

MEMORIA ESCRITA PRÁCTICA 2
BIBLIOTECA POO

ASIGNATURA

Programación orientada a objetos

Presentado por:

Medina Reyes, Juan Leonardo jumedinare@unal.edu.co

Gomez Hincapie, Paulina pgomezh@unal.edu.co

Hincapie Cardona, Daniel dhincapieca@unal.edu.co

Avendaño Serna, Alejandro aavendano@unal.edu.co

Sierra Durango, Valentina vsierrad@unal.edu.co

Grupo 1 - Equipo 9

Profesor:

Jaime Alberto Guzman Luna

jaguzman@unal.edu.co

UNIVERSIDAD NACIONAL DE COLOMBIA

Medellín

2024



2. Descripción general de la solución:

Para este proyecto, se planteó la creación de un sistema de bibliotecas universitario diseñado para gestionar de manera eficiente y centralizada la información de las diferentes sedes asociadas a la universidad, junto con los detalles específicos de cada una. Este sistema busca integrar de forma organizada todos los aspectos relacionados con las bibliotecas, como la disponibilidad de recursos (libros, computadores, entre otros), el historial de préstamos, las multas de los usuarios y la administración de las colecciones de cada sede.

El diseño se basa en un enfoque orientado a objetos, implementando clases que representan los elementos clave del sistema, como usuarios, recursos, préstamos y sedes. Estas clases permiten modelar de manera realista las relaciones y comportamientos propios de un sistema de bibliotecas. Además, se incluyen funcionalidades que buscan emular operaciones habituales en la vida real, como la consulta de disponibilidad de recursos, el préstamo y la devolución de elementos, la gestión de multas y la actualización de inventarios.

1.1 Identificador de Problemas y Necesidades:

Este sistema representa una solución robusta y escalable para la gestión de bibliotecas universitarias. Al centralizar la información y permitir una interacción fluida entre las diferentes sedes, el sistema facilita tareas que, de otra manera, podrían ser tediosas y propensas a errores si se realizaran de forma manual. Además, el uso de clases bien definidas mejora la mantenibilidad del código, permitiendo realizar actualizaciones o integrar nuevas funcionalidades con mayor facilidad.

Desde el punto de vista del usuario, el sistema ofrece una experiencia más personalizada y eficiente, ya que los estudiantes, profesores y demás usuarios pueden consultar la disponibilidad de recursos en tiempo real, gestionar sus préstamos y resolver multas sin necesidad de intermediarios. Por otro lado, los administradores de las bibliotecas pueden tener una visión general y detallada de los recursos y las actividades de las sedes, lo que les permite tomar decisiones informadas y optimizar procesos.

Beneficios del sistema

1. **Gestión centralizada:** Unifica toda la información en una base de datos general, reduciendo la redundancia y mejorando la consistencia de los datos entre las sedes.
2. **Eficiencia operativa:** Automatiza tareas comunes, como la consulta de disponibilidad de recursos y el seguimiento de préstamos y devoluciones.
3. **Accesibilidad:** Permite a los usuarios interactuar con el sistema de manera sencilla, consultando información desde cualquier sede o incluso de forma remota.

4. **Personalización:** Facilita la personalización de la experiencia del usuario al permitirle acceder a recursos específicos y gestionar sus propios historiales.
5. **Escalabilidad:** El diseño modular basado en clases hace que sea fácil añadir nuevas sedes, recursos o funcionalidades conforme crezcan las necesidades de la universidad.

Problemas que podría solucionar

- **Errores humanos:** Reduce la probabilidad de errores en el manejo de inventarios, como préstamos no registrados o recursos mal asignados.
- **Falta de coordinación entre sedes:** Facilita la comunicación y el intercambio de información entre las diferentes bibliotecas, asegurando que los recursos sean utilizados de manera óptima.
- **Ineficiencia en la búsqueda de recursos:** Al implementar funciones de búsqueda avanzadas, los usuarios pueden localizar rápidamente los recursos disponibles en cualquier sede.
- **Gestión manual ineficiente:** Al automatizar los procesos administrativos, se reduce el tiempo y el esfuerzo necesarios para gestionar las operaciones diarias de las bibliotecas.
- **Falta de transparencia:** Los usuarios tienen acceso en tiempo real a su historial de préstamos, multas y disponibilidad de recursos, lo que mejora la confianza en el sistema.

3. Descripción del diseño estático del sistema en la especificación UML.

Clase GestionMultas

la clase se encarga de gestionar las multas asociadas a el usuario. Permite visualizar las multas pendientes, consultar sus detalles y realizar el pago de las mismas la relacion con el sistemas se utiliza para obtener información sobre las bibliotecas disponibles y el usuario actual se utiliza.

self.sistema.get_user(): para interactúa con la entidad de usuario para recuperar la lista de multas asociadas.

self.sistema.get_user().get_multas(): Accede a la colección de multas.

Clase GestionPrestamos

Permite gestionar los préstamos de materiales Los usuarios pueden visualizar sus préstamos activos, consultar detalles como la fecha de vencimiento y la sede del préstamo, extender la fecha de devolución o devolver el material antes de la fecha estipulada, se conecta con el sistema general de la biblioteca para obtener información sobre los préstamos y el usuario actual se utiliza.

- **self.sistema.get_user().get_prestamos():** Accede a la lista de préstamos del usuario.
- **Prestamo:** Representa los préstamos gestionados dentro del sistema.
- **Copia y PC:** Tipos de materiales prestados, que pueden ser devueltos y reinsertados en la sede correspondiente.
- **Calendar de tkcalendar:** Permite seleccionar una nueva fecha al extender un préstamo.

Clase ReservaEvento

ReservaEvento esta es una interfaz gráfica que permite a los usuarios realizar reservas de salas para eventos en bibliotecas. Proporciona opciones para buscar salas mediante diferentes métodos de búsqueda (búsqueda básica, por criterios y por lista). Además, permite reservar materiales que pueden ser útiles en el evento.

ReservaEvento hereda de Frame,

Utiliza sistema que es un objeto pasado como argumento en el constructor y se almacena en self.sistema, utilizado para obtener la lista de bibliotecas y sus nombres.

la clase accede a una lista de bibliotecas a través de **sistema.get_bibliotecas()**, lo que sugiere que sistema gestiona múltiples objetos de tipo Biblioteca.

Relación con Clases de Búsqueda (BusquedaBasica, BusquedaPorCriterio, BusquedaPorLista):

- o ReservaEvento crea instancias de estas clases en los métodos funcBusquedaBasica, funcBusquedaPorCriterio y funcBusquedaPorLista

Se crea las clases:

La clase BusquedaBasica representa una interfaz gráfica, permite a los usuarios seleccionar una sede, elegir una sala disponible y reservar material adicional como libros o computadoras. Su función principal es gestionar el proceso de reserva de recursos de manera interactiva y sencilla.

BusquedaBasica hereda de Frame, recibe un objeto sistema que contiene la información de bibliotecas, salas, libros y computadoras.

la clase obtiene una lista de bibliotecas del sistema y permite a los usuarios seleccionar una.

Cuando un usuario selecciona una sede, se accede a las salas disponibles mediante self.sedeSelObj.get_salas().

El usuario puede seleccionar material adicional para el evento, lo que conecta la clase con Libro y Computador.

Se validan las reservas verificando disponibilidad en sedeSelObj.get_copias() para libros y sedeSelObj.get_PCs() para computadoras.

Cuando se confirma una reserva, se crea un objeto de tipo Prestamo que se asocia con el usuario actual (self.sistema.get_user()).

La Clase BusquedaPorCriterio esta clase representa una interfaz gráfica en tkinter que permite a los usuarios buscar y reservar salas en una biblioteca para eventos, así como reservar recursos como libros y computadores. Facilita la selección de sedes, salas y materiales mediante menús desplegables e interactúa con el sistema de bibliotecas para verificar la disponibilidad y procesar reservas.

Recibe una instancia de Sistema, lo que le permite acceder a las bibliotecas, libros, computadores y la información del usuario.

Tras seleccionar una sede, accede a sus salas mediante get_salas() para listar las opciones disponibles.

Dependiendo del tipo de recurso seleccionado (Libro o Computador), la clase busca entre las instancias respectivas usando get_libros() y get_computadores().

Compara atributos como nombre, ID, ISBN, autor, año (para libros) y modelo, marca, gama (para computadores) para encontrar coincidencias.

Si un usuario confirma la reserva de un libro o computador, se crea una instancia de Prestamo y se agrega a la lista de préstamos del usuario get_multas().append(Prestamo()).

Accede a la información del usuario activo a través de self.sistema.get_user() para registrar las reservas y préstamos.

Si no hay copias disponibles del recurso solicitado, se lanzan estas excepciones y se muestra un mensaje de error con messagebox.showerror().

La clase BusquedaPorLista esta clase que permite a los usuarios:

Seleccionar una sede de biblioteca para eventos, elegir una sala disponible para reservar, y seleccionar y reservar un recurso adicional (libro o computador) para el evento.

Facilita la interacción con el sistema de bibliotecas, verificando la disponibilidad de salas y recursos antes de confirmar una reserva.

Recibe una instancia de Sistema para acceder a la lista de bibliotecas, libros, computadores y usuarios.
Muestra las bibliotecas disponibles en el sistema y permite al usuario seleccionar una sede. Obtiene información mediante `get_nombre()` y `get_salas()`.
Una vez seleccionada una sede, accede a sus salas disponibles y permite la reserva de una de ellas, Utiliza `get_nombre()` y `get_capacidad()`.
Tras seleccionar un tipo de recurso, obtiene la lista de libros o computadores disponibles en el sistema.
Se accede a los atributos como nombre e ID para listar las opciones y seleccionar un recurso.
La reserva se completa verificando la disponibilidad en las bibliotecas.
Si un usuario reserva un libro o computador, se crea una instancia de Prestamo y se añade a su historial de préstamos `get_multas().append(Prestamo())`.
Se accede a los datos del usuario mediante `self.sistema.get_user()` para registrar la reserva del recurso.
Con `FormField` Se usa para capturar la entrada del usuario cuando se solicita un ID para la selección del libro o computador.

Clase BaseDeDatos

Se encarga de gestionar la administración de los recursos dentro del sistema de la biblioteca. Permite agregar y eliminar (Libros, Copias, Computadores y PCs) en función de la sede de la biblioteca seleccionada. Además, interactúa con el usuario mediante la interfaz gráfica.
interactúa con la clase Sistema, la cual maneja la administración general de la biblioteca, incluyendo las sedes y los recursos disponibles.

- Con Biblioteca: Tiene una relación uno a muchos ya que una BaseDeDatos puede interactuar con múltiples bibliotecas dentro del sistema.
- Con Libro: Relación uno a muchos, ya que una biblioteca puede contener múltiples libros, los cuales pueden ser agregados o eliminados por BaseDeDatos.
- Con Copia: Relación uno a muchos, porque cada libro puede tener varias copias en la biblioteca, y estas pueden ser administradas mediante BaseDeDatos.
- Con Computador: Relación uno a muchos, ya que una biblioteca puede tener múltiples computadores, los cuales pueden ser agregados o eliminados.
- Con PC: Relación uno a muchos, puesto que cada Computador puede estar asociado a múltiples PCs, los cuales también pueden ser administrados.

Clase PrestamoRecursos

Se utiliza para consultar la disponibilidad de recursos en una biblioteca y facilitar su préstamo. Proporciona diferentes métodos de búsqueda (básica, por criterios y por lista) para encontrar libros o computadores dentro del sistema.

Relaciones:

- Sistema (1:1) → PrestamoRecursos depende de un objeto Sistema, que gestiona los datos y operaciones de la biblioteca.
- busquedaBasica (1:1) → Contiene una instancia de busquedaBasica para realizar búsquedas simples.
- busquedaPorCriterio (1:1, dinámico) → Puede crear instancias de busquedaPorCriterio según el tipo de recurso seleccionado.
- busquedaPorLista (1:1, dinámico) → Puede crear una instancia de busquedaPorLista para realizar búsquedas mediante una lista de recursos.

Dentro de esta se crean las clases

La clase `busquedaBasica` permite a los usuarios consultar la disponibilidad de recursos en la biblioteca mediante una búsqueda simple. Los usuarios pueden buscar libros o computadores ingresando su título o modelo respectivamente y, si están disponibles, reservarlos.

Interactúa con el objeto del sistema para acceder a la lista de recursos disponibles.

Puede generar objetos `Préstamo` si el usuario decide reservar un recurso.

Consulta las bibliotecas en sistema para verificar la disponibilidad de copias de libros o computadores.

Puede lanzar excepciones si el recurso buscado no se encuentra.

NoHayCopia y NoHayPC Se activan cuando un recurso existe pero no hay copias disponibles para préstamo.

La clase `busquedaPorCriterio` permite a los usuarios buscar y reservar recursos en una biblioteca basándose en múltiples criterios. Funciona como una interfaz gráfica que recibe valores de búsqueda detallados, valida la entrada y, si el recurso está disponible, permite su reserva.

Accede a sistema para consultar la disponibilidad de recursos.

Busca coincidencias en la lista de libros o computadores dentro del sistema.

Puede generar objetos `Préstamo` cuando un usuario decide reservar un recurso.

Interactúa con bibliotecas para verificar la disponibilidad y ubicar el recurso en una sede específica.

La clase `BusquedaPorLista` permite a los usuarios buscar y reservar libros o computadores en un sistema de biblioteca. Su propósito principal es mostrar una lista de recursos disponibles y facilitar la selección y reserva de estos. Permite acceder a los métodos `get_libros()` y `get_computadores()`, que devuelven listas de libros y computadores disponibles.

Proporciona acceso a `get_bibliotecas()`, donde se almacenan copias de los recursos.

Permite obtener al usuario actual mediante `get_user()`, lo que es clave para gestionar los préstamos.



El **FieldFrame** Se usa para ingresar el ID del recurso seleccionado, permite al usuario confirmar su selección antes de proceder con la reserva.

4. Descripción de la implementación de características de programación orientada a objetos en el proyecto (indicando los lugares y el modo en que se implementaron).

Clase y métodos abstractos:

En la implementación en Python, utilizamos clases abstractas para definir una estructura base en nuestro sistema de bibliotecas. La clase `Recurso` es una clase abstracta que actúa como una plantilla para los diferentes tipos de recursos de la biblioteca, como `Libro` y `Computador`.

- La clase `Recurso` se define usando `ABC` (Abstract Base Class) de la librería `abc`.
- Contiene atributos comunes como `nombre`, `id_recurso` y `tipo_de_recurso`.
- Se define un método abstracto `tipo_recurso()`, el cual es reescrito en las clases hijas (`Libro` y `Computador`) para proporcionar una implementación específica según el tipo de recurso.
- Se incluyen métodos `getters` y `setters` para acceder y modificar los atributos de cada recurso.
- El método `__str__()` devuelve el nombre del recurso, facilitando su representación en la interfaz gráfica con `tkinter`.

```
src > gestorAplicacion > clasesDeBiblioteca >  Recurso.py >  Recurso
1  from abc import ABC, abstractmethod
2
3  class Recurso(ABC):
4      def __init__(self, nombre, id_recurso, tipo_de_recurso):
5          self.nombre = nombre
6          self.id_recurso = id_recurso
7          self.tipo_de_recurso = tipo_de_recurso
8
```

Las siguientes imágenes muestran cómo la clase abstracta **Recurso** es heredada e implementada en sus respectivas clases hijas:

→ Clase Libro:

- ◆ Extiende la clase Recurso, indicando que es un tipo de recurso de tipo "Libro".
- ◆ Define atributos específicos como isbn, autor y año.
- ◆ Implementa el método abstracto tipo_recurso(), devolviendo "Libro".
- ◆ Se usa un atributo de clase totalLibros para llevar el conteo total de los libros creados, asignando un ID único a cada uno.
- ◆ El método __str__() devuelve el nombre del libro, facilitando su representación en la interfaz gráfica.

src > gestorAplicacion > clasesDeBiblioteca > Libro.py > Libro

```
1  from .Recurso import Recurso
2
3  class Libro(Recurso):
4      totalLibros = 0
5
6      def __init__(self, nombre, idRecurso, isbn, autor, año):
7          super().__init__(nombre, idRecurso, "Libro")
8          self.isbn = isbn
9          self.autor = autor
10         self.año = año
11         autor.get_obras().append(self)
12         Libro.totalLibros += 1
13         self.idRecurso = Libro.totalLibros
```

→ Clase Computador:

- ◆ También hereda de Recurso y define el tipo de recurso como "Computador".
- ◆ Agrega atributos específicos como marca y gama, que permiten diferenciar los computadores según sus características.
- ◆ Implementa el método abstracto tipo_recurso(), devolviendo "Computador".
- ◆ Utiliza un atributo de clase totalPCs para contar cuántos computadores se han creado, asignando un ID único.
- ◆ El método __str__() retorna el nombre del computador, lo que facilita su uso en la interfaz gráfica con tkinter.

src > gestorAplicacion > clasesDeBiblioteca > Computador.py > Computador > __str__

```
1  from .Recurso import Recurso
2
3  class Computador(Recurso):
4      # Atributo de clase que lleva el conteo total de objetos Computador creados
5      totalPCs = 0
6
7      def __init__(self, nombre, idRecurso, marca, gama):
8          # Llama al constructor de la clase base Recurso, asignando tipo "Computador"
9          super().__init__(nombre, idRecurso, "Computador")
10         self.marca = marca # Marca del computador (ejemplo: Dell, HP)
11         self.gama = gama # Gama del computador (ejemplo: alta, media, baja)
```

Implementación de Interfaces en el Proyecto

En Python, no existen interfaces como en Java, pero se pueden simular utilizando clases abstractas de la librería `abc`. En nuestro proyecto, la funcionalidad de una interfaz se implementa mediante la clase abstracta `Prestable`, que define los métodos que deben ser implementados por las clases que hereden de ella.

La clase `Prestable` actúa como una interfaz para los elementos que pueden ser prestados, como `Copia` y `PC`. Se define con la clase `ABC` y el decorador `@abstractmethod`, lo que obliga a las clases hijas a implementar los métodos:

- `get_id()`: Retorna el identificador único del recurso.
- `is_prestado()`: Indica si el recurso está actualmente prestado.
- `is_disponible_evento()`: Verifica si el recurso está disponible para eventos.
- `is_disponible_particular()`: Verifica si el recurso está disponible para préstamo a una persona.

Las clases `Copia` y `PC` implementan estos métodos, asegurando que cada recurso prestable tenga la información necesaria sobre su disponibilidad y estado.

Este enfoque permite aplicar los principios de la Programación Orientada a Objetos, asegurando que las clases que representan recursos prestables cumplan con una estructura común sin necesidad de una implementación concreta en la clase base.

```
src > gestorAplicacion > clasesDeBiblioteca > Prestable.py > Prestable
1  from abc import ABC, abstractmethod
~
```

```
src > gestorAplicacion > clasesDeBiblioteca > Copia.py > ...
1  from .Libro import Libro
2  from .Prestable import Prestable
3
4  class Copia(Libro, Prestable):
```

```
src > gestorAplicacion > clasesDeBiblioteca > PC.py > ...
1  from .Computador import Computador
2  from .Prestable import Prestable
3
4  class PC(Computador, Prestable):
```


Implementación de Herencia en el Proyecto

En nuestro proyecto, se implementa el concepto de **herencia** para estructurar los diferentes tipos de recursos dentro del sistema. La clase base **Recurso** define atributos y métodos comunes a todos los recursos de la biblioteca, como el **nombre**, el **ID** y el **tipo de recurso**.

Las clases **Libro** y **Computador** heredan de **Recurso**, agregando atributos específicos:

- Libro añade atributos como **ISBN**, **autor** y **año de publicación**.
- Computador incorpora **marca** y **gama**.

A su vez, las clases **Copia** y **PC** heredan de **Libro** y **Computador** respectivamente, y también implementan la interfaz **Prestable**, asegurando que estos recursos puedan ser prestados. Gracias a la herencia, **Copia** y **PC** mantienen los atributos y métodos de sus clases base, evitando código duplicado y permitiendo la reutilización eficiente de funcionalidades comunes.

```
src > gestorAplicacion > clasesDeBiblioteca > Libro.py > Libro
```

```
1  from .Recurso import Recurso
2
3  class Libro(Recurso):
4      totalLibros = 0
5
6      def __init__(self, nombre, idRecurso, isbn, autor, año):
7          super().__init__(nombre, idRecurso, "Libro")
8          self.isbn = isbn
9          self.autor = autor
10         self.año = año
11         autor.get_obras().append(self)
12         Libro.totalLibros += 1
13         self.idRecurso = Libro.totalLibros
```

```
src > gestorAplicacion > clasesDeBiblioteca > Computador.py > Computador > __str__
```

```
1  from .Recurso import Recurso
2
3  class Computador(Recurso):
4      # Atributo de clase que lleva el conteo total de objetos Computador creados
5      totalPCs = 0
6
7      def __init__(self, nombre, idRecurso, marca, gama):
8          # Llama al constructor de la clase base Recurso, asignando tipo "Computador"
9          super().__init__(nombre, idRecurso, "Computador")
10         self.marca = marca # Marca del computador (ejemplo: Dell, HP)
11         self.gama = gama # Gama del computador (ejemplo: alta, media, baja)
```

```
src > gestorAplicacion > clasesDeBiblioteca > Copia.py > ...
```

```
1 from .Libro import Libro
2 from .Prestable import Prestable
3
4 class Copia(Libro, Prestable):
```

```
src > gestorAplicacion > clasesDeBiblioteca > PC.py > ...
```

```
1 from .Computador import Computador
2 from .Prestable import Prestable
3
4 class PC(Computador, Prestable):
```

Ligadura Dinámica:

En el sistema de bibliotecas, la **ligadura dinámica** se encarga de garantizar que no haya problemas de ambigüedad entre las clases Recurso, Libro, Computador, Copia y PC. En Python, esto se logra de manera automática debido a su tipado dinámico y a la resolución de métodos en tiempo de ejecución.

Todas estas clases sobrescriben el método `tipo_recurso()`, el cual devuelve un **string** con el tipo de recurso correspondiente según la clase a la que pertenece el objeto. Gracias a la ligadura dinámica, cuando se invoca `tipo_recurso()` en un objeto, Python selecciona automáticamente la implementación correcta dependiendo del tipo real del objeto en tiempo de ejecución.

Un caso análogo ocurre con el método `__str__()`, el cual actúa como el equivalente de `toString()` en Java. Este método se ha redefinido en las diferentes clases (Libro, Computador, Copia y PC), permitiendo que cada tipo de recurso tenga su propia representación en forma de cadena. Así, cuando se imprime un objeto de estas clases, Python ejecuta automáticamente la versión del método correspondiente a la clase específica, resolviendo cualquier posible ambigüedad.

Recurso

```
src > gestorAplicacion > clasesDeBiblioteca > Recurso.py > Recurso
```

```
3 class Recurso(ABC):
23
24     @abstractmethod
25     def tipo_recurso(self):
26         pass
27
```

Libro

```
src > gestorAplicacion > clasesDeBiblioteca > Libro.py > Libro
```

```
3 class Libro(Recurso):
36     def tipo_recurso(self):
37         return "Libro"
```

Computador

```
src > gestorAplicacion > clasesDeBiblioteca > Computador.py > Computador > __str__
```

```
3 class Computador(Recurso):
16
17     def tipo_recurso(self):
18         """Devuelve el tipo de recurso como 'Computador'."""
19         return "Computador"
20
```

PC

```
src > gestorAplicacion > clasesDeBiblioteca > PC.py > ...
```

```
4 class PC(Computador, Prestable):
48
49     def tipo_recurso(self):
50         return "PC"
51
```

Atributos de Clase y Métodos

En la implementación del sistema de bibliotecas, se han utilizado **atributos de clase** y **métodos de clase** para mantener un control global sobre ciertos elementos clave.

Uno de los atributos de clase más importantes es `totalSedes`, definido en la clase `Biblioteca`. Este atributo **estático** permite registrar el número total de sedes de la biblioteca, incrementándose cada vez que se crea una nueva instancia de `Biblioteca`.

```
src > gestorAplicacion > clasesDeBiblioteca > Biblioteca.py >
```

```
1 class Biblioteca:
2     totalSedes = 0
```

Para acceder a este valor, se implementa el **método estático** `get_total_sedes()`, el cual devuelve la cantidad total de sedes registradas:

```
src > gestorAplicacion > clasesDeBiblioteca > Biblioteca.py > ...
```

```
66 @staticmethod
67 def get_total_sedes():
68     return Biblioteca.totalSedes
```

Este patrón de atributos de clase también se encuentra en las clases Libro y Computador. En Libro, el atributo totalLibros permite rastrear la cantidad total de copias de libros disponibles en la biblioteca. En Computador, totalPCs mantiene el registro del número total de modelos de computadoras disponibles en todas las sedes.

Libro

```
src > gestorAplicacion > clasesDeBiblioteca > Libro.py > Libro
```

```
1  from .Recurso import Recurso
2
3  class Libro(Recurso):
4      totalLibros = 0
```

```
src > gestorAplicacion > clasesDeBiblioteca > Libro.py > Libro
```

```
3  class Libro(Recurso):
41
42      @staticmethod
43      def get_total_libros():
44          return Libro.totalLibros
```

Computador

```
src > gestorAplicacion > clasesDeBiblioteca > Computador.py > Computador > __str__
```

```
1  from .Recurso import Recurso
2
3  class Computador(Recurso):
4      # Atributo de clase que lleva el conteo total de objetos Computador creados
5      totalPCs = 0
```

```
src > gestorAplicacion > clasesDeBiblioteca > Computador.py > Computador > __str__
```

```
3  class Computador(Recurso):
44
45      @staticmethod
46      def get_total_pcs():
47          """Devuelve la cantidad total de objetos Computador creados."""
48          return Computador.totalPCs
49
```

Constantes

En la clase Usuario, se define la constante PRESTAMOS_MAXIMOS, la cual establece el número máximo de préstamos simultáneos que un usuario puede realizar.

```
src > gestorAplicacion > clasesDeAdministracion > Usuario.py > ...
1  class Usuario:
2      PRESTAMOS_MAXIMOS = 3  # Número máximo de préstamos permitidos
3
```

En Python, no existen verdaderas constantes, pero por convención, los nombres de variables escritas en **mayúsculas** indican que su valor no debería cambiar a lo largo de la ejecución del programa. Aunque técnicamente el valor de PRESTAMOS_MAXIMOS podría modificarse, el uso de mayúsculas advierte a los desarrolladores de que esta variable debe tratarse como una constante.

Encapsulamiento

En Python, el encapsulamiento se maneja a través de convenciones con guiones bajos:

- **Público:** Los atributos y métodos sin guiones bajos (self.isbn, self.autor, etc.) son accesibles desde cualquier parte del código.
- **Protegido:** Se usa un guion bajo (_atributo), lo que indica que no debería modificarse fuera de la clase o sus subclasses, aunque sigue siendo accesible.
- **Privado:** Se usa doble guion bajo (__atributo), lo que oculta el atributo al hacer *name mangling* (cambia el nombre internamente para evitar acceso directo).

Sobrecarga

En Python, no existe sobrecarga de métodos o constructores de forma nativa como en Java o C++.

Manejo de this en Python

En Python, **no existe this como en Java**, sino que se usa self.

El self sirve para referirse a los atributos y métodos de un objeto dentro de la misma clase. Es como decir: "esto le pertenece a este objeto".

```
src > gestorAplicacion > clasesDeBiblioteca > Libro.py > Libro > get_autor
3  class Libro(Recurso):
6      def __init__(self, nombre, idRecurso, isbn, autor, año):
8          self.isbn = isbn
9          self.autor = autor
10         self.año = año
```

Módulo enum

El módulo enum en Python sirve para crear listas de valores fijos que no cambian. En lugar de usar cadenas de texto o números sueltos, usamos nombres más claros.

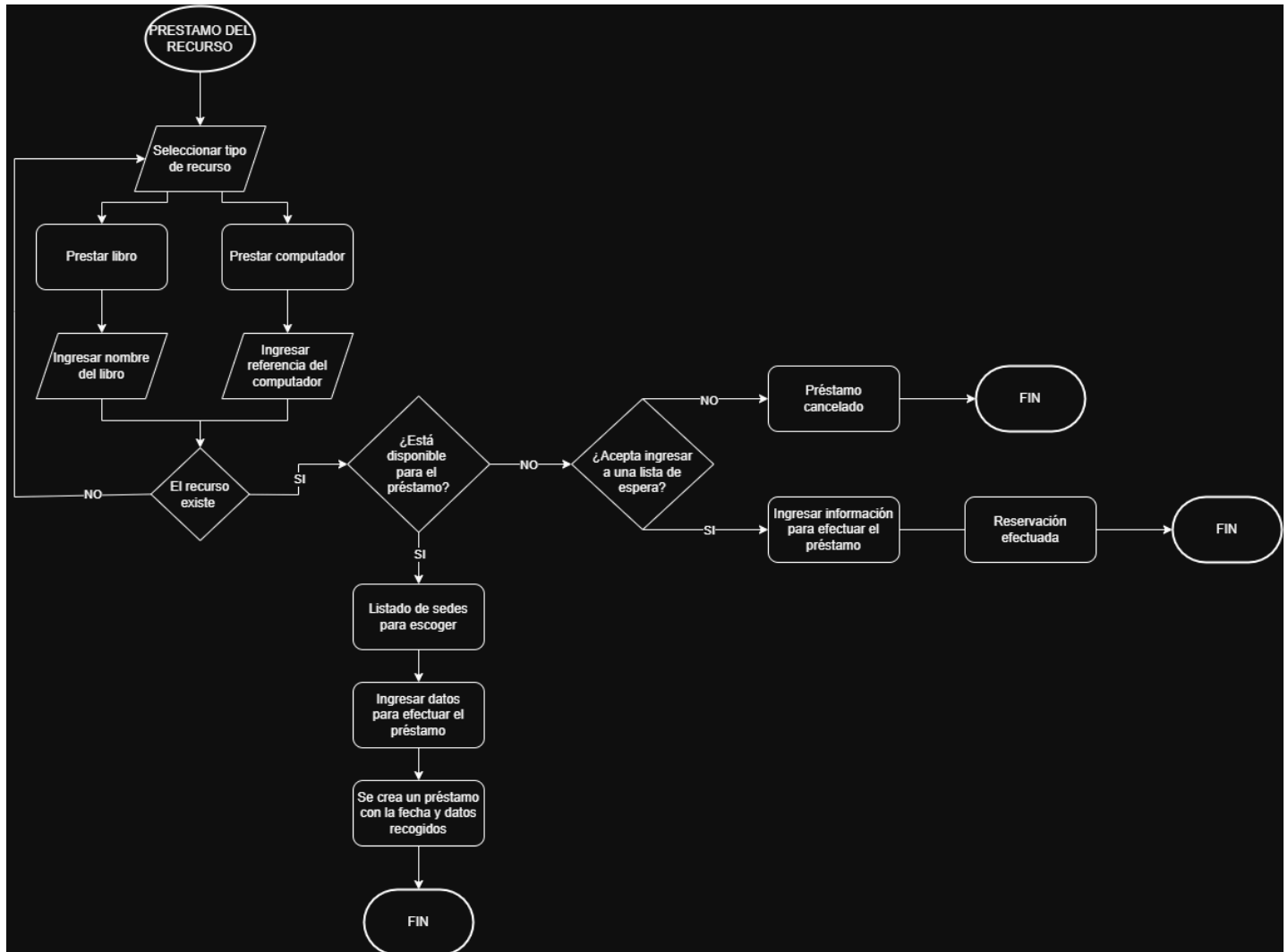
```
gestionMultas.py X
uiMain > gestionMultas.py > GestionMultas > mostrarMultas
11 class GestionMultas(Frame):
40
47 def mostrarMultas(self):
48     try:
49         if not self.sistema.get_user().get_multas():
50             raise NoHayMultas
51         self.kill(self.frame4)
52         multas = self.sistema.get_user().get_multas()
53         lista = ""
54         for i, multa in enumerate(multas):
55             lista += f"{i}. Multa de costo: {multa.get_costo()} con fecha impuesta de pago: {multa.get_fecha_impuesta()} \n"
56         listaMultas = Text(self.frame4, border=False, font=("Arial", 11), borderwidth=2, highlightthickness=3, highlightbackground="gray")
57         listaMultas.grid(row=0, column=0, columnspan=2, pady=5)
58         listaMultas.delete("1.0", "end")
59         listaMultas.config(height=5)
60         listaMultas.insert(INSERT, lista)
61
62         self.seleccion = FieldFrame(self.frame4, "Criterio", ["ID: ", "Valor"])
63         self.seleccion.grid(row=1, column=0, columnspan=2, pady=5)
64         self.seleccion.crearBoton("Enviar", self.confirmar, 0)
65
66     except NoHayMultas:
67         messagebox.showerror("Error", NoHayMultas().getError())
```

4. Descripción de cada una de las 5 funcionalidades implementadas.

El Sistema cuenta con cinco funcionalidades principales, cada una diseñada para mejorar la experiencia del usuario en la gestión de la biblioteca. A continuación, se describen en detalle:

1ra funcionalidad. Préstamo de Recursos:

Diagrama de Flujo:



La funcionalidad permite a un usuario gestionar el préstamo de recursos disponibles en la biblioteca, incluyendo libros y computadoras. El sistema realiza un proceso estructurado para verificar la disponibilidad del recurso solicitado y, en función de su estado, permite realizar el préstamo o una reserva.

1. Selección del recurso:

El flujo comienza con la selección del tipo de recurso que el usuario desea solicitar:

- Libro: El sistema solicitará el título o nombre del libro que se busca.
- Computadora: El sistema pedirá información específica sobre el equipo a buscar (como el número o ubicación del computador).

The screenshot shows a web application window titled "Sistema de Gestión de Bibliotecas". The menu bar includes "Archivo", "Procesos y Consultas", and "Ayuda". The main content area is titled "Consulta de disponibilidad para prestamo" and contains the following text: "En esta opcion podras consultar la disponibilidad para prestamo de los diferentes recursos de la biblioteca, usando criterios como Sede, Titulo, Autor, Fecha. Para de esta manera generar un prestamo en la biblioteca." Below this text are three buttons: "Busqueda básica", "Busqueda por criterios", and "Busqueda por lista". Further down, there is a label "Seleccione el material que desea consultar:" followed by a dropdown menu. The dropdown menu is open, showing "Libro" and "Computador" as options, with "Computador" currently selected. Below the dropdown is a label "Ingrese el titulo del libro a consultar:". At the bottom of the form are two buttons: "Buscar" and "Limpiar campos".

Sistema de Gestión de Bibliotecas

Archivo Procesos y Consultas Ayuda

Consulta de disponibilidad para prestamo

En esta opcion podras consultar la disponibilidad para prestamo de los diferentes recursos de la biblioteca, usando criterios como Sede, Titulo, Autor, Fecha. Para de esta manera generar un prestamo en la biblioteca.

Busqueda básica Busqueda por criterios Busqueda por lista

Seleccione el material que desea consultar: Libro

Libro
Computador

Ingrese el titulo del libro a consultar:

Buscar Limpiar campos

2. Verificación de disponibilidad:

El sistema realiza una consulta para verificar si el recurso está disponible en la base de datos:

- Si el recurso no existe: Se informa al usuario que el recurso no está disponible en la biblioteca.
- Si el recurso existe: El sistema evalúa el estado del recurso:
 - o Disponible: Permite continuar con el proceso de préstamo.
 - o En préstamo: Informa al usuario que el recurso ya está en uso y ofrece la opción de realizar una reserva.



Consulta de disponibilidad para prestamo

En esta opcion podras consultar la disponibilidad para prestamo de los diferentes recursos de la biblioteca, usando criterios como Sede, Titulo, Autor, Fecha. Para de esta manera generar un prestamo en la biblioteca.

Busqueda básica Busqueda por criterios Busqueda por lista

Error

Error en la aplicacion: El libro 'una visa' no ha sido encontrado

Aceptar

Seleccione el material que desea consultar: Libro

Ingresa el titulo del libro a consultar: una visa

Buscar Limpiar campos



Sistema de Gestión de Bibliotecas

ArchivoProcesos y ConsultasAyuda

Consulta de disponibilidad para prestamo

En esta opcion podras consultar la disponibilidad para prestamo de los diferentes recursos de la biblioteca, usando criterios como Sede, Titulo, Autor, Fecha. Para de esta manera generar un prestamo en la biblioteca.

Busqueda básicaBusqueda por criteriosBusqueda por lista

Confirmar

¿Desea reservar '1984'?

SíNo

Seleccione el material que desea consultar:

Ingrese el titulo del libro a consultar:


1984

BuscarLimpiar campos

3. Gestión de reserva:

Si el recurso se encuentra en préstamo, el sistema consulta al usuario si desea realizar una reserva para que el recurso se preste automáticamente cuando vuelva a estar disponible:

- Si no desea reservar: El proceso finaliza.
- Si desea reservar: Se solicita al usuario ingresar los datos requeridos para la reserva, y esta se registra en el sistema.

 Sistema de Gestión de Bibliotecas

—

□

×

Archivo Procesos y Consultas Ayuda

Consulta de disponibilidad para préstamo

En esta opción podrás consultar la disponibilidad para préstamo de los diferentes recursos de la biblioteca, usando criterios como Sede, Título, Autor, Fecha. Para de esta manera generar un préstamo en la biblioteca.

Busqueda básica

Busqueda por criterios

Busqueda por lista

Seleccione el material que desea consultar:

Libro

Ingrese el título del libro a consultar:

1984

Buscar

Limpiar campos

Reserva de recurso

Seleccione la sede en la cual desea realizar el préstamo:

Ingrese la fecha hasta la cual desea realizar el préstamo:

February 2025

	Mon	Tue	Wed	Thu	Fri	Sat	Sun
5	27	28	29	30	31	1	2
6	3	4	5	6	7	8	9
7	10	11	12	13	14	15	16
8	17	18	19	20	21	22	23
9	24	25	26	27	28	1	2
10	3	4	5	6	7	8	9

Reservar

4. Proceso de préstamo:

En caso de que el recurso esté disponible, el sistema continúa con el préstamo:

1. El usuario ingresa los detalles del préstamo, como el plazo de devolución (fecha límite).
2. Se crea una instancia de la clase Préstamo, que incluye información detallada como:
 - o Identificación del usuario.
 - o Identificación del recurso.
 - o Fecha de inicio y fecha límite de devolución.
3. El recurso se asocia al usuario, se elimina de la lista de recursos disponibles y se actualiza su estado en la base de datos.

5. Finalización:

- El flujo concluye cuando se completa el préstamo o la reserva.
- El sistema garantiza que la información esté actualizada, ya sea en las tablas de préstamos, reservas o usuarios.

The screenshot displays a web application titled "Sistema de Gestión de Bibliotecas". A modal dialog box titled "Reserva realizada" is open, displaying a question mark icon and the message: "¡Su reserva ha sido realizada con éxito! No olvide devolver su recurso :)". Below the message are "Aceptar" and "Cancelar" buttons. In the background, the main interface shows a search section with buttons for "Busqueda básica", "Busqueda por criterios", and "Busqueda por lista". The search criteria include a dropdown for "Seleccione el material que desea consultar:" (set to "Libro") and a text input for "Ingrese el título del libro a consultar:" (containing "1984"). Below these are "Buscar" and "Limpiar campos" buttons. The "Reserva de recurso" section includes a dropdown for "Seleccione la sede en la cual desea realizar el préstamo:" (set to "Efe Gomez") and a date picker for "Ingrese la fecha hasta la cual desea realizar el préstamo:". The date picker shows a calendar for February 2025, with the 24th highlighted. A "Reservar" button is located at the bottom.

Sistema de Gestión de Bibliotecas

Archivo Procesos y Consultas Ayuda

Reserva realizada

¿Su reserva ha sido realizada con éxito! No olvide devolver su recurso :)

Aceptar Cancelar

En esta opción podrás consultar los recursos de la biblioteca, usando criterios como Sede, o en la biblioteca.

Busqueda básica Busqueda por criterios Busqueda por lista

Seleccione el material que desea consultar: Libro

Ingrese el título del libro a consultar: 1984

Buscar Limpiar campos

Reserva de recurso

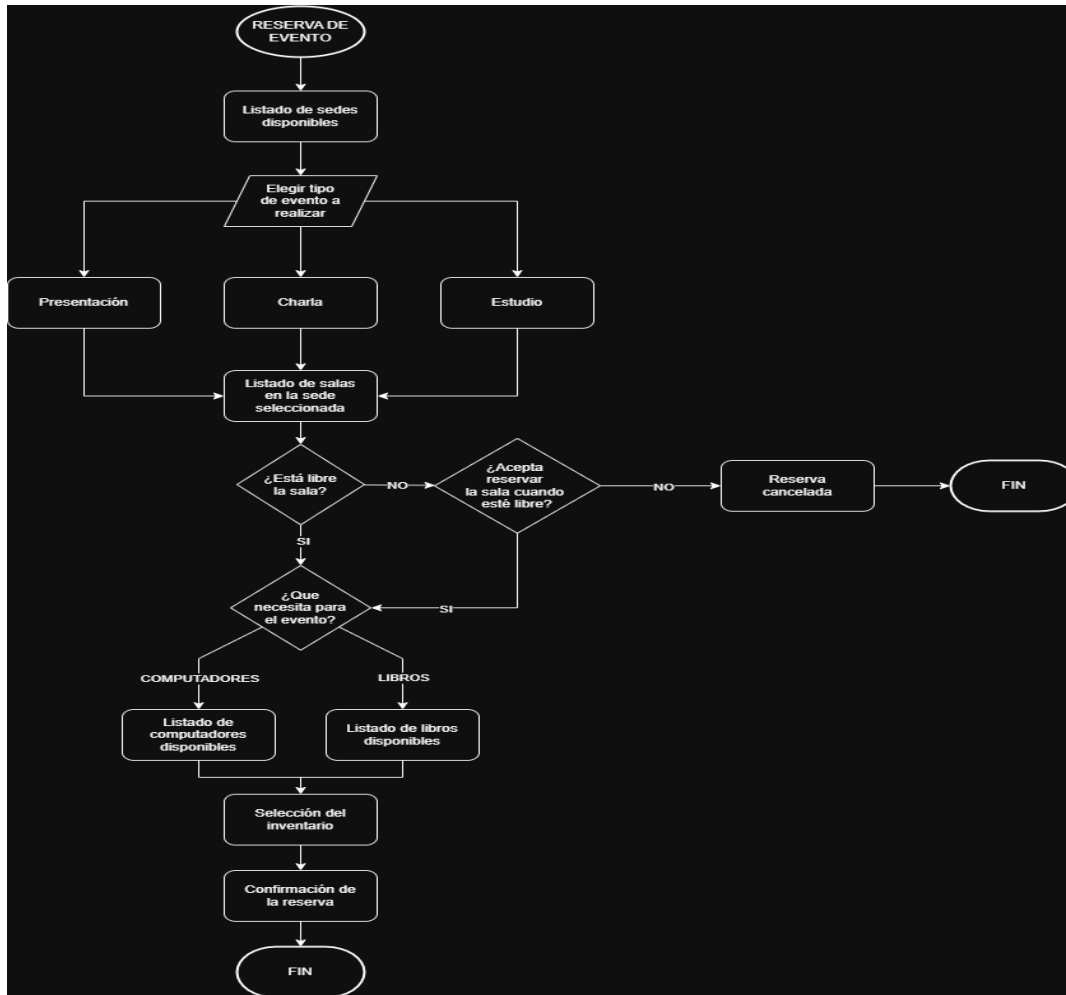
Seleccione la sede en la cual desea realizar el préstamo: Efe Gomez

Ingrese la fecha hasta la cual desea realizar el préstamo:

	Mon	Tue	Wed	Thu	Fri	Sat	Sun
5	27	28	29	30	31	1	2
6	3	4	5	6	7	8	9
7	10	11	12	13	14	15	16
8	17	18	19	20	21	22	23
9	24	25	26	27	28	1	2
10	3	4	5	6	7	8	9

Reservar

2. Consulta de disponibilidad y reserva de recursos para eventos:



El sistema de consulta de disponibilidad y reserva de recursos para eventos funciona de la siguiente manera:

1. Inicio del Proceso:

- El proceso comienza cuando un usuario necesita organizar un evento en una de las sedes de la biblioteca
- El sistema primero recopila los datos básicos necesarios del nivel inicial

2. Recopilación de Información Base:

- Se solicita al usuario que especifique:
 - o La sede donde desea realizar el evento
 - o El tipo específico de evento a realizar
 - o Los recursos o materiales necesarios para el evento

3. Proceso de Verificación:

- El sistema realiza una verificación triple paralela entre:
 - o Disponibilidad de salas en la sede seleccionada
 - o Verificación de los recursos solicitados
 - o Revisión del estado de disponibilidad general

4. Proceso de Validación:

- El sistema verifica si hay salas disponibles que cumplan con los requisitos

- Si no hay disponibilidad, se informa al usuario y el proceso termina
- Si existe disponibilidad, se procede a una segunda verificación de los materiales requeridos

5. Selección de Recursos:

- Si hay disponibilidad, el usuario puede elegir entre:
 - o Las salas que cumplen con los requisitos del evento
 - o Los materiales disponibles (copias de libros, computadoras, etc.)
 - o Los horarios disponibles para la fecha requerida

6. Finalización de la Reserva:

- Una vez seleccionados todos los elementos:
 - o Se genera un registro de préstamo que incluye todos los recursos reservados
 - o Se vincula la información del usuario, sala, materiales y horario
 - o Se confirma la reserva en el sistema

7. Interacción con las Clases del Sistema:

- El proceso involucra la interacción entre múltiples clases:
 - o Usuario: Mantiene la información del solicitante
 - o Recurso: Clase base para los diferentes tipos de recursos
 - o Libro y Copia: Gestionan los materiales bibliográficos
 - o Computador y PC: Manejan los recursos tecnológicos
 - o Préstamo: Registra la reserva final
 - o Biblioteca: Coordina la gestión general
 - o Sala: Administra los espacios físicos disponibles

Este sistema está diseñado para asegurar una gestión eficiente de los recursos de la biblioteca, permitiendo una coordinación efectiva entre los diferentes tipos de recursos y asegurando que no haya conflictos en las reservas.



Reserva de sala para evento

En esta opción podras realizar una reserva de evento en una de las salas de nuestras bibliotecas, ademas de esto, podras reservar material que pueda ser de utilidad en tu evento


Busqueda básica


Busqueda por criterios

Busqueda por lista

Seleccione la sede en la que desea realizar su evento:

Efe Gomez
Gabriel Garcia Marquez

 Confirmación

 ¿Deseas Reservar el Auditorio 2 con capacidad para 50 personas?

Reserva de sala para evento

En esta opción podras realizar una reserva de evento en una de las salas de nuestras bibliotecas, ademas de esto, podras reservar material que pueda ser de utilidad en tu evento

Busqueda básica

Busqueda por criterios

Busqueda por lista

Seleccione la sede en la que desea realizar su evento:

Las salas disponibles para evento en esta biblioteca son:

Reserva de sala para evento

En esta opción podrás realizar una reserva de evento en una de las salas de nuestras bibliotecas, además de esto, podrás reservar material que pueda ser de utilidad en tu evento

Busqueda básica

Busqueda por criterios

Busqueda por lista

Seleccione la sede en la que desea realizar su evento:

Efe Gomez ▾

Las salas disponibles para evento en esta biblioteca son:

Auditorio 2 ▾

Seleccione el material que desea reservar para el evento:

Libro ▾

Criterios

Valor

Nombre

ID

ISBN

Autor

Año

Buscar

Limpiar campos

Reserva de sala para evento

En esta opción podras realizar una reserva de evento en una de las salas de nuestras bibliotecas, ademas de esto, podras reservar material que pueda ser de utilidad en tu evento

Busqueda básica

Busqueda por criterios

Busqueda por lista

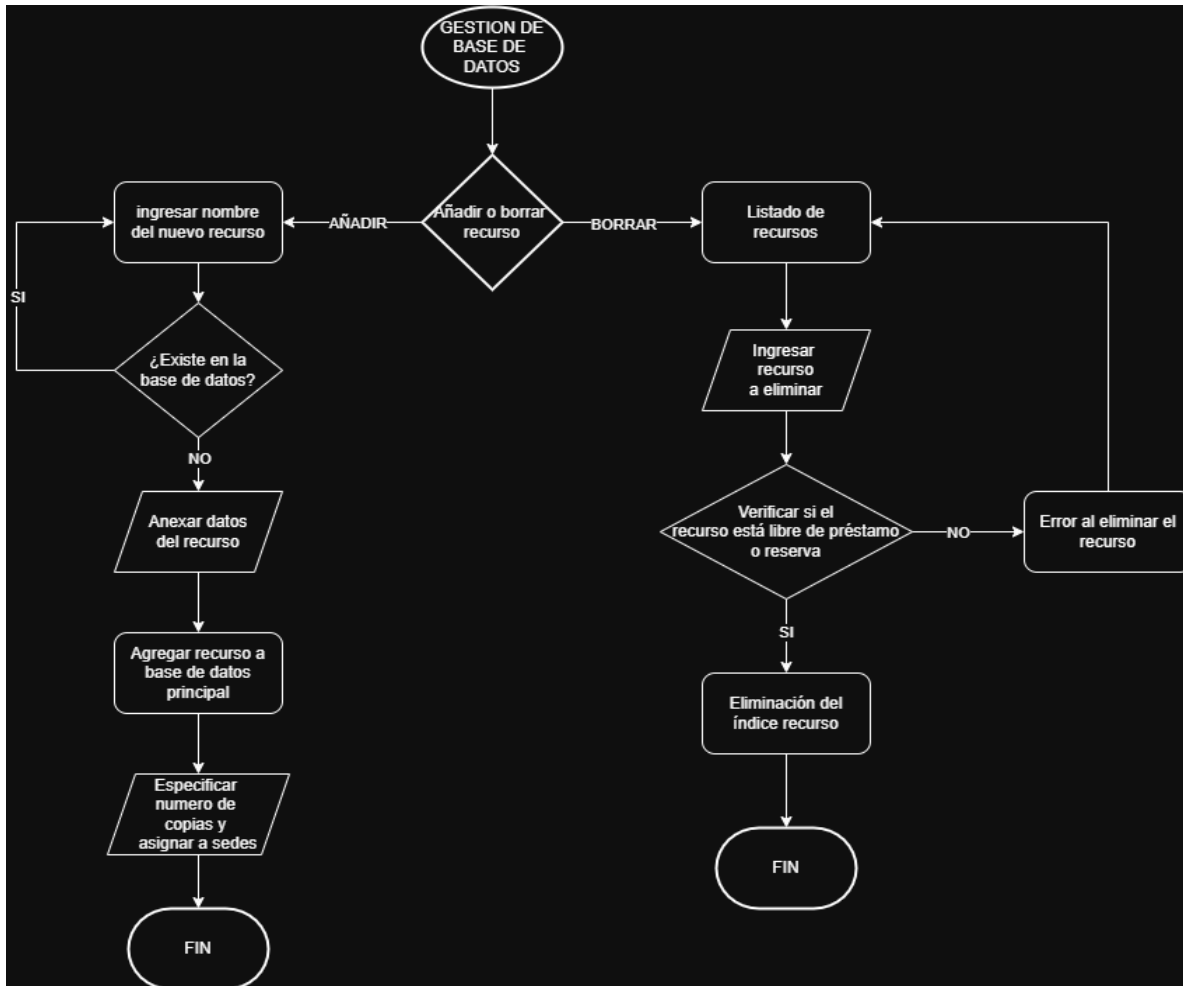
Error



Error en la aplicacion: No hay computadores disponibles del modelo '1984'

Aceptar

3. Gestión base de datos de biblioteca:



El sistema de gestión de la base de datos de la biblioteca es una funcionalidad diseñada específicamente para los administradores, que permite la manipulación integral de los recursos bibliotecarios. El proceso sigue el siguiente flujo:

1. Inicio del Proceso:

- El administrador accede al sistema de gestión de base de datos
- El sistema solicita verificar los permisos e intenciones de la base de datos

2. Selección y Validación:

- El sistema requiere que el administrador especifique el tipo de recurso a gestionar (libro o computador)
- Se verifica la existencia del recurso en la base de datos
- Si el recurso no existe, el sistema redirige al proceso de agregar nuevo recurso

3. Opciones de Manipulación: Para recursos existentes, el administrador puede:

- Modificar información existente
- Agregar nuevos elementos
- Eliminar registros
- Gestionar copias o unidades

4. Proceso de Modificación:

- El sistema verifica la disponibilidad del recurso
- Comprueba si hay préstamos o reservas activas

- Permite la actualización de datos específicos como:
 - Para libros: título, autor, editorial, ISBN, ubicación
 - Para computadores: especificaciones, ubicación, estado

5. Proceso de Eliminación:

- Verifica que el recurso no tenga préstamos activos
- Comprueba la existencia de copias o unidades relacionadas
- Solicita confirmación antes de la eliminación definitiva

6. Proceso de Agregación:

- Requiere la entrada de todos los datos obligatorios
- Valida la no duplicación de identificadores únicos
- Permite la asociación con otros elementos (como autores para libros)

7. Control de Inventario:

- Mantiene un registro de todas las modificaciones
- Actualiza automáticamente el estado de disponibilidad
- Gestiona la información de copias y ejemplares

8. Validaciones de Seguridad:

- Verifica los permisos del administrador para cada operación
- Mantiene logs de todas las modificaciones realizadas
- Implementa confirmaciones para operaciones críticas

Gestión de Base de Datos

En este apartado podrás realizar cambios en la Base de Datos de la biblioteca, añadiendo o eliminando recursos.

Sede:

Medellín

Acción:

☒ Agregar☐ Eliminar

Recurso:

Libro

Criterios

Valor

Nombre

hola

ISBN

12

Autor

vale

Año

2025

Aceptar

Limpiar campos

Gestión de Base de Datos

En este apartado podrás realizar cambios en la Base de Datos de la biblioteca, añadiendo o eliminando recursos.

Sede:

Medellín

Acción:

- ☒ Agregar
☐ Eliminar

Recurso:

Libro

Éxito



Se ha agregado el Libro a la base de datos con éxito.

Aceptar

Criterios

Valor

Nombre

hola

ISBN

12

Autor

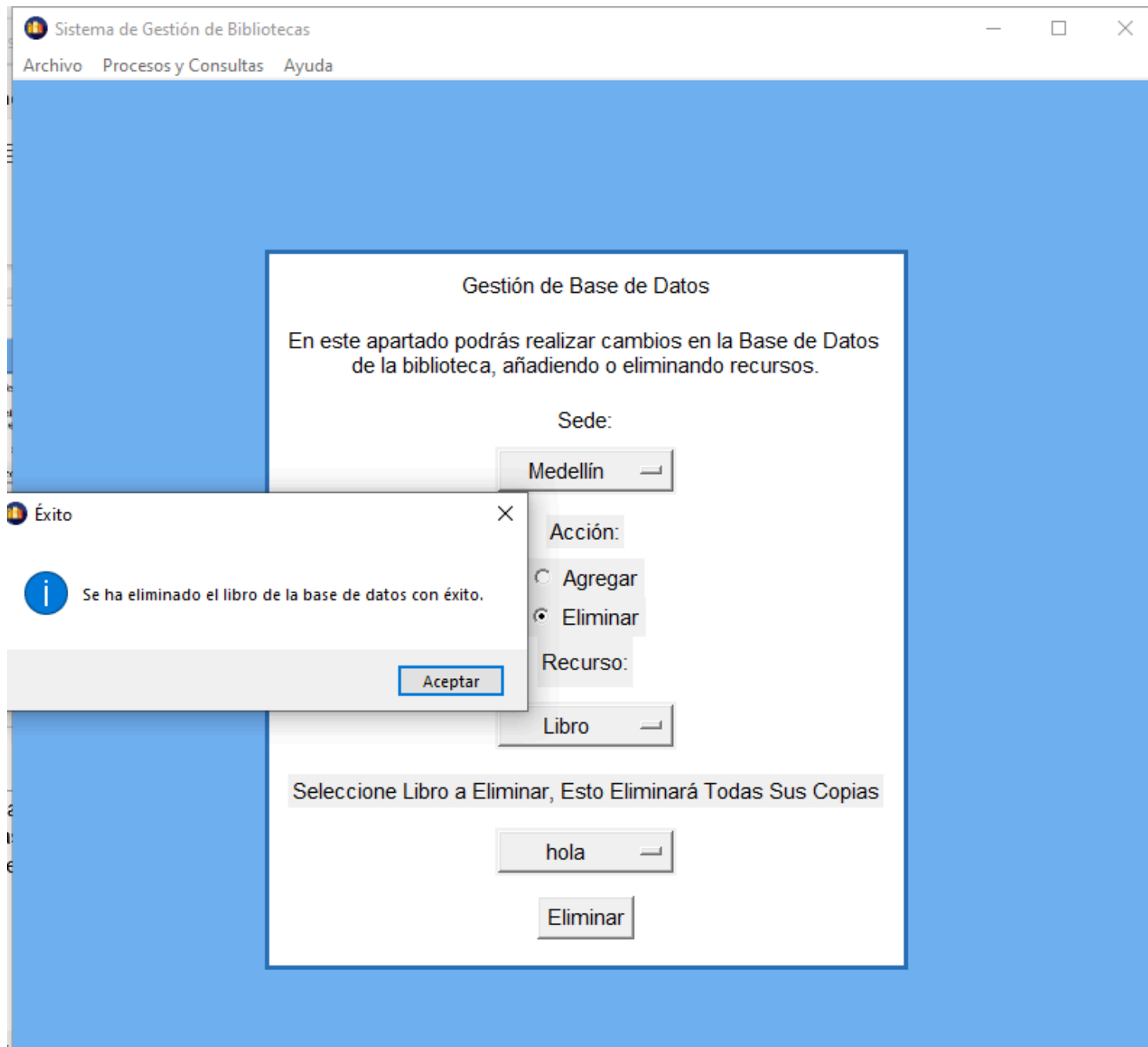
vale

Año

2025

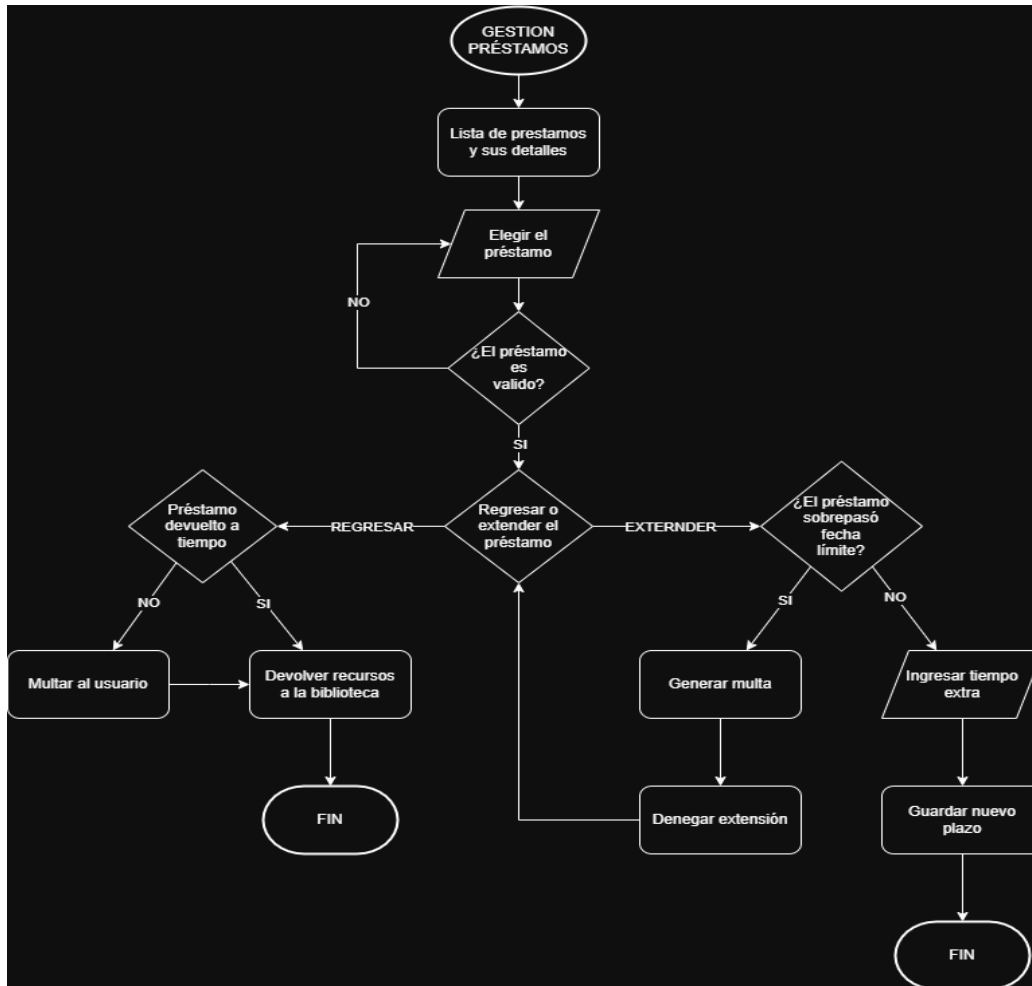
Aceptar

Limpiar campos



Esta funcionalidad es fundamental para mantener actualizada y precisa la base de datos de recursos de la biblioteca, asegurando que tanto los usuarios como el personal puedan acceder a información confiable sobre la disponibilidad y estado de los recursos bibliotecarios.

4. Gestión de Préstamos y Reservas



Esta funcionalidad tiene como objetivo principal administrar de manera eficiente los préstamos y reservas realizados por los usuarios, permitiendo tanto la devolución correcta de materiales como la extensión de plazos cuando sea aplicable. La funcionalidad está diseñada para evitar penalizaciones en la cuenta del usuario, asegurando que se cumplan los plazos y las condiciones de entrega de los recursos.

Clases Involucradas:

- Usuario: Representa al individuo que realiza préstamos o reservas en el sistema.
- Recurso: Clase genérica que abarca todos los tipos de materiales disponibles para préstamo o reserva.
- Libro: Subtipo de Recurso que representa libros físicos.
- Copia: Representa una copia específica de un libro.
- Computador / PC: Recursos tecnológicos que los usuarios pueden reservar o utilizar temporalmente.
- Préstamo: Clase que gestiona los detalles de los préstamos, como fechas, estado, y el recurso asociado.

Descripción del Proceso

1. Inicio del Proceso

- El sistema permite al usuario gestionar sus préstamos y reservas. Se muestra una lista de todos los materiales vigentes, ya sea en préstamo o en reserva, junto con sus detalles

2. Selección del Material a Gestionar:

- El usuario selecciona uno o varios materiales de la lista para proceder con su administración.

3. Verificación del Material:

- El sistema valida que el material seleccionado sea un préstamo o reserva válido. Si el préstamo no es válido, el proceso termina para ese recurso.

4. Decisión del Usuario: Regresar o Extender el Préstamo:

- El sistema solicita al usuario indicar si desea devolver el material o solicitar una extensión del plazo.

Devolución del Material:

- El sistema verifica si el usuario ha cumplido con el plazo y las condiciones de entrega.
- Si el préstamo está fuera de plazo o existen irregularidades en el estado del material (como daños), se genera una multa y se aplica una penalización en la cuenta del usuario.
- Si todo está en orden, el material se registra como devuelto correctamente, y se actualiza su disponibilidad en el sistema.

Extensión del Préstamo:

- El sistema verifica si la fecha límite del préstamo ya ha pasado.
- Si el plazo ya venció, se genera una multa y no se permite la extensión.
- Si el plazo no ha vencido, el usuario ingresa el nuevo tiempo de préstamo, y el sistema actualiza la fecha límite.

5. Registro de las Acciones:

Todas las acciones realizadas (devolución, extensión, generación de multas) se registran en el historial del usuario para garantizar trazabilidad.

Gestion de prestamos

En esta opcion podras gestionar tus prestamos actuales, pudiendo consultar detalles como el material prestado, la fecha de devolucion o realizar acciones como devolver antes de la fecha estipulada

Gestionar prestamos

- 0. Prestamo de: Cien años de soledad con fecha de vencimiento: 2023-12-31
- 1. Prestamo de: Dell Inspiron con fecha de vencimiento: 2023-12-31
- 2. Prestamo de: 1984 con fecha de vencimiento: 2025-02-24

Criterio

Valor

ID:

Enviar

Limpiar campos

Gestion de prestamos

En esta opcion podras gestionar tus prestamos actuales, pudiendo consultar detalles como el material prestado, la fecha de devolucion o realizar acciones como devolver antes de la fecha estipulada

[Gestionar prestamos](#)

Detalles del prestamo:

Detalle

Material prestado:

1984

Tipo:

Particular

Fecha de inicio:

2025-02-24


Fecha de vencimiento:

2025-02-24

Sede del prestamo:


Medellín

[Devolver prestamo](#)[Extender prestamo](#)



Prestamo devuelto

×



¡Su préstamo ha sido devuelto con éxito!, Gracias por su puntualidad :)

Aceptar

Cancelar

Gestion de prestamos

En esta opcion podras gestionar tus prestamos actuales, pudiendo consultar detalles como el material prestado, la fecha de devolucion o realizar acciones como devolver antes de la fecha estipulada

Gestionar prestamos

Detalles del prestamo:

Detalle

Material prestado: 1984

Tipo: Particular

Fecha de inicio: 2025-02-24

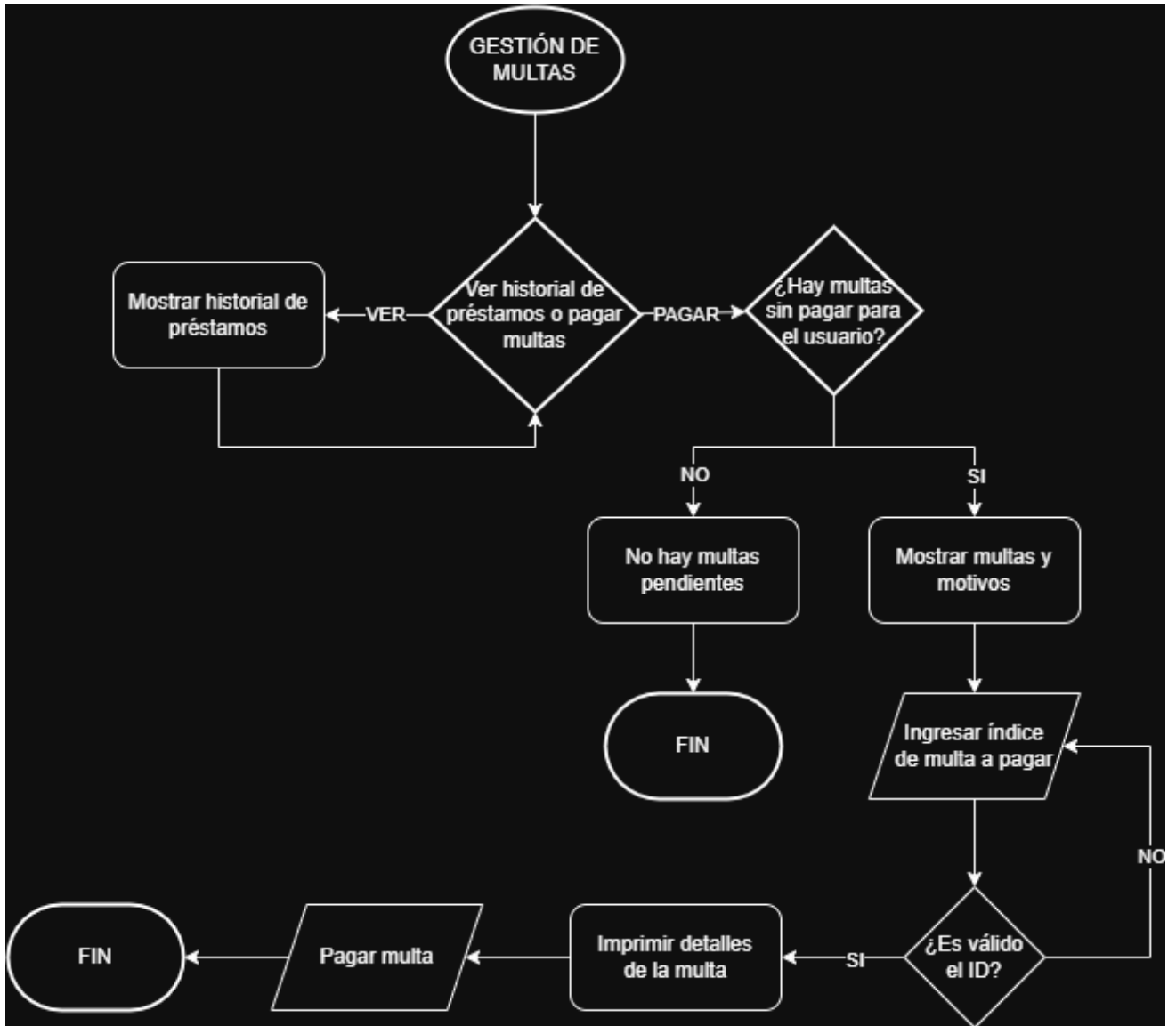
Fecha de vencimiento: 2025-02-24

Sede del prestamo: Medellín

Devolver prestamo

Extender prestamo

5. Gestión de Multas Detallada



El sistema de gestión de multas permite a los usuarios gestionar las penalizaciones generadas a partir de sus interacciones con recursos como libros, copias físicas o digitales, y equipos de cómputo (por ejemplo, computadores personales). Esta funcionalidad está diseñada para brindar una experiencia clara y sencilla, guiando al usuario paso a paso para revisar, comprender y, eventualmente, resolver sus multas.

Flujo de la Funcionalidad

1. Acceso al Sistema de Gestión de Multas:

- o El usuario ingresa al módulo de gestión de multas, donde se le presentan dos opciones principales:
 - **Pagar multas:** Permite al usuario gestionar y realizar pagos de las multas pendientes.
 - **Ver historial de préstamos:** Muestra un registro detallado de los recursos que el usuario ha solicitado y utilizado previamente.

2. Gestión de Pagos de Multas:

- o Si el usuario selecciona "Pagar multas", el sistema primero verifica si existen multas pendientes:
 - **Sin multas pendientes:** Si no hay multas asociadas al usuario, se muestra un mensaje indicando que no hay nada que gestionar y el flujo termina.
 - **Con multas pendientes:** Si hay multas activas, el sistema muestra una lista detallada con:
 - La descripción de cada multa.
 - El motivo que la originó (por ejemplo, retraso en la devolución de un libro o mal uso de un equipo).

3. Selección de Multa a Pagar:

- o El usuario ingresa el ID de la multa que desea pagar.
- o El sistema valida la información ingresada:
 - **ID Inválido:** Si el ID ingresado no corresponde a una multa existente, se solicita al usuario que lo reingrese.
 - **ID Válido:** Si el ID es correcto, se muestra un detalle completo de la multa, incluyendo:
 - El monto a pagar.
 - La razón de la multa.
 - Fecha de emisión.

4. Proceso de Pago:

- o Una vez confirmada la multa seleccionada, el sistema habilita la opción para realizar el pago.
- o Al completarse el pago, la multa es eliminada del estado del usuario y este puede continuar con el uso del sistema sin restricciones.

5. Historial de Préstamos:

- o Si el usuario selecciona "Ver historial de préstamos", el sistema genera un informe que detalla:
 - Recursos solicitados (libros, copias, equipos).
 - Fechas de préstamo y devolución.
 - Multas asociadas a esos préstamos, si las hubiera

Gestion de multas

En esta opción podrás gestionar tus multas actuales además de pagarlas, en las siguientes opciones podrás acceder a los detalles de cada multa asociada y realizar acciones sobre ellas

Gestionar multas

0. Multa de costo: 3000 con fecha impuesta de pago: 2023-11-21
1. Multa de costo: 10000 con fecha impuesta de pago: 2023-11-21

Criterio

Valor

ID:

Enviar

Limpiar campos

Gestion de multas

En esta opción podrás gestionar tus multas actuales además de pagarlas, en las siguientes opciones podrás acceder a los detalles de cada multa asociada y realizar acciones sobre ellas

Gestionar multas

Detalles de la multa:

Detalle

Tipo:

Retraso

Fecha:

2023-11-21

Costo:

3000

Pagar

Gestion de multas

En esta opción podrás gestionar tus multas actuales además de pagarlas, en las siguientes opciones podrás acceder a los detalles de cada multa asociada y realizar acciones sobre ellas

Multa pagada

×

?

Su multa ha sido pagada con éxito.

Aceptar

Cancelar

Detalle

Tipo:

Fecha:

Costo:

2023-11-21

3000

Pagar

6. Manejo de excepciones

Descripción

En el desarrollo del sistema, se han implementado diversas excepciones personalizadas para mejorar la estabilidad, robustez y seguridad de la aplicación. Estas excepciones permiten manejar situaciones inesperadas, garantizando que el usuario reciba retroalimentación clara y adecuada en caso de errores. Se han dividido en dos grupos principales:

Excepciones de usuario: Manejan errores relacionados con la interacción del usuario, como campos vacíos, recursos inexistentes o elementos no disponibles.

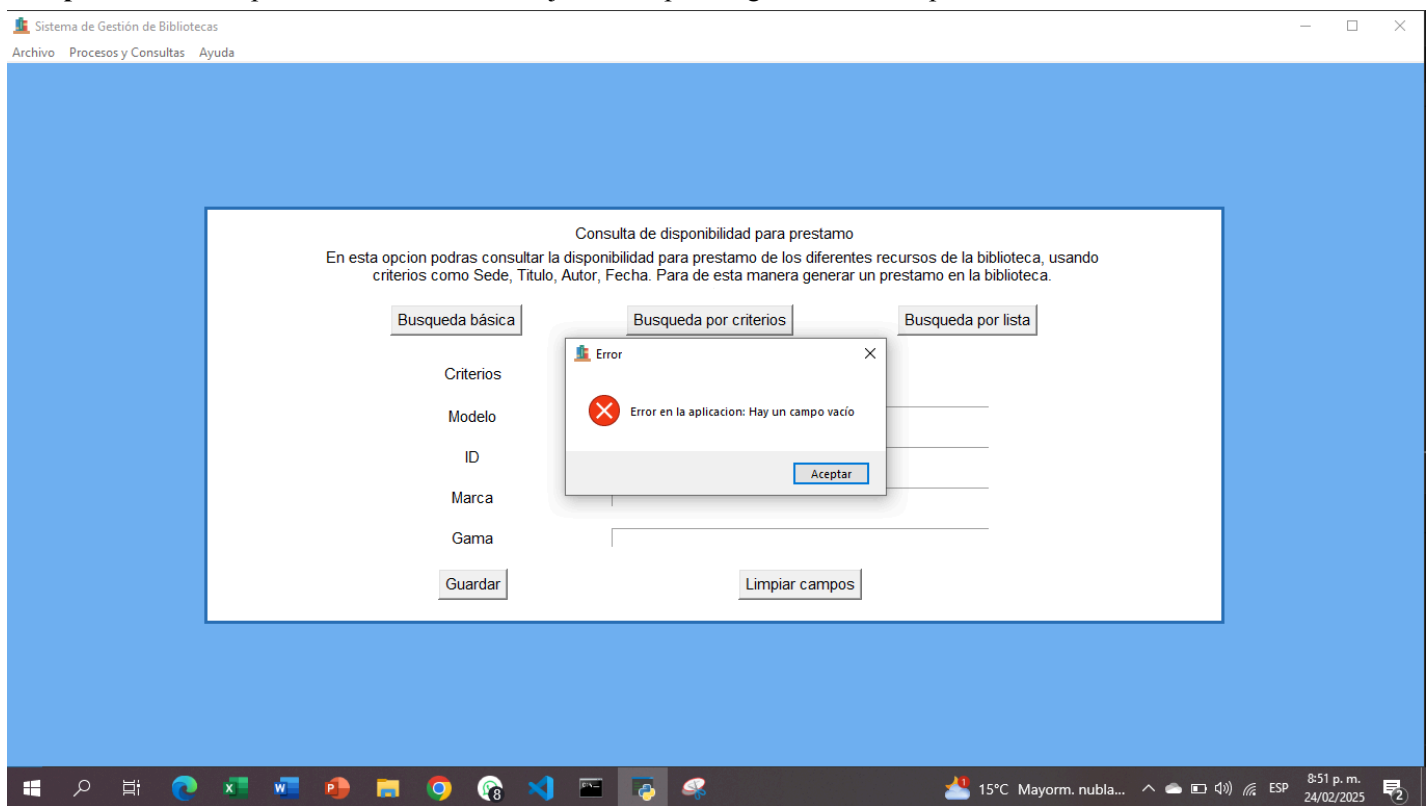
Excepciones de Python: Controlan errores técnicos y de lógica del sistema, como problemas de índices fuera de rango o datos incorrectos.

A continuación, se describen las principales excepciones utilizadas en el sistema:

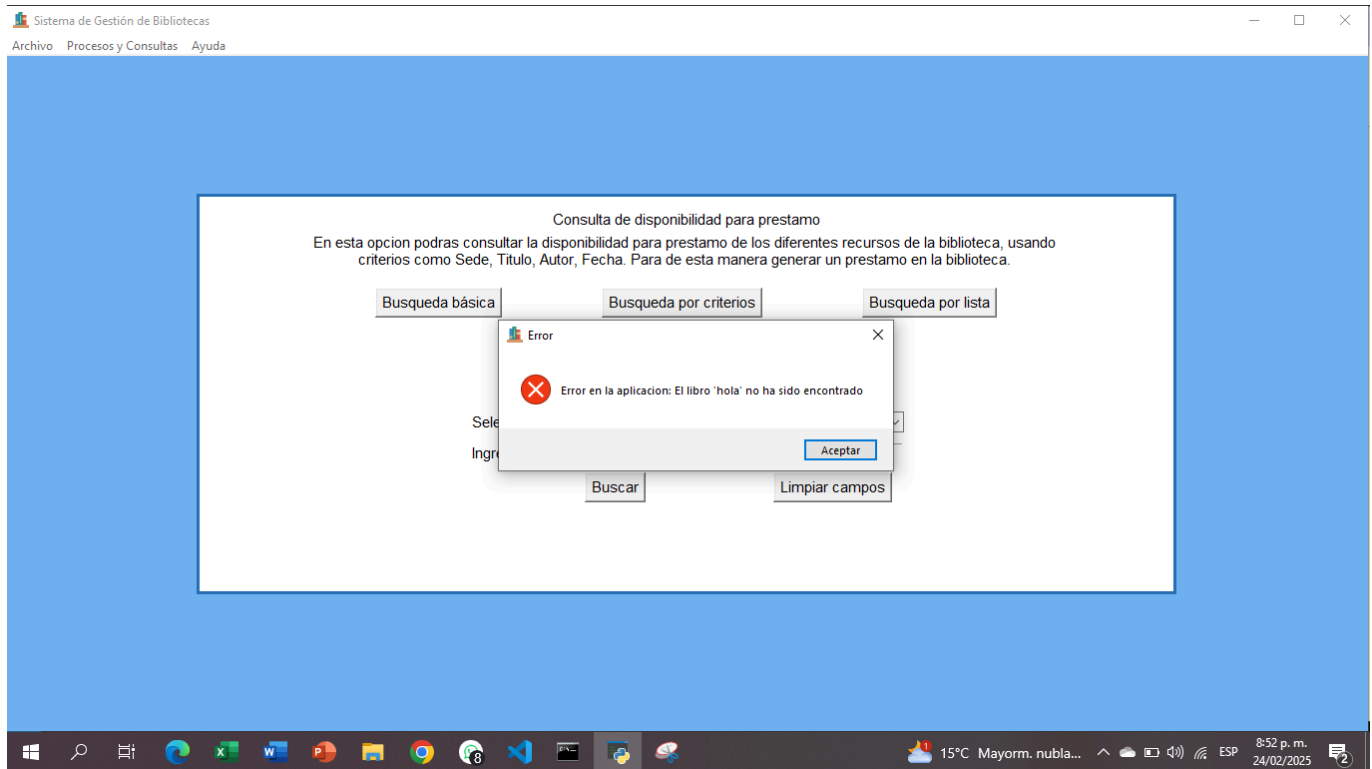
Excepciones de usuario

Estas excepciones derivan de la clase `ErrorDeUsuario`, que a su vez hereda de `ErrorAplicacion`.

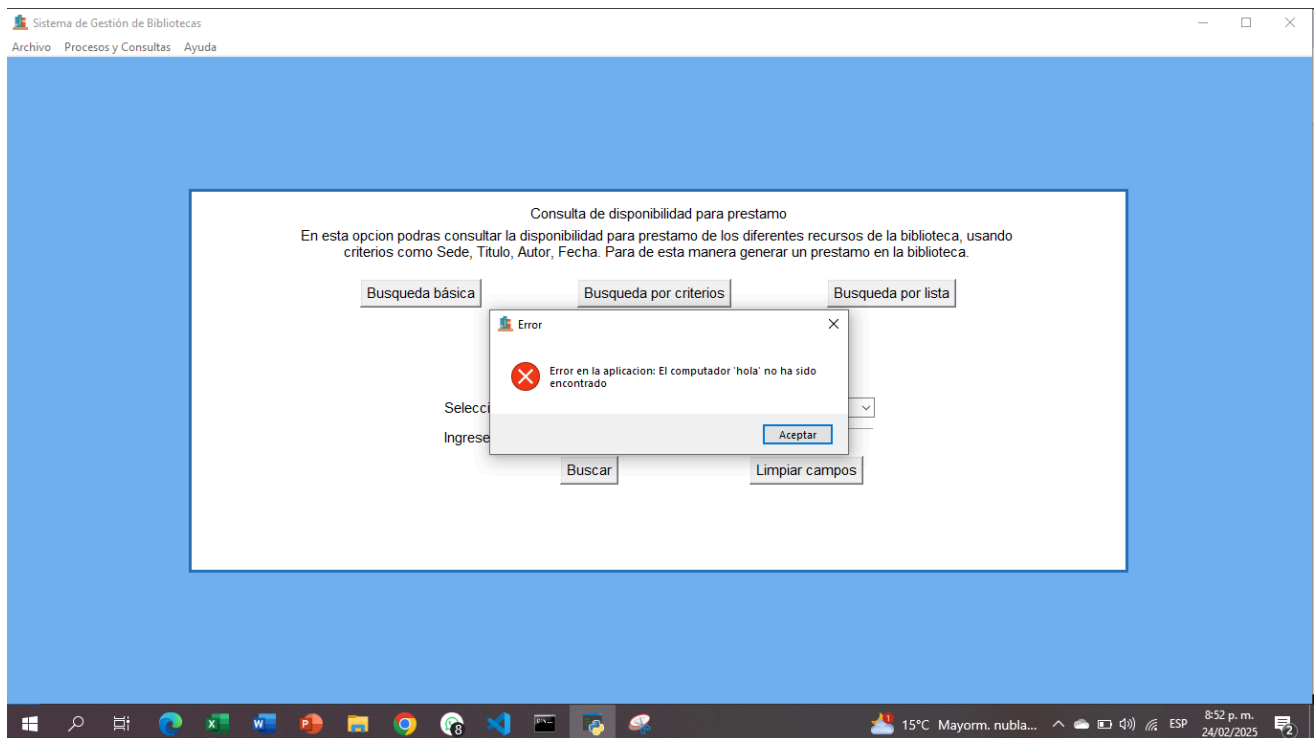
CampoVacio: Se dispara cuando el usuario deja un campo obligatorio sin completar.



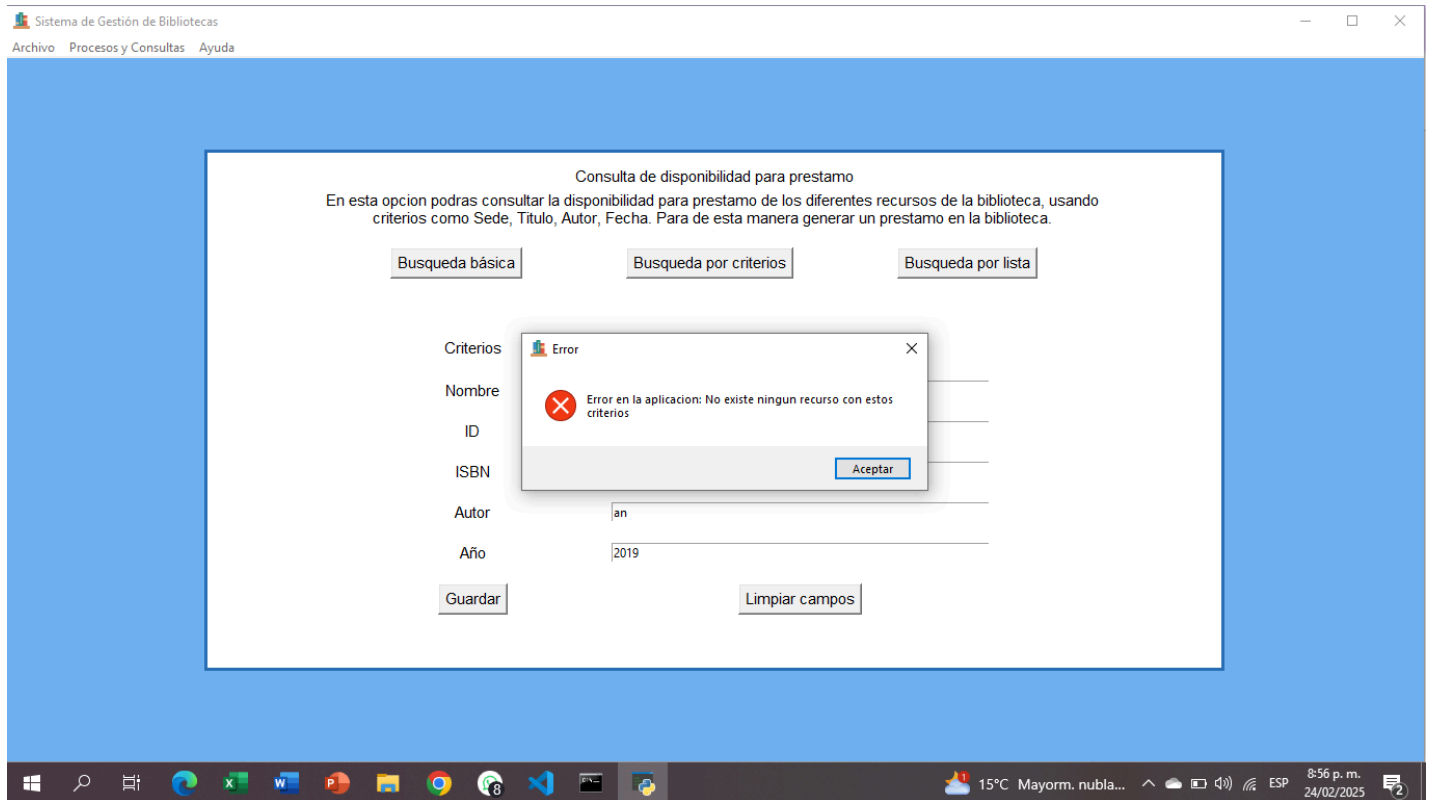
libroNoEncontrado: Indica que el libro ingresado no está registrado en la base de datos.



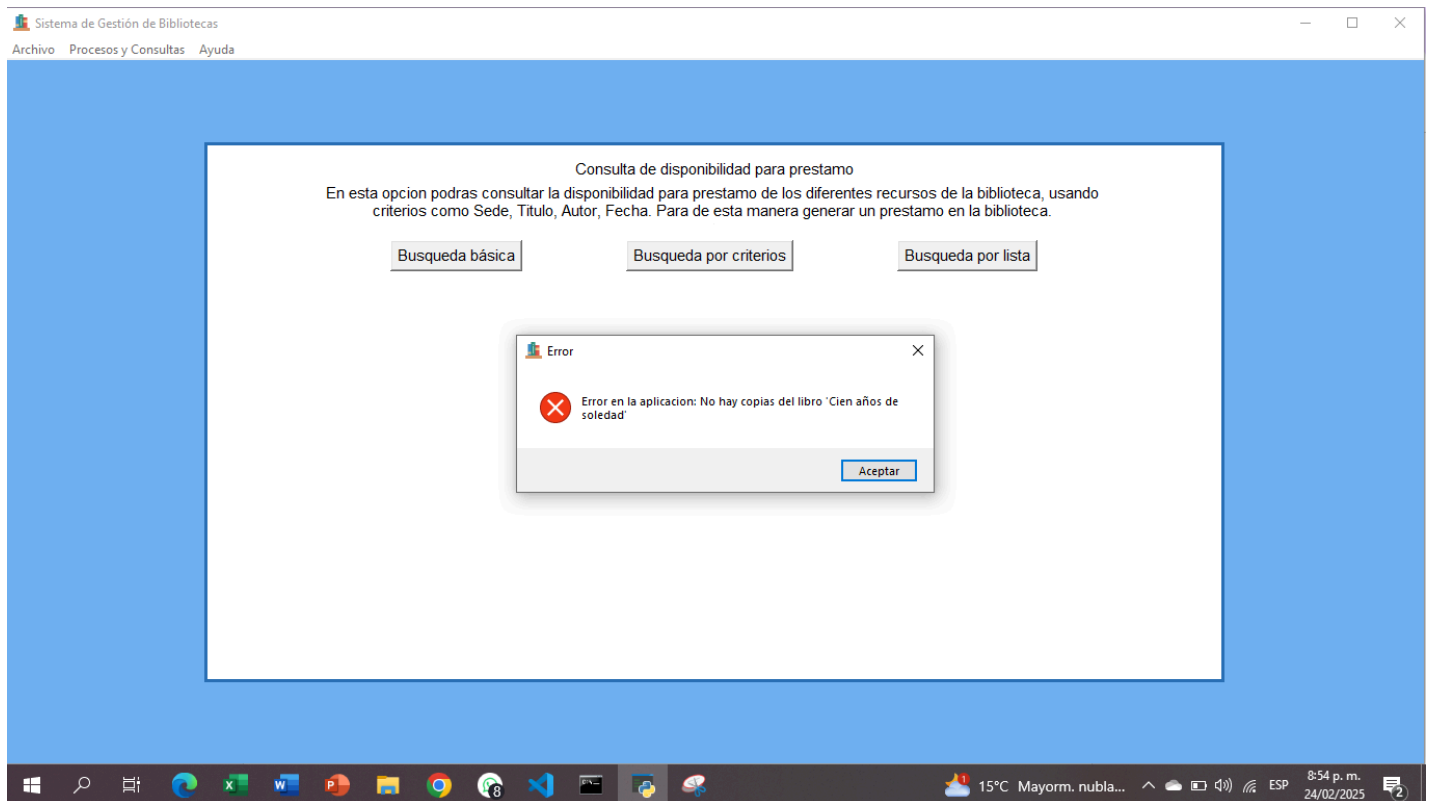
ComputadorNoEncontrado: Se lanza si el usuario intenta acceder a un modelo de computador existente.



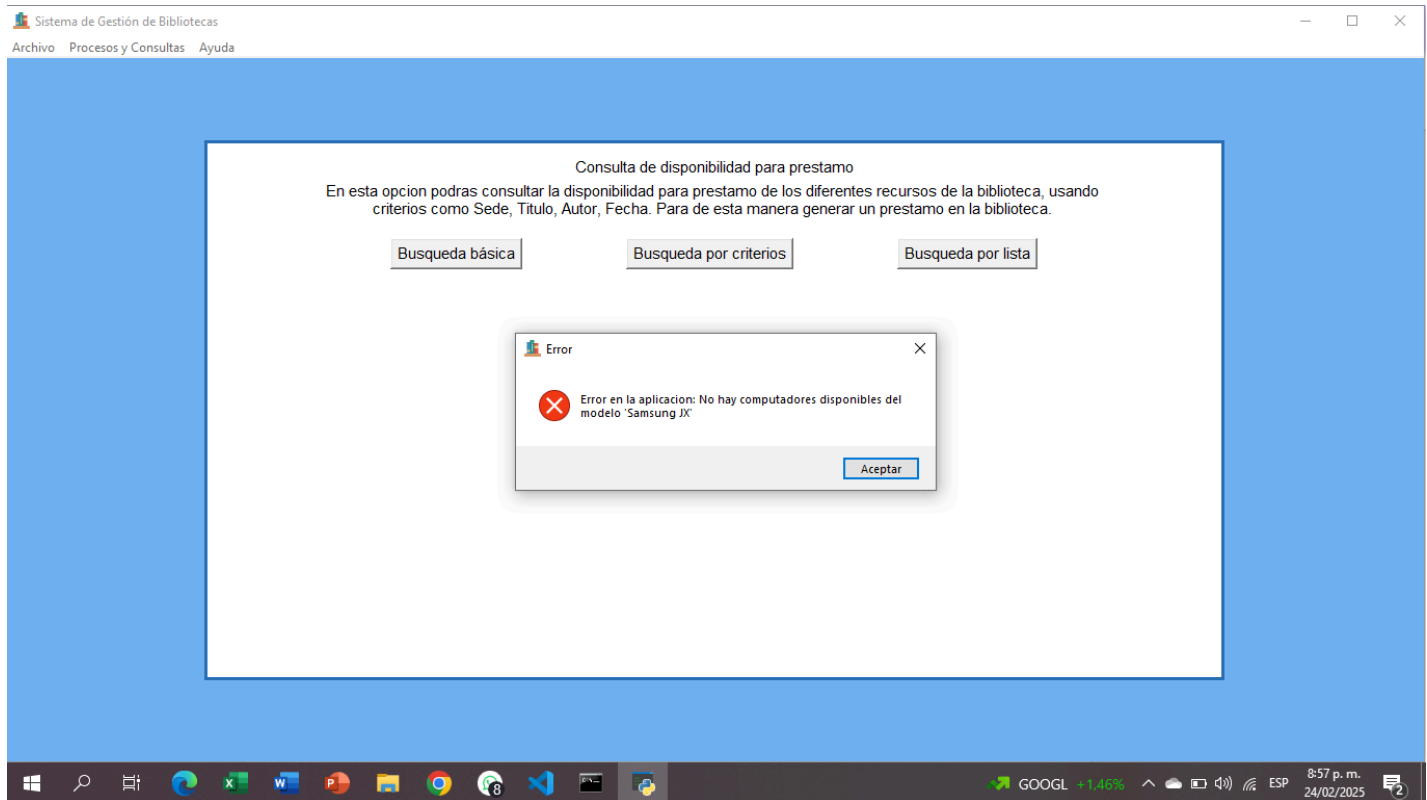
NingunRecurso: Se genera cuando una búsqueda no arroja resultados.



NoHayCopia: Indica que no hay copias disponibles de un libro en la biblioteca.



NoHayPC: Se activa si no hay computadoras disponibles en la sede seleccionada.



NoHayPrestamos: Se dispara cuando el usuario intenta gestionar préstamos y no tiene ninguno activo.

NoHayMultas: Se lanza cuando el usuario intenta consultar multas y no tiene ninguna registrada.

Implementación en el Código

A continuación, se muestra un fragmento de código donde se implementa CampoVacio:

```
15     class CampoVacio(ErrorDeUsuario):
16         def __init__(self):
17             super().__init__(f"Hay un campo vacio")
```

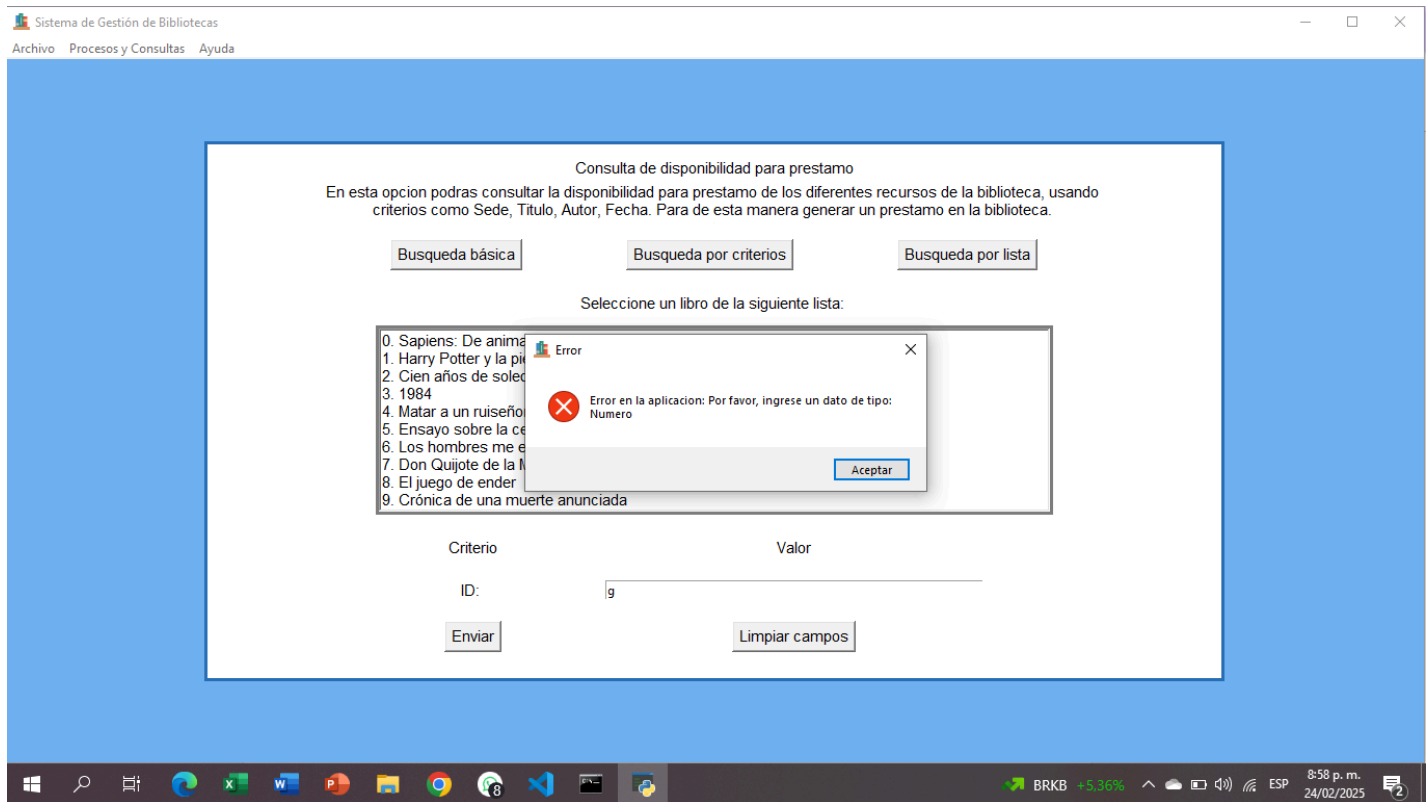
Este bloque verifica si el usuario dejó vacío un campo y muestra un mensaje de error en caso de que así sea.

Excepciones de Python

Estas excepciones derivan de ErrorDePython y están relacionadas con errores internos del sistema.

IndexFuera: Se lanza cuando un usuario ingresa un índice fuera del rango de opciones disponibles.

DatoIncorrecto: Se dispara cuando se ingresa un dato con un tipo incorrecto (ejemplo: ingresar texto en un campo numérico).



VariableNone: Indica que se ha intentado acceder a una variable que no tiene un valor asignado.

RutaIncorrecta: Se usa cuando la aplicación no encuentra un archivo o directorio necesario.

ErrorSerializacion: Se genera si ocurre un fallo en la serialización o deserialización de datos.

Implementación en el Código

Ejemplo de IndexFuera en la gestión de préstamos:

```

76         try:
77             if int(self.seleccion.getValue("ID: ")) < 0:
78                 raise IndexFuera
79             self.prestamo = self.sistema.get_user().get_prestamos()[int(self.seleccion.getValue("ID: "))]
88         except (IndexError, IndexFuera):
89             messagebox.showerror("Error", IndexFuera().getError())

```

Este fragmento previene que un usuario seleccione un préstamo con un ID fuera de los disponibles.

7. FieldFrame

La clase FieldFrame se ubica en el uiMain y se implementó en las funcionalidades baseDatos.py, gestionMultas.py, gestionPrestamos.py, prestamoRecursos.py y reservaEvento.py, es un contenedor que cumple la función de organizar y gestionar campos de entrada con etiquetas en un marco de referencia visual, en este caso particularmente facilita la gestión de las listas atributo-valor de tal manera que organizándose en pares clave-valor pueda mostrar la información indexada de forma más clara, intuitiva y amigable para el usuario final.

Organizar los valores de esta forma es conveniente tanto para posicionar visualmente los atributos con sus respectivos valores como para permitir la modificación de estos con el control Entry, también facilita agregar o borrar filas de forma dinámica y permite almacenar los datos en una estructura en forma de diccionario según convenga.

Componentes y alcance de FieldFrame:

root: Contenedor en donde se ubicará el FieldFrame.

títuloCriterios: Título de columna de criterios.

Criterios: Lista de criterios que se mostrarán como etiquetas o Labels.

títuloValores: Título de la columna de valores.

valores: Lista con valores predefinidos para cada Entry.

La clase FieldFrame permite organizar información en forma de tabla con etiquetas y entradas de datos, facilita la manipulación de las mismas con métodos para obtener, limpiar y bloquear campos.

Métodos de la clase:

Método **crearBoton:** Línea 41, sirve para crear un botón y ubicarlo en la columna.

```
def crearBoton(self, texto, comando, col):
```

```
    Button(self, text=texto, command=comando, font=("Arial", 11)).grid(row=(len(self.criterios)+1), column=col,
padx=50, pady=10)
```

Método **añadirBoton:** Línea 44, sirve para añadir un botón externo al FieldFrame.

```
def añadirBoton(self, boton, col):
```

```
    boton.grid(row=(len(self.criterios)+1), column=col, padx=50, pady=10)
```

Método **limpiarEntradas:** Línea 47, sirve para limpiar el contenido en los campos Entry.

```
def limpiarEntradas(self):
```

```
    for entrada in self.entradas:
```

```
        entrada.delete(0, last=END)
```

Método **getValores:** Línea 51, sirve para guardar en self.valores los valores en los campos Entry.

```
def getValores(self):
```

```
    self.valores = [valor.get() for valor in self.entradas]
```

Método **getValue:** Línea 54, sirve para devolver el valor del campo Entry que corresponda algún criterio.

```
def getValue(self, criterio):  
  
    i = self.criterios.index(criterio)  
  
    return self.entradas[i].get()
```

Implementación en las demás clases:

- Ubicación en baseDatos.py:

Se ubica en la línea 4 que es donde se importa con “from uiMain.FieldFrame import FieldFrame” y posteriormente entre las líneas 347-351

Su función será contener dentro del marco de la interfaz las referencias correspondientes a los objetos criteriosTitulo, lista, valorTitulo, permitiendo realizar búsquedas dentro de la base de datos para ítems bajo estos índices. Consecuentemente el argumento habilitará la interacción de sistema y recurso con la interfaz.

- Ubicación en gestionMultas.py:

Se importa en línea 7, se construye desde la línea 62 y finalmente entre las líneas 82 y 86.

En línea 62 su función es facilitar posteriormente la búsqueda de multas por criterio de número de identificación o valor.

Su función es mostrar las multas y los detalles asociados a estas de forma organizada y entendible, tales como tipo, fecha y costo de la multa, a su vez permite interactuar con el listado permitiendo pagar la multa y por consecuencia eliminarla de la base de datos.

- Ubicación en gestionPrestamos.py:

Se importa en línea 8, se construye desde la línea 63 y entre las líneas 81,87.

Su función en línea 63 es muy similar en el sentido en que mediante la interfaz facilita la búsqueda en el apartado de préstamos filtrada por criterio ID y valor.

A partir de la línea 81 su función es brindar detalles del préstamo como material prestado, tipo de préstamo y fecha de inicio, permite interactuar con la base de datos en la medida en que se devuelva el préstamo para eliminarlo del registro o se extienda para mantenerlo indexado.

- Ubicación en prestamoRecursos.py:

Se importa en línea 7, se construye desde la línea 246 y por último entre las líneas 415,442.

Su función en la línea 246 es contener la opción de realizar una búsqueda del recurso a prestar filtrada por tipo de recurso.

A partir de la línea 415 hasta 427 su función es en el caso de seleccionar libro, enseñar sus detalles, como criterio, ID y valor del mismo, también contiene al botón enviar que permite registrar el préstamo y añadirlo a la base de datos, por último desde línea 428 hasta 442, en el caso de seleccionar computador hará lo propio pero con el tipo de recurso computador (mostrará criterio ID, y valor del recurso) permitiendo también registrar el préstamo en la base de datos.

· Ubicación en reservaEvento.py:

Se importa en línea 6, se construye entre las líneas 245, 252 y finalmente entre 430, 457.

Su función entre las líneas 245 y 248 es expandir los detalles de criterio del libro a prestar como nombre, ID, ISBN, autor, año de publicación y valor de este, en caso de seleccionar computador, la funcionalidad desde 249 hasta 252 es la de expandir los detalles de criterio con respecto al computador, detalles como modelo, ID, marca, gama y valor.

La función desde 430 hasta 442 es enseñar detalles del artículo como ID y valor, y también la de permitir interactuar con el botón enviar para registrar el préstamo del libro en la base de datos, desde la línea 443 hasta la 457 la función es la misma pero para el artículo computador.

8. Manual de usuario.

Este sistema ha sido diseñado teniendo en cuenta la necesidad de que el acceso y la manipulación del sistema sean gestionados por un usuario administrador. Este administrador tiene la capacidad de acceder a todas las funcionalidades del sistema sin necesidad de introducir contraseñas o realizar ningún tipo de comprobación de identidad. Esto implica que el sistema puede ser utilizado de manera directa, sin la obligación de ingresar credenciales para acceder a sus recursos.

En el contexto de nuestra biblioteca, existen dos tipos de recursos principales que se gestionan: los libros y los computadores. Los objetos de las clases correspondientes, Libro y Computador, representan los elementos que la biblioteca conoce y gestiona, es decir, aquellos recursos que están registrados en el sistema como disponibles para su uso. Estos objetos permiten representar de manera detallada los libros y computadores disponibles en la biblioteca.

Por otro lado, las clases Copia y PC son utilizadas para representar las copias de los libros existentes y los equipos de computadores disponibles, respectivamente. Los objetos de estas clases son los que se asignan a una sede específica de la biblioteca y se prestan a los usuarios. Es importante señalar que un libro puede estar registrado en el sistema, pero esto no implica necesariamente que haya una copia disponible en ese momento para ser prestada. De igual manera, puede haber computadores registrados, pero no siempre estarán disponibles para su uso. Esta diferencia es lo que justifica la existencia de las clases Copia y PC, las cuales permiten gestionar de manera más precisa la disponibilidad de los recursos en la biblioteca.

Además de estos, se ha desarrollado una clase FieldFrame la cual hereda de Frame. Esta clase genera automáticamente entradas de valores asociadas a criterios en específico. Esta clase se usa en todas las funcionalidades para permitir al usuario el ingreso de datos al sistema. Para el uso de esta clase se debe ingresar los títulos de las dos columnas y una lista con los criterios. Además, se tiene la opción de definir el contenido de las entradas y permitir su edición o no.

Para crear un entorno que sea fácil de explorar y que permita probar las funcionalidades del sistema, se han cargado al mismo una serie de objetos predefinidos, con sus respectivos atributos, de las clases Biblioteca, Libro, Computador y Autor. Además, se han creado instancias de las clases Copia y PC para probar de manera efectiva las funcionalidades relacionadas con el préstamo de los recursos. De este modo, el sistema está preparado para simular situaciones reales de préstamo y gestión de los recursos dentro de la biblioteca.

Bibliotecas: Los objetos de la clase biblioteca están contruidos de la siguiente manera:

Biblioteca(Nombre, Sede). Y los que ya se encuentran cargados en el sistema son los siguientes:

- Efe Gomez, Medellin

- Gabriel Garcia Marquez, Bogota

Libros: Los objetos de la clase Libro están contruidos de la siguiente manera: Libro(Titulo). Y los que ya están cargados en el sistema son:

- Sapiens: De animales a dioses
- Harry Potter y la piedra filosofal
- Cien años de soledad
- 1984
- Matar a un ruiseñor
- Ensayo sobre la ceguera
- Los hombres me explican cosas
- Don Quijote de la Mancha
- El juego de ender
- Crónica de una muerte anunciada
- Rayuela
- El gran Gatsby
- El amor en los tiempos del cólera
- To Kill a Mockingbird

(Donde “autor#” corresponde a los objetos de la clase Autor creados anteriormente, por favor, refiérase a la parte de Autores para saber cual es cual)

Computadores: Los objetos de la clase computador están contruidos de la siguiente manera Computador(Nombre). Y los que ya están cargados en el sistema son:

- Samsung JX
- HP Pavilion
- Dell Inspiron
- Lenovo ThinkPad
- Asus VivoBook
- Acer Aspire

Prestamos: Los objetos de tipo préstamo están contruidos de la siguiente manera: Prestamo (recursoPrestado, fechaFin) y los que ya están cargados al sistema son los siguientes:

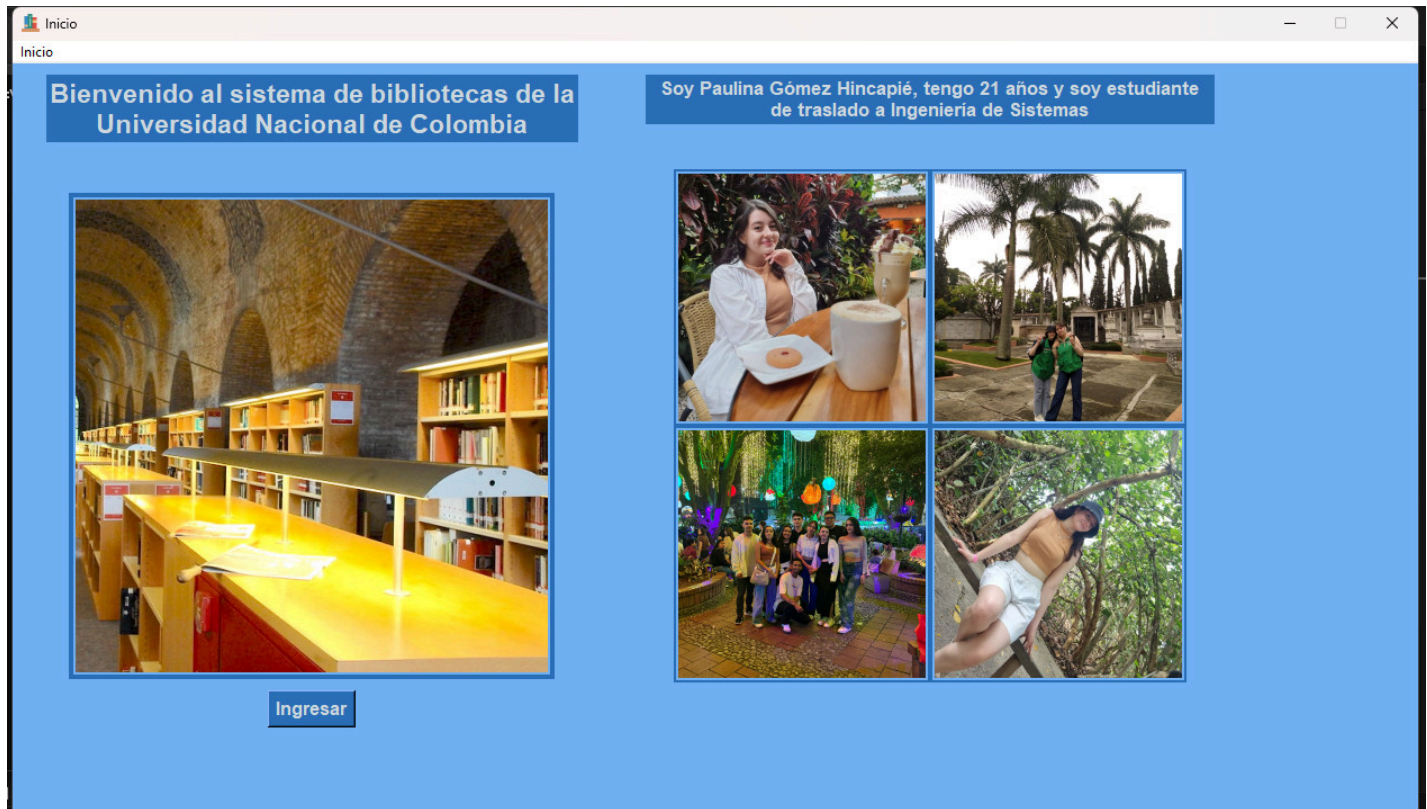
- 0, “Cien años de soledad”, Particular, fechaFin
- 1, “Dell Inspiron”, fechaFin.

Los objetos de tipo multa están contruidos de la siguiente manera: Multa(Tipo, costo, fecha) y los que ya están cargados al sistema son los siguientes:

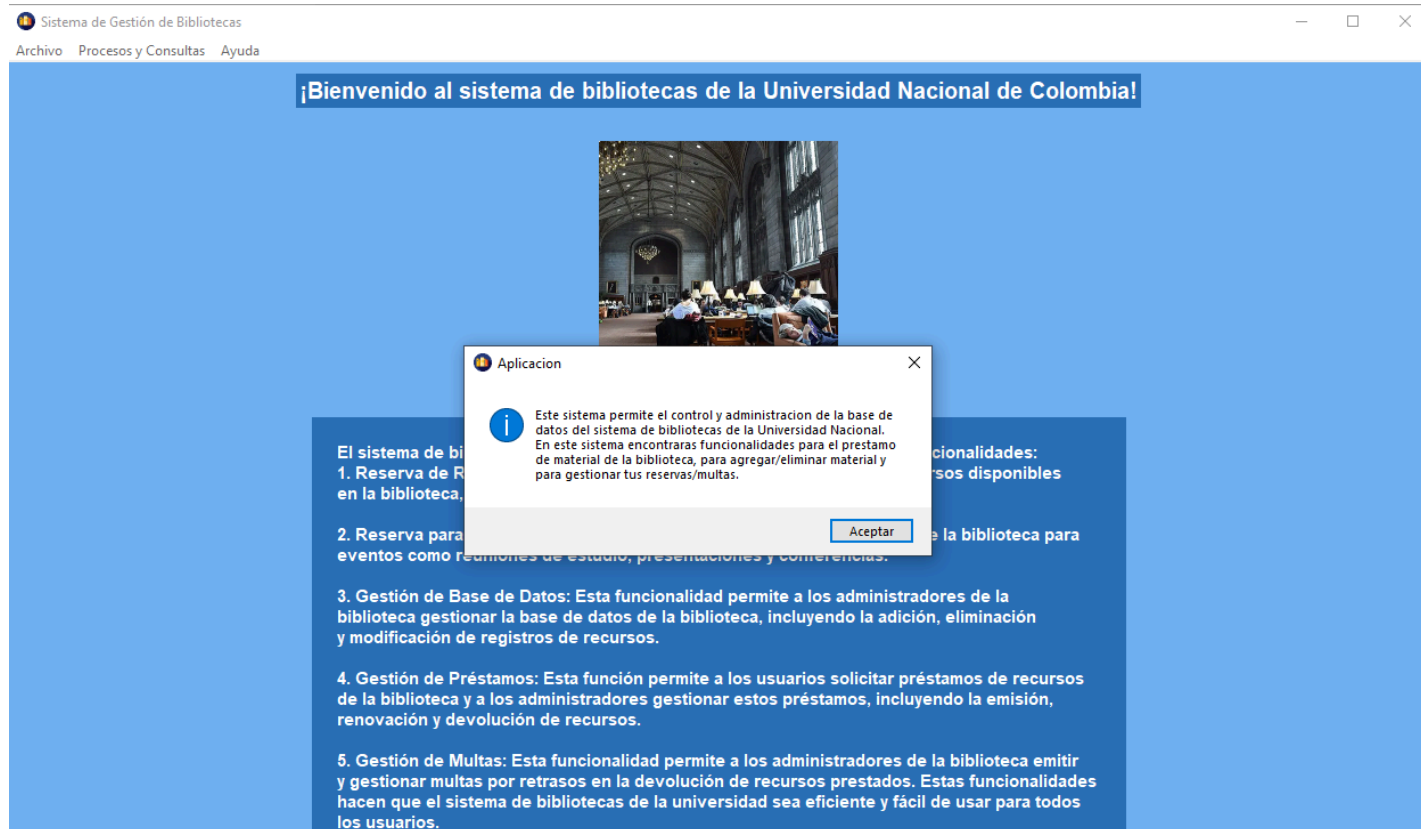
- 0. Multa de costo: 3000 con fecha impuesta de pago: 2023-11-21
- 1. Multa de costo: 10000 con fecha impuesta de pago: 2023-11-21

Ejecutable

Cuando el usuario abre el archivo ejecutable del programa, lo primero que ve en la pantalla es el siguiente encabezado con la bienvenida y una imagen de la biblioteca en la parte izquierda y en la derecha encontrará la biografía de cada uno de los miembros, cuenta con un menú en la parte superior del inicio donde hay dos opciones una llamada descripción la cual muestra cómo funciona el sistema y la opción salir que cierra por completo el sistema:



Después de ingresar al sistema se le va a dar la bienvenida al sistema bibliotecas de la universidad el usuario podrá ver las cinco funcionalidades que hay en la parte superior se encuentra un menú con las opciones archivo proceso en consulta y ayuda en la parte de archivo se va a encontrar la opción aplicación que nos dará un mensaje de que nos permite realizar el sistema y la opción salir que nos devuelve a la página principal



En la siguiente opción de **procesos y consultas** se desplegarán las opciones

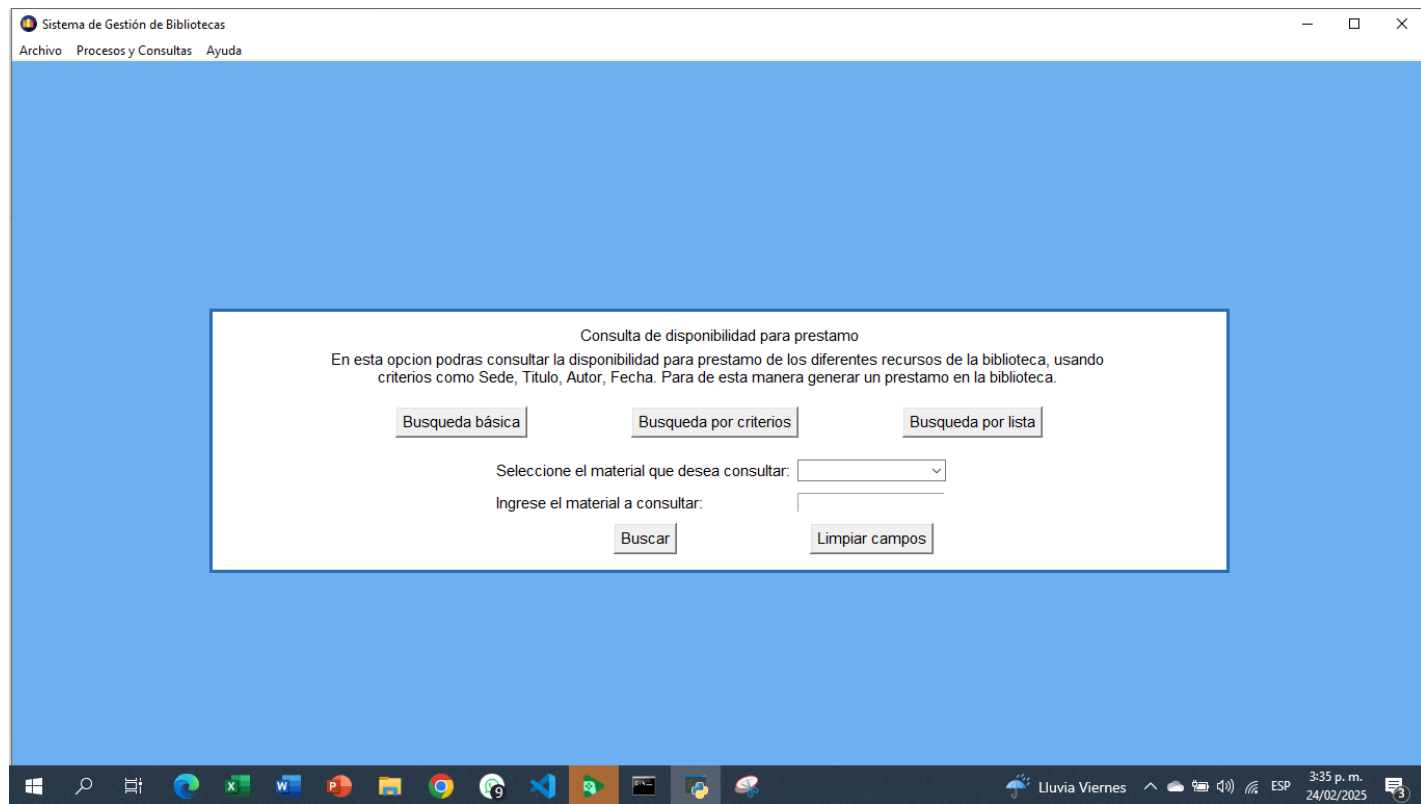
- préstamo de recursos.
- Reserva de recursos para eventos.
- Gestión de base de datos.
- Gestión de préstamos y reservas.
- Gestión de multas.

Aquí el usuario podrá escoger la opción que necesite.

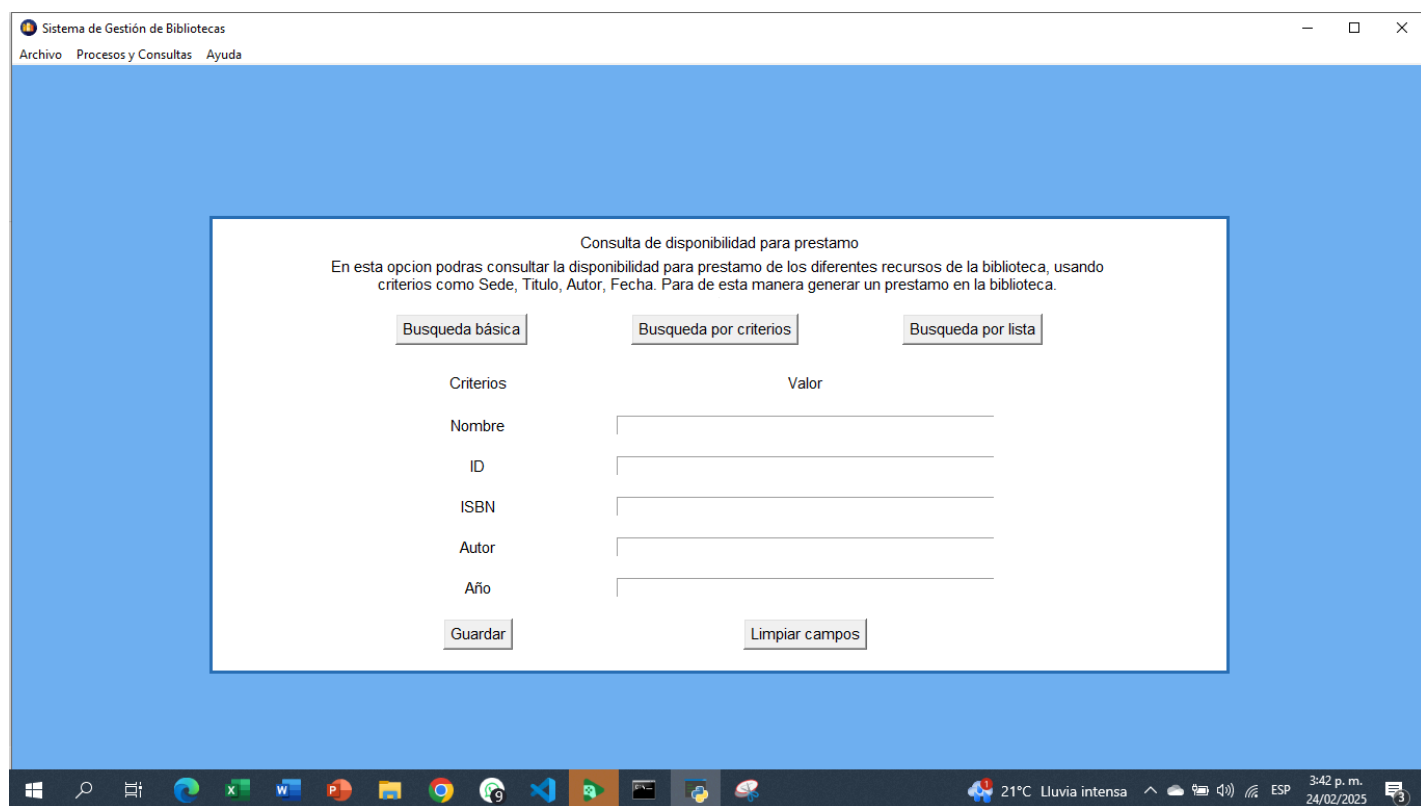


Prestamo de Recursos

Cuando el usuario selecciona Prestamo de Recursos, el sistema lo envia a la siguiente pagina donde se daran tres opciones de búsqueda, la opción de búsqueda básica donde pondrá el material que desea consultar (libro O computador) seguido el usuario pondrá el material que desea consultar



La opción de búsqueda por criterio donde el usuario podrá Buscar por nombre, ID ISBN autor o año.



La opción búsqueda por lista hay el usuario escoger entre computadores o libros le saldrá un listado de los objetos con los que cuenta la biblioteca según el caso.

Sistema de Gestión de Bibliotecas

Archivo Procesos y Consultas Ayuda

Consulta de disponibilidad para préstamo

En esta opción podrás consultar la disponibilidad para préstamo de los diferentes recursos de la biblioteca, usando criterios como Sede, Título, Autor, Fecha. Para de esta manera generar un préstamo en la biblioteca.

Búsqueda básica Búsqueda por criterios Búsqueda por lista

Seleccione un libro de la siguiente lista:

- 0. Sapiens: De animales a dioses
- 1. Harry Potter y la piedra filosofal
- 2. Cien años de soledad
- 3. 1984
- 4. Matar a un ruiseñor
- 5. Ensayo sobre la ceguera
- 6. Los hombres me explican cosas
- 7. Don Quijote de la Mancha
- 8. El juego de ender
- 9. Crónica de una muerte anunciada

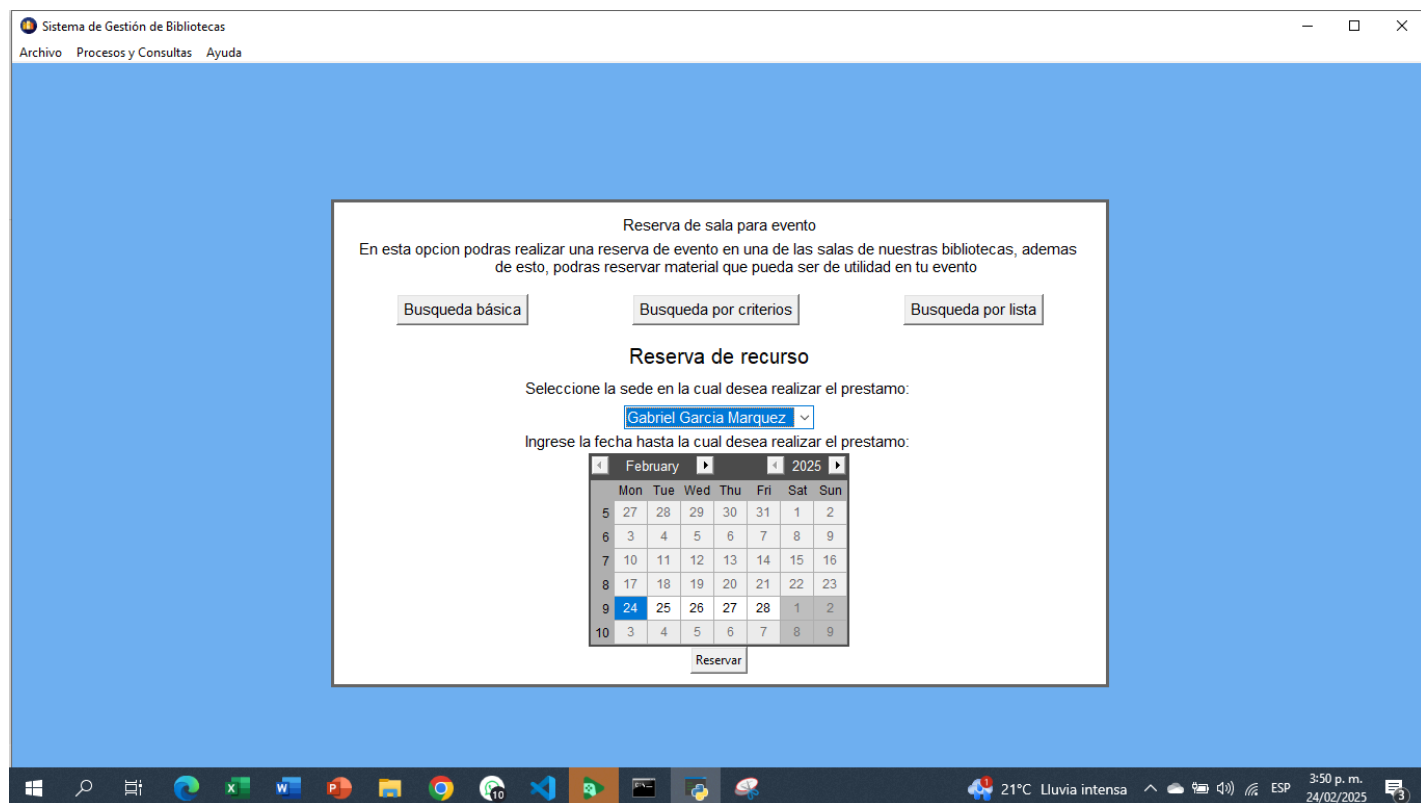
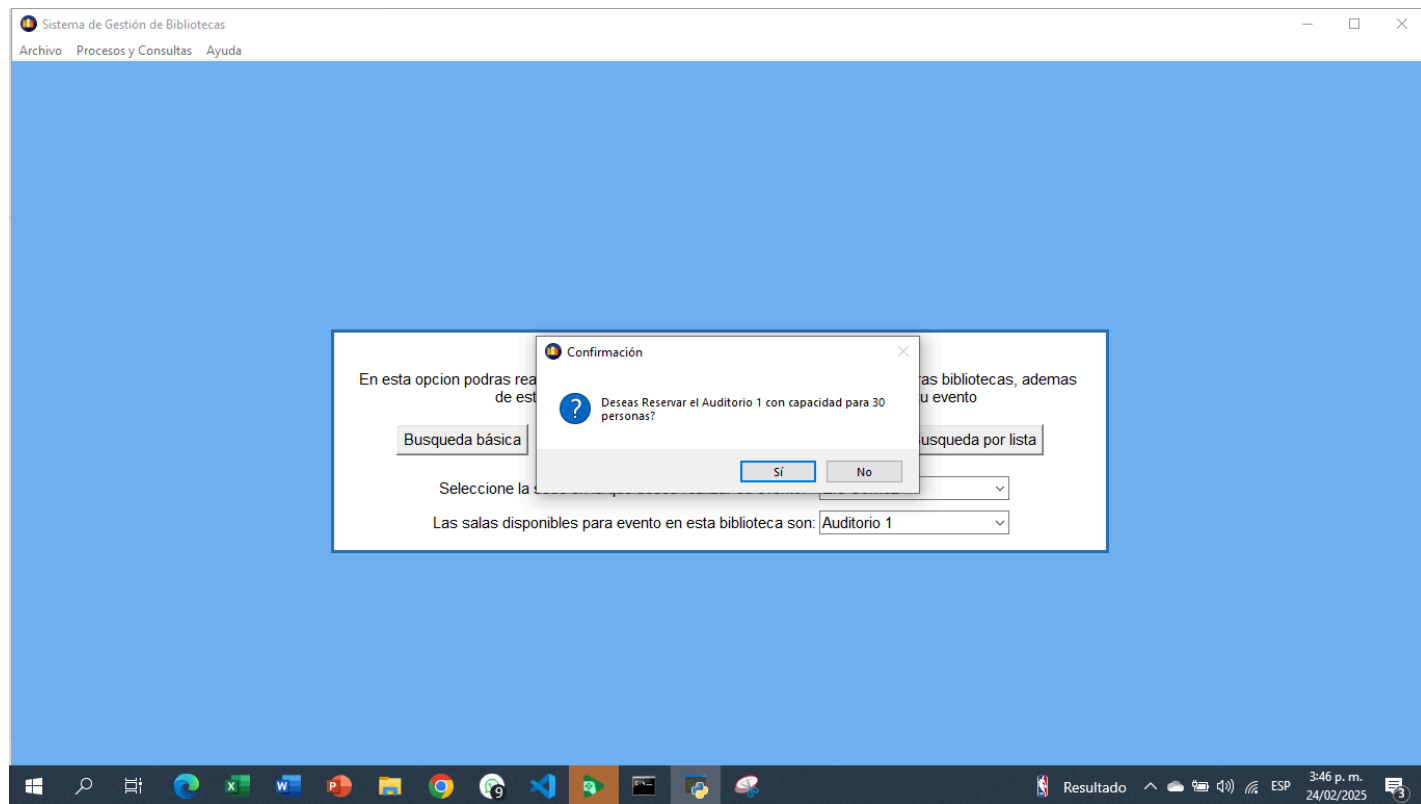
Criterio Valor

ID:

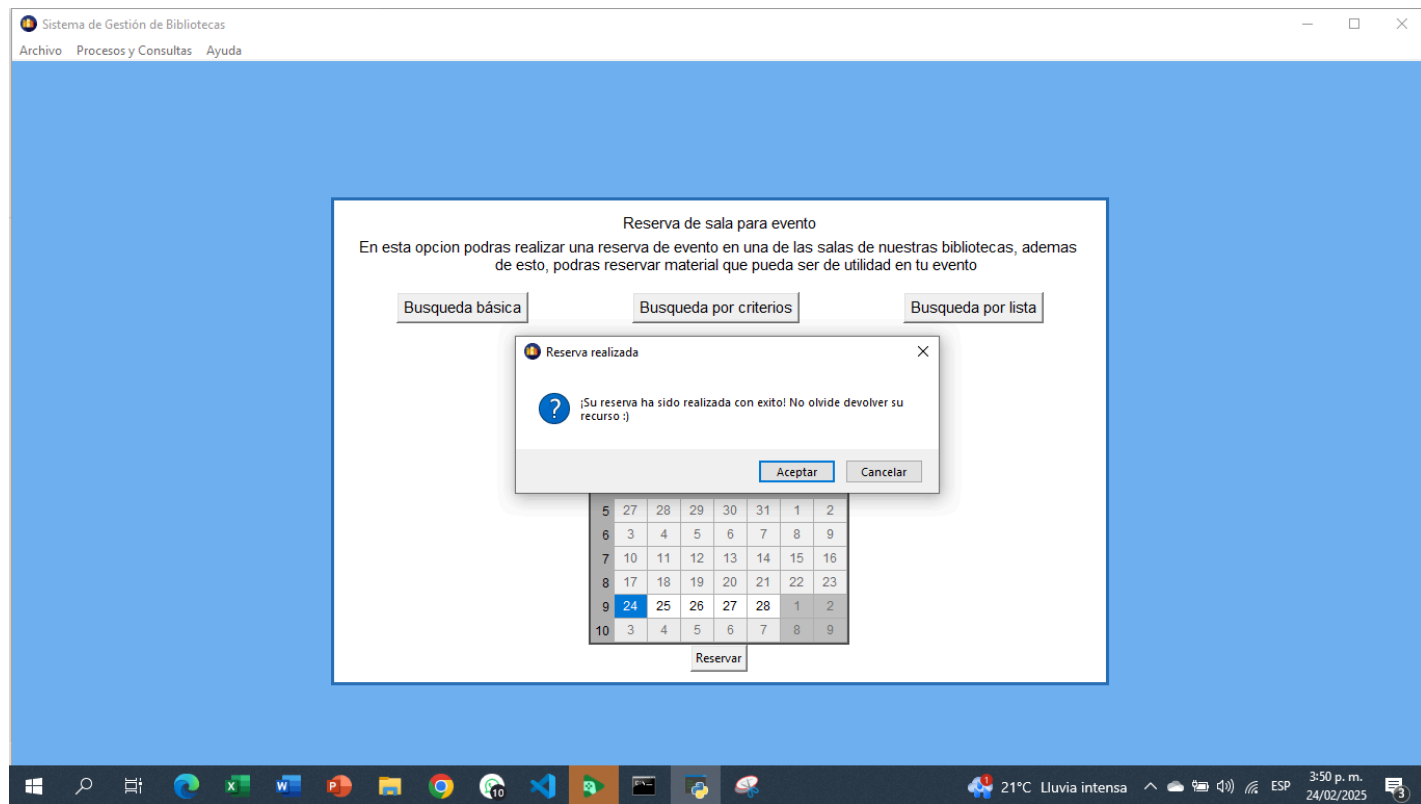
Enviar Limpiar campos

Reserva de Recursos para Eventos

En esta página se darán las tres opciones igualmente búsqueda básica búsqueda por criterio y búsqueda por lista donde escogerán el lugar donde desean el evento el auditorio al escoger el auditorio se les dará la capacidad que tiene el auditorio en un mensaje emergente y si el usuario le da aceptar continuará a elegir el material que desea reservar igual que en el proceso anterior al escoger el libro o computador que desea utilizar aparecerá una imagen donde va a seleccionar la fecha en la que sea realizar el evento.

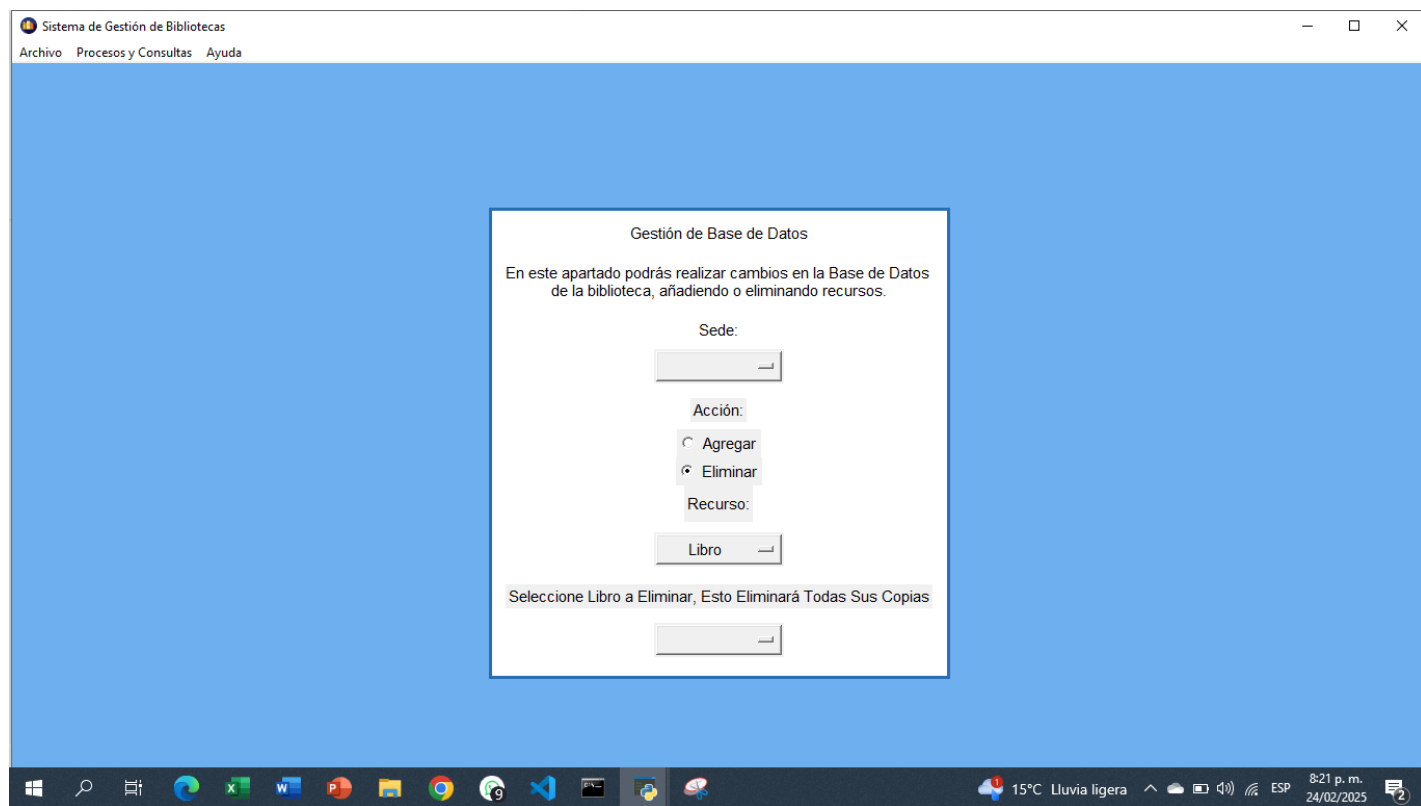


Al finalizar se enviara un mensaje donde indica que la reserva fue realizada con exito.



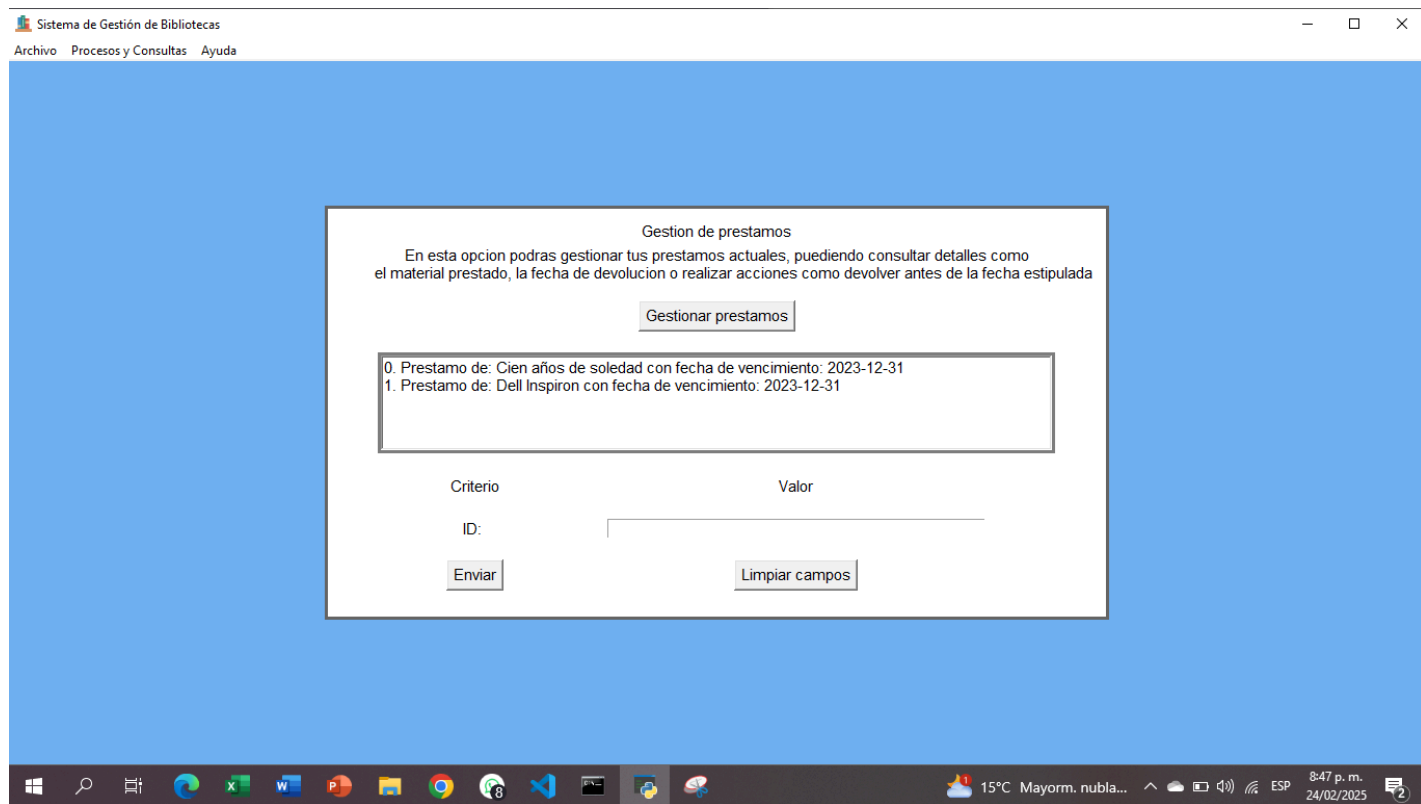
Gestion Base de Datos

En la opción de gestión base de datos vamos a encontrar la opción sede donde se escoger entre Medellín y Bogotá se podrá agregar o eliminar un recurso los recursos



Gestion de Prestamo y Reservas

En esta pestaña vamos a encontrar los recursos que tenemos prestados acompañados de la fecha de vencimiento podemos escoger con el ID si deseamos entregar alguno de los recursos



Gestion de Multas

En la ventana gestión de multas vamos a encontrar el motivo de la multa el valor y la fecha en la que se generó la multa y con el ID podemos realizar el pago

Gestion de multas

En esta opcion podras gestionar tus multas actuales ademas de pagarlas, en las siguientes opciones podras acceder a los detalles de cada multa asociada y realizar acciones sobre ellas

Gestionar multas

0. Multa de costo: 3000 con fecha impuesta de pago: 2023-11-21
1. Multa de costo: 10000 con fecha impuesta de pago: 2023-11-21

Criterio

Valor

ID:

Enviar

Limpiar campos