**INTRODUCCIÓN A LA PROGRAMACIÓN ORIENTADA A OBJETOS Y UML**

**BIBIOTECA**

**HANNA KATHERINE ABRIL GÓNGORA**

**KAROL ASLEY ORJUELA MAPE**

**UNIVERSIDAD MANUELA BELTRÁN**

**PROGRAMACIÓN ORIENTADA A OBJECTOS**

**DOCENTE**

**DIANA MARCELA TOQUICA RODRÍGUEZ**

**BOGOTÁ DC VIERNES 23 DE FEBRERO**

1. **PREGUNTAS ORIENTADORAS**
   1. En este apartado se realiza el análisis de los datos obtenidos, estos pueden ser de

formacualitativa o cuantitativa según la naturaleza de la práctica.

* + En el código, se adoptó una estructura de código organizada en clases y métodos. Esto nos permite modularizar la funcionalidad, facilitando la comprensión, el mantenimiento y la reutilización del código.
  + En el código, se desarrollaron funciones específicas para inscribir usuarios, consultar disponibilidad de libros y gestionar la biblioteca de manera eficiente. Esto nos permite ofrecer un servicio más ágil y satisfactorio para nuestros usuarios.
  + En el código se utilizaron bucles para iterar sobre colecciones de datos y automatizar procesos. Esto mejora la eficiencia y la legibilidad del código.
  1. ¿Cuáles fueron los aprendizajes obtenidos al realizar esta guía?, liste como mínimo 3 aprendizajes y relaciónelos con su futuro que hacer profesional.
  + Durante este trabajo, ampliamos nuestras habilidades en Python y profundizamos en los principios de la programación orientada a objetos. Esto nos permite abordar proyectos más complejos y escribir código más estructurado y mantenible.
  + Durante nuestro aprendizaje, adquirimos habilidades en la creación y uso de diagramas UML, aplicándolos con éxito en proyectos de desarrollo en Python. Esto nos permite diseñar y comunicar de manera más efectiva la arquitectura de nuestros programas.
  + Saber diseñar las estructuras básicas para sistemas simples como en este caso el de la biblioteca, donde se implementaron acciones básicas en un sistema de ese tipo.

1. **PREGUNTAS**
   1. ¿Cómo escribir programas que tengan un alto nivel de flexibilidad?

Para escribir programas con un alto nivel de flexibilidad en el análisis orientado a objetos:

* + Abstracción y Modularidad:

La abstracción representa las características esenciales de un objeto sin preocuparse por los detalles internos. Diseña clases y métodos que sean lo más abstractos posible. La modularidad divide tu código en módulos o clases independientes. Cada clase debe tener una responsabilidad específica. Esto facilita la reutilización y el mantenimiento.

* + Encapsulamiento:

Oculta los detalles internos de una clase y expone solo lo necesario. Utiliza modificadores de acceso (como public, private y protected) para controlar la visibilidad de los miembros de la clase.

El encapsulamiento protege el estado interno de los objetos y permite cambios internos sin afectar el código externo.

* + Herencia:

Utiliza la herencia para crear jerarquías de clases. Una clase puede heredar propiedades y métodos de otra.

La herencia permite reutilizar código y definir relaciones entre clases (por ejemplo, una clase “Empleado” hereda de una clase “Persona”).

* + Polimorfismo:

El polimorfismo permite que un objeto se comporte de diferentes maneras según el contexto. Utiliza interfaces, clases abstractas y métodos sobrescritos para lograr polimorfismo. Por ejemplo, una interfaz “Figura” puede tener métodos como “calcularÁrea()” que se implementan de manera diferente en las clases “Círculo” y “Rectángulo”.

* + Interfaces y Composición:

Las interfaces definen contratos que las clases deben cumplir. Utiliza interfaces para establecer comportamientos comunes. La composición implica combinar objetos más pequeños para formar objetos más complejos. Por ejemplo, una clase “Coche” puede contener objetos de las clases “Motor”, “Rueda” y “Transmisión”.

* + Manejo de Excepciones:

Implementa manejo de excepciones para lidiar con errores de manera elegante. Esto evita que el programa se bloquee y permite una recuperación adecuada. Utiliza bloques try, catch y finally para controlar situaciones excepcionales.

* 1. ¿Qué tipos de objetos deben existir en el programa?

En un programa de una biblioteca simple, varios tipos de objetos son esenciales para

su funcionamiento. Aquí están algunos de ellos:

* + Libros:

Representan las publicaciones impresas que contienen información sobre temas específicos. Cada libro tiene atributos como su código, título, autor y disponibilidad. Pueden ser prestados a los socios de la biblioteca.

* + Socios:

Los socios son las personas que se dan de alta en la biblioteca. Cada socio está caracterizado por un número de membresía, nombre y email. Pueden tomar prestados libros de la biblioteca.

* + Préstamos:

Los préstamos representan la acción de prestar un libro a un socio.

Cada préstamo está caracterizado por el código del libro prestado, el número de socio y la fecha del préstamo.

* + Clases y Métodos:

Utiliza programación orientada a objetos (POO) para crear clases como Libro, Usuario y Préstamo. Cada clase debe tener métodos para registrar, consultar y gestionar los libros, socios y préstamos.

* 1. ¿Cuáles son las propiedades (atributos) y comportamientos (métodos) de los objetos a crear?

Un sistema de biblioteca simple requiere una estructura organizada para gestionar los procesos relacionados con la biblioteca. A continuación, te presento algunos atributos y métodos que podrían ser necesarios para un sistema de biblioteca básico:

* + 1. Atributos:
  + Libro:
* Título: El nombre del libro.
* Autor: El autor o autores del libro.
* Género: La categoría a la que pertenece el libro (por ejemplo, ficción, no ficción, ciencia ficción, etc.).
* ISBN: El número de identificación único del libro.
* Disponibilidad: Indica si el libro está disponible para préstamo.
  + Prestatario:
* Nombre: El nombre completo del prestatario.
* Número de identificación: Un identificador único para el prestatario (por ejemplo, número de estudiante o cédula).
* Historial de préstamos: Registra los libros que el prestatario ha tomado prestados anteriormente.
  + Préstamo:
* Fecha de préstamo: La fecha en que se prestó el libro.
* Fecha de devolución: La fecha en que se debe devolver el libro.
* Estado del préstamo: Indica si el libro ha sido devuelto o está pendiente de devolución.
  + 1. Métodos:
  + Agregar libro: Permite registrar un nuevo libro en la biblioteca.
  + Eliminar libro: Elimina un libro de la base de datos.
  + Buscar libro por título o autor: Facilita la búsqueda de libros específicos.
  + Registrar préstamo: Registra un préstamo de libro a un prestatario.
  + Devolver libro: Actualiza el estado del préstamo cuando un libro se devuelve.

1. **EXPLICACIÓN DEL CÓDIGO EN PYTHON**

Este código es una simulación simple de un sistema de gestión de una biblioteca que permite a los usuarios inscribirse, buscar libros, solicitarlos y devolverlos, así como también permite al bibliotecario administrar los libros y usuarios de la biblioteca. Veamos cada parte del código:

* + *Clase: Biblioteca,* define atributos estáticos para almacenar información sobre usuarios (nombre, documento, email, etc.) y libros (titulo, categoría, autor, etc.).

Imagen que contiene Texto

Descripción generada automáticamente

* + *inscribir():* Agrega un nuevo usuario a la biblioteca, esta permite agregar al usuario mediante la petición de alguna información al usuario y el almacenaje de esta en sus respectivas listas.

Texto

Descripción generada automáticamente

* + *consultar():* Busca y muestra información de un usuario por su documento, la información mostrada es la que se almaceno respectivamente al momento del usuario inscribirse mediante el método *inscribir()* se imprime toda la información correspondiente excepto información importante y privada como la contraseña.

Texto

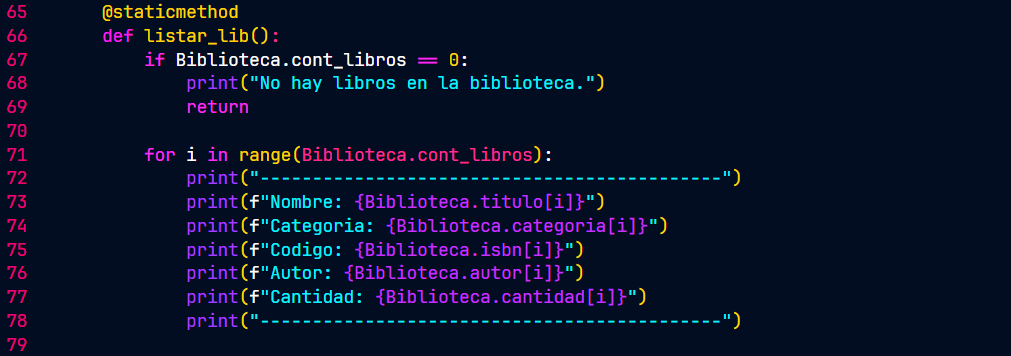
Descripción generada automáticamente

* + *consultar\_libro():* Busca y muestra información de un libro por título, categoría o autor, hay que tener en cuenta que tuvo que haberse inscrito un libro o libros con anterioridad para poder consultarlos, este método funciona de manera similar al de *consultar()* la diferencia es que en este método la búsqueda puede ser de diferentes maneras.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

* + *listar\_lib():* Muestra una lista de todos los libros en la biblioteca. Si no hay libros inscritos se arrojara que no hay ningún libro correspondiente a ese parámetro de búsqueda. Este hace una búsqueda en donde se almaceno el libro al momento de agregarlo en este caso una lista.



* + *listar\_usu*(): Muestra una lista de todos los usuarios inscritos, los cuales fueron almacenados al momento de inscribir en una lista.

Texto

Descripción generada automáticamente

* + *agregar():* Agrega un nuevo libro a la biblioteca, los cuales los usuarios podrán acceder mas adelante y los bibliotecarios podrán hacer consultas sobre estos.

Texto

Descripción generada automáticamente

* + *eliminar():* Remueve o almacena un libro de la biblioteca, se usa en el caso de que quieran descontinuar un libro o lo quieran almacenar.

Texto

Descripción generada automáticamente

* + *eliminar\_libro\_pedido():* Remueve un libro de la lista de libros pedidos por un usuario, este se usa al momento de que el usuario devuelva el libro, si este no lo devuelve este método no se usa, por lo que este método es dependiente del método *devolver\_lib()*

Texto

Descripción generada automáticamente

* + *Clase: Usuarios*, define un atributo estático para almacenar los libros pedidos por los usuarios, y tiene métodos que permiten realizar prestamos, devoluciones y revisar los libros que ese usuario pueda tener en su poder.

Imagen que contiene Patrón de fondo

Descripción generada automáticamente

* + *pedir\_lib():* Permite a un usuario solicitar un libro, este revisa si el libro esta disponible en la lista de libros en la biblioteca y así poder prestarlo.

Texto

Descripción generada automáticamente

* + *devolver\_lib():* Permite a un usuario devolver un libro, se debe tener en cuenta que si el libro no esta en la lista de libros prestados a dicho usuario no se va a poder devolverlo, ya que este método hace la comparación directa con ese método.

Texto

Descripción generada automáticamente

* + *listar\_lib\_usu():* Lista los libros que ha pedido un usuario, la lista incluye toda la información de libro.

Texto

Descripción generada automáticamente

* + Bucle principal: Un bucle *while* True que presenta un menú de opciones. Dependiendo de la opción elegida por el usuario, se llama a los métodos correspondientes de las clases Biblioteca o Usuarios.

Cada opción tiene un submenú, en especial Biblioteca y Usuarios, ya que estos tienes sus funciones por aparte.

En este bucle se llaman todas las clases y métodos definidos anterior mente para que el código funcione correctamente.

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

1. **EXPLICACION DEL CODIGO UML**

**Texto

Descripción generada automáticamente**

* + *@startuml* Biblioteca inicia un diagrama UML llamado “Biblioteca”.
  + *class Biblioteca* , comienza la definición de una clase llamada “Biblioteca”.
  + Las siguientes líneas definen los atributos de la clase “Biblioteca”, que incluyen listas de strings y datetimes, así como enteros.
  + *\_\_init\_\_(...)* es el constructor de la clase que inicializa los atributos con los valores proporcionados cuando se crea un objeto de esta clase.
  + Las siguientes líneas son métodos (funciones) que pertenecen a la clase “Biblioteca”. Estos métodos no retornan nada (None).
  + *class Usuarios*, comienza la definición de otra clase llamada “Usuarios”.
  + Nuevamente, las siguientes líneas definen los atributos y métodos para esta nueva clase.
  + *@enduml* indica el final del diagrama UML.

1. **DIAGRAMA DEL CODIGO EN UML**

**Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente**

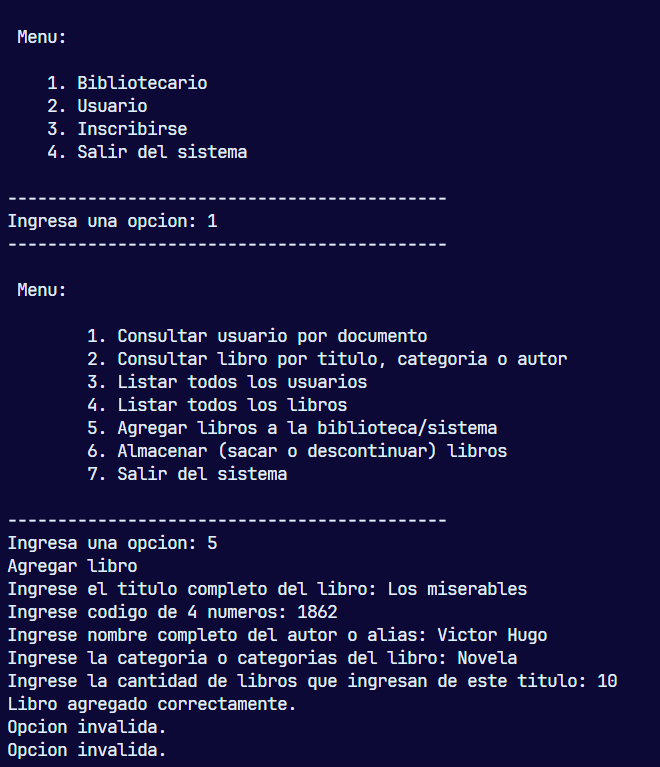
1. **EXPLICACION Y USO**

Es un proyecto en donde se implementa un pequeño sistema de biblioteca, con usuarios, libros y la biblioteca en sí.

En este sistema cada uno tiene accesos a cosas diferentes, por ejemplo, la biblioteca tiene accesos a funciones como inscribir usuarios, inscribir libros, revisar la lista tanto de usuarios inscritos como de libros agregados, hacer una búsqueda en especifico de los dos, siendo la de usuario mediante su número de identificación y la de los libros mediante más parámetros como lo es el título, autor o categoría, la biblioteca también tiene la habilidad de almacenar o retirar libros.

En cuanto al usuario después de inscrito este va a necesitar una contraseña para poder acceder al sistema de usuarios, después de ingresado este podrá realizar procesos como pedir libros, devolver libros, revisar libros que estén en su poder.

1. **DEMOSTRACION DE PRUEBAS Y FUNCIONAMIENTO DEL CODIGO**
   1. Prueba 1.



En esta prueba se demuestra la inscripción de un libro, en donde se accede al menú de bibliotecario, y después encuentra el menú de las acciones que este puede realizar.

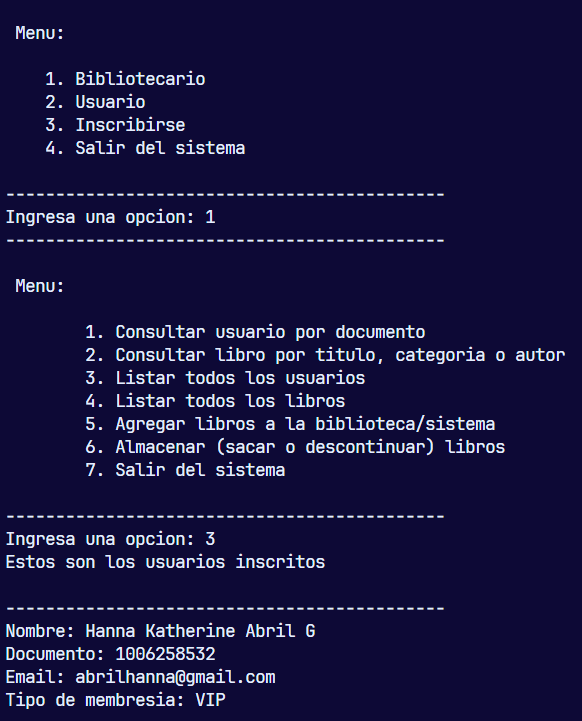
* 1. Prueba 2.

Texto

Descripción generada automáticamente

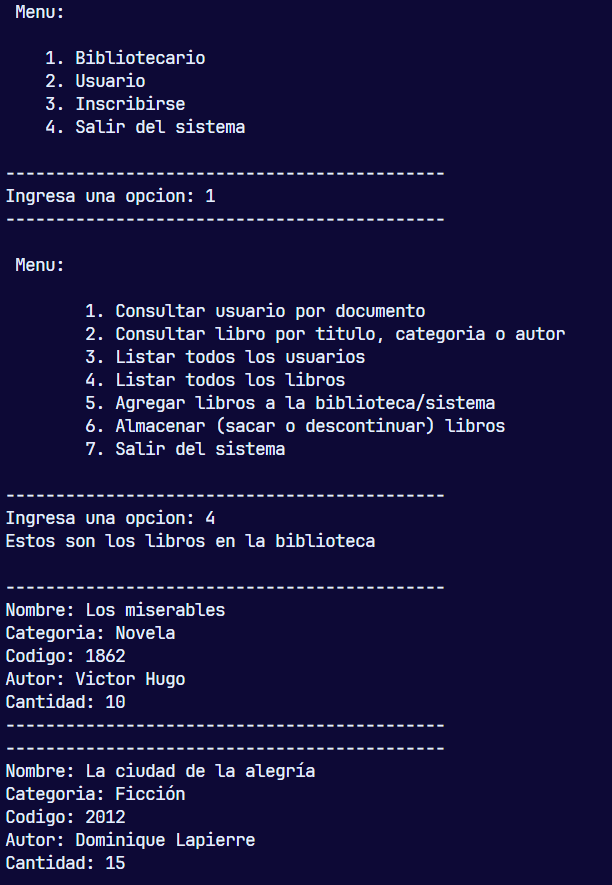
En esta prueba se muestra la inscripción de un usuario nuevo, directamente desde el menú principal.

* 1. Prueba 3.



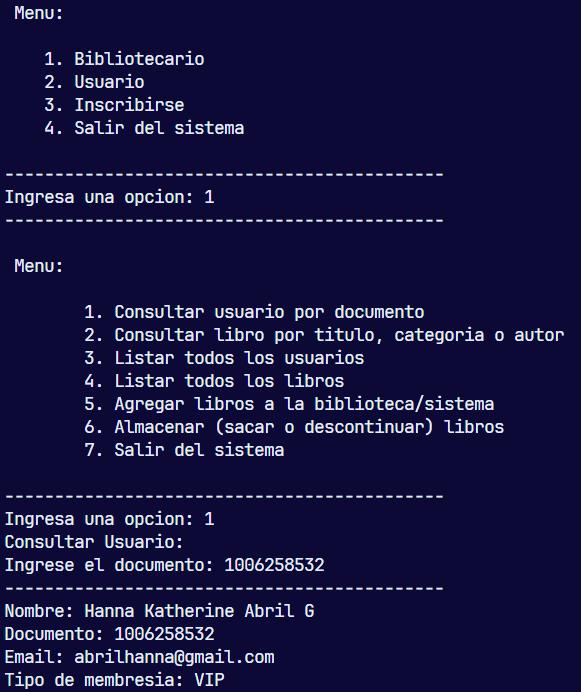
Se lista los usuarios inscritos, en este caso como solo se inscribió uno anteriormente, solo muestra ese usuario y toda su información, excepto información importante como la contraseña.

* 1. Prueba 4.



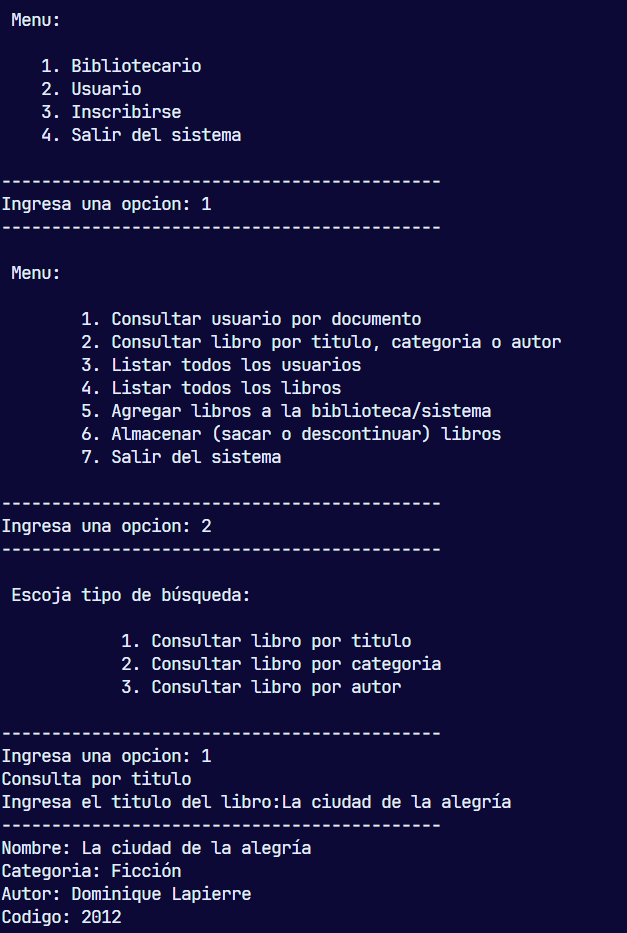
En esta prueba se listan todos los libros agregados, en este caso podemos apreciar que hay dos libros que se agregaron anteriormente, para motivos de esta prueba.

* 1. Prueba 5.



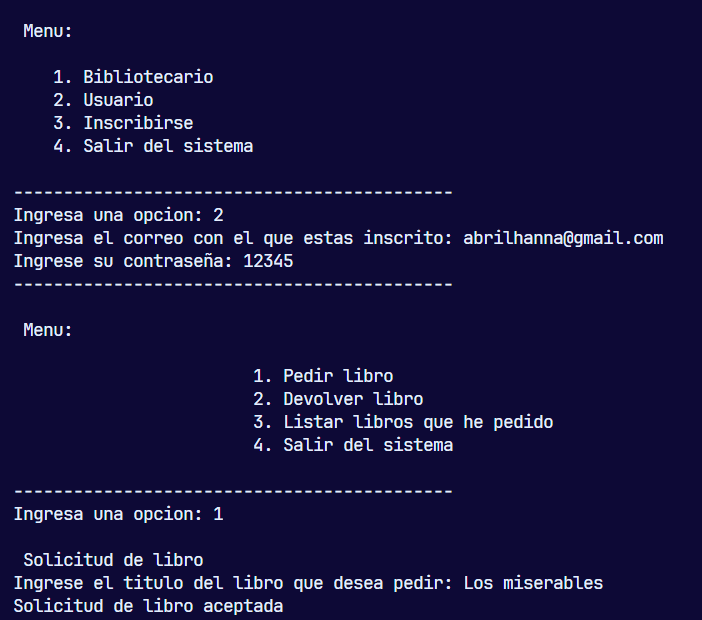
En esta prueba se consulta el usuario mediante el numero de su documento, el programa hace una búsqueda en las listas de la información del usuario y retorna la coincidencia de este.

* 1. Prueba 6.



Se consulta un libro en este caso por el título, la búsqueda también se puede hacer con categoría o autor.

* 1. Prueba 7.



En esta prueba se realiza la comprobación en donde un usuario pide un libro de la biblioteca, para eso debe ingresar con sus credenciales las cuales puso en el momento de la inscripción y después solicita el libro si este esta disponible le acepta el préstamo, si no esta disponible le dice q el titulo no esta disponible.

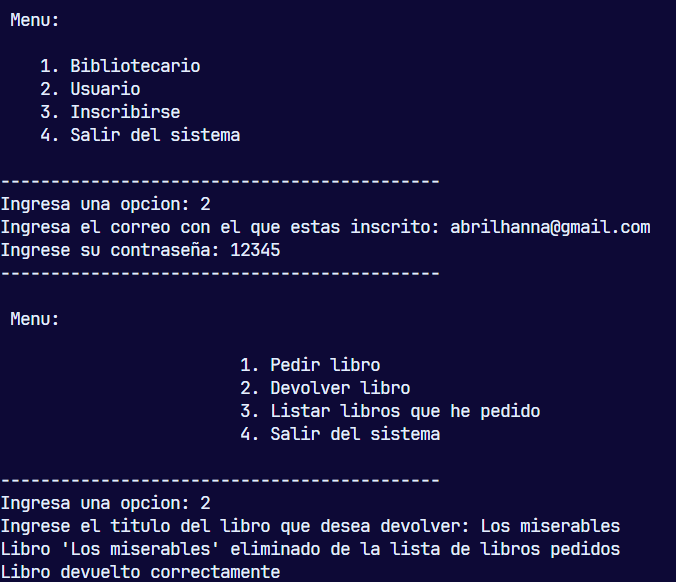
* 1. Prueba 8.

Texto

Descripción generada automáticamente

Se lista los libros que el usuario a pedido hasta el momento.

* 1. Prueba 9.



El usuario devuelve un libro, el sistema revisa en la lista de prestados que tiene el usuario lo remueve de ahí y lo devuelve a la lista de biblioteca.

* 1. Prueba 10.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Se comprueba que se puede salir del sistema.

1. **DECISIÓN DE DISEÑO O CONSIDERACION IMPORTANTE**

La principal decisión o consideración importante al hacer el diseño, fue tratar de saber que atributo pertenecía a que clase, y que métodos era pertenecientes a cada clase. Después de considerar eso fue bastante fácil el resto, también se le agregaron mas funciones que nosotros consideramos son necesarios de un sistema básico de biblioteca.

1. **PILARES DE POO IMPLEMENTADOS** 
   1. Clases: En el código, la clase Biblioteca actúa como un molde que contiene propiedades y métodos para construir objetos relacionados con usuarios y libros. Las listas como nombre, documento, email, etc., representan las características de los usuarios, mientras que las listas como título, categoría, autor, etc., representan las características de los libros. Las clases son la base para crear múltiples instancias u objetos.
   2. Polimorfismo: Aunque no se implementa explícitamente aquí, el polimorfismo permite definir métodos con el mismo nombre, pero con diferentes comportamientos en clases derivadas. Por ejemplo, podríamos tener un método consultar\_libro que se comporte de manera diferente según el tipo de libro o la categoría.

En resumen, este código muestra algunos aspectos de la POO, como la creación de una clase, la definición de propiedades y métodos, y la búsqueda de usuarios y libros. Para una implementación más completa, podríamos explorar más a fondo los pilares de la POO y aplicarlos de manera más extensa en el diseño de la biblioteca virtual.

1. **CONCLUCIONES**

En conclusión, creemos que este trabajo fue bastante útil para nosotros comprender mejor la programación orientada a objetos, y a perfeccionar mejor nuestros conocimientos sobre este e implementarlos en un caso aplicado a la vida real.