Informe Explicativo del Código – Sistema de Biblioteca

Presentado por:

JOHAN STEVEN CARDENAS GONZALEZ

Presentado a:

Jesús Ariel González Bonilla

Ingeniería mecatrónica

Semestre III

Corporación universitaria del Huila

06 septiembre 2025

Introducción

Este proyecto consiste en el desarrollo de un sistema de biblioteca digital en consola utilizando Python. Su propósito principal es aprender los fundamentos de la programación orientada a objetos, así como el uso de listas, diccionarios y la interacción con el usuario mediante un menú.

El programa permitirá:

- Crear usuarios.
- Registrar materiales (libros y revistas).
- Prestar y devolver materiales.
- Generar reportes de préstamos.

El enfoque es **pedagógico**: cada parte del código está diseñada para que un estudiante novato entienda la lógica paso a paso.

Paso a paso.

2. Paso 1 – Definir las bases del programa

Antes de escribir el código, debemos pensar qué elementos tendrá una biblioteca:

1. **Usuarios**: personas que piden prestados libros.
2. **Materiales**: libros o revistas que pueden prestarse.
3. **Préstamos**: acciones de dar un material a un usuario.
4. **Reportes**: información sobre lo que está prestado o vencido.

Esto nos lleva a pensar en clases que organicen estos elementos.

3. Paso 2 – Constantes y Excepciones

Código:

```python
# Simulación de constantes
class ItemType:
    BOOK = "Libro"
    MAGAZINE = "Revista"


# Excepciones simuladas
class LoanError(Exception):
    pass
```

EXPLICACION

- ItemType: clase que **simula constantes**. En vez de escribir “Libro” o “Revista” en todas partes, usamos ItemType.BOOK
- LoanError: define un **tipo de error propio** para cuando algo falle como un usuario que no exista

4. Paso 3 – Clase InventoryService (Inventario)

Código

```python
# Servicios simplificados
class InventoryService:
    def __init__(self):
        self.users = {}      # dict con {documento: nombre}
        self.items = {}      # dict con {titulo: {"tipo":..., "stock":...}}

    def create_user(self, name, document_id):
        if document_id in self.users:
            raise ValueError("Documento ya registrado")
        self.users[document_id] = name

    def list_users(self):
        return [(doc, name) for doc, name in self.users.items()]

    def create_item(self, item_type, title, stock):
        if title in self.items:
            raise ValueError("Ya existe un material con ese título")
        self.items[title] = {"tipo": item_type, "stock": stock}

    def list_items(self):
        return [(title, data["tipo"], data["stock"]) for title, data in self.items.items()]
```

Explicación:

- self.users: es un **diccionario** que guarda a los usuarios.
    - Clave = documento ○ Valor = nombre ○

      Ejemplo: {"1010": "Ana"}
- self.items: es otro **diccionario** que guarda materiales.
    - Clave = título ○ Valor = tipo y stock ○

      Ejemplo: {"Python Básico": {"tipo": "Libro",

      "stock": 2}}

Métodos:

- create_user: registra un usuario, el cual no permite que hayan números repetidos
- list_users: devuelve los usuarios que están registrados.
- create_item: agrega un libro/revista con stock inicial.
- list_items: lista los materiales disponibles.

5. Paso 4 – Clase LoanService (Préstamos)

Código

```python
class LoanService:
    def __init__(self, inventory):
        self.inventory = inventory
        self.loans = []  # lista de préstamos activos

    def loan_item(self, document_id, title):
        if document_id not in self.inventory.users:
            raise LoanError("Usuario no existe")
        if title not in self.inventory.items:
            raise LoanError("Material no existe")
        if self.inventory.items[title]["stock"] <= 0:
            raise LoanError("Sin stock disponible")

        self.inventory.items[title]["stock"] -= 1
        self.loans.append({"doc": document_id, "title": title})
        return {"doc": document_id, "title": title}

    def return_item(self, document_id, title):
        found = None
        for loan in self.loans:
            if loan["doc"] == document_id and loan["title"] == title:
                found = loan
                break
        if not found:
            raise LoanError("Préstamo no encontrado")

        self.loans.remove(found)
        self.inventory.items[title]["stock"] += 1
        return 0  # penalización ficticia
```

Explicación:

- inventory: conecta esta clase con los usuarios y materiales ya creados.
- self.loans: lista que guarda préstamos activos.

Métodos:

1. loan_item:
    - Verifica que el usuario exista. ○ Verifica que el material exista. ○ Verifica que haya stock. ○ Si todo está bien:

- Resta 1 al stock.
- Registra el préstamo en la lista.

2. return_item:
    - Busca el préstamo en la lista. ○ Si lo encuentra: lo elimina y devuelve el stock.
    - Retorna una “penalización ficticia” (siempre 0 en este ejemplo).

.

6. Paso 5 – Clase ReportService (Reportes)

Código:

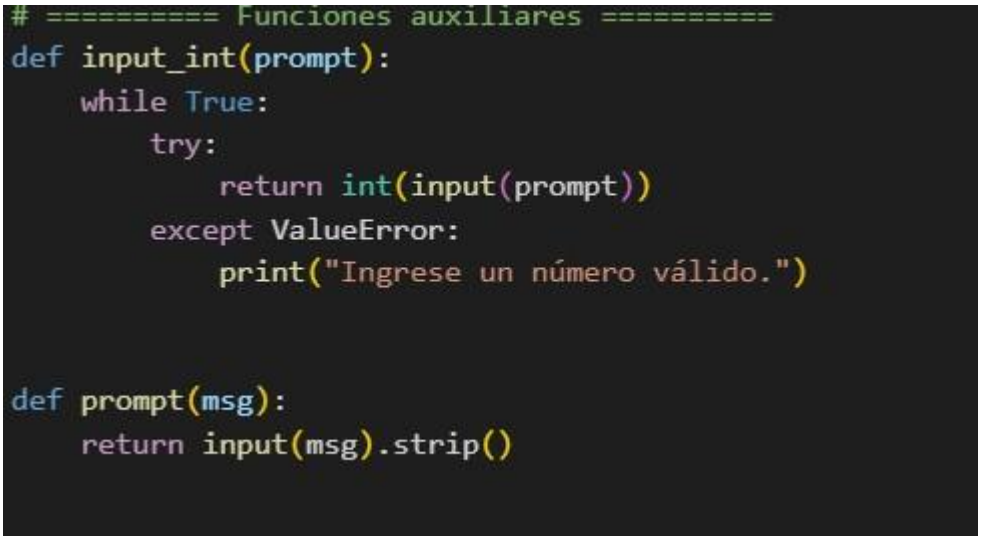
```
class ReportService:
    def __init__(self, loans):
        self.loans = loans

    def active_loans(self):
        return self.loans.loans

    def overdue_loans(self):
        # simplificado: no calculamos fechas, solo simulamos vacío
        return []
```

return [] # simplificado

- active_loans: devuelve todos los préstamos activos.
- overdue_loans: simula los vencidos

7. Paso 6 – Funciones Auxiliares

Código:



```python
# ========== Funciones auxiliares ==========
def input_int(prompt):
    while True:
        try:
            return int(input(prompt))
        except ValueError:
            print("Ingrese un número válido.")


def prompt(msg):
    return input(msg).strip()
```

- input_int: garantiza que el usuario ingrese con un número válido.
- prompt: pide un texto y lo limpia de espacios.

8. Paso 7 – Menú de Opciones

Código (fragmento):

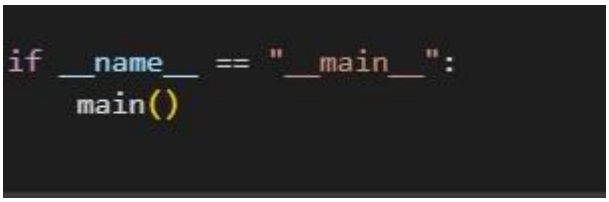
```python
# ========== Acciones del menú ==========
def create_user(inv):
    name = prompt("Nombre: ")
    doc = prompt("Documento (único): ")
    try:
        inv.create_user(name, doc)
        print("✓ Usuario creado.")
    except ValueError as e:
        print(f"✗ Error: {e}")


def list_users(inv):
    users = inv.list_users()
    if not users:
        print("(sin usuarios)")
    for doc, name in users:
        print(f"- {name} (doc: {doc})")


def create_item(inv):
    title = prompt("Título del material: ")
    print("Tipo: 1) Libro  2) Revista")
    choice = prompt("Seleccione tipo: ")
    item_type = ItemType.BOOK if choice == "1" else ItemType.MAGAZINE if choice == "2" else None
    if not item_type:
        print("Tipo inválido.")
        return
    stock = input_int("Stock (>=0): ")
    try:
        inv.create_item(item_type, title, stock)
        print("✓ Material creado.")
    except ValueError as e:
        print(f"✗ Error: {e}")
```

```
    MENU = """
================ Biblioteca Digital (CLI) ================
1. Crear usuario
2. Listar usuarios
3. Crear material (Libro/Revista)
4. Listar materiales
5. Prestar material
6. Devolver material
7. Reporte: Préstamos activos
8. Reporte: Préstamos vencidos
0. Salir
==========================================================
"""
```

Este menú muestra al usuario todas las acciones posibles.

9. Paso 8 – Programa Principal

Código:



```
if __name__ == "__main__":
    main()
```

- Garantiza que el programa se ejecute desde este archivo.
- Dentro de main() se crean los objetos (inventory, loans, reports) y se controla el flujo con el menú.

Aquí el estudiante aprende a estructurar un programa completo.

Anexo del video

https://drive.google.com/file/d/1SNd9eLtclGuf-jb8HPDUmnM399wSpBrZ/view?usp=drive_link