



# 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**



# UEA UNIVERSIDAD ESTATAL AMAZÓNICA

## 1. CÓDIGO DE GITHUB

[https://github.com/EDFANOBA19/PROGRAMACION\\_ORIENTADA\\_A\\_OBJETOS](https://github.com/EDFANOBA19/PROGRAMACION_ORIENTADA_A_OBJETOS)

[.git](#)

## 2. REPOSITORIO DE GITHUB

github.com/EDFANOBA19/PROGRAMACION\_ORIENTADA\_A\_OBJETOS

PROGRAMACION\_ORIENTADA\_A\_OBJETOS Public

main 1 Branch 0 Tags

Go to file Add file Code

EDFANOBA Implementación completa del Sistema de Gestión de Biblioteca Digital ... 7ab668e · 4 minutes ago 29 Commits

.idea	feat: Añadir clase Cliente con atributos y método para mostr...	3 months ago
CAPTURAS SEMANA 8	Implementación completa del sistema de gestión de inventa...	3 weeks ago
EJERCICIO EN CLASE SEMANA 07	Implementación de ejercicios de POO en Python:	2 months ago
SEMANA_10_POO_Sistema de Gestión de Inve...	Implementación completa del Sistema de Gestión de Invent...	2 weeks ago
SEMANA_11_POO_Sistema Avanzado de Gesti...	feat: Implementación completa de sistema de inventario en ...	last week
SEMANA_12_POO_Sistema de Gestión de Bibli...	Implementación completa del Sistema de Gestión de Bibliot...	4 minutes ago
SEMANA_2_POO_PILARES DE PROGRAMACIO...	Add files via upload	3 months ago
SEMANA_3_POO_Comparación de Programaci...	Add files via upload	3 months ago
SEMANA_4_POO_EJEMPLOMUNDOREAL	Merge remote-tracking branch 'origin/main'	2 months ago
SEMANA_5_POO_Tipos de datos, Identificadores	Add files via upload	3 months ago
SEMANA_6_POO_Clases, objetos, herencia, enc...	Add files via upload	2 months ago
SEMANA_7_POO_Constructores y Destructores	Merge remote-tracking branch 'origin/main'	2 months ago
SEMANA_9_POO_Sistema de Gestión de Inven...	Implementación completa del sistema de gestión de inventa...	3 weeks ago

EDFANOBA19/PROGRAMACION\_ORIENTADA\_A\_OBJETOS/tree/main/SEMANA\_12\_POO\_Sistema de Gestión de Biblioteca Digital

About

No description, website, or topics provided.

Readme Activity 0 stars 0 watching 0 forks

Releases

No releases published [Create a new release](#)

Packages

No packages published [Publish your first package](#)

Contributors 2

EDFANOBA EDFANOBA19

EDFANOBA19 / PROGRAMACION\_ORIENTADA\_A\_OBJETOS

Type to search

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

Files

main

Go to file

SEMANA\_12\_POO\_Sistema de Gestión de Biblioteca Digital

Sistema de Gestión de Bibliotec...

PROGRAMACION\_ORIENTADA\_A\_OBJETOS / SEMANA\_12\_POO\_Sistema de Gestión de Biblioteca Digital / Sistema de Gestión de Bibliotec

EDFANOBA Implementación completa del Sistema de Gestión de Biblioteca Digital ... 7ab668

Code Blame 276 lines (239 loc) · 11.2 KB

```
1 # Clase Libro
2 class Libro:
3     def __init__(self, titulo, autor, categoria, isbn):
4         """
5         Inicializa un libro con titulo y autor como tupla inmutable,
6         categoria e ISBN.
7         La tupla (titulo, autor) es inmutable porque estos datos no cambian.
8         """
9         self.titulo_autor = (titulo, autor) # Tupla inmutable
10        self.categoria = categoria
11        self.isbn = isbn
12
13    def __str__(self):
14        """
15        Representación amigable del libro para impresión y debugging.
16        """
17        return f'{self.titulo_autor[0]} por {self.titulo_autor[1]} (ISBN: {self.isbn}, Categoría: {self.categoria})'
18
19
20 # Clase Usuario
21 class Usuario:
22     def __init__(self, nombre, id_usuario):
```



### 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 x
1  # Clase Libro
2  class Libro: 3 usages new*
3      def __init__(self, titulo, autor, categoria, isbn): new*
4          """
5          Inicializa un libro con titulo y autor como tupla inmutable,
6          categoría e ISBN.
7          La tupla (titulo, autor) es inmutable porque estos datos no cambian.
8          """
9          self.titulo_autor = (titulo, autor) # Tupla inmutable
10         self.categoria = categoria
11         self.isbn = isbn
12
13     def __str__(self): new*
14         """
15         Representación amigable del libro para impresión y debugging.
16         """
17         return f'{self.titulo_autor[0]} por {self.titulo_autor[1]} (ISBN: {self.isbn}, Categoría: {self.categoria})'
18
19
```



# UEA

## UNIVERSIDAD ESTATAL AMAZÓNICA

- **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 x
19
20 # Clase Usuario
21 class Usuario: 2 usages new *
22     def __init__(self, nombre, id_usuario): new *
23         """
24         Inicializa un usuario con un nombre, un ID único, y una lista vacía para los libros prestados.
25         """
26         self.nombre = nombre
27         self.id_usuario = id_usuario
28         self.libros_prestados = [] # Lista que guarda objetos Libro prestados
29
30     def listar_libros_prestados(self): 1 usage (1 dynamic) new *
31         """
32         Devuelve una lista con la representación en string de los libros prestados.
33         Si no hay libros prestados, notifica que el usuario no tiene ninguno.
34         """
35         if not self.libros_prestados:
36             return f"{self.nombre} no tiene libros prestados."
37         return [str(libro) for libro in self.libros_prestados]
38
39     def __str__(self): new *
40         """
41         Representación amigable del usuario para impresión y debugging.
42         """
43         return f"Usuario: {self.nombre}, ID: {self.id_usuario}"
44
```



# UEA

## UNIVERSIDAD ESTATAL AMAZÓNICA

- **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 x
46 # Clase Biblioteca
47 class Biblioteca:
48     def __init__(self):
49         """
50         Inicializa la biblioteca con:
51         - Un diccionario libros donde clave es ISBN y valor es el objeto Libro
52         - Un conjunto usuarios_ids para asegurar unicidad de IDs
53         - Un diccionario usuarios con clave ID y valor objeto Usuario
54         """
55         self.libros = {} # Diccionario para almacenar libros {isbn: Libro}
56         self.usuarios_ids = set() # Conjunto para IDs únicos de usuarios
57         self.usuarios = {} # Diccionario para usuarios {id_usuario: Usuario}
58
```

### 3.2.2. Funcionalidades:

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

```
58
59 # Gestión de libros
60 def añadir_libro(self, libro):
61     """
62     Añade un libro al diccionario de libros si no existe ya.
63     Retorna mensaje informativo.
64     """
65     if libro.isbn in self.libros:
66         return f"El libro con ISBN {libro.isbn} ya existe."
67     self.libros[libro.isbn] = libro
68     return f"Libro '{libro.titulo_autor[0]}' añadido a la biblioteca."
69
70 def quitar_libro(self, isbn):
71     """
72     Quita un libro identificándolo por ISBN.
73     Solo permite quitarlo si no está prestado a algún usuario.
74     """
75     if isbn not in self.libros:
76         return f"No existe un libro con ISBN {isbn}."
77     # Verificar si libro está prestado
78     for usuario in self.usuarios.values():
79         if any(libro.isbn == isbn for libro in usuario.libros_prestados):
80             return f"El libro con ISBN {isbn} está actualmente prestado y no puede ser eliminado."
81     del self.libros[isbn]
82     return f"Libro con ISBN {isbn} eliminado de la biblioteca."
83
```



# UEA

## UNIVERSIDAD ESTATAL AMAZÓNICA

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

```
84 # Gestión de usuarios
85 def registrar_usuario(self, usuario): 3 usages new *
86     """
87     Registra un usuario si su ID no está ya en uso.
88     """
89     if usuario.id_usuario in self.usuarios_ids:
90         return f"El ID de usuario {usuario.id_usuario} ya está registrado."
91     self.usuarios_ids.add(usuario.id_usuario)
92     self.usuarios[usuario.id_usuario] = usuario
93     return f"Usuario '{usuario.nombre}' registrado con ID {usuario.id_usuario}."
94
95 def dar_baja_usuario(self, id_usuario) 2 usages new *
96     """
97     Da de baja a un usuario si existe y no tiene libros prestados.
98     """
99     if id_usuario not in self.usuarios_ids:
100         return f"No existe un usuario con ID {id_usuario}."
101     usuario = self.usuarios[id_usuario]
102     if usuario.libros_prestados:
103         return f"El usuario tiene libros prestados y no puede ser dado de baja."
104     self.usuarios_ids.remove(id_usuario)
105     del self.usuarios[id_usuario]
106     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.

```
108 # Préstamos y devoluciones
109 def prestar_libro(self, id_usuario, isbn): 3 usages new *
110     """
111     Presta un libro identificado por ISBN a un usuario identificado por ID.
112     Verifica que el usuario y libro existan y que el libro no esté actualmente prestado.
113     """
114     if id_usuario not in self.usuarios_ids:
115         return f"No existe un usuario con ID {id_usuario}."
116     if isbn not in self.libros:
117         return f"No existe un libro con ISBN {isbn} en la biblioteca."
118
119     usuario = self.usuarios[id_usuario]
120     libro = self.libros[isbn]
121
122     # Verificar si el libro ya está prestado a cualquier usuario
123     for u in self.usuarios.values():
124         if libro in u.libros_prestados:
125             return f"El libro con ISBN {isbn} ya está prestado a otro usuario."
126
127     usuario.libros_prestados.append(libro)
128     return f"Libro '{libro.titulo_autor[0]}' prestado a usuario {usuario.nombre}."
129
```





```
130 def devolver_libro(self, id_usuario, isbn): 3 usages new*
131 """
132     Permite la devolución de un libro por parte de un usuario.
133     Verifica que el usuario tenga ese libro prestado.
134     """
135     if id_usuario not in self.usuarios_ids:
136         return f"No existe un usuario con ID {id_usuario}."
137
138     usuario = self.usuarios[id_usuario]
139
140     for libro in usuario.libros_prestados:
141         if libro.isbn == isbn:
142             usuario.libros_prestados.remove(libro)
143             return f"Libro '{libro.titulo_autor[0]}' devuelto por usuario {usuario.nombre}."
144     return f"El usuario no tiene prestado el libro con ISBN {isbn}."
```

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

```
146 # Métodos de búsqueda
147 def buscar_por_titulo(self, titulo): 1 usage new*
148 """
149     Retorna una lista de libros cuyo título contenga el texto buscado (no sensible a mayúsculas).
150     """
151     resultados = [libro for libro in self.libros.values() if titulo.lower() in libro.titulo_autor[0].lower()]
152     if not resultados:
153         return "No se encontraron libros con ese título."
154     return [str(libro) for libro in resultados]
155
156 def buscar_por_autor(self, autor): 1 usage new*
157 """
158     Retorna una lista de libros cuyo autor contenga el texto buscado (no sensible a mayúsculas).
159     """
160     resultados = [libro for libro in self.libros.values() if autor.lower() in libro.titulo_autor[1].lower()]
161     if not resultados:
162         return "No se encontraron libros con ese autor."
163     return [str(libro) for libro in resultados]
164
165 def buscar_por_categoria(self, categoria): 1 usage new*
166 """
167     Retorna una lista de libros cuya categoría coincida exactamente (no sensible a mayúsculas).
168     """
169     resultados = [libro for libro in self.libros.values() if libro.categoria.lower() == categoria.lower()]
170     if not resultados:
171         return "No se encontraron libros en esa categoría."
172     return [str(libro) for libro in resultados]
173
```



# UEA

## UNIVERSIDAD ESTATAL AMAZÓNICA

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

```
174 # Listar libros prestados de un usuario
175 def listar_libros_prestados_usuario(self, id_usuario): 1 usage new *
176     """
177     Muestra los libros que tiene prestados un usuario dado su ID, o mensaje si no tiene.
178     """
179     if id_usuario not in self.usuarios_ids:
180         return f"No existe un usuario con ID {id_usuario}."
181     usuario = self.usuarios[id_usuario]
182     return usuario.listar_libros_prestados()
183
```

### 3.2.3. Implementación:

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

```
# 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.

```
# 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.

```
# 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.



```
# 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:

```
184 # Pruebas del Sistema de Biblioteca con salidas claras
185
186 # Crear la instancia de la biblioteca
187 biblioteca = Biblioteca()
188
189 # Crear algunos libros (título, autor, categoría, ISBN)
190 libro1 = Libro(título: "Cien Años de Soledad", autor: "Gabriel García Márquez", categoría: "Novela", isbn: "1234567890")
191 libro2 = Libro(título: "El Principito", autor: "Antoine de Saint-Exupéry", categoría: "Infantil", isbn: "1234567891")
192 libro3 = Libro(título: "Don Quijote de la Mancha", autor: "Miguel de Cervantes", categoría: "Novela", isbn: "1234567892")
193
194 # Añadir libros a la biblioteca
195 print("---- Añadir Libros ----")
196 print(biblioteca.añadir_libro(libro1))
197 print(biblioteca.añadir_libro(libro2))
198 print(biblioteca.añadir_libro(libro3))
199 print(biblioteca.añadir_libro(libro1)) # Intento de añadir libro duplicado
200 print()
```



```
Run Sistema de Gestión de Biblioteca Digital x

"C:\Users\hp\PycharmProjects\SEMANA_2_P00_PILARES DE PROGRAMACION ORIENTADA
---- Añadir Libros ----
Libro 'Cien Años de Soledad' añadido a la biblioteca.
Libro 'El Principito' añadido a la biblioteca.
Libro 'Don Quijote de la Mancha' añadido a la biblioteca.
El libro con ISBN 1234567890 ya existe.
```

## 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:

```
Run Sistema de Gestión de Biblioteca Digital x

---- Libros Disponibles en Biblioteca ----
'Cien Años de Soledad' por Gabriel García Márquez (ISBN: 1234567890, Categoría: Novela) - Disponible
'El Principito' por Antoine de Saint-Exupéry (ISBN: 1234567891, Categoría: Infantil) - Disponible
'Don Quijote de la Mancha' por Miguel de Cervantes (ISBN: 1234567892, Categoría: Novela) - Disponible
```



**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()
220
```

### Consola:

```
Run Sistema de Gestión de Biblioteca Digital x
---- 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()
227
```



```
Run Sistema de Gestión de Biblioteca Digital x

---- Préstamos de Libros ----
Libro 'Cien Años de Soledad' prestado a usuario Ana Pérez.
El libro con ISBN 1234567890 ya está prestado a otro usuario.
Libro 'El Principito' prestado a usuario Juan López.
```

### 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:**

```
Run Sistema de Gestión de Biblioteca Digital x

---- Libros Prestados por Usuario ----
Usuario Juan López tiene prestados:
  - 'El Principito' por Antoine de Saint-Exupéry (ISBN: 1234567891, Categoría: Infantil)
Usuario Ana Pérez tiene prestados:
  - 'Cien Años de Soledad' por Gabriel García Márquez (ISBN: 1234567890, Categoría: Novela)
```



**Código:**

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

**Consola:**

```
Sistema de Gestión de Biblioteca Digital x
---- 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:**

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

**Consola:**

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





# UEA

## UNIVERSIDAD ESTATAL AMAZÓNICA

### 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()
258
```

Consola:

```
un Sistema de Gestión de Biblioteca Digital x
---- 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()
265
```

Consola:

```
Run Sistema de Gestión de Biblioteca Digital x
---- 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.
```



# UEA

UNIVERSIDAD  
ESTATAL AMAZÓNICA

## Estado final de libros y usuarios

**Código:**

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

**Consola:**

```
Run Sistema de Gestión de Biblioteca Digital x
---- 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:**

```
273
274 print("---- Usuarios Registrados Actualmente ----")
275 for usuario in biblioteca.usuarios.values():
276     print(f"{usuario} con libros prestados: {len(usuario.libros_prestados)}")
277
```

**Consola:**

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

Process finished with exit code 0
```