fecha_actual): Aplica la penalización si el préstamo está retrasado.
- pagar_penalizacion(self): Marca la penalización como pagada.

11. Clase: Catalogo
Atributos:

- materiales: Lista de materiales disponibles en el catálogo.

Métodos:

- __init__(self): Constructor para inicializar la lista de materiales.
- agregar_material(self, material): Agrega un material al catálogo.
- buscar_por_criterio(self, criterio, valor): Busca materiales por un criterio específico.
- mostrar_materiales(self): Muestra todos los materiales disponibles en el catálogo.

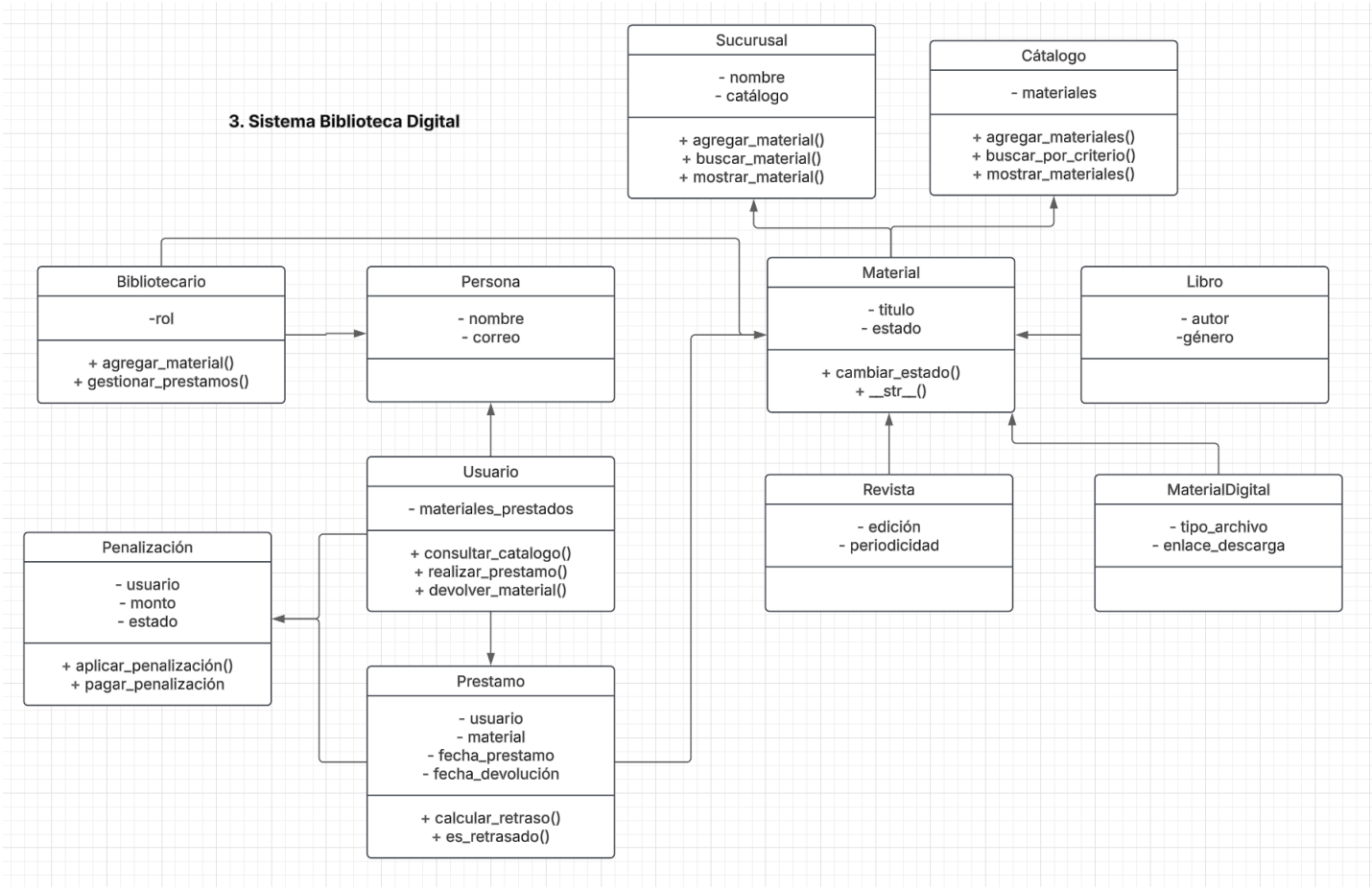
CAPTURA DE PANTALLA DEL CÓDIGO EJECUTADO

```
El Quijote ha sido agregado a la sucursal Sucursal Central.
National Geographic ha sido agregado a la sucursal Sucursal Central.
Python para todos ha sido agregado a la sucursal Sucursal Norte.
Juan Pérez ha realizado un préstamo de El Quijote.
['Libro: El Quijote, Autor: Miguel de Cervantes, Género: Novela, Estado: disponible', 'Revista: National Geographic, Edición: 2023,
Penalización de 5 aplicada a Juan Pérez por retraso.
Estado de El Quijote cambiado a: disponible
Juan Pérez ha devuelto El Quijote.
El Quijote ha sido agregado a la sucursal Sucursal Norte.
Carlos Gómez ha agregado El Quijote a la sucursal.
```

continuacion:

```
2023, Periodicidad: Mensual, Estado: disponible', 'Material Digital: Python para todos, Tipo: PDF, Enlace: http://descargarpython.com, Estado: disponible']
```

DIAGRAMA UML TERCER SISTEMA



LINK DE GITHUB

https://github.com/Alexander117161/Pr-ctica-Unidad-1-Prog.-Avanzada