UEA
UNIVERSIDAD
ESTATAL AMAZÓNICA

UNIVERSIDAD ESTATAL AMAZÓNICA

INGENIERÍA EN TECNOLOGÍAS DE INFORMACIÓN

PROGRAMACIÓN ORIENTADA A OBJETOS

TEMA:

SISTEMA DE GESTIÓN DE BIBLIOTECA DIGITAL

ESTUDIANTE:

NORIEGA BALDEON EDWIN FABIÁN

DOCENTE:

ING. GALES GUERRERO SANTIAGO ISRAEL NO

NIVEL:

SEGUNDO NIVEL_PARALELO “A”

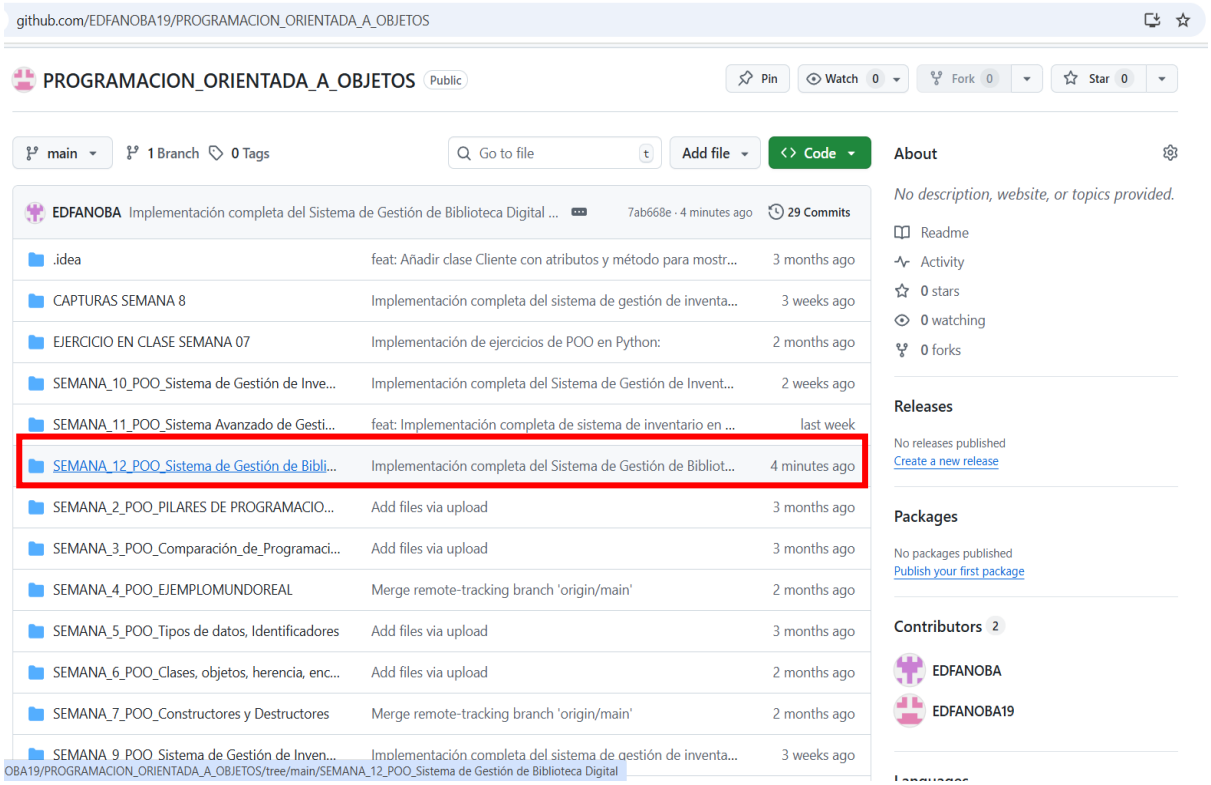
2025-2025

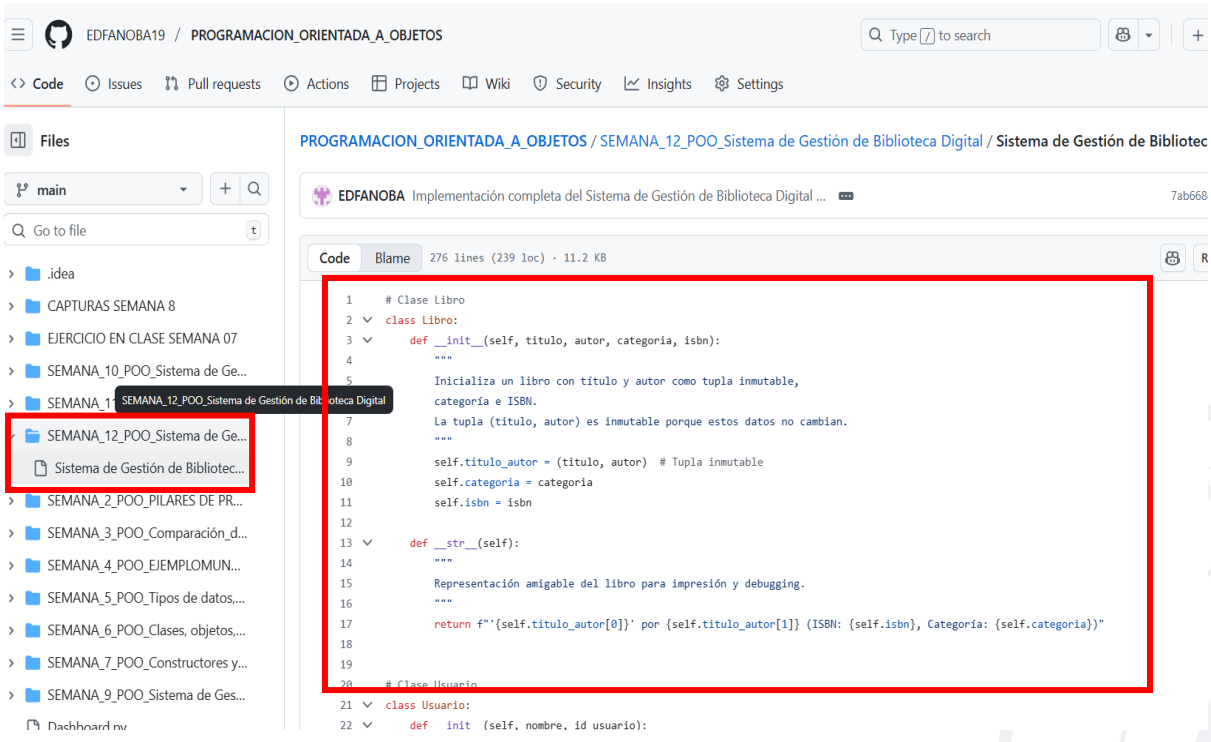
PUYO-ECUADOR

1. CÓDIGO DE GITHUB

https://github.com/EDFANOBA19/PROGRAMACION_ORIENTADA_A_OBJETOS.git

2. REPOSITORIO DE GITHUB





3. DESARROLLO DE CÓDIGO CON SUS DIFERENTES PEDIDOS Y REQUISITOS

3.1. objetivo de la tarea

Desarrollar un sistema para gestionar una biblioteca digital. El sistema permitirá administrar los libros disponibles, las categorías de libros, los usuarios registrados y el historial de préstamos.

3.2. Requisitos:

3.2.1. Clases Principales:

- **Clase Libro:** Representa un libro con atributos como título, autor, categoría y ISBN. Utiliza una tupla para almacenar el autor y el título, ya que estos no cambiarán una vez creados.

Sistema de Gestión de Biblioteca Digital.py

```python
# Clase Libro
class Libro:  3 usages  new *
    def __init__(self, titulo, autor, categoria, isbn):  new *
        """
        Inicializa un libro con título y autor como tupla inmutable,
        categoría e ISBN.
        La tupla (titulo, autor) es inmutable porque estos datos no cambian.
        """
        self.titulo_autor = (titulo, autor)  # Tupla inmutable
        self.categoria = categoria
        self.isbn = isbn

    def __str__(self):  new *
        """
        Representación amigable del libro para impresión y debugging.
        """
        return f"'{self.titulo_autor[0]}' por {self.titulo_autor[1]} (ISBN: {self.isbn}, Categoría: {self.categoria})"
```

- **Clase Usuario:** Representa a un usuario de la biblioteca con atributos como nombre, ID de usuario (único) y una lista de libros actualmente prestados.

Sistema de Gestión de Biblioteca Digital.py

```python
# Clase Usuario
class Usuario:  2 usages  new *
    def __init__(self, nombre, id_usuario):  new *
        """
        Inicializa un usuario con un nombre, un ID único, y una lista vacía para los libros prestados.
        """
        self.nombre = nombre
        self.id_usuario = id_usuario
        self.libros_prestados = []  # Lista que guarda objetos Libro prestados

    def listar_libros_prestados(self):  1 usage (1 dynamic)  new *
        """
        Devuelve una lista con la representación en string de los libros prestados.
        Si no hay libros prestados, notifica que el usuario no tiene ninguno.
        """
        if not self.libros_prestados:
            return f"{self.nombre} no tiene libros prestados."
        return [str(libro) for libro in self.libros_prestados]

    def __str__(self):  new *
        """
        Representación amigable del usuario para impresión y debugging.
        """
        return f"Usuario: {self.nombre}, ID: {self.id_usuario}"
```

- **Clase Biblioteca:** Gestiona las colecciones de libros, usuarios y préstamos. Utiliza un diccionario para almacenar los libros disponibles, con el ISBN como clave y el objeto Libro como valor, para búsquedas eficientes. Usa un conjunto para manejar los IDs de usuarios únicos.

Sistema de Gestión de Biblioteca Digital.py

```
# Clase Biblioteca
class Biblioteca:  1 usage  new *
    def __init__(self):  new *
        """
        Inicializa la biblioteca con:
        - Un diccionario libros donde clave es ISBN y valor es el objeto Libro
        - Un conjunto usuarios_ids para asegurar unicidad de IDs
        - Un diccionario usuarios con clave ID y valor objeto Usuario
        """
        self.libros = {}         # Diccionario para almacenar libros {isbn: Libro}
        self.usuarios_ids = set()  # Conjunto para IDs únicos de usuarios
        self.usuarios = {}       # Diccionario para usuarios {id_usuario: Usuario}
```

3.2.2. Funcionalidades:

- **Añadir/quitar libros:** Permitir añadir o quitar libros de la biblioteca.

```
    # Gestión de libros
    def añadir_libro(self, libro):  4 usages  new *
        """
        Añade un libro al diccionario de libros si no existe ya.
        Retorna mensaje informativo.
        """
        if libro.isbn in self.libros:
            return f"El libro con ISBN {libro.isbn} ya existe."
        self.libros[libro.isbn] = libro
        return f"Libro '{libro.titulo_autor[0]}' añadido a la biblioteca."

    def quitar_libro(self, isbn):  2 usages  new *
        """
        Quita un libro identificándolo por ISBN.
        Solo permite quitarlo si no está prestado a algún usuario.
        """
        if isbn not in self.libros:
            return f"No existe un libro con ISBN {isbn}."
        # Verificar si libro está prestado
        for usuario in self.usuarios.values():
            if any(libro.isbn == isbn for libro in usuario.libros_prestados):
                return f"El libro con ISBN {isbn} está actualmente prestado y no puede ser eliminado."
        del self.libros[isbn]
        return f"Libro con ISBN {isbn} eliminado de la biblioteca."
```

- **Registrar/dar de baja usuarios:** Permitir el registro de nuevos usuarios y la baja de usuarios existentes.

```python
    # Gestión de usuarios
    def registrar_usuario(self, usuario):  3 usages  new *
        """
        Registra un usuario si su ID no está ya en uso.
        """
        if usuario.id_usuario in self.usuarios_ids:
            return f"El ID de usuario {usuario.id_usuario} ya está registrado."
        self.usuarios_ids.add(usuario.id_usuario)
        self.usuarios[usuario.id_usuario] = usuario
        return f"Usuario '{usuario.nombre}' registrado con ID {usuario.id_usuario}."

    def dar_baja_usuario(self, id_usuario):  2 usages  new *
        """
        Da de baja a un usuario si existe y no tiene libros prestados.
        """
        if id_usuario not in self.usuarios_ids:
            return f"No existe un usuario con ID {id_usuario}."
        usuario = self.usuarios[id_usuario]
        if usuario.libros_prestados:
            return f"El usuario tiene libros prestados y no puede ser dado de baja."
        self.usuarios_ids.remove(id_usuario)
        del self.usuarios[id_usuario]
        return f"Usuario con ID {id_usuario} dado de baja."
```

- **Prestar/devolver libros:** Facilitar el préstamo de libros a usuarios y la devolución de los mismos.

```python
    # Préstamos y devoluciones
    def prestar_libro(self, id_usuario, isbn):  3 usages  new *
        """
        Presta un libro identificado por ISBN a un usuario identificado por ID.
        Verifica que el usuario y libro existan y que el libro no esté actualmente prestado.
        """
        if id_usuario not in self.usuarios_ids:
            return f"No existe un usuario con ID {id_usuario}."
        if isbn not in self.libros:
            return f"No existe un libro con ISBN {isbn} en la biblioteca."

        usuario = self.usuarios[id_usuario]
        libro = self.libros[isbn]

        # Verificar si el libro ya está prestado a cualquier usuario
        for u in self.usuarios.values():
            if libro in u.libros_prestados:
                return f"El libro con ISBN {isbn} ya está prestado a otro usuario."

        usuario.libros_prestados.append(libro)
        return f"Libro '{libro.titulo_autor[0]}' prestado a usuario {usuario.nombre}."
```

```python
    def devolver_libro(self, id_usuario, isbn):  3 usages  new *
        """
        Permite la devolución de un libro por parte de un usuario.
        Verifica que el usuario tenga ese libro prestado.
        """
        if id_usuario not in self.usuarios_ids:
            return f"No existe un usuario con ID {id_usuario}."

        usuario = self.usuarios[id_usuario]

        for libro in usuario.libros_prestados:
            if libro.isbn == isbn:
                usuario.libros_prestados.remove(libro)
                return f"Libro '{libro.titulo_autor[0]}' devuelto por usuario {usuario.nombre}."
        return f"El usuario no tiene prestado el libro con ISBN {isbn}."
```

- **Buscar libros:** Implementar búsquedas de libros por título, autor o categoría.

```python
    # Métodos de búsqueda
    def buscar_por_titulo(self, titulo):  1 usage  new *
        """
        Retorna una lista de libros cuyo título contenga el texto buscado (no sensible a mayúsculas).
        """
        resultados = [libro for libro in self.libros.values() if titulo.lower() in libro.titulo_autor[0].lower()]
        if not resultados:
            return "No se encontraron libros con ese título."
        return [str(libro) for libro in resultados]

    def buscar_por_autor(self, autor):  1 usage  new *
        """
        Retorna una lista de libros cuyo autor contenga el texto buscado (no sensible a mayúsculas).
        """
        resultados = [libro for libro in self.libros.values() if autor.lower() in libro.titulo_autor[1].lower()]
        if not resultados:
            return "No se encontraron libros con ese autor."
        return [str(libro) for libro in resultados]

    def buscar_por_categoria(self, categoria):  1 usage  new *
        """
        Retorna una lista de libros cuya categoría coincida exactamente (no sensible a mayúsculas).
        """
        resultados = [libro for libro in self.libros.values() if libro.categoria.lower() == categoria.lower()]
        if not resultados:
            return "No se encontraron libros en esa categoría."
        return [str(libro) for libro in resultados]
```

- **Listar libros prestados:** Mostrar una lista de todos los libros actualmente prestados a un usuario.

```python
    # Listar libros prestados de un usuario
    def listar_libros_prestados_usuario(self, id_usuario):  1 usage  new *
        """
        Muestra los libros que tiene prestados un usuario dado su ID, o mensaje si no tiene.
        """
        if id_usuario not in self.usuarios_ids:
            return f"No existe un usuario con ID {id_usuario}."
        usuario = self.usuarios[id_usuario]
        return usuario.listar_libros_prestados()

```

3.2.3. Implementación:

- Utiliza listas para gestionar los libros prestados a cada usuario.

```python
# Clase Usuario
class Usuario:  2 usages  new *
    def __init__(self, nombre, id_usuario):  new *
        """
        Inicializa un usuario con un nombre, un ID único, y una lista vacía para los libros prestados
        """
        self.nombre = nombre
        self.id_usuario = id_usuario
        self.libros_prestados = []  # Lista que guarda objetos Libro prestados

    def listar_libros_prestados(self):  1 usage (1 dynamic)  new *
        """
        Devuelve una lista con la representación en string de los libros prestados.
        Si no hay libros prestados, notifica que el usuario no tiene ninguno.
        """
        if not self.libros_prestados:
            return f"{self.nombre} no tiene libros prestados."
        return [str(libro) for libro in self.libros_prestados]

    def __str__(self):  new *
        """
        Representación amigable del usuario para impresión y debugging.
        """
        return f"Usuario: {self.nombre}, ID: {self.id_usuario}"
```

- Emplea tuplas para los atributos inmutables de los libros.

```python
# Clase Libro
class Libro:  3 usages  new *
    def __init__(self, titulo, autor, categoria, isbn):  new *
        """
        Inicializa un libro con título y autor como tupla inmutable,
        categoría e ISBN.
        La tupla (titulo, autor) es inmutable porque estos datos no cambian.
        """
        self.titulo_autor = (titulo, autor)  # Tupla inmutable
        self.categoria = categoria
        self.isbn = isbn
```

- Usa diccionarios para almacenar y acceder eficientemente a los libros por ISBN.

```python
# Clase Biblioteca
class Biblioteca:  1 usage  new *
    def __init__(self):  new *
        """
        Inicializa la biblioteca con:
        - Un diccionario libros donde clave es ISBN y valor es el objeto Libro
        - Un conjunto usuarios_ids para asegurar unicidad de IDs
        - Un diccionario usuarios con clave ID y valor objeto Usuario
        """
        self.libros = {}        # Diccionario para almacenar libros {isbn: Libro}
        self.usuarios_ids = set()  # Conjunto para IDs únicos de usuarios
        self.usuarios = {}      # Diccionario para usuarios {id_usuario: Usuario}
```

- Aplica conjuntos para asegurar IDs de usuario únicos y gestionar los usuarios registrados.

```python
# Gestión de usuarios
def registrar_usuario(self, usuario):  3 usages  new *
    """
    Registra un usuario si su ID no está ya en uso.
    """
    if usuario.id_usuario in self.usuarios_ids:
        return f"El ID de usuario {usuario.id_usuario} ya está registrado."
    self.usuarios_ids.add(usuario.id_usuario)
    self.usuarios[usuario.id_usuario] = usuario
    return f"Usuario '{usuario.nombre}' registrado con ID {usuario.id_usuario}."
```

3.2.4. Instrucciones:

- Prueba tu sistema creando objetos de cada clase y ejecutando varias operaciones del sistema de biblioteca:

Añadir libros

Código:

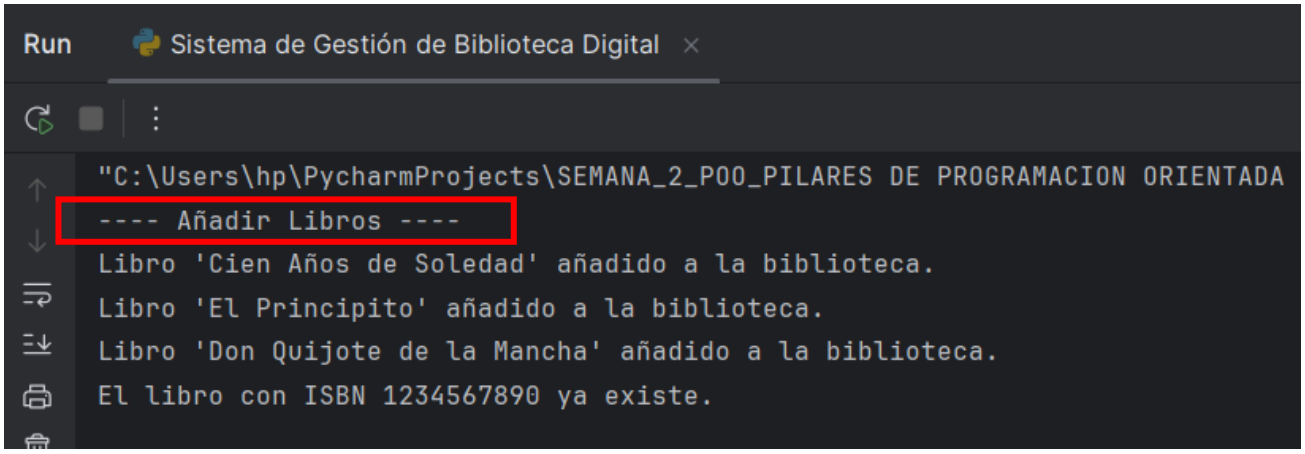
```python
# Pruebas del Sistema de Biblioteca con salidas claras

# Crear la instancia de la biblioteca
biblioteca = Biblioteca()

# Crear algunos libros (título, autor, categoría, ISBN)
libro1 = Libro( titulo: "Cien Años de Soledad",  autor: "Gabriel García Márquez",  categoria: "Novela",  isbn: "1234567890")
libro2 = Libro( titulo: "El Principito",  autor: "Antoine de Saint-Exupéry",  categoria: "Infantil",  isbn: "1234567891")
libro3 = Libro( titulo: "Don Quijote de la Mancha",  autor: "Miguel de Cervantes",  categoria: "Novela",  isbn: "1234567892")

# Añadir libros a la biblioteca
print("---- Añadir Libros ----")
print(biblioteca.añadir_libro(libro1))
print(biblioteca.añadir_libro(libro2))
print(biblioteca.añadir_libro(libro3))
print(biblioteca.añadir_libro(libro1))  # Intento de añadir libro duplicado
print()
```

Consola:

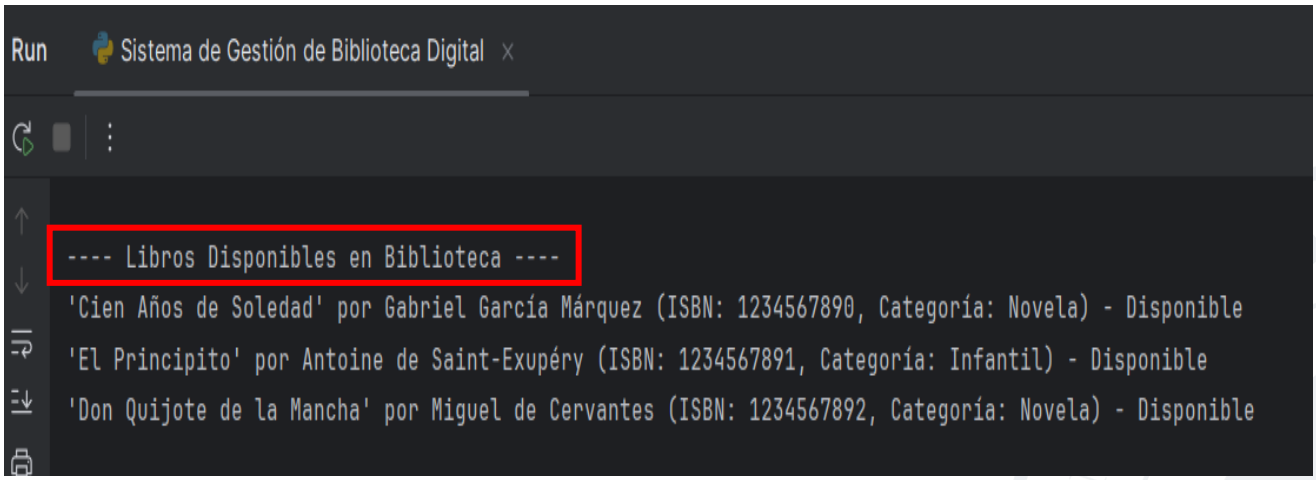


Mostrar libros disponibles

Código:

```
202  # Mostrar todos los libros disponibles (estado)
203  print("---- Libros Disponibles en Biblioteca ----")
204  for libro in biblioteca.libros.values():
205      # Verifica si el libro está prestado a algún usuario
206      prestado = any(libro in u.libros_prestados for u in biblioteca.usuarios.values())
207      estado = "Prestado" if prestado else "Disponible"
208      print(f"{libro} - {estado}")
209  print()
210
```

Consola:



Como registrar usuarios

Código:

```
211  # Registrar usuarios
212  print("---- Registrar Usuarios ----")
213  usuario1 = Usuario( nombre: "Ana Pérez",  id_usuario: "user001")
214  usuario2 = Usuario( nombre: "Juan López",  id_usuario: "user002")
215
216  print(biblioteca.registrar_usuario(usuario1))
217  print(biblioteca.registrar_usuario(usuario2))
218  print(biblioteca.registrar_usuario(usuario1))  # Intento con ID repetido
219  print()
```

Consola:

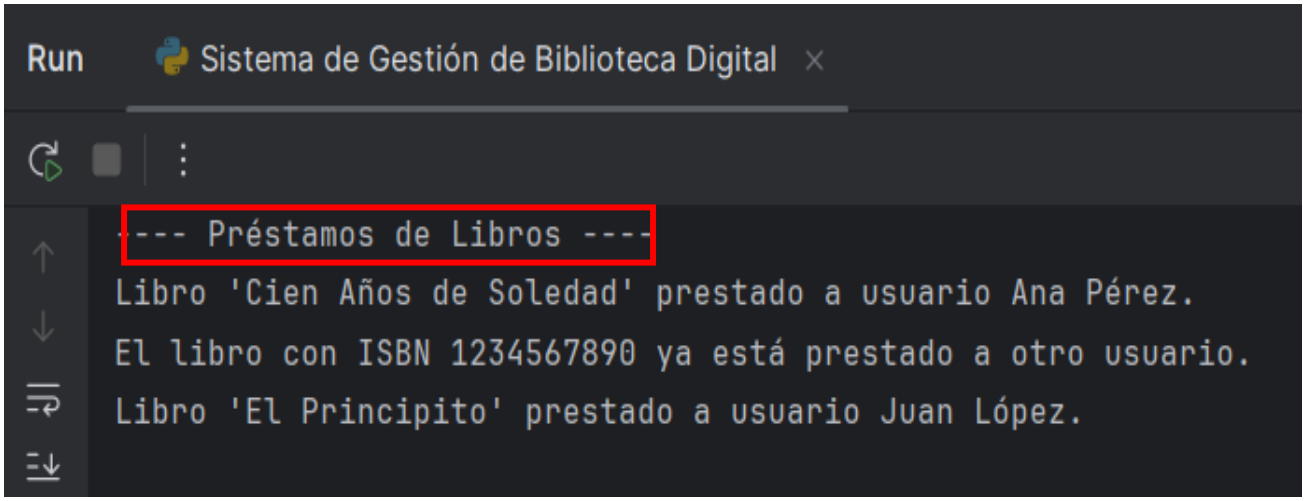
```
Run    Sistema de Gestión de Biblioteca Digital
---- Registrar Usuarios ----
Usuario 'Ana Pérez' registrado con ID user001.
Usuario 'Juan López' registrado con ID user002.
El ID de usuario user001 ya está registrado.
```

Prestar libros

Código:

```
221  # Prestar libros a usuarios
222  print("---- Préstamos de Libros ----")
223  print(biblioteca.prestar_libro( id_usuario: "user001",  isbn: "1234567890"))  # Ana presta Cien Años de Soledad
224  print(biblioteca.prestar_libro( id_usuario: "user002",  isbn: "1234567890"))  # Juan intenta préstamo ya ocupado
225  print(biblioteca.prestar_libro( id_usuario: "user002",  isbn: "1234567891"))  # Juan presta El Principito
226  print()
```

Consola:

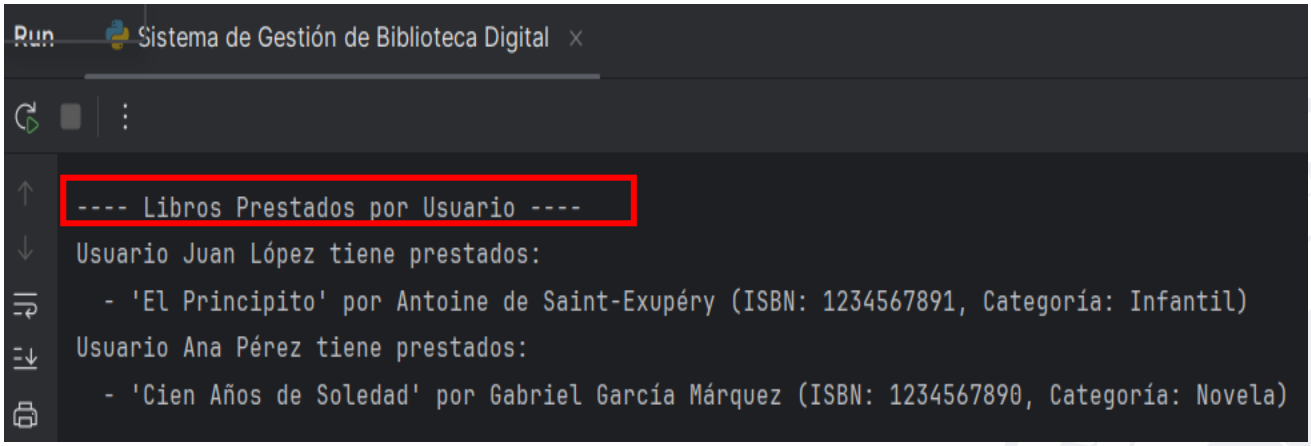


Mostrar libros prestados

Código:

```
228  # Mostrar libros prestados por usuario
229  print("---- Libros Prestados por Usuario ----")
230  for id_user in biblioteca.usuarios_ids:
231      libros = biblioteca.listar_libros_prestados_usuario(id_user)
232      if isinstance(libros, list):
233          print(f"Usuario {biblioteca.usuarios[id_user].nombre} tiene prestados:")
234          for l in libros:
235              print("  -", l)
236      else:
237          print(libros)
238  print()
```

Consola:



Buscar en el catálogo

Código:

```
# Búsquedas en la biblioteca
print("---- Búsqueda de Libros ----")
print("Buscar por título 'Don':", biblioteca.buscar_por_titulo("Don"))
print("Buscar por autor 'Gabriel':", biblioteca.buscar_por_autor("Gabriel"))
print("Buscar por categoría 'Novela':", biblioteca.buscar_por_categoria("Novela"))
print()
```

Consola:

```
Sistema de Gestión de Biblioteca Digital

---- Búsqueda de Libros ----
Buscar por título 'Don': ["'Don Quijote de la Mancha' por Miguel de Cervantes (ISBN: 1234567892, Categoría: Novela)"]
Buscar por autor 'Gabriel': ["'Cien Años de Soledad' por Gabriel García Márquez (ISBN: 1234567890, Categoría: Novela)"]
Buscar por categoría 'Novela': ["'Cien Años de Soledad' por Gabriel García Márquez (ISBN: 1234567890, Categoría: Novela)",
```

Devolver libros

Código:

```
# Devolver libros
print("---- Devolución de Libros ----")
print(biblioteca.devolver_libro( id_usuario: "user001",  isbn: "1234567890"))  # Ana devuelve su libro
print(biblioteca.devolver_libro( id_usuario: "user002",  isbn: "1234567890"))  # Juan intenta devolver libro no prestado
print()
```

Consola:

```
Run  Sistema de Gestión de Biblioteca Digital

---- Devolución de Libros ----
Libro 'Cien Años de Soledad' devuelto por usuario Ana Pérez.
El usuario no tiene prestado el libro con ISBN 1234567890.
```

Dar de baja usuarios

Código:

```
253  # Dar de baja usuarios
254  print("---- Dar de Baja Usuarios ----")
255  print(biblioteca.dar_baja_usuario("user001"))  # Ana se puede dar de baja sin libros prestados
256  print(biblioteca.dar_baja_usuario("user002"))  # Juan aún tiene libro prestado, no puede darse de baja
257  print()
```

Consola:

```
un    Sistema de Gestión de Biblioteca Digital  ×

---- Dar de Baja Usuarios ----
Usuario con ID user001 dado de baja.
El usuario tiene libros prestados y no puede ser dado de baja.
```

Intentar quitar libros prestados y no prestados

Código:

```
259  # Intentar quitar libros prestados y no prestados
260  print("---- Quitar Libros ----")
261  print(biblioteca.quitar_libro("1234567891"))  # El Principito está prestado a Juan, no se elimina
262  print(biblioteca.devolver_libro( id_usuario: "user002",  isbn: "1234567891"))  # Juan devuelve el libro
263  print(biblioteca.quitar_libro("1234567891"))  # Ahora sí se puede eliminar
264  print()
```

Consola:

```
Run    Sistema de Gestión de Biblioteca Digital  ×

---- Quitar Libros ----
El libro con ISBN 1234567891 está actualmente prestado y no puede ser eliminado.
Libro 'El Principito' devuelto por usuario Juan López.
Libro con ISBN 1234567891 eliminado de la biblioteca.
```

Estado final de libros y usuarios

Código:

```
# Estado final de libros y usuarios
print("---- Estado Final de Libros en Biblioteca ----")
for libro in biblioteca.libros.values():
    prestado = any(libro in u.libros_prestados for u in biblioteca.usuarios.values())
    estado = "Prestado" if prestado else "Disponible"
    print(f"{libro} - {estado}")
print()
```

Consola:

```
Run  Sistema de Gestión de Biblioteca Digital
---- Estado Final de Libros en Biblioteca ----
'Cien Años de Soledad' por Gabriel García Márquez (ISBN: 1234567890, Categoría: Novela) - Disponible
'Don Quijote de la Mancha' por Miguel de Cervantes (ISBN: 1234567892, Categoría: Novela) - Disponible
```

Usuarios registrados actualmente

Código:

```
print("---- Usuarios Registrados Actualmente ----")
for usuario in biblioteca.usuarios.values():
    print(f"{usuario} con libros prestados: {len(usuario.libros_prestados)}")
```

Consola:

```
un  Sistema de Gestión de Biblioteca Digital
---- Usuarios Registrados Actualmente ----
Usuario: Juan López, ID: user002 con libros prestados: 0

Process finished with exit code 0
```