

29 DE NOVIEMBRE DE 2023



# Sistema de Gestión de Biblioteca

## REPORTE ESCRITO DEL PROYECTO DESARROLLO EN JAVA

VALENTINA PINEDA BARRÓN  
215365  
Instituto Tecnológico Autónomo de México

## Contenido

INTRODUCCIÓN .....	2
Contexto.....	2
Identificación del problema.....	2
Objetivos.....	2
Organización del documento.....	2
ANÁLISIS.....	3
Requisitos funcionales.....	3
Restricciones .....	3
DISEÑO.....	4
Descripción completa del diseño.....	4
Estándares Utilizados .....	6
IMPLEMENTACION.....	9
Descripción completa de la implementación.....	9
GUI.....	9
Manual Usuario .....	9
PRUEBAS Y RESULTADOS .....	10
Descripción de las pruebas aplicadas (casos de prueba).....	13
Resultados Obtenidos.....	13
Análisis de los resultados .....	20
CONCLUSIONES.....	20
REFERENCIAS .....	21
APÉNDICE (código).....	21

## INTRODUCCIÓN

### Contexto

En la actualidad, las bibliotecas juegan un papel crucial como centros de investigación y conocimiento. Las grandes cantidades de información que manejan nos han permitido obtener una fructífera y variada gama de pensamientos innovadores, de los cuales podemos referenciarlos en nuestro aprendizaje. Aunque la administración manual de una biblioteca presenta desafíos para aquellos encargados de gestionarla, debido a la información que manejan diariamente, por lo que desde la década de los 80, las bibliotecas han implementado sistemas automatizados<sup>1</sup> en sus instalaciones, mejorando así su eficiencia y capacidad para brindar servicios más efectivos.

### Identificación del problema

La administración manual de una biblioteca puede ser un proceso muy demandante en tiempo, propenso a errores y poco eficiente. Por esta razón, el sistema de gestión de una biblioteca está dirigido hacia el bibliotecario. Los bibliotecarios se encargan de diversas tareas, como dar de alta a nuevos usuarios y a los materiales que se incluyan en el inventario, así como dar de baja a los mismos; buscar información solicitada; verificar la disponibilidad de la información requerida; y proporcionar una lista de la información dentro del inventario a los usuarios, entre otras funciones.

### Objetivos

Al concluir el desarrollo de mi proyecto, me propongo reflejar en mi trabajo las habilidades que he adquirido a lo largo del curso:

- Realizar una interfaz amigable, sencilla y eficaz, para el uso del bibliotecario.
- Implementar funciones básicas al sistema que permitan recuperar características singulares del local.
- Automatizar el seguimiento de inventario, préstamos y devoluciones para mejorar la eficiencia operativa del personal bibliotecario.
- Facilitar la creación de perfiles de usuario.
- Obtención de un registro de los préstamos, tanto de usuarios como materiales.

### Organización del documento

El informe se divide en ocho secciones, cada una subdividida en subsecciones detalladas a continuación:

- I. Introducción: presenta un breve contexto de por qué elegí el tema al que va dirigido mi proyecto, aborda la problemática por la cual una biblioteca requiere un sistema automatizado, establece los objetivos del proyecto y, por último, explica la organización del documento.
- II. Análisis del proyecto: enumera los métodos para la función del proyecto, seguido de las restricciones que presenta el proyecto, considerando que la persona que interactuará con el sistema lo hará de manera adecuada.

---

<sup>1</sup> González Fernández-Villavicencio, N. (2009, p. 13)

- III. Diseño del proyecto: descripción detallada del diseño, explicando el uso de las clases y atributos, así como la relación entre los métodos. Se incluirá un diagrama UML que ejemplificará de manera gráfica los estándares utilizados.
- IV. Implementación del proyecto: detalla cómo se llevó a cabo la codificación, describe el diseño y funcionamiento de la interfaz gráfica y proporciona instrucciones detalladas para la interacción del usuario con el sistema.
- V. Pruebas y resultados: enumera y describe las pruebas utilizadas para evaluar el funcionamiento del programa, muestra los resultados de las pruebas y, finalmente, analiza los resultados obtenidos y discute desviaciones o problemas encontrados durante la prueba.
- VI. Conclusiones: resume logros, aprendizajes y posibles áreas de mejora.
- VII. Referencias: expone las fuentes bibliográficas, documentos o recursos que consulté durante el desarrollo del proyecto.
- VIII. Código fuente del proyecto.

## ANÁLISIS

### Requisitos

### funcionales

El proyecto que deseaba implementar requiere una clase principal llamada "Biblioteca" y dos clases secundarias denominadas "Usuario" e "Informacion". La clase "Biblioteca" posee diversos métodos, incluyendo getters y setters que permiten recuperar o modificar información esencial de la biblioteca, es decir, atributos. Además, cuenta con funcionalidades mínimas y, esencialmente, proporciona la base para una buena sistematización de una biblioteca.

La clase "Biblioteca" incluye métodos que permiten interactuar tanto individualmente como en conjunto con el inventario y los usuarios. Para interactuar con el inventario, se han implementado funciones como añadir material al inventario, quitar material del inventario, añadir un ejemplar a un material específico, recuperar el historial de préstamos de un material y realizar búsquedas por código, título, autor o año.

En cuanto a la interacción con los usuarios, la clase "Biblioteca" proporciona métodos para crear un usuario, eliminar un usuario, recuperar el historial de préstamos de un usuario, recuperar los préstamos actuales de un usuario, y buscar usuarios por nombre o cuenta.

Finalmente, la clase "Biblioteca" también ofrece métodos para interactuar con usuarios e inventario en conjunto, como realizar préstamos y regresar préstamos. Este diseño proporciona una estructura organizada y funcional para el manejo completo de una biblioteca.

### Restricciones

Las restricciones del programa están vinculadas a las limitaciones específicas de diseño o, al menos, hasta donde alcanza mi conocimiento. Las búsquedas en la biblioteca deben realizarse de manera individual, es decir, mediante botones dedicados a diferentes parámetros. Cada parámetro de búsqueda tiene su propio método asociado. Además, se espera que las búsquedas sean exactas, sin permitir errores ortográficos o búsquedas por palabras clave.

En esta biblioteca, se limita exclusivamente a libros para simplificar el proyecto. El inventario tiene un límite de 500 libros, y el sistema tiene un límite de 100 usuarios. No se ha implementado un

sistema de contraseñas para garantizar el acceso exclusivo del bibliotecario a la interfaz, y los usuarios tampoco requieren una contraseña para realizar préstamos.

Adicionalmente, es importante destacar que los campos de entrada para la información no pueden ser muy largos, ya que el sistema trabaja con *Strings* e *ints*, no con *longs*. No hay un límite establecido para el año actual, permitiendo la inclusión de cualquier año. El sistema tampoco verifica si se registra dos veces el mismo libro o usuario, lo que podría generar duplicados. Además, no existe un límite de duración para los préstamos adquiridos.

## DISEÑO

### Descripción completa del diseño

#### INFORMACION

La clase “Informacion” representa los materiales de la biblioteca. Cada instancia de esta clase tiene atributos como título (título del material), autor (autor del material), año (el año de publicación del material), ejemplares (el número total de ejemplares del material), ejemplares disponibles (la cantidad de ejemplares disponibles para préstamo), historial de préstamos (un historial de préstamos que registra las cuentas de los usuarios que han tomado prestado este material y la cantidad de veces que se ha prestado), y código (un identificador único para cada material).

Esta clase principalmente está formada por *getters* para acceder a los diferentes atributos de cada material particular en la clase de biblioteca, también cuenta con funcionalidad mínima. Sin embargo, los métodos más relevantes podrían ser:

- `getDisponibilidad()`: devuelve un resultado *boolean* (*true* o *false*), si los ejemplares disponibles son mayores a cero.
- `añadirEjemplarDisponible()`: incrementa la cantidad de ejemplares disponible una unidad. Este método va a ser usada cuando un ejemplar se ha regresado
- `retirarEjemplarDisponible()`: reduce la cantidad de ejemplares disponibles una unidad. Cuando en la biblioteca se haga un préstamo
- `añadirHistorialInfo(int)`: Se ingresa la cuenta de la persona que realizó el préstamo al historial de préstamos del material, e indica la cantidad de veces que esa persona ha solicitado ese material.

#### USUARIO

La clase “Usuario” representa los usuarios de la biblioteca. Cada instancia de esta clase tiene atributos como nombre (nombre de usuario), correo (correo del usuario), historial de préstamos (un historial de préstamos que registra los códigos de los materiales que ha tomado prestado el usuario y la cantidad de veces que lo ha pedido prestado), préstamos actuales (clave de los materiales que tiene en posesión el usuario), MAX (la cantidad máxima de materiales que un usuario puede tener en préstamo al mismo tiempo), contador préstamos (la cantidad actual de materiales que el usuario tiene en préstamo) y cuenta (un identificador único para cada usuario).

Esta clase principalmente está formada por *getters* para acceder a los diferentes atributos de cada usuario particular en la clase de biblioteca, también cuenta con funcionalidad mínima. Sin embargo, los métodos más relevantes:

- `anadirHistorialUsuario(int)`: Se ingresa el código del material que está siendo prestado al historial de préstamos del usuario.
- `anadirPrestamo(int)`: Añade el código del material prestado en préstamos actuales y utiliza el método “`anadirHistorialUsuario()`”. Siempre y cuando, el usuario tenga menos de diez préstamos actuales y no este solicitando el mismo material.
- `regresarPrestamo(int)`: Elimina el código del material prestado en el atributo préstamos actuales. Cuando se realiza una devolución de un préstamo en la biblioteca.

## BIBLIOTECA

La clase “Biblioteca” parece ser la clase principal que gestiona la biblioteca en su conjunto. Cada instancia de esta clase tiene atributos como nombre (nombre de la biblioteca), dirección (dirección de la biblioteca), bibliotecario (nombre del bibliotecario), usuarios (un arreglo de usuarios que la biblioteca tiene registrados), inventario (un arreglo de materiales disponibles en la biblioteca), contador usuarios (la cantidad actual de usuarios registrados), contador inventario (la cantidad actual de materiales en el inventario).

La clase se conforma por métodos: *getters* y *setters*, funcionalidad mínima, funcionalidad de apoyo y funcionalidad del sistema. Los métodos más destacables son:

### Funcionalidad Información

- `añadirMaterial(String, String, int)`: añade un material al inventario de forma ordenada por título.
- `quitarMaterial(int)`: elimina un material del inventario dado su código. Devuelve un resultado *boolean* (*true* o *false*), dependiendo si el código está en el inventario. (Se apoya en el método “`getCodigo()`” de la clase “*Informacion*”)
- `anadirEjemplar(int)`: añade un ejemplar al material en el inventario dado su código. Devuelve un resultado *boolean* (*true* o *false*), dependiendo si el código está en el inventario. (Se apoya de los métodos “`getCodigo()`” y “`añadirEjemplar()`” de la clase “*Informacion*”).
- `historialPrestamosInformacion(int)`: Devuelve el historial de préstamos de un material dado su código. Devuelve un resultado *boolean* (*true* o *false*), dependiendo si el código está en el inventario. (Se apoya de los métodos “`getCodigo()`” y “`getHistorialPrestamos()`” de la clase “*Informacion*”).
- `buscarInformacionCodigo(int)`: devuelve la información completa de un material dado su código. (Se apoya de los métodos “`getCodigo()`” y “`to String()`” de la clase “*Informacion*”).
- `buscarInformacionTitulo(String)`: devuelve la información completa de un material dado su título. (Se apoya en los métodos “`getTitulo()`” y “`to String()`” de la clase “*Informacion*”).
- `buscarInformacionAutor(String)`: devuelve la información completa de un material dado su autor. (Se apoya en los métodos “`getAutor()`” y “`to String()`” de la clase “*Informacion*”).
- `buscarInformacionAnio(int)`: devuelve la información completa de un material dado su año de publicación. (Se apoya en los métodos “`getAnio()`” y “`to String()`” de la clase “*Informacion*”).

### Funcionalidad Usuario

- crearUsuario(String, String): añade un usuario al sistema y regresa la cuenta del usuario. (Se apoya en el método “getCuenta()” de la clase “Usuario”).
- quitarUsuario(int): Elimina un usuario del sistema dado su cuenta. (Se apoya en el método “getCuenta()” de la clase “Usuario”).
- historialPrestamosUsuario(int): Devuelve el historial de préstamos de un usuario dado su número de cuenta. (Se apoya en los métodos “getHistorialPrestamo()” y “getCuenta()” de la clase “Usuario”).
- historialPrestamosUsuario(int): Devuelve el historial de préstamos actuales de un usuario dado su número de cuenta. (Se apoya en los métodos “getPrestamosActuales()” y “getCuenta()” de la clase “Usuario”).
- buscarUsuarioNombre(String): devuelve la información completa de un usuario dado su nombre. (Se apoya en los métodos “getNombre()” y “to String()” de la clase “Usuarios”).
- buscarUsuarioCuenta(int): devuelve la información completa de un usuario dado su cuenta. (Se apoya en los métodos “getCuenta()” y “to String()” de la clase “Usuario”).

### Funcionalidad Información y Usuarios

- realizarPrestamo(int,int): Realiza un préstamo dado el número de cuenta del usuario y el código del material. (Se apoya en los métodos “getDisponibilidad()”, “getEjemplares()”, “getEjemplaresDisponibles()”, “anadirPrestamo()”, “retirarEjemplarDisponible()” y “anadirHistorialInfo” de las clases “Informacion” y “Usuario”).
- regresarPrestamo(int,int): Regresa un préstamo dado el número de cuenta del usuario y el código del material. (Se apoya en los métodos “getEjemplares()”, “getEjemplaresDisponibles()”, “regresarPrestamo()” y “anadirEjemplarDisponible()” de las clases “Informacion” y “Usuario”).

## Estándares Utilizados

Informacion
<ul style="list-style-type: none"><li>- titulo: String</li><li>- autor: String</li><li>- anio: int</li><li>- ejemplares: int</li><li>- ejemplaresDisponibles: int</li><li>- historialPrestamos: ArrayList &lt;Integer&gt;</li><li>- consecutivo: static int</li><li>- codigo: int</li></ul>

```

+ Informacion(String, String, int)
+ getTitulo():String
+ getAutor():String
+ getAnio():int
+ getEjemplares():int
+ getEjemplaresDisponibles():int
+ getDisponibilidad():boolean
+ getHistorialPrestamos():String
+ getConsecutivo():int
+ getCodigo():int
+ toString():String
+ equals(Obj):boolean
+ compareTo(Obj):int
+ anadirEjemplarDisponible()
+ retirarEjemplarDisponible()
+ anadirHistorialInfo(int)
    
```

Usuario
<ul style="list-style-type: none"> <li>- nombre: String</li> <li>- correo: String</li> <li>- historialPrestamos: ArrayList&lt;Integer&gt;</li> <li>- prestamosActuales: int[ ]</li> <li>- MAX: final int</li> <li>- consecutivo: static int</li> <li>- contPrestamos: int</li> <li>- cuenta: int</li> </ul>
<pre> +Usuario(String,String) +getNombre():String +getCuenta():int +getHistorialPrestamos():String +getConsecutivo():int +getContPrestamos():int + toString():String + equals(Obj):boolean + compareTo(Obj):int +anadirHistorialUsuario(int) +anadirPrestamo(int):boolean +regresarPrestamo(int):boolean         </pre>

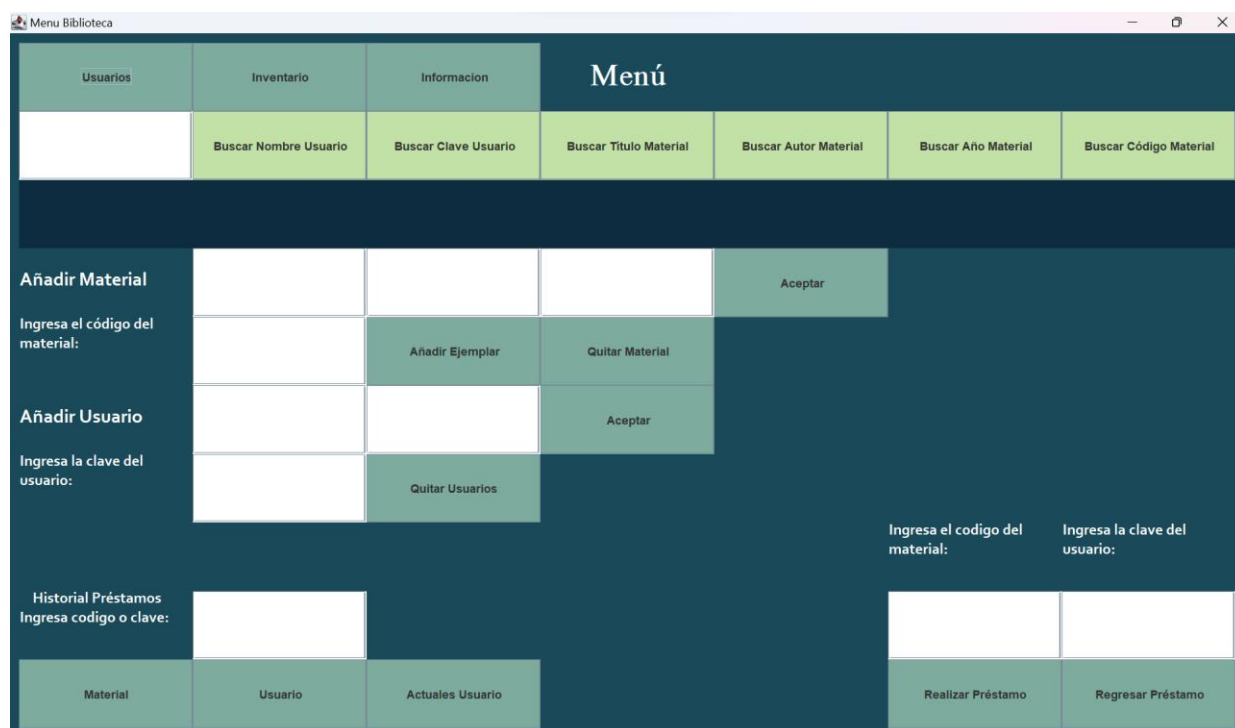
Biblioteca
<ul style="list-style-type: none"> <li>- nombre: String</li> <li>- direccion: String</li> <li>- bibliotecario: String</li> <li>- usuarios: Usuario[ ]</li> <li>- inventario: Informacion[ ]</li> <li>- USU: final int</li> <li>- INV: final int</li> <li>- contUsu: int</li> <li>- contInv: int</li> </ul>
<pre> +Biblioteca(String,String,String) +getNombre():String +getDireccion():String +getBibliotecario():String +getUsuarios():String +getInformacionParticular():String +getUSU():String +getINV():String +getcontUSU():String +getcontINV():String +setNombre(String) +setDireccion(String) + toString():String +equals(Obj):boolean +compareTo(Obj):int +posicionOrdenada(String):int +indiceMaterial(int):int +anadirMaterial(String,String,int):boolean +quitarMaterial(int):boolean +anadirEjemplar(int):boolean +historialPrestamosInformacion(int):String +buscarInformacionCodigo(int):String +buscarInformacionTitulo(String):String +buscarInformacionAutor(String):String +buscarInformacionAnio(int):String +indiceUsuario(int):int +crearUsuario(String,String):int +quitarUsuario(int):boolean +historialPrestamosUsuario(int):String +historialPrestamosActualesUsuario(int):String +buscarUsuarioNombre(String):String +buscarUsuarioCuenta(int):String +realizarPrestamos(int, int):int +regresarPrestamos(int,int):int </pre>

## IMPLEMENTACION

### Descripción completa de la implementación

[https://drive.google.com/file/d/1\\_C5SGwxw\\_qjTiV\\_tbYihhUHpcAGpW3IR/view?usp=drive\\_link](https://drive.google.com/file/d/1_C5SGwxw_qjTiV_tbYihhUHpcAGpW3IR/view?usp=drive_link)

### GUI



### Manual Usuario

[https://drive.google.com/file/d/1DoctE0BXFINzPI-jjWLUbn8Eo9f9y\\_3ED/view?usp=sharing](https://drive.google.com/file/d/1DoctE0BXFINzPI-jjWLUbn8Eo9f9y_3ED/view?usp=sharing)

### Diseño de la Interfaz Gráfica:

La interfaz utiliza una combinación de colores personalizados para crear un diseño atractivo y coherente.

#1A415A (26, 65, 90)	#7DAB9E (125, 171, 158)	#C1E1A7 (193, 225, 167)	#0E2C40 (14, 44, 64)
-------------------------	----------------------------	----------------------------	-------------------------

La disposición de los componentes se organiza en un panel con un diseño de cuadrícula de 10 filas y 7 columnas, proporcionando una estructura ordenada y fácil de entender. Se utilizan etiquetas (JLabel) para identificar secciones y proporcionar información importante. El texto se formatea en diferentes tamaños y estilos para mejorar la legibilidad.

Los botones (JButton) se han coloreado (setBackground) para mejorar la apariencia y se han distribuido estratégicamente para representar distintas acciones. Se utilizan campos de texto (JTextField) para la entrada del usuario. Se ajusta el fondo de las áreas de texto (setBackground) para un efecto visual. La ventana tiene dimensiones específicas (1280x720), proporcionando una presentación visual específica. La fuente del texto se ajusta usando diferentes estilos para las etiquetas y áreas de texto, añadiendo variedad y destacando información crucial. Se ha añadido un borde con márgenes alrededor del panel para mejorar la presentación visual y el espaciado.

#### Funcionamiento de la Interfaz Gráfica:

Los botones representan diversas acciones, como buscar usuarios o materiales, realizar préstamos, añadir o quitar materiales o usuarios, y más.

Los campos de texto permiten al usuario ingresar información relevante, como nombres de usuario, claves, títulos de materiales, etc.

El cambio de colores en los botones proporciona feedback visual sobre las diferentes secciones o acciones disponibles.

La interfaz está dividida en secciones claramente identificadas, como la búsqueda de usuarios, búsqueda de materiales, añadir materiales o usuarios, y realizar préstamos, facilitando la navegación.

Se utilizan áreas de texto (JTextArea) para proporcionar instrucciones claras al usuario, indicando qué información se espera que ingresen.

## PRUEBAS Y RESULTADOS

```
//Valentina Pineda Barrón
//23/11/2023
//Pruebas realizadas a las clases (Biblioteca, Informacion y Usuarios)
import java.io.File;
import java.util.Scanner;

public class PruebasClases {
    public static void main(String[] args) {
        /*
         * Pruebas Biblioteca
         */
        Biblioteca biblio = new Biblioteca ("Pineda", "San Jerónimo",
"Julian");
        File archivo;
        Scanner lectura;
        String renglon, titulo, autor;
        int n, anio;
        archivo = new File("info.txt");
        boolean resultado;

        System.out.println(biblio.anadirMaterial("Abc", "Gerardo",
2023));
        System.out.println(biblio.anadirMaterial("Bca", "Gerardo",
2021));
```

```

        try {
            lectura = new Scanner (archivo);
            renglon = lectura.nextLine();
            n = Integer.parseInt(renglon);

            int i;
            i=1;

            do {
                renglon=lectura.nextLine();
                titulo=renglon;
                renglon=lectura.nextLine();
                autor=renglon;
                renglon=lectura.nextLine();
                anio=Integer.parseInt(renglon);

                resultado=biblio.anadirMaterial(titulo,autor,anio);
                i++;

            }

            while (resultado==true && i<=n);

            lectura.close();
        }

        catch (Exception error) {
            System.err.println("error");
        }

        System.out.println(biblio.buscarInformacionAnio(2023));
        System.out.println(biblio.buscarInformacionAutor("Gerardo"));
        System.out.println(biblio.buscarInformacionTitulo("Abc"));

        System.out.println(biblio.crearUsuario("Valeboo",
"valentinapineda.com"));
        System.out.println(biblio.crearUsuario("Cristina",
"crisbarron.com"));
        System.out.println(biblio.getUsuarios());

        System.out.println(biblio.buscarInformacionAnio(2021));
        System.out.println(biblio.getInventario());
        System.out.println(biblio.getContInv());

        System.out.println(biblio.realizarPrestamo(1500, 1002));
        System.out.println(biblio.historialPrestamosUsuario(1500));

        System.out.println(biblio.historialPrestamosActualesUsuario(1500));

        System.out.println(biblio.historialPrestamosInformacion(1002));
        System.out.println(biblio.anadirEjemplar(1002));

```

```

        System.out.println(biblio.realizarPrestamo(1550, 1002));
        System.out.println(biblio.regresarPrestamo(1500, 1002));
        System.out.println(biblio.realizarPrestamo(1500, 1002));
        System.out.println(biblio.historialPrestamosUsuario(1500));

        System.out.println(biblio.historialPrestamosInformacion(1002));

        Informacion u= new Informacion("Hola", "Pedro", 2022);
        Usuario val= new Usuario("paty", "paty@gmail.com");
        System.out.println(val.anadirPrestamo(1001));
        val.anadirPrestamo(1002);
        val.anadirPrestamo(1003);

        System.out.println(val.getContPrestamos());
        System.out.println(val.getPrestamosactuales()+"\n");
        System.out.println(val.regresarPrestamo(1001));
        System.out.println(val.getContPrestamos());
        System.out.println(val.getPrestamosactuales());
        System.out.println(u.getHistorialPrestamos());
        /*
        *
        *Pruebas Informacion
        */

        Informacion r= new Informacion("Hola", "Pedro", 2022);

        System.out.println(r.toString());

        /*
        *
        *Pruebas Usuario
        */

        Informacion m= new Informacion("Hola", "Pedro", 2022);
        Usuario va= new Usuario("paty", "paty@gmail.com");
        System.out.println(va.anadirPrestamo(1001));
        va.anadirPrestamo(1002);
        va.anadirPrestamo(1003);

        System.out.println(va.getContPrestamos());
        System.out.println(va.getPrestamosactuales()+"\n");
        System.out.println(va.regresarPrestamo(1001));
        System.out.println(va.getContPrestamos());
        System.out.println(va.getPrestamosactuales());
        System.out.println(m.getHistorialPrestamos());

        /*
        *
        */

    }

```

```
}
```

## Descripción de las pruebas aplicadas (casos de prueba)

### Pruebas de la clase “Biblioteca”

1. Se crea una instancia de la clase “Biblioteca” llamada *biblio* con información específica.
2. Se intenta añadir dos materiales a la biblioteca, con títulos en desorden alfabético para ver si se ordenaban alfabéticamente y se imprime el resultado.
3. Se lee información de un archivo llamado "info.txt" y se intenta añadir más materiales a la biblioteca a partir de ese archivo.
4. Se realizan búsquedas de información por año, autor y título.
5. Se crea y muestra información sobre usuarios.
6. Se realiza un préstamo, se imprime el historial de préstamos del usuario, la información de los préstamos actuales y el historial de préstamos de la información.
7. Se añade un ejemplar, se realiza otro préstamo, se devuelve un préstamo y se realiza un nuevo préstamo.
8. Finalmente, se imprimen varios resultados relacionados con los préstamos.

### Pruebas de la clase “Informacion”

1. Se crea una instancia de la clase “Informacion” llamada con información específica.
2. Se imprime la representación en cadena de la información utilizando el método `toString()`.

### Pruebas de la clase “Usuario”

1. Se crea una instancia de la clase “Usuario” llamada y una instancia de la clase “Usuario” llamada *val* con información específica.
2. Se añaden tres préstamos al usuario y se imprime el conteo de préstamos y los préstamos actuales.
3. Se realiza la devolución de uno de los préstamos y se imprime nuevamente el conteo de préstamos y los préstamos actuales.
4. Se imprime el historial de préstamos de la información asociada al usuario

## Resultados Obtenidos

```
true
true
El libro de codigo: 1001
tiene los siguientes datos
Título: "Abc"
```

Autor: Gerardo  
Año de publicación: 2023  
Número de ejemplares: 1

El libro de codigo: 1017  
tiene los siguientes datos  
Título: "Claudia Sheinbaum : presidenta"  
Autor: Cano, Arturo  
Año de publicación: 2023  
Número de ejemplares: 1

El libro de codigo: 1022  
tiene los siguientes datos  
Título: "El sexenio se me hace chiquito"  
Autor: El Fisgón  
Año de publicación: 2023  
Número de ejemplares: 1

El libro de codigo: 1025  
tiene los siguientes datos  
Título: "Que nunca se sepa : el intento de asesinato de Díaz Ordaz y la brutal respuesta del Estado mexicano"  
Autor: Cossío, José Ramón  
Año de publicación: 2023  
Número de ejemplares: 1

El libro de codigo: 1015  
tiene los siguientes datos  
Título: "Resiliencia : camino para transformar la adversidad"  
Autor: Belausteguigoitia Rius  
Año de publicación: 2023  
Número de ejemplares: 1

El libro de codigo: 1010  
tiene los siguientes datos  
Título: "Ternuritas : el linchamiento lingüístico de AMLO"  
Autor: Bak Geler, David  
Año de publicación: 2023  
Número de ejemplares: 1

El libro de codigo: 1001  
tiene los siguientes datos  
Título: "Abc"  
Autor: Gerardo  
Año de publicación: 2023  
Número de ejemplares: 1

El libro de codigo: 1002  
tiene los siguientes datos  
Título: "Bca"  
Autor: Gerardo  
Año de publicación: 2021  
Número de ejemplares: 1

El libro de codigo: 1001  
tiene los siguientes datos

```
Título: "Abc"
Autor: Gerardo
Año de publicación: 2023
Número de ejemplares: 1

1500
1550
El usuario 1500 tiene los siguientes datos
Nombre: Valeboo
Correo electronico: valentinapineda.com
El usuario 1550 tiene los siguientes datos
Nombre: Cristina
Correo electronico: crisbarron.com

El libro de codigo: 1002
tiene los siguientes datos
Título: "Bca"
Autor: Gerardo
Año de publicación: 2021
Número de ejemplares: 1

El libro de codigo: 1021
tiene los siguientes datos
Título: "C++ cómo programar : introducción al nuevo C++14 estándar"
Autor: Deitel, Paul
Año de publicación: 2021
Número de ejemplares: 1

El libro de codigo: 1008
tiene los siguientes datos
Título: "Reconciliación : tendiendo puentes"
Autor: Alemany, Jesús M
Año de publicación: 2021
Número de ejemplares: 1

El libro de codigo: 1001
tiene los siguientes datos
Título: "Abc"
Autor: Gerardo
Año de publicación: 2023
Número de ejemplares: 1

El libro de codigo: 1007
tiene los siguientes datos
Título: "Administración y control de empresas agropecuarias"
Autor: Aguirre Pérez, José Heber
Año de publicación: 2018
Número de ejemplares: 1

El libro de codigo: 1016
tiene los siguientes datos
Título: "Antología del humor : 1961-1962"
Autor: Caballero Robredo, Agustín
Año de publicación: 1962
Número de ejemplares: 1
```

```
El libro de codigo: 1011
tiene los siguientes datos
Título: "Atlas lingüístico del Estado de Puebla"
Autor: Barbosa Cano, Manlio
Año de publicación: 1980
Número de ejemplares: 1

El libro de codigo: 1002
tiene los siguientes datos
Título: "Bca"
Autor: Gerardo
Año de publicación: 2021
Número de ejemplares: 1

El libro de codigo: 1021
tiene los siguientes datos
Título: "C++ cómo programar : introducción al nuevo C++14 estándar"
Autor: Deitel, Paul
Año de publicación: 2021
Número de ejemplares: 1

El libro de codigo: 1017
tiene los siguientes datos
Título: "Claudia Sheinbaum : presidenta"
Autor: Cano, Arturo
Año de publicación: 2023
Número de ejemplares: 1

El libro de codigo: 1019
tiene los siguientes datos
Título: "Dioniso y la diosa Tierra"
Autor: Daraki, Maria
Año de publicación: 2005
Número de ejemplares: 1

El libro de codigo: 1022
tiene los siguientes datos
Título: "El sexenio se me hace chiquito"
Autor: El Fisgón
Año de publicación: 2023
Número de ejemplares: 1

El libro de codigo: 1013
tiene los siguientes datos
Título: "En el régimen de Echeverría : rebelión e independencia"
Autor: Basurto, Jorge
Año de publicación: 1989
Número de ejemplares: 1

El libro de codigo: 1024
tiene los siguientes datos
Título: "Estudios de ciencia política"
Autor: Ferrando Badía, Juan
Año de publicación: 1976
Número de ejemplares: 1
```

El libro de codigo: 1012  
tiene los siguientes datos  
Título: "Ideologías indigenistas y movimientos indios"  
Autor: Barre, Marie-Chantal  
Año de publicación: 1988  
Número de ejemplares: 1

El libro de codigo: 1023  
tiene los siguientes datos  
Título: "La diplomacia consular mexicana en tiempos de Trump"  
Autor: Fernández Aguado, Javier,  
Año de publicación: 2018  
Número de ejemplares: 1

El libro de codigo: 1004  
tiene los siguientes datos  
Título: "La guerra de Galio"  
Autor: Aguilar Camín, Héctor  
Año de publicación: 1992  
Número de ejemplares: 1

El libro de codigo: 1003  
tiene los siguientes datos  
Título: "La guerra de Galio"  
Autor: Aguilar Camín, Héctor  
Año de publicación: 2012  
Número de ejemplares: 1

El libro de codigo: 1006  
tiene los siguientes datos  
Título: "La provincia perdida"  
Autor: Aguilar Camín, Héctor  
Año de publicación: 2012  
Número de ejemplares: 1

El libro de codigo: 1014  
tiene los siguientes datos  
Título: "Orígenes de la comunidad primitiva en Patagonia"  
Autor: Bate, Luis Felipe  
Año de publicación: 1982  
Número de ejemplares: 1

El libro de codigo: 1005  
tiene los siguientes datos  
Título: "Pasado pendiente y otras historias conversadas"  
Autor: Aguilar Camín, Héctor  
Año de publicación: 2012  
Número de ejemplares: 1

El libro de codigo: 1020  
tiene los siguientes datos  
Título: "Poesías escogidas"  
Autor: Darío, Rubén  
Año de publicación: 1972  
Número de ejemplares: 1

El libro de codigo: 1025  
tiene los siguientes datos  
Título: "Que nunca se sepa : el intento de asesinato de Díaz Ordaz y la brutal respuesta del Estado mexicano"  
Autor: Cossío, José Ramón  
Año de publicación: 2023  
Número de ejemplares: 1

El libro de codigo: 1008  
tiene los siguientes datos  
Título: "Reconciliación : tendiendo puentes"  
Autor: Alemany, Jesús M  
Año de publicación: 2021  
Número de ejemplares: 1

El libro de codigo: 1015  
tiene los siguientes datos  
Título: "Resiliencia : camino para transformar la adversidad"  
Autor: Belausteguigoitia Rius  
Año de publicación: 2023  
Número de ejemplares: 1

El libro de codigo: 1009  
tiene los siguientes datos  
Título: "Roland Bacri"  
Autor: Bacri, Roland  
Año de publicación: 1976  
Número de ejemplares: 1

El libro de codigo: 1018  
tiene los siguientes datos  
Título: "Sol en los pomares : (poemas de Asturias)"  
Autor: Conde, Matías  
Año de publicación: 1948  
Número de ejemplares: 1

El libro de codigo: 1010  
tiene los siguientes datos  
Título: "Ternuritas : el linchamiento lingüístico de AMLO"  
Autor: Bak Geler, David  
Año de publicación: 2023  
Número de ejemplares: 1

25

1

Historial de prestamos del usuario 1500:  
1002 (1)

Historial de prestamos actuales del usuario 1500:  
1002

Historial de usuarios que han hecho prestamos del material 1002:  
1500 (1)

```
true
1
1
1
Historial de prestamos del usuario 1500:
1002 (2)

Historial de usuarios que han hecho prestamos del material 1002:
1500 (2)
1550 (1)

true
3
Historial de prestamos actuales del usuario 1600:
1001
1002
1003

true
2
Historial de prestamos actuales del usuario 1600:
1002
1003

El material 1026 no ha sido solicitado
El libro de codigo: 1027
tiene los siguientes datos
Título: "Hola"
Autor: Pedro
Año de publicación: 2022
Número de ejemplares: 1

true
3
Historial de prestamos actuales del usuario 1650:
1001
1002
1003

true
2
Historial de prestamos actuales del usuario 1650:
1002
1003

El material 1028 no ha sido solicitado
```

## Análisis de los resultados

Los resultados presentados en este informe reflejan el éxito de las pruebas realizadas en las clases "Biblioteca", "Informacion" y "Usuario". Aunque no todos los métodos tuvieron éxito en la primera ejecución, el logro se atribuye al proceso de prueba y error, donde se realizaron ajustes y correcciones en los métodos de las tres clases, convirtiendo así el sistema en una estructura más eficiente y a prueba de errores.

Se logró la correcta creación e impresión de instancias de las clases, así como la manipulación efectiva de la información asociada. Las pruebas relacionadas con la Biblioteca demostraron la capacidad del sistema para ordenar materiales alfabéticamente y manejar la carga de información desde un archivo externo ("info.txt").

Los resultados obtenidos en relación con la clase "Usuario" muestran un manejo adecuado de préstamos y devoluciones, con detalles sobre los ejemplares y un historial de préstamos bien estructurado. La capacidad para gestionar varios usuarios y sus interacciones con los materiales.

## CONCLUSIONES

Durante este semestre, me enfrenté al desafío de desarrollar un proyecto, y este proceso, junto con la práctica constante, me proporcionó conocimientos sólidos en programación. Esta experiencia ha sido fundamental para comprender la importancia de mantener una estructura de código organizada, así como la capacidad de modelar objetos de manera efectiva y asignar responsabilidades a cada clase.

La implementación de la interfaz gráfica ha representado un avance significativo, permitiéndome proyectar el código de una manera más visual y estéticamente diseñada. La disposición lógica de elementos y la elección de colores funcionales han agregado una dimensión adicional a la presentación del proyecto, alejándolo de la simple ejecución en la consola.

Asimismo, he experimentado los beneficios de reducir tareas manuales propensas a errores, gracias a la automatización y optimización de procesos. Esta capacidad ha mejorado la eficiencia del proyecto y ha contribuido a la calidad del código desarrollado.

La resolución de problemas en tiempo real, especialmente la corrección de errores durante las pruebas ha sido una lección práctica fundamental. Aprender a identificar y abordar problemas de manera eficiente ha fortalecido mi habilidad para enfrentar desafíos en el desarrollo de software.

En resumen, este semestre no solo ha sido académico, sino también altamente práctico. Las lecciones aprendidas no se limitan solo a la teoría, sino que se han aplicado directamente en la realización de un proyecto real.

## REFERENCIAS

González Fernández-Villavicencio, N. (2009). *La biblioteca expandida en código abierto*. Boletín de la Asociación Andaluza de Bibliotecarios, 24 (96-97), 11-35.

<https://dialnet.unirioja.es/descarga/articulo/3347383.pdf>

ChatGPT. (2023). Chat con ChatGPT. <https://chat.openai.com/>

## APÉNDICE (código)

```
//Valentina Pineda Barrón
//21/11/2023
// Clase que hace función de recopilar los datos de los diferentes
materiales en la biblioteca

import java.util.ArrayList;

public class Informacion{
    private String titulo;
    private String autor;
    private int anio;
    private int ejemplares;
    private int ejemplaresDisponibles;
    private ArrayList<Integer> historialPrestamos;
    private static int consecutivo = 1001;
    private int codigo;

    public Informacion(String titulo, String autor, int anio) {
        this.titulo=titulo;
        this.autor=autor;
        this.anio=anio;
        ejemplares=1;
        historialPrestamos = new ArrayList<Integer>();
        ejemplaresDisponibles=ejemplares;
        codigo=consecutivo;
        consecutivo++;
    }

    public String getTitulo() {
        return titulo;
    }

    public String getAutor() {
        return autor;
    }

    public int getAnio() {
        return anio;
    }

    public int getEjemplares() {
        return ejemplares;
    }
}
```

```

    public int getEjemplaresDisponibles() {
        return ejemplaresDisponibles;
    }

    public boolean getDisponibilidad() {
        if (ejemplaresDisponibles>0)
            return true;
        else
            return false;
    }

    public String getHistorialPrestamos() {
        StringBuilder texto= new StringBuilder();
        StringBuilder textoalt = new StringBuilder();
        String textoini;

        textoini="Historial de usuarios que han hecho prestamos del
material "+codigo+":\n";
        texto.append(textoini);

        /*El for imprime la cuenta de los usuarios
        * posteriormente entre parentesis la cantidad de veces
prestadas
        * hasta que la i es menor al tamaño del arreglo historial de
prestamos
        */
        for(int i=0;i<historialPrestamos.size();i+=2) {
            texto.append(historialPrestamos.get(i)+"
("+historialPrestamos.get(i+1)+")\n");
        }

        if(texto.toString().equals(textoini)) {
            textoalt.append("El material "+codigo+" no ha sido
solicitado");
            texto = textoalt;
        }

        return texto.toString();
    }

    public int getConsecutivo() {
        return consecutivo;
    }

    public int getCodigo() {
        return codigo;
    }

    public String toString() {
        StringBuilder texto= new StringBuilder ();

        texto.append("El libro de codigo: " +codigo);
        texto.append("\ntiene los siguientes datos\n");
        /* En estos ejemplos, la barra invertida (\)
        * se utiliza como carácter de escape antes de la comilla
doble ("")

```

```

        * para indicar que se trata de un carácter literal
        * y no como el final del string.
        */
        texto.append("Título: "+"\""+titulo+"\""+"\n");
        texto.append("Autor: "+autor+"\n");
        texto.append("Año de publicación: "+anio+"\n");
        texto.append("Número de ejemplares: "+ejemplares+"\n");

        return texto.toString();
    }

    public boolean equals(Informacion otra) {
        if(this.codigo==otra.codigo)
            return true;
        else
            return false;
    }

    public int compareTo(Informacion otra) {
        if(equals(otra)==true)
            return 0;
        else
            if(this.codigo<otra.codigo)
                return -1;
            else
                return 1;
    }

    public void anadirEjemplar() {
        ejemplares++;
        ejemplaresDisponibles++;
    }

    public void anadirEjemplarDisponibile() {
        ejemplaresDisponibles++;
    }
    public void retirarEjemplarDisponibile() {
        ejemplaresDisponibles--;
    }

    public void anadirHistorialInfo(int cuenta) {
        boolean cuentaEncontrada;
        cuentaEncontrada=false;
        for(int i=0; i<historialPrestamos.size();i+=2) {
            /*Si la cuenta ingresada ya existe en el historial
            * se le añade una unidad a la casilla de a lado
            * que indica cuantas veces se ha pedido prestado
            */
            if(historialPrestamos.get(i)==cuenta) {
                cuentaEncontrada=true;
                historialPrestamos.set(i+1,
historialPrestamos.get(i+1)+1);
                break;
                //break indica que si se cumple el if se termina
el for
            }
        }
    }

```

```
    }  
    if(!cuentaEncontrada) {  
        historialPrestamos.add(cuenta);  
        historialPrestamos.add(1);  
    }  
}  
  
}
```

```
//Valentina Pineda Barrón  
//21/11/2023  
// Clase que hace función de recopilar los datos de los diferentes  
// usuarios que hacen prestamos en la biblioteca  
  
import java.util.ArrayList;  
  
public class Usuario {  
  
    // Atributos  
  
    private String nombre;  
    private String correo;  
    private ArrayList<Integer> historialPrestamos;  
    private int[] prestamosActuales;  
    private final int MAX=10; //Máximo de prestamos actuales por  
persona  
    private static int consecutivo = 1450;  
    private int contPrestamos;  
    private int cuenta;  
  
    // Constructores  
  
    public Usuario(String nombre, String correo) {  
        this.nombre = nombre;  
        this.correo=correo;  
        historialPrestamos=new ArrayList<Integer>();  
        prestamosActuales = new int[MAX];  
        consecutivo +=50;  
        cuenta=consecutivo;  
    }  
  
    // Funcionalidad Mínima  
  
    public String getNombre() {  
        return nombre;  
    }  
    public String getCorreo () {  
        return correo;  
    }  
    public int getCuenta() {  
        return cuenta;  
    }  
    public String getHistorialPrestamos() {
```

```

        StringBuilder texto= new StringBuilder();
        StringBuilder textoalt=new StringBuilder ();
        String textoini;

        textoini="Historial de prestamos del usuario "+cuenta+":\n";
        texto.append(textoini);

        /*El for imprime el codigo de la información
        * posteriormente entre parentesis la cantidad de veces
prestadas
        * hasta que la i es menor al tamaño del arreglo historial de
prestamos
        */
        for(int i=0;i<historialPrestamos.size();i+=2) {
            texto.append(historialPrestamos.get(i)+"
("+historialPrestamos.get(i+1)+")\n");
        }
        // Esta modificacion lo añadi despues de probarlo en la
interfaz gráfica
        if(texto.toString().equals(textoini)) {
            textoalt.append("El usuario "+cuenta+" no ha realizado
prestamos");
            texto = textoalt;
        }
        return texto.toString();
    }

    public int getConsecutivo() {
        return consecutivo;
    }

    /* Prestamos Actuales es pensando que
    * no se va a prestar la misma información más de una vez al mismo
tiempo
    */
    public String getPrestamosactuales() {
        StringBuilder texto=new StringBuilder ();
        StringBuilder textoalt=new StringBuilder ();
        String textoini;

        textoini="Historial de prestamos actuales del usuario
"+cuenta+":\n";
        texto.append(textoini);

        for(int i=0;i<contPrestamos;i++) {
            texto.append(prestamosActuales[i)+"\n");
        }

        /*
        * La variable textoini y textoalt los añadi para incorporar
        * un mejor mensaje en la interfaz gráfica
        */
        if(texto.toString().equals(textoini)) {
            textoalt.append("El usuario "+cuenta+" no cuenta con
prestamos actuales");

```

```

        texto = textoalt;
    }
    return texto.toString();
}
public int getContPrestamos() {
    return contPrestamos;
}

public String toString() {
    StringBuilder texto = new StringBuilder();
    texto.append("El usuario "+cuenta+" tiene los siguientes
datos \n");
    texto.append("Nombre: "+nombre+"\n");
    texto.append("Correo electronico: "+correo);
    return texto.toString();
}

public boolean equals(Usuario otro) {
    if(this.cuenta==otro.cuenta)
        return true;
    else
        return false;
}

public int compareTo(Usuario otro) {
    if(equals(otro)==true)
        return 0;
    else
        if(this.cuenta<otro.cuenta)
            return -1;
        else
            return 1;
}

public void anadirHistorialUsuario(int codigo) {
    boolean codigoEncontrado;
    codigoEncontrado=false;

    for(int i=0; i<historialPrestamos.size();i+=2) {
        /*Si el codigo ingresado ya existe en el historial
        * se le añade una unidad a la casilla de a lado
        * que indica cuantas veces se ha pedido prestado
        */
        if(historialPrestamos.get(i)==codigo) {
            codigoEncontrado=true;
            historialPrestamos.set(i+1,
historialPrestamos.get(i+1)+1);
            break;
            //break indica que si se cumple el if se termina
el for
        }

    }

    if(!codigoEncontrado) {
        historialPrestamos.add(codigo);
        historialPrestamos.add(1);
    }
}

```

```

    }

}

public boolean anadirPrestamo(int n) {
    boolean resultado;
    resultado=false;
    /*Inicialmente el resultado es falso
    * Por si no cumple con el if
    * ya que el máximo de prestamos
    * que puedes al mismo tiempo es 10
    */

    if (contPrestamos<MAX) {
        resultado=true;
        for(int i=0; i<contPrestamos;i++) {
            if(prestamosActuales[i]==n) {
                resultado = false;
                break;
            }
            /*
            * Si el codigo del prestamo se repite,
            */
        }
        if(resultado) {
            prestamosActuales[contPrestamos]=n;
            anadirHistorialUsuario(n);
            contPrestamos++;
        }
    }

    return resultado;
}

public boolean regresarPrestamo(int clave) {
    int indiceElemento;
    indiceElemento=-1;

    for(int i=0;i<contPrestamos;i++) {
        if(prestamosActuales[i] == clave) {
            indiceElemento = i;
            break;
        }
    }

    if (indiceElemento!=-1) {
        contPrestamos--;
        while(indiceElemento<contPrestamos) {
            prestamosActuales[indiceElemento]=prestamosActuales[indiceElemento+
1];
            indiceElemento++;
        }
        return true;
    }
}

```

```
    }  
  
    else  
        return false;  
  
    }  
  
}
```

```
//Valentina Pineda Barrón  
//21/11/2023  
// Clase que a través de las clases Informacion y Usuarios, establece el  
sistema operativo de una biblioteca  
  
public class Biblioteca {  
  
    private String nombre;  
    private String direccion;  
    private String bibliotecario;  
    private Usuario[] usuarios;  
    private Informacion [] inventario;  
    private final int USU=100;  
    private final int INV=500;  
    private int contUsu;  
    private int contInv;  
  
    //Constructor  
    public Biblioteca (String nombre, String direccion,String  
bibliotecario) {  
        this.nombre=nombre;  
        this.direccion=direccion;  
        this.bibliotecario=bibliotecario;  
        usuarios = new Usuario[USU];  
        inventario = new Informacion[INV];  
    }  
  
    //Get's  
    public String getNombre() {  
        return nombre;  
    }  
  
    public String getDireccion() {  
        return direccion;  
    }  
  
    public String getBibliotecario() {  
        return bibliotecario;  
    }  
  
    public String getUsuarios() {  
        StringBuilder texto= new StringBuilder();  
  
        for(int i=0;i<contUsu;i++) {  
            texto.append(usuarios[i].toString()+"\n");  
        }  
        return texto.toString();  
    }  
}
```

```

    }

    public String getInventario() {

        StringBuilder texto = new StringBuilder();

        for(int i=0;i<contInv;i++) {
            texto.append(inventario[i].toString()+"\n");
        }
        return texto.toString();
    }
    public String getInformacionParticular(int i) {
        return (inventario[i].toString()+"\n");
    }

    public int getUSU() {
        return USU;
    }

    public int getINV() {
        return INV;
    }

    public int getContUsu() {
        return contUsu;
    }

    public int getContInv() {
        return contInv;
    }

    //Set's

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public void setDireccion(String direccion) {
        this.direccion = direccion;
    }

    //Funcionalidad Minima

    public boolean equals(Biblioteca otra) {
        if (nombre.equalsIgnoreCase(otra.nombre))
            return true;
        else
            return false;
    }

    public int compareTo(Biblioteca otra) {
        if (nombre.equalsIgnoreCase(otra.nombre))
            return 0;
        else
            if (nombre.compareToIgnoreCase(otra.nombre) > 0)
                return 1;
    }

```

```

        else
            return -1;
    }

    public String toString() {
        StringBuilder texto;

        texto = new StringBuilder();
        texto.append("El nombre de la biblioteca es: " +
nombre+"\n");
        texto.append("La direccion de la biblioteca es: " +
direccion+"\n");
        texto.append("El/La bibliotecari@ es: " +
bibliotecario+"\n");
        texto.append("Número de usuarios registrados: " +
contUsu+"\n");
        texto.append("Número de material actual: " + contInv);

        return texto.toString();
    }

    // FUNCIONALIDAD
    // Funcionalidad de Informacion

    // Funcionalidad de apoyo

    // Este método me ayuda a saber en que posición van los materiales
    // que se van incluyendo en el inventario
    // ya que el inventario esta ordenado en orden alfabetico
    // y su parametro es el título
    public int posicionOrdenada(String x) {
        int i = 0;
        // se realiza la b↯squeda mientras todav↯a no se termine el
arreglo y no se
        // encuentre todav↯a un elemento mayor que el que se busca
        while (i < contInv &&
inventario[i].getTitulo().compareTo(x)<0)
            i++;
        return i;
    }

    // Este método obtiene el índice de la casilla donde se encuentra el
material con la clave
    public int indiceMaterial(int n) {
        int i;
        for (i=0; i<contInv;i++) {
            if(n==inventario[i].getCodigo())
                break;
        }
        return i;
    }

    // Este método añade materiales al inventario de forma alfabética,
    // tomando como criterio el título del material

```

```

    public boolean anadirMaterial(String titulo, String autor, int
anio) {
        int pos;
        boolean resultado;
        //el boolean es falso en caso de que el inventario este lleno
        resultado=false;
        if (contInv < INV-1) {
            resultado=true;
            pos = posicionOrdenada(titulo);
            //si la posicion el el arreglo es la del contador
            //se añade en esa posicion porque , en teoría, es la
            //del arreglo y no tendría que recorrer a la derecha
            if(pos==contInv) {
                inventario[contInv]= new Informacion
                (titulo,autor, anio);
                contInv++;
            }
            else {
                //este metodo recorre a la derecha desde la
                //donde se añade el material
                //y despues añade el material
                for(int i = INV - 1; i>pos; i--) {
                    inventario[i]=inventario[i-1];
                }
                inventario[pos]= new Informacion (titulo,autor,
                anio);
                contInv++;
            }
        }
        return resultado;
    }

    //Método que elimina un material del inventario
    // n es el código del material
    public boolean quitarMaterial(int n) {
        boolean resultado;
        //el resultado inicial es false en caso
        // de que el código ingresado no este en el inventario
        resultado=false;

        // si el indice es igual a contInv o mayor
        //significa que no se encuentra dentro del arreglo
        if(indiceMaterial(n)<contInv) {
            resultado=true;
            //disminuyo contador inventario
            contInv--;
            //recorre a la izquierda el inventario
            // de esta forma elimina el material con el indice
            for(int i=indiceMaterial(n);i<contInv;i++) {
                inventario[i]=inventario[i+1];
            }
            // el único error de este método es que

```

```

        // el último material ingresado es doble
        // en las últimas dos casillas
        // pero solo imprimira hasta el nuevo contador
inventario
        // y el último repetido se sustituirá cuando se
encuentre un título
        // mayor a este
    }

    return resultado;
}

/*
 * Si no hay un material con el código indicado
 * regresa false
 * Al contrario, si encuentra el código
 * añade un ejemplar al libro y regresa true
 *
 * Esto sirve para que más usuarios puedan hacer
 * un préstamo del mismo libro
 */
public boolean anadirEjemplar( int codigo) {
    if(indiceMaterial(codigo)<contInv) {
        inventario[indiceMaterial(codigo)].anadirEjemplar();
        return true;
    }
    return false;
}

/*
 * Recupera el historial préstamos del material
 * Usa de parametr el código y a través
 * de indiceMaterial, busca el índice e imprime el historial
 * de esa casilla
 */
public String historialPrestamosInformacion(int codigo) {
    int indice;
    indice=indiceMaterial(codigo);

    // si el índice es igual a contInv
    //significa que no se encuentra dentro del arreglo
    if(indice!=contInv)
        return inventario[indice].getHistorialPrestamos();
    else
        return "";
}

/*
 * Recupera toda la información de un elemento
 * Su parametro es el código
 * Se apoya del método indice material
 */
public String buscarInformacionCodigo(int codigo) {
    int indice;
    indice=indiceMaterial(codigo);

    // si el índice es igual a contInv

```

```

        //significa que no se encuentra dentro del arreglo
        if(indice!=contInv)
            return inventario[indice].toString();
        else
            return "";
    }
    /*
    * Busca materiales con el mismo título
    * El problema con este método es que el título
    * debe estar igual
    *
    * Este método trate de hacerlo con un while, ya que
    * el arreglo esta ordenado de forma alfabética. Uno de los
    criterios
    * era que el título ingresado fuera mayor
    * o igual a los títulos evaluados. Pero no funciona, así que use
    un for
    */
    public String buscarInformacionTitulo(String titulo) {

        StringBuilder texto = new StringBuilder();

        for(int i=0;i<contInv;i++) {
            //El lower case es para que se imprima incluso si se
            escribio mal
            //en terminos de mayusculas o minúsculas

            if(inventario[i].getTitulo().toLowerCase().equals(titulo.toLowerCase()))
                texto.append(inventario[i].toString());
        }
        return texto.toString();
    }
    /*
    * Busca materiales con el mismo autor
    * El problema con este método es que el autor
    * debe estar igual
    */
    public String buscarInformacionAutor(String autor) {

        StringBuilder texto = new StringBuilder();

        for(int i=0;i<contInv;i++) {
            //El lower case es para que se imprima incluso si se
            escribio mal
            //en terminos de mayusculas o minúsculas

            if(inventario[i].getAutor().toLowerCase().equals(autor.toLowerCase()))
                texto.append(inventario[i].toString());
        }
        return texto.toString();
    }
    /*

```

```

    * Busca materiales con el mismo año
    */
    public String buscarInformacionAnio(int anio) {
        StringBuilder texto = new StringBuilder();

        for(int i=0;i<contInv;i++) {
            if(inventario[i].getAnio()==(anio))
                texto.append(inventario[i].toString()+"\n");
        }
        return texto.toString();
    }
    /*
    * Los últimos métodos no tienen mensaje, ya que
    * el mensaje correspondiente se evalúa en la interfaz, si
    * el StringBuilder está vacío
    */

    //Funcionalidad Usuario

    //Funcionalidad de apoyo

    //Busca el índice usuario, tiene de parametro su clave/cuenta
    public int indiceUsuario(int cuenta) {
        int i;
        for (i=0; i<contUsu;i++) {
            if(cuenta==usuarios[i].getCuenta())
                break;
        }
        return i;
    }

    /*
    * Crea un usuario
    * El arreglo está desordenado
    */

    public int crearUsuario(String nombre, String correo) {
        int n;
        //n=-1 si el arreglo usuarios está lleno
        n=-1;
        if(contUsu<USU) {
            usuarios[contUsu]= new Usuario (nombre, correo);
            n=usuarios[contUsu].getCuenta();
            contUsu++;
        }
        return n;
    }
    /*
    * Elimina un usuario
    * n es la clave o cuenta del usuario
    */

    public boolean quitarUsuario(int n) {
        boolean resultado;
        //resultado es false inicialmente
        // por si el usuario no está en el arreglo
    }

```

```

        resultado=false;
        //si el indice es mayor o igual a contador usuarios
        //la cuenta/clave del usuario no esta en el arreglo
        if(indiceUsuario(n)<contUsu) {
            resultado=true;
            contUsu--;
            for(int i=indiceUsuario(n);i<contUsu;i++) {
                usuarios[i]=usuarios[i+1];
            }
        }

        return resultado;
    }

    //Imprime el historial de prestamos de un usuario
    public String historialPrestamosUsuario(int cuenta) {
        int indice;
        String texto;
        texto="";
        indice=indiceUsuario(cuenta);
        // si el indice es igual a contUsu
        // el usuario no esta en el arreglo
        if(indice!=contUsu)
            texto = usuarios[indice].getHistorialPrestamos();

        return texto;
    }

    //Imprime el historial de prestamos actuales del usuario
    public String historialPrestamosActualesUsuario(int cuenta) {
        int indice;
        String texto;
        texto="";
        indice=indiceUsuario(cuenta);
        // si el indice es igual a contUsu
        // el usuario no esta en el arreglo
        if(indice!=contUsu)
            texto = usuarios[indice].getPrestamosactuales();

        return texto;
    }

    //Busca usuarios con ese nombre en el arreglo
    public String buscarUsuarioNombre(String nombre) {

        StringBuilder texto = new StringBuilder();

        for(int i=0;i<contUsu;i++) {

            if(usuarios[i].getNombre().toLowerCase().equals(nombre.toLowerCase(
)))
                texto.append(usuarios[i].toString());
            }
        return texto.toString();
    }

    //Busca usuarios con esa clave en el arreglo

```

```

public String buscarUsuarioCuenta(int clave) {
    String texto;
    texto="";
    int indice;
    indice=indiceUsuario(clave);
    if(indice!=contUsu)
        texto=usuarios[indice].toString();
    return texto;
}
/*
 * Los últimos métodos no tienen mensaje, ya que
 * el mensaje correspondiente se evalúa en la interfaz, si
 * el String está vacío
 */

//Funcionalidad de Informacion y Usuarios

/*originalmente este era un boolean que regresaba true solo si
 * cumplía todos los if
 * pero en el main decidí cambiarlo a un int indicador para
 * enviar un mensaje apropiado dependiendo del problema
 */

//Realizar Prestamo
//los parametros son clave del usuario
//y código del material
public int realizarPrestamo(int cuenta, int codigo){
    int indicador;
    int indiceCuenta;
    int indiceCodigo;
    indiceCuenta=indiceUsuario(cuenta);
    indiceCodigo=indiceMaterial(codigo);
    indicador=-1; //indicador=-1, si el usuario o material no
    estan en los arreglos

    if(indiceCuenta!=contUsu && indiceCodigo!=contInv) {
        Usuario usuario;
        Informacion info;
        usuario=usuarios[indiceCuenta];
        info=inventario[indiceCodigo];
        indicador=0;
        /*
         * indicador=0, si el material no está disponible
         * si el usuario ya tiene más de 10 préstamos actuales
         * o el usuario ya tiene posesión de un ejemplar de
         este libro
         */

        if(info.getDisponibilidad() &&
        usuario.anadirPrestamo(codigo)) {
            info.anadirHistorialInfo(cuenta);
            info.retirarEjemplarDisponible();
            indicador=1; //indicador=1, si se realizó el
            prestamo
        }
    }
}

```

```

        return indicador;
    }

    //Regresar prestamo
    //los parametros son clave del usuario
    //y codigo del material
    public int regresarPrestamo(int cuenta, int codigo) {
        int indicador;
        int indiceCuenta;
        int indiceCodigo;
        indiceCuenta=indiceUsuario(cuenta);
        indiceCodigo=indiceMaterial(codigo);
        indicador=-1; //indicador=-1, si el usuario o material no
        estan en los arreglos

        if(indiceCuenta!=contUsu && indiceCodigo!=contInv) {
            indicador=0;
            /*
             * indicador=0, si el material cuenta con todos sus
            ejemplares en el inventario
             * si el usuario no realizo un prestamo del libro
             */
            Usuario usuario;
            Informacion info;
            usuario=usuarios[indiceCuenta];
            info=inventario[indiceCodigo];

            if(info.getEjemplares()>info.getEjemplaresDisponibles() && usuario.re
            gresarPrestamo(codigo)) {
                info.anadirEjemplarDisponible();
                indicador=1; //indicador=1, si se regreso el
            prestamo
            }
        }
        return indicador;
    }
}

```

```

//Valentina Pineda Barrón
//23/11/2023
//Vista del menu de la biblioteca
import java.awt.Color;
import java.awt.Font;
import java.awt.GridLayout;
import javax.swing.*;
import javax.swing.border.Border;

public class VistaMenu extends JFrame{
    /*
     * Un amigo de otra clase me dijo que añadiera esto
     * pero realmente no se que es
     */
    private static final long serialVersionUID = 1L;

```

```

protected JPanel panel;
protected Border borde;
protected JLabel lblTitulo, lblAnadir, lblVacia, lblRespuesta;
//Etiquetas del panel
protected JTextField txtBuscar, txtInfo1, txtInfo2,
txtInfo3,txtInfo4,txtInfo5, txtInfo6, txtInfo7,
txtInfo8,txtInfo9,txtInfo10;
protected JTextArea txtVacia, txtIndicacion;
protected JButton btNombre, btClave, btCodigo, btUsuarios,
btInventario, btInformacion, btAceptar1, btAceptar2, btAutor, btTitulo,
btAnio, btMaterial, btEjemplar, btQuitar, btQuitar1, btHistorialM,
btHistorialUA, btHistorialU, btRealizar, btRegresar;

public VistaMenu() {
    super("Menu Biblioteca");
    //titulo
    panel = new JPanel(new GridLayout(10,7));
    //Divide el panel en 10 renglones y 7 columnas
    borde = BorderFactory.createEmptyBorder(10,10,10,10); //
    Distancia de los bordes de la pantalla (mueve a la derecha, mueve hacia
    bajo, instancia la base, instancia la altura)
    panel.setBorder(borde);
    // Convertir el código hexadecimal a valores RGB y cree un
    objeto Color con los valores RGB
    Color color1 = new
    Color(Integer.parseInt("1A",16),Integer.parseInt("4A",16),Integer.parseIn
    t("5A",16));
    Color color2 = new
    Color(Integer.parseInt("7d",16),Integer.parseInt("ab",16),Integer.parseIn
    t("9e",16));
    Color color3 = new
    Color(Integer.parseInt("C1",16),Integer.parseInt("E1",16),Integer.parseIn
    t("A7",16));
    Color color4 = new
    Color(Integer.parseInt("0e",16),Integer.parseInt("2c",16),Integer.parseIn
    t("40",16));

    panel.setBackground(color1);

    btUsuarios = new JButton ("Usuarios");
    // puse el color2 de fondo en los botones
    btUsuarios.setBackground(color2);
    panel.add(btUsuarios);
    btInventario = new JButton ("Inventario");
    // puse el color2 de fondo en los botones
    btInventario.setBackground(color2);
    panel.add(btInventario);
    btInformacion = new JButton ("Informacion");
    // puse el color2 de fondo en los botones
    btInformacion.setBackground(color2);
    panel.add(btInformacion);
    // SwingConstants.CENTER es para poner el texto en medio
    lblTitulo = new JLabel("Menú", SwingConstants.CENTER);
    // Me permite cambiar la tipografía del letrero
    lblTitulo.setFont(new Font("Mongolian Baiti", Font.BOLD, 32));

```

```

//Cambie el color de la letra
lblTitulo.setForeground(Color.WHITE);
panel.add(lblTitulo);
lblVacia = new JLabel(" ");
panel.add(lblVacia);
lblVacia = new JLabel(" ");
panel.add(lblVacia);
lblVacia = new JLabel(" ");
panel.add(lblVacia);

txtBuscar = new JTextField(); //Número
de Carácteres en el campo de texto
panel.add(txtBuscar);
btNombre = new JButton ("Buscar Nombre Usuario");
// puse el color2 de fondo en los botonos
btNombre.setBackground(color3);
panel.add(btNombre);
btClave = new JButton ("Buscar Clave Usuario");
// puse el color2 de fondo en los botonos
btClave.setBackground(color3);
panel.add(btClave);
btTitulo = new JButton ("Buscar Título Material");
// puse el color2 de fondo en los botonos
btTitulo.setBackground(color3);
panel.add(btTitulo);
btAutor = new JButton ("Buscar Autor Material");
// puse el color2 de fondo en los botonos
btAutor.setBackground(color3);
panel.add(btAutor);
btAnio = new JButton ("Buscar Año Material");
// puse el color2 de fondo en los botonos
btAnio.setBackground(color3);
panel.add(btAnio);
btCodigo = new JButton ("Buscar Código Material");
// puse el color2 de fondo en los botonos
btCodigo.setBackground(color3);
panel.add(btCodigo);

txtVacia = new JTextArea("");
// puse el color de fondo en los JTextArea
txtVacia.setBackground(color4);
panel.add(txtVacia);
txtVacia = new JTextArea("");
txtVacia.setBackground(color4);
panel.add(txtVacia);
txtVacia = new JTextArea("");
txtVacia.setBackground(color4);
panel.add(txtVacia);
txtVacia = new JTextArea("");
txtVacia.setBackground(color4);
panel.add(txtVacia);
txtVacia = new JTextArea("");
txtVacia.setBackground(color4);
panel.add(txtVacia);

```

```

        txtVacia = new JTextArea("");
        txtVacia.setBackground(color4);
        panel.add(txtVacia);
        txtVacia = new JTextArea("");
        txtVacia.setBackground(color4);
        panel.add(txtVacia);

        lblAnadir = new JLabel("Añadir Material");
        //Cambie el color de la letra
        lblAnadir.setForeground(Color.WHITE);
        lblAnadir.setFont(new Font("Corbel", Font.BOLD, 20));
        panel.add(lblAnadir);
        txtInfo1 = new JTextField();
        panel.add(txtInfo1);
        txtInfo2 = new JTextField();
        panel.add(txtInfo2);
        txtInfo3 = new JTextField();
        panel.add(txtInfo3);
        btAceptar1 = new JButton("Aceptar");
        // puse el color2 de fondo en los botones
        btAceptar1.setBackground(color2);
        panel.add(btAceptar1);
        lblVacia = new JLabel(" ");
        panel.add(lblVacia);
        lblVacia = new JLabel(" ");
        panel.add(lblVacia);

        txtIndicacion = new JTextArea("Ingresa el código del
material: ");
        // font
        txtIndicacion.setFont(new Font("Corbel", Font.BOLD, 16));
        // puse el color de fondo en los JTextArea
        txtIndicacion.setBackground(color1);
        //Cambie el color de la letra
        txtIndicacion.setForeground(Color.WHITE);
        txtIndicacion.setEditable(false);
        txtIndicacion.setWrapStyleWord(true);
        txtIndicacion.setLineWrap(true);
        panel.add(txtIndicacion);
        txtInfo4 = new JTextField();
        panel.add(txtInfo4);
        btEjemplar = new JButton("Añadir Ejemplar");
        // puse el color2 de fondo en los botones
        btEjemplar.setBackground(color2);
        panel.add(btEjemplar);
        btQuitar = new JButton("Quitar Material");
        // puse el color2 de fondo en los botones
        btQuitar.setBackground(color2);
        panel.add(btQuitar);
        lblVacia = new JLabel(" ");
        panel.add(lblVacia);
        lblVacia = new JLabel(" ");
        panel.add(lblVacia);
        lblVacia = new JLabel(" ");

```

```

panel.add(lblVacía);

lblAnadir = new JLabel("Añadir Usuario");
lblAnadir.setFont(new Font("Corbel", Font.BOLD, 20));
panel.add(lblAnadir);
//Cambie el color de la letra
lblAnadir.setForeground(Color.WHITE);
txtInfo5 = new JTextField();
panel.add(txtInfo5);
txtInfo6 = new JTextField();
panel.add(txtInfo6);
btAceptar2 = new JButton("Aceptar");
// puse el color2 de fondo en los botones
btAceptar2.setBackground(color2);
panel.add(btAceptar2);
lblVacía = new JLabel(" ");
panel.add(lblVacía);
lblVacía = new JLabel(" ");
panel.add(lblVacía);
lblVacía = new JLabel(" ");
panel.add(lblVacía);

txtIndicacion = new JTextArea("Ingresa la clave del usuario:");

//cambie la tipografía y el tamaño
txtIndicacion.setFont(new Font("Corbel", Font.BOLD, 16));
txtIndicacion.setBackground(color1);
//Cambie el color de la letra
txtIndicacion.setForeground(Color.WHITE);
txtIndicacion.setEditable(false);
txtIndicacion.setWrapStyleWord(true);
txtIndicacion.setLineWrap(true);
panel.add(txtIndicacion);
txtInfo7 = new JTextField();
panel.add(txtInfo7);
btQuitar1 = new JButton("Quitar Usuarios");
// puse el color2 de fondo en los botones
btQuitar1.setBackground(color2);
panel.add(btQuitar1);
lblVacía = new JLabel(" ");
panel.add(lblVacía);
lblVacía = new JLabel(" ");
panel.add(lblVacía);
lblVacía = new JLabel(" ");
panel.add(lblVacía);
lblVacía = new JLabel(" ");
panel.add(lblVacía);

lblVacía = new JLabel(" ");
panel.add(lblVacía);
lblVacía = new JLabel(" ");
panel.add(lblVacía);
lblVacía = new JLabel(" ");
panel.add(lblVacía);
lblVacía = new JLabel(" ");
panel.add(lblVacía);

```

```

        lblVacía = new JLabel(" ");
        panel.add(lblVacía);
        txtIndicacion = new JTextArea("Ingresa el código del
material: ");
        txtIndicacion.setFont(new Font("Corbel", Font.BOLD, 16));
        // puse el color de fondo en los JTextArea
        txtIndicacion.setBackground(color1);
        //Cambie el color de la letra
        txtIndicacion.setForeground(Color.WHITE);
        txtIndicacion.setEditable(false);
        txtIndicacion.setWrapStyleWord(true);
        txtIndicacion.setLineWrap(true);
        panel.add(txtIndicacion);
        txtIndicacion = new JTextArea("Ingresa la clave del usuario:
");

        txtIndicacion.setFont(new Font("Corbel", Font.BOLD, 16));
        // puse el color de fondo en los JTextArea
        txtIndicacion.setBackground(color1);
        //Cambie el color de la letra
        txtIndicacion.setForeground(Color.WHITE);
        txtIndicacion.setEditable(false);
        txtIndicacion.setWrapStyleWord(true);
        txtIndicacion.setLineWrap(true);
        panel.add(txtIndicacion);

        txtIndicacion = new JTextArea("    Historial Préstamos
Ingresa código o clave:");
        txtIndicacion.setFont(new Font("Corbel", Font.BOLD, 16));
        // puse el color de fondo en los JTextArea
        txtIndicacion.setBackground(color1);
        //Cambie el color de la letra
        txtIndicacion.setForeground(Color.WHITE);
        txtIndicacion.setEditable(false);
        txtIndicacion.setWrapStyleWord(true);
        txtIndicacion.setLineWrap(true);
        panel.add(txtIndicacion);
        txtInfo8 = new JTextField(); //Número
de Carácteres en el campo de texto
        panel.add(txtInfo8);
        lblVacía = new JLabel(" ");
        panel.add(lblVacía);
        lblVacía = new JLabel(" ");
        panel.add(lblVacía);
        lblVacía = new JLabel(" ");
        panel.add(lblVacía);
        txtInfo9 = new JTextField(); //Número
de Carácteres en el campo de texto
        panel.add(txtInfo9);
        txtInfo10 = new JTextField(); //Número
de Carácteres en el campo de texto
        panel.add(txtInfo10);

        btHistorialM = new JButton("Material");
        // puse el color2 de fondo en los botones
        btHistorialM.setBackground(color2);
        panel.add(btHistorialM);

```

```

        btHistorialU = new JButton("Usuario");
        // puse el color2 de fondo en los botones
        btHistorialU.setBackground(color2);
        panel.add(btHistorialU);
        btHistorialUA = new JButton("Actuales Usuario");
        // puse el color2 de fondo en los botones
        btHistorialUA.setBackground(color2);
        panel.add(btHistorialUA);
        lblVacía = new JLabel(" ");
        panel.add(lblVacía);
        lblVacía = new JLabel(" ");
        panel.add(lblVacía);
        btRealizar = new JButton("Realizar Préstamo");
        // puse el color2 de fondo en los botones
        btRealizar.setBackground(color2);
        panel.add(btRealizar);
        btRegresar = new JButton("Regresar Préstamo");
        // puse el color2 de fondo en los botones
        btRegresar.setBackground(color2);
        panel.add(btRegresar);

        super.add(panel);
        super.setBounds(0, 0, 1280, 720); //Moves and resizes this
component. The new location of the top-leftcorner is specified by x and
y, and thenew size is specified by width and height.
        super.setDefaultCloseOperation(EXIT_ON_CLOSE);
        super.setVisible(true);
    }
}

```

```

//Valentina Pineda Barrón
//23/11/2023
//Controlador del menu de la biblioteca
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.File;
import java.util.Scanner;

import javax.swing.JOptionPane;

public class ControladorMenu extends VistaMenu{

    /**
     *
     */
    private static final long serialVersionUID = 1L;

    private static Biblioteca bailleres = new Biblioteca ("Biblioteca
Raúl Baillères Jr.", "Río Hondo 1, Altavista, Álvaro
Obregón", "Valentina");

```

```

// Da un mensaje de los usuarios en el sistema
public class EscuchaBotonUsuarios implements ActionListener {

    @Override
    public void actionPerformed(ActionEvent e) {
        // TODO Auto-generated method stub
        if(bailleres.getUsuarios().isEmpty())
            JOptionPane.showMessageDialog(null,"No hay
usuarios registrados","Usuarios
Registrados",JOptionPane.INFORMATION_MESSAGE);
        else

            JOptionPane.showMessageDialog(null,bailleres.getUsuarios(),"Usuario
s Registrados",JOptionPane.INFORMATION_MESSAGE);
    }

}

//Da un mensaje de los materiales en el inventario
public class EscuchaBotonInventario implements ActionListener {

    @Override
    public void actionPerformed(ActionEvent e) {
        // TODO Auto-generated method stub
        if(bailleres.getUsuarios().isEmpty())
            JOptionPane.showMessageDialog(null,"No hay libros
registrados","Inventario Registrado",JOptionPane.INFORMATION_MESSAGE);
        else

            JOptionPane.showMessageDialog(null,bailleres.getInventario(),"Inven
tario Registrado",JOptionPane.INFORMATION_MESSAGE);
    }

}

//Da un mensaje de la informacion de la biblioteca
public class EscuchaBotonInformacion implements ActionListener {

    @Override
    public void actionPerformed(ActionEvent e) {
        // TODO Auto-generated method stub

        JOptionPane.showMessageDialog(null,bailleres.toString(),"Informació
n Biblioteca",JOptionPane.INFORMATION_MESSAGE);
    }

}

//Usa el método buscar usuario por nombre
// Regresa el usuario/usuarios con ese nombre
public class EscuchaBotonNombre implements ActionListener {

    @Override

```

```

        public void actionPerformed(ActionEvent e) {
            // TODO Auto-generated method stub
            String sRespuesta;

            sRespuesta = txtBuscar.getText();

            if(!sRespuesta.isEmpty()) {
                txtBuscar.setText("");
                /*todos los métodos o la mayoría
                * borran la casilla utilizada despues de ingresar
datos validos
                */

                if(bailleres.buscarUsuarioNombre(sRespuesta).isEmpty())
                    JOptionPane.showMessageDialog(null,"No hay usuarios con el nombre ingresado","Buscar nombre",JOptionPane.INFORMATION_MESSAGE);
                else

                    JOptionPane.showMessageDialog(null,bailleres.buscarUsuarioNombre(sRespuesta),"Buscar nombre",JOptionPane.INFORMATION_MESSAGE);
            }
            else {
                //si no se ingresa nada y pica el boton, envia un
mensaje que solicita
                // el método para evaluar
                txtBuscar.setText("");
                JOptionPane.showMessageDialog(null,"Error: Ingresa un nombre de usuario válido","Error",JOptionPane.INFORMATION_MESSAGE);
            }
        }

        //Usa el metodo busca clave usuario
        //Regresa el usuario con esa clave
        public class EscuchaBotonClave implements ActionListener {

            @Override
            public void actionPerformed(ActionEvent e) {
                // TODO Auto-generated method stub
                String sRespuesta;
                int numRespuesta;

                sRespuesta = txtBuscar.getText();

                //El try es por si no se ingresan numeros en el campo o
esta vacío

                //envie un mensaje de error
                try {
                    numRespuesta = Integer.parseInt(sRespuesta);
                    txtBuscar.setText("");

                    if(bailleres.buscarUsuarioCuenta(numRespuesta).isEmpty())
                        JOptionPane.showMessageDialog(null,"No hay usuarios con la clave ingresada","Buscar clave",JOptionPane.INFORMATION_MESSAGE);
                }
            }
        }
    }

```

```

        else

            JOptionPane.showMessageDialog(null,bailleres.buscarUsuarioCuenta(nu
mRespuesta),"Buscar clave",JOptionPane.INFORMATION_MESSAGE);
        } catch (NumberFormatException ex) {
            txtBuscar.setText("");
            JOptionPane.showMessageDialog(null,"Error: Ingresa una
clave válida","Error",JOptionPane.INFORMATION_MESSAGE);
            return;
        }
    }

    // Usa el metodo buscar material por titulo
    public class EscuchaBotonTitulo implements ActionListener {

        @Override
        public void actionPerformed(ActionEvent e) {
            // TODO Auto-generated method stub
            String sRespuesta;

            sRespuesta = txtBuscar.getText();

            if(!sRespuesta.isEmpty()) {
                txtBuscar.setText("");

                if(bailleres.buscarInformacionTitulo(sRespuesta).isEmpty())
                    JOptionPane.showMessageDialog(null,"No hay
libros con el título ingresado","Buscar
título",JOptionPane.INFORMATION_MESSAGE);
                else

                    JOptionPane.showMessageDialog(null,bailleres.buscarInformacionTitul
o(sRespuesta),"Buscar título",JOptionPane.INFORMATION_MESSAGE);
            }
            else {
                txtBuscar.setText("");
                JOptionPane.showMessageDialog(null,"Error: Ingresa
un título válido","Error",JOptionPane.INFORMATION_MESSAGE);
            }
        }

    }

    //Usa el metodo buscar material por autor
    public class EscuchaBotonAutor implements ActionListener {

        @Override
        public void actionPerformed(ActionEvent e) {
            // TODO Auto-generated method stub
            String sRespuesta;

            sRespuesta = txtBuscar.getText();

            if(!sRespuesta.isEmpty()) {
                txtBuscar.setText("");

                if(bailleres.buscarInformacionAutor(sRespuesta).isEmpty())

```

```

                                JOptionPane.showMessageDialog(null, "No hay
libros con el autor ingresado", "Buscar
autor", JOptionPane.INFORMATION_MESSAGE);
                                else

                                JOptionPane.showMessageDialog(null, bailleres.buscarInformacionAutor
(sRespuesta), "Buscar autor", JOptionPane.INFORMATION_MESSAGE);
                                }
                                else {
                                    txtBuscar.setText("");
                                    JOptionPane.showMessageDialog(null, "Error: Ingresa
un autor válido", "Error", JOptionPane.INFORMATION_MESSAGE);
                                }
                            }

//Usa el método buscar material por año
public class EscuchaBotonAnio implements ActionListener {

    @Override
    public void actionPerformed(ActionEvent e) {
        // TODO Auto-generated method stub
        String sRespuesta;
        int numRespuesta;

        sRespuesta = txtBuscar.getText();

        try {
            numRespuesta = Integer.parseInt(sRespuesta);
            txtBuscar.setText("");

            if(bailleres.buscarInformacionAnio(numRespuesta).isEmpty())
                JOptionPane.showMessageDialog(null, "No hay
libros con el año ingresado", "Buscar
autor", JOptionPane.INFORMATION_MESSAGE);
            else

                JOptionPane.showMessageDialog(null, bailleres.buscarInformacionAnio(
numRespuesta), "Buscar año", JOptionPane.INFORMATION_MESSAGE);
        } catch (NumberFormatException ex) {
            txtBuscar.setText("");
            JOptionPane.showMessageDialog(null, "Error: Ingresa un
año válido", "Error", JOptionPane.INFORMATION_MESSAGE);
            return;
        }
    }

//busca material por código
public class EscuchaBotonCodigo implements ActionListener {

    @Override
    public void actionPerformed(ActionEvent e) {
        // TODO Auto-generated method stub
        String sRespuesta;
        int numRespuesta;

```

```

        sRespuesta = txtBuscar.getText();

        try {
            numRespuesta = Integer.parseInt(sRespuesta);
            txtBuscar.setText("");

            if(bailleres.buscarInformacionCodigo(numRespuesta).isEmpty())
                JOptionPane.showMessageDialog(null,"No hay
libros con el código ingresado","Buscar
código",JOptionPane.INFORMATION_MESSAGE);
            else

                JOptionPane.showMessageDialog(null,bailleres.buscarInformacionCodigo(numRespuesta),"Buscar código",JOptionPane.INFORMATION_MESSAGE);
        } catch (NumberFormatException ex) {
            txtBuscar.setText("");
            JOptionPane.showMessageDialog(null,"Error: Ingresa un
código válido","Error",JOptionPane.INFORMATION_MESSAGE);;
            return;
        }
    }

}

// Añade un material al inventario
public class EscuchaBotonAceptar1 implements ActionListener {

    @Override
    public void actionPerformed(ActionEvent e) {
        // TODO Auto-generated method stub
        String sTitulo, sAutor, sAnio;
        int numAnio;
        boolean respuesta;

        sTitulo = txtInfo1.getText();
        sAutor = txtInfo2.getText();
        sAnio= txtInfo3.getText();

        if(!sTitulo.isEmpty()||!sAutor.isEmpty()||!sAnio.isEmpty()) {
            try {
                numAnio = Integer.parseInt(sAnio);
                respuesta=bailleres.anadirMaterial(sTitulo,
sAutor, numAnio);

                txtInfo1.setText("");
                txtInfo2.setText("");
                txtInfo3.setText("");
                if(respuesta)

                    JOptionPane.showMessageDialog(null,"Alta Exitosa","Añadir
Material",JOptionPane.INFORMATION_MESSAGE);
                else

                    JOptionPane.showMessageDialog(null,"El inventario ha alcanzado su
capacidad, borra un material para volver a añadir","Añadir
Material",JOptionPane.INFORMATION_MESSAGE);
            } catch (NumberFormatException ex) {

```

```

// si la casilla año no tiene un número
//tambien indica que tienes que escribir en
cada casilla
// el único problema es que el usuario
despues debe borrar lo ingresado
txtInfo1.setText("Título");
txtInfo2.setText("Autor");
txtInfo3.setText("Año");

JOptionPane.showMessageDialog(null,"Error: Ingresa un año
valido","Error",JOptionPane.INFORMATION_MESSAGE);;
return;
}
}
else {
// si alguna casilla esta vacia envia
mensaje
//tambien indica que tienes que escribir en
cada casilla
// el único problema es que el usuario despues
debe borrar lo ingresado
JOptionPane.showMessageDialog(null,"Error:
Ingresa datos válidos","Error",JOptionPane.INFORMATION_MESSAGE);
txtInfo1.setText("Título");
txtInfo2.setText("Autor");
txtInfo3.setText("Año");
}
}
}
//añade ejemplares a un material
//usa el metodo añadirEjemplar
public class EscuchaBotonAnadirE implements ActionListener {

@Override
public void actionPerformed(ActionEvent e) {
// TODO Auto-generated method stub
String sRespuesta;
int numRespuesta;

sRespuesta = txtInfo4.getText();
try {
numRespuesta = Integer.parseInt(sRespuesta);
txtInfo4.setText("");
if(!bailleres.anadirEjemplar(numRespuesta))
JOptionPane.showMessageDialog(null,"No hay
libros con el codigo ingresado","Añadir
ejemplar",JOptionPane.INFORMATION_MESSAGE);
else
JOptionPane.showMessageDialog(null,"El
ejemplar ya ha sido añadido","Añadir
ejemplar",JOptionPane.INFORMATION_MESSAGE);
} catch (NumberFormatException ex) {
JOptionPane.showMessageDialog(null,"Error: Ingresa un
código válido","Error",JOptionPane.INFORMATION_MESSAGE);
return;
}
}
}

```

```

    }

    // quita un material del inventario
    // usa el metodo quitar material
    public class EscuchaBotonQuitarM implements ActionListener {

        @Override
        public void actionPerformed(ActionEvent e) {
            // TODO Auto-generated method stub
            String sRespuesta;
            int numRespuesta;

            sRespuesta = txtInfo4.getText();
            try {
                numRespuesta = Integer.parseInt(sRespuesta);
                txtInfo4.setText("");
                if(!bailleres.quitarMaterial(numRespuesta))
                    JOptionPane.showMessageDialog(null, "No hay
libros con el codigo ingresado", "Eliminar
Material", JOptionPane.INFORMATION_MESSAGE);
                else
                    JOptionPane.showMessageDialog(null, "El
material ha sido retirado", "Eliminar
Material", JOptionPane.INFORMATION_MESSAGE);
            } catch (NumberFormatException ex) {
                txtInfo4.setText("");
                JOptionPane.showMessageDialog(null, "Error: Ingresa un
código válido", "Error", JOptionPane.INFORMATION_MESSAGE);
                return;
            }
        }

    }

    // Añade un usuario al inventario
    public class EscuchaBotonAceptar2 implements ActionListener {

        @Override
        public void actionPerformed(ActionEvent e) {
            // TODO Auto-generated method stub
            String sNombre, sCorreo;
            int respuesta;

            sNombre = txtInfo5.getText();
            sCorreo = txtInfo6.getText();

            if(!sNombre.isEmpty() & !sCorreo.isEmpty()) {
                txtInfo5.setText("");
                txtInfo6.setText("");
                respuesta=bailleres.crearUsuario(sNombre,sCorreo);
                if(respuesta!=-1) {
                    JOptionPane.showMessageDialog(null, "Alta
Exitosa, la clave del nuevo usuario es: "+respuesta, "Añadir
Usuario", JOptionPane.INFORMATION_MESSAGE);
                }
            }
            else {

```

```

        JOptionPane.showMessageDialog(null, "Usuarios
ha alcanzado su capacidad, borra un usuario para añadir", "Añadir
Usuario", JOptionPane.INFORMATION_MESSAGE);
    }
}
else {
    JOptionPane.showMessageDialog(null, "Error: Ingresa
datos válidos", "Error", JOptionPane.INFORMATION_MESSAGE);
    txtInfo5.setText("Nombre");
    txtInfo6.setText("Correo");
}
}
}
// quita un usuario del inventario
public class EscuchaBotonQuitarU implements ActionListener {

    @Override
    public void actionPerformed(ActionEvent e) {
        // TODO Auto-generated method stub
        String sRespuesta;
        int numRespuesta;

        sRespuesta = txtInfo7.getText();
        try {
            numRespuesta = Integer.parseInt(sRespuesta);
            txtInfo7.setText("");
            if(!bailleres.quitarUsuario(numRespuesta))
                JOptionPane.showMessageDialog(null, "No hay
usuarios con la clave ingresada", "Eliminar
Usuario", JOptionPane.INFORMATION_MESSAGE);
            else
                JOptionPane.showMessageDialog(null, "El
usuario ha sido retirado", "Eliminar
Usuario", JOptionPane.INFORMATION_MESSAGE);
        } catch (NumberFormatException ex) {
            txtInfo7.setText("");
            JOptionPane.showMessageDialog(null, "Error: Ingresa una
clave válida", "Error", JOptionPane.INFORMATION_MESSAGE);
            return;
        }
    }
}
//Regresa el historial de un material
public class EscuchaBotonHistorialM implements ActionListener {

    @Override
    public void actionPerformed(ActionEvent e) {
        // TODO Auto-generated method stub
        String sRespuesta, resultado;
        int numRespuesta;

        sRespuesta = txtInfo8.getText();
        try {
            numRespuesta = Integer.parseInt(sRespuesta);
            resultado=bailleres.historialPrestamosInformacion(numRespuesta);

```

```

        txtInfo8.setText("");
        if(resultado.isEmpty())
            JOptionPane.showMessageDialog(null,"No hay
libros con el código ingresado","Historial Prestamos
Material",JOptionPane.INFORMATION_MESSAGE);
        else

            JOptionPane.showMessageDialog(null,resultado,"Historial Prestamos
Material",JOptionPane.INFORMATION_MESSAGE);
        } catch (NumberFormatException ex) {
            txtInfo8.setText("");
            JOptionPane.showMessageDialog(null,"Error: Ingresa un
código válido","Error",JOptionPane.INFORMATION_MESSAGE);
            return;
        }
    }

    //Regresa el historial de un usuario
    public class EscuchaBotonHistorialU implements ActionListener {

        @Override
        public void actionPerformed(ActionEvent e) {
            // TODO Auto-generated method stub
            String sRespuesta, resultado;
            int numRespuesta;

            sRespuesta = txtInfo8.getText();
            try {
                numRespuesta = Integer.parseInt(sRespuesta);

                resultado=bailleres.historialPrestamosUsuario(numRespuesta);
                txtInfo8.setText("");
                if(resultado.isEmpty())
                    JOptionPane.showMessageDialog(null,"No hay
usuarios con la clave ingresada","Historial Prestamos
Usuario",JOptionPane.INFORMATION_MESSAGE);
                else

                    JOptionPane.showMessageDialog(null,resultado,"Historial Prestamos
Usuario",JOptionPane.INFORMATION_MESSAGE);
            } catch (NumberFormatException ex) {
                txtInfo8.setText("");
                JOptionPane.showMessageDialog(null,"Error: Ingresa una
clave válida","Error",JOptionPane.INFORMATION_MESSAGE);
                return;
            }
        }

    }

    //Regresa el historial actual de un usuario
    public class EscuchaBotonHistorialUA implements ActionListener {

        @Override
        public void actionPerformed(ActionEvent e) {
            // TODO Auto-generated method stub
            String sRespuesta, resultado;
            int numRespuesta;

```

```

        sRespuesta = txtInfo8.getText();
        try {
            numRespuesta = Integer.parseInt(sRespuesta);

            resultado=bailleres.historialPrestamosActualesUsuario(numRespuesta);
            txtInfo8.setText("");
            if(resultado.isEmpty())
                JOptionPane.showMessageDialog(null,"No hay usuarios con la clave ingresada","Historial Prestamos Actuales del Usuario",JOptionPane.INFORMATION_MESSAGE);
            else

                JOptionPane.showMessageDialog(null,resultado,"Historial Prestamos Actuales del Usuario",JOptionPane.INFORMATION_MESSAGE);
        } catch (NumberFormatException ex) {
            txtInfo8.setText("");
            JOptionPane.showMessageDialog(null,"Error: Ingresa una clave válida","Error",JOptionPane.INFORMATION_MESSAGE);
            return;
        }
    }

    //realiza prestamos
    public class EscuchaBotonRealizar implements ActionListener {

        @Override
        public void actionPerformed(ActionEvent e) {
            // TODO Auto-generated method stub
            String sCodigo, sClave;
            int numCodigo, numClave, respuesta;

            sCodigo = txtInfo9.getText();
            sClave = txtInfo10.getText();

            try {
                numCodigo = Integer.parseInt(sCodigo);
                numClave = Integer.parseInt(sClave);
                respuesta=bailleres.realizarPrestamo(numClave, numCodigo);

                txtInfo9.setText("");
                txtInfo10.setText("");

                //En esta parte se usa el indicador del metodo para imprimir un mensaje apropiado
                if(respuesta==1)
                    JOptionPane.showMessageDialog(null,"El préstamo del material "+sCodigo+" se ha realizado correctamente","Realizar Préstamos",JOptionPane.INFORMATION_MESSAGE);
                else
                    if(respuesta==0)
                        JOptionPane.showMessageDialog(null,"El préstamo no ha sido concluido por tres posibles razones: \nEl usuario cuenta con más de 10 prestamos actuales\nEl usuario ya esta ejerciendo un préstamo de este material\nEl material no esta disponible por el momento","Realizar Prestamos",JOptionPane.INFORMATION_MESSAGE);
            }
        }
    }

```

```

        else
            JOptionPane.showMessageDialog(null, "El
préstamo no ha sido concluido por que el usuario o el material no se
encuentran dentro del sistema", "Realizar
Prestamos", JOptionPane.INFORMATION_MESSAGE);

    } catch (NumberFormatException ex) {
        txtInfo9.setText("");
        txtInfo10.setText("");
        JOptionPane.showMessageDialog(null, "Error: Ingresa un
código y clave válidos", "Error", JOptionPane.INFORMATION_MESSAGE);
        return;
    }
}

//Regresar prestamo
public class EscuchaBotonRegresar implements ActionListener {

    @Override
    public void actionPerformed(ActionEvent e) {
        // TODO Auto-generated method stub
        String sCodigo, sClave;
        int numCodigo, numClave, respuesta;

        sCodigo = txtInfo9.getText();
        sClave = txtInfo10.getText();

        try {
            numCodigo = Integer.parseInt(sCodigo);
            numClave = Integer.parseInt(sClave);
            respuesta=bailleres.regresarPrestamo(numClave,
numCodigo);

            txtInfo9.setText("");
            txtInfo10.setText("");
            if(respuesta==1)
                JOptionPane.showMessageDialog(null, "El regreso del
material "+sCodigo+" se ha realizado correctamente", "Regresar
Préstamos", JOptionPane.INFORMATION_MESSAGE);
            else
                if(respuesta==0)
                    JOptionPane.showMessageDialog(null, "El
regreso no ha sido concluido por dos posibles razones: \nTodos los
ejemplares del material estan en el inventario\nEl usuario no tiene esta
clave en su historial de prestamos actuales", "Regresar
Prestamos", JOptionPane.INFORMATION_MESSAGE);
                else
                    JOptionPane.showMessageDialog(null, "El
préstamo no ha sido concluido por que el usuario o el material no se
encuentran dentro del sistema", "Regresar
Prestamos", JOptionPane.INFORMATION_MESSAGE);

        } catch (NumberFormatException ex) {
            txtInfo9.setText("");
            txtInfo10.setText("");

```

```

        JOptionPane.showMessageDialog(null, "Error: Ingresa un
código y clave válidos", "Error", JOptionPane.INFORMATION_MESSAGE);
        return;
    }
}

public void menu() {
    File archivo, archivov1;
    Scanner lectura;
    String renglon, titulo, autor, nombre, correo;
    int n, anio, num;
    archivo = new File("info.txt");
    archivov1 = new File("usuarios.txt");
    boolean resultado;

    try {
        lectura = new Scanner (archivo);
        renglon = lectura.nextLine();
        n = Integer.parseInt(renglon);

        int i;
        i=1;

        do {
            renglon=lectura.nextLine();
            titulo=renglon;
            renglon=lectura.nextLine();
            autor=renglon;
            renglon=lectura.nextLine();
            anio=Integer.parseInt(renglon);

            resultado=bailleres.anadirMaterial(titulo, autor, anio);
            i++;
        }

        while (resultado==true && i<=n);

        lectura.close();
    }

    catch (Exception error) {
        System.err.println("error");
    }

    try {
        lectura = new Scanner (archivov1);
        renglon = lectura.nextLine();
        n = Integer.parseInt(renglon);

        int i;
        i=1;

```

```

        do {
            renglon=lectura.nextLine();
            nombre=renglon;
            renglon=lectura.nextLine();
            correo=renglon;
            num=bailleres.crearUsuario(nombre,correo);
            i++;
        }

        while (num!=-1 && i<=n);

        lectura.close();
    }

    catch (Exception error) {
        System.err.println("error");
    }

}

public ControladorMenu() {
    super();
    menu();
    btUsuarios.addActionListener(new EscuchaBotonUsuarios());
    btInventario.addActionListener(new EscuchaBotonInventario());
    btInformacion.addActionListener(new
EscuchaBotonInformacion());
    btNombre.addActionListener(new EscuchaBotonNombre());
    btClave.addActionListener(new EscuchaBotonClave());
    btTitulo.addActionListener(new EscuchaBotonTitulo());
    btAutor.addActionListener(new EscuchaBotonAutor());
    btAnio.addActionListener(new EscuchaBotonAnio());
    btCodigo.addActionListener(new EscuchaBotonCodigo());
    btAceptar1.addActionListener(new EscuchaBotonAceptar1());
    btEjemplar.addActionListener(new EscuchaBotonAnadirE());
    btQuitar.addActionListener(new EscuchaBotonQuitarM());
    btAceptar2.addActionListener(new EscuchaBotonAceptar2());
    btQuitar1.addActionListener(new EscuchaBotonQuitarU());
    btHistorialM.addActionListener(new EscuchaBotonHistorialM());
    btHistorialU.addActionListener(new EscuchaBotonHistorialU());
    btHistorialUA.addActionListener(new
EscuchaBotonHistorialUA());
    btRealizar.addActionListener(new EscuchaBotonRealizar());
    btRegresar.addActionListener(new EscuchaBotonRegresar());
}
}

```

```

//Valentina Pineda Barrón
//23/11/2023
//Prueba de la interfaz gráfica de la biblioteca

```

```
public class PruebaMenu {  
    public static void main(String[] args) {  
        /*  
        * VistaCine Cine;  
        * Cine = new VistaCine();  
        */  
  
        ControladorMenu Biblioteca;  
        Biblioteca=new ControladorMenu();  
    }  
}
```