Informe técnico
Taller 5

Integrantes: Antonia Flores
Claudia Lovera
Profesor: Álvaro Castillo
Ayudante: Helmer Pizarro
Paralelo: C2

Explicación del código

Para desarrollar el código, utilizamos la herramienta **Java Swing** para crear las ventanas que aparecen dependiendo de las opciones que el usuario ingrese.

La figura 1 se desplegará al iniciar el programa, pidiendo al usuario que ingrese su contraseña y Rut. Se comparan los datos ingresados con los datos leídos desde el archivo "usuarios.txt" con el código N°1, que desplegará un mensaje dependiendo de la validación que se arroje (Figura 2 o 3)

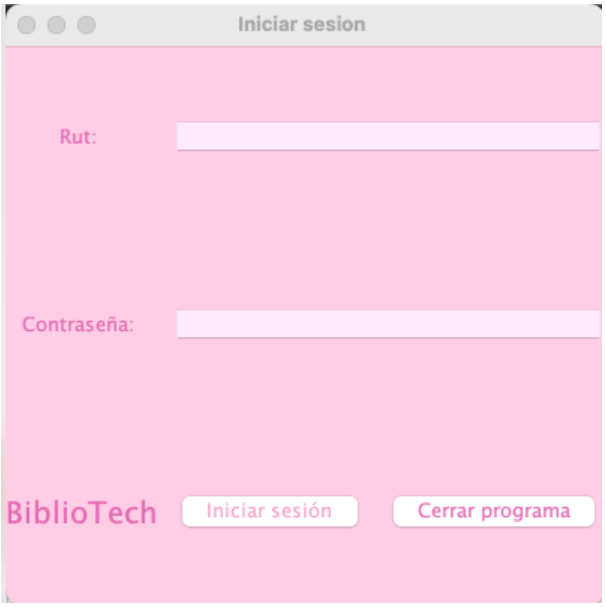


Figura 1

Figura 2

Figura 3

1) Código validación

```
private void ingresar() {
    String rut = rutField.getText();
    String contrasenia = ContraseniaField.getText();
    //Buscar al usuario en la lista según el RUT y la contraseña
    Usuario buscarUsuario = buscarUsuario(rut, contrasenia);
    //Si el usuario es valido, muestra un mensaje y abre la ventan de Menú Inicial
    if (buscarUsuario != null) {
        JOptionPane.showMessageDialog( parentComponent: null,  message: "¡Ingreso Valido!");

        if (menuInicial != null) {
            //Mostrar por ventana si ya existe
            menuInicial.setVisible(true);
        } else {
            //Nueva instancia en Menú Inicial
            menuInicial = new MenuInicial(libroList);
        }
    } else {
        //Muestra un mensaje de error si los datos son incorrectos
        JOptionPane.showMessageDialog( parentComponent: null,  message: "Ha ingresado un dato erroneo...");

    }
}

/**Método para buscar un usuario en la lista según el RUT y la contraseña ingresada
 *
 * @param rut : RUT del usuario
 * @param contrasenia : Contraseña del usuario
 * @return Retorna usuario si se encuentra, o null si no
 */
1 usage   ± claudialovera *
private Usuario buscarUsuario(String rut, String contrasenia) {
    for (Usuario usuario : usuariosList) {
        if (usuario.getRut().equalsIgnoreCase(rut) && usuario.getPassword().equalsIgnoreCase(contrasenia)) {
            return usuario; //Se retorna el usuario si se encuentra
        }
    }
    return null; //Retorna null si el usuario no se encuentra
}
```

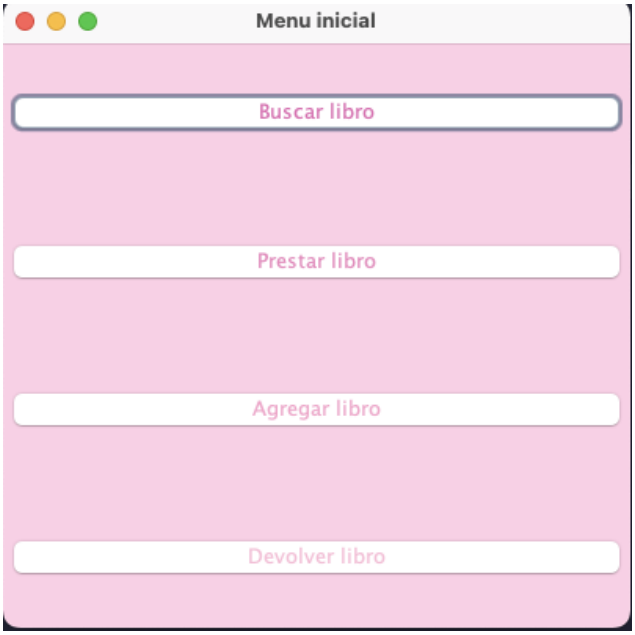
-Se trae la lista que contiene los usuarios y se compara con los datos ingresados desde teclado
-Si el usuario se encuentra en la lista, se desplegará el mensaje “Ingreso valido”

Una vez la validación es correcta se desplegará la siguiente ventana, que mostrará el menú principal, que se realizó de la siguiente manera:

```
if(menuInicial != null){
    menuInicial.setVisible(true);
}else{
    menuInicial = new MenuInicial();
}
```

Se realizó un ciclo IF que establece que cuando es ***‘menuInicial’*** sea distinto de null, este será visible.

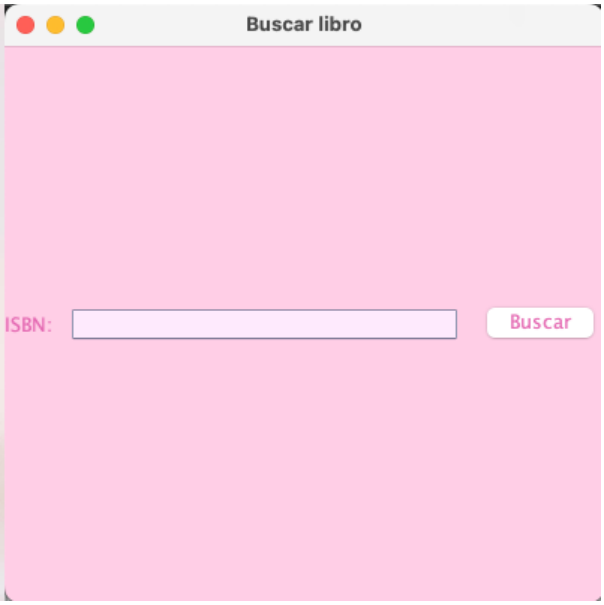
Que tendrá 4 botones que el usuario podrá elegir.



El usuario podrá elegir la opción que quiera y se desplegarán las siguientes ventanas según su elección.

1) *Buscar libro*

Donde el usuario tendrá que ingresar el ISBN del libro que está buscando, si el libro que está buscando corresponde con alguno de los isbn leídos del archivo, se desplegar el título, autor, categoría y número de copias disponibles.



Si no está disponible, se le mostrará una ventana indicando que el libro no existe o no está disponible.

Código: Clase BuscarLibro

- Descripción: *La clase **'BuscarLibro'** representa la ventana de búsqueda de libros en el sistema. Se utiliza **'JFrame'** para crear una ventana*.
- Atributos: ***'libroList'** es una lista que almacena los datos de los libros.*
- Función: *En el método **'buscar()'**, se obtiene el código ISBN ingresado por el usuario y se busca el libro correspondiente en la lista **'libroList'**. Si se encuentra, se muestra la información del libro en un cuadro de diálogo, de lo contrario, se muestra un mensaje de error.*
- Relacións: *La clase **'BuscarLibro'** interactúa con la clase **'Libro'** para buscar libros en la lista.*

```
1 usage
private void agregarLibro() {
    String isbn = textField2.getText();
    String titulo = TituloField.getText();
    String autor = AutorField.getText();
    String categoria = CategoriaField.getText();

    if (libroExiste(isbn)) {
        JOptionPane.showMessageDialog( parentComponent: AgregarLibro.this,  message: "El libro con el ISBN: " + isbn + " ya se encuentra registrado",  title: "Error", JOptionPane.ERROR_MESSAGE);

    } else {
        //Agregar nuevo libro a la lista
        Libro nuevoLibro = new Libro(isbn, titulo, autor, categoria,  copies: 0,  price: 0);
        libroList.add(nuevoLibro);
        JOptionPane.showMessageDialog( parentComponent: AgregarLibro.this,  message: "¡El libro ha sido agregado exitosamente!",  title: "Información", JOptionPane.INFORMATION_MESSAGE);
        reestablecerEntrada();
    }
}
```

```
1 usage
private Libro buscarLibroISBN(List<Libro> libroList, String isbn) {
    for (Libro libro : libroList) {
        if (libro.getIsbn().equalsIgnoreCase(isbn)) {
            return libro;
        }
    }

    return null;
}
```

2) *Prestar libro*

El usuario deberá ingresar el isbn del libro que desea arrendar, si el isbn pertenece a alguno de los libros leídos desde archivo, el sistema desplegará una ventana que le indicará al usuario que se ha realizado el préstamo exitosamente, en caso contrario se desplegará una ventana indicando que el libro no existe o no tiene copias disponibles según sea el caso.

Código: Clase PrestarLibro

- Descripción: *La clase **'PrestarLibro'** representa la ventana para prestar un libro del sistema al usuario. Se utiliza la clase **'JFrame'** para crear la ventana.*
- Atributos: ***'libroList'** es la lista que almacena los datos de los libros.*
- Función: *En el método **'buscar()'**, se obtiene el código ISBN ingresado por el usuario y se busca el libro correspondiente en la lista **'libroList'**. Si se encuentra y existen copias disponibles en el sistema, se realiza el préstamo del libro y se muestra un mensaje de éxito, de lo contrario, se muestra un mensaje de error.*
- Relación: *La clase **'PrestarLibro'** interactúa con la clase **'Libro'** para buscar libros en la lista y realizar el préstamo.*

```
1 usage
private void buscar() {
    String isbn = ISBNField.getText();
    Libro libro = buscarLibroISBN(isbn);

    if (libro != null && libro.getCopies() > 0) {
        //Realizar el préstamo del libro
        libro.setCopies(libro.getCopies() - 1);
        JOptionPane.showMessageDialog( parentComponent: PrestarLibro.this,  message: "¡Préstamo exitoso!",  title: "Información", JOptionPane.INFORMATION_MESSAGE);

    } else {
        JOptionPane.showMessageDialog( parentComponent: PrestarLibro.this,  message: "No se puede prestar el libro con el ISBN "+ isbn +" ingresado",  title: "Error", JOptionPane.ERROR_MESSAGE);
    }
}

1 usage
private Libro buscarLibroISBN(String isbn) {
    for (Libro libro : libroList) {
        if (libro.getIsbn().equalsIgnoreCase(isbn)) {
            return libro;
        }
    }

    return null;
}
```

3) Agregar libro

El sistema pedirá al usuario que ingrese Los datos del libro que quiere agregar, que serán:

-Titulo
-Autor
-Categoría
-ISBN

En caso de que el libro que quiera agregar el usuario ya exista, se desplegará un mensaje indicando que el libro ya existe.

Código: Clase AgregarLibro

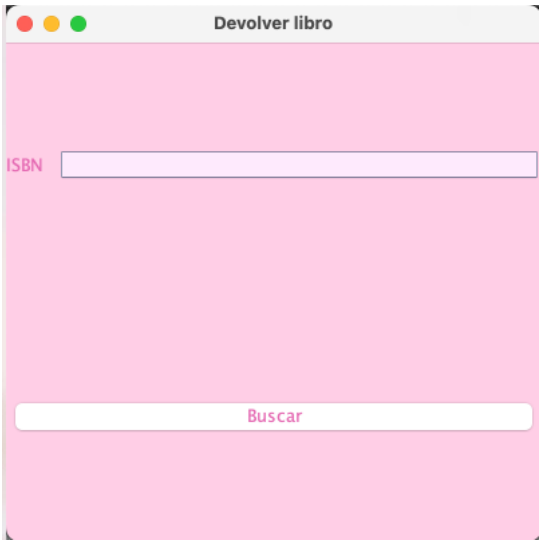
- Descripción: *La clase **'AgregarLibro'** representa la ventana para agregar un nuevo libro al sistema. Se utiliza **'JFrame'** para crear la ventana.*
- Atributos: ***'libroList'** es una lista que almacena los datos de los libros para su uso.*
- Función: *En el método **'agregarLibro()'**, se obtiene los datos ingresados por el usuario que serían; el ISBN, Título, Autor, Categoría, Número de copias y el precio. Se verifica si el libro existe en la lista **'libroList'**, si no existe, se crea una nueva instancia de la clase **'Libro'** y se agrega a la lista, de lo contrario, se muestra un mensaje de error.*

- Relación: *La clase **'AgregarLibro'** interactúa con la clase **'Libro'** para agregar libros a la lista **'libroList'**.*

```
1 usage
private void agregarLibro() {
    String isbn = textField2.getText();
    String titulo = TituloField.getText();
    String autor = AutorField.getText();
    String categoria = CategoriaField.getText();

    if (libroExiste(isbn)) {
        JOptionPane.showMessageDialog( parentComponent: AgregarLibro.this,  message: "El libro con el ISBN: " + isbn + " ya se encuentra registrado",  title: "Error", JOptionPane.ERROR_MESSAGE);

    } else {
        //Agregar nuevo libro a la lista
        Libro nuevoLibro = new Libro(isbn, titulo, autor, categoria,  copies: 0,  price: 0);
        libroList.add(nuevoLibro);
        JOptionPane.showMessageDialog( parentComponent: AgregarLibro.this,  message: "¡El libro ha sido agregado exitosamente!",  title: "Información", JOptionPane.INFORMATION_MESSAGE);
        reestablecerEntrada();
    }
}

1 usage
private boolean libroExiste(String isbn) {
    for (Libro libro : libroList) {
        if (libro.getIsbn().equalsIgnoreCase(isbn)) {
            return true;
        }
    }

    return false;
}
```

4) Devolver Libro

El sistema pedirá al usuario el ISBN del libro que quiere devolver, en caso de no existir el libro, se desplegará un mensaje de error.



Código: Clase DevolverLibro

- Descripción: *La clase **'DevolverLibro'** representa la ventana para devolver un libro al sistema. Utiliza **'JFrame'** para crear una ventana.*
- Atributos: ***'libroList'** es una lista que almacena los datos de los libros.*
- Función: *En el método **'buscar()'**, se obtiene el código ISBN ingresado por el usuario y se busca el libro correspondiente en la lista **'libroList'**. Si se encuentra y hay al menos una copia prestada, se realiza la devolución del libro, aumentando el número de copias disponibles. Se muestra un mensaje de éxito y se cierra la ventana, si no se encuentra el libro o no hay copias prestadas, se muestra un mensaje de error.*
- Relación: *La clase **'DevolverLibro'** interactúa con la clase **'Libro'** para buscar libros en la lista y realizar la devolución.*

```java
1 usage
private void buscar() {
    String isbn = ISBNField.getText();
    Libro libro = buscarLibroISBN(libroList,isbn);

    if(libro != null && libro.getNumCopias() > 0 ){
        //Aumenta el numero de copias
        libro.setNumCopias(libro.getNumCopias() + 1);
        JOptionPane.showMessageDialog( parentComponent: null, message: "'DEVOLUCION EXITOSA;: ");
        dispose();


    }else{
        JOptionPane.showMessageDialog( parentComponent: DevolverLibro.this,  message: "No se puede prestar el libro con el ISBN "+ isbn +" ingresado",  title: "Error", JOptionPane.ERROR_MESSAGE);
    }
}

1 usage
private Libro buscarLibroISBN(List<Libro> libroList, String isbn) {
    for (Libro libro : libroList) {
        if (libro.getIsbn().equalsIgnoreCase(isbn)) {
            return libro;
        }
    }

    return null;
}
```

5) Main

Se encarga de coordinar el inicio y la configuración inicial del sistema de biblioteca. Desde esta clase, se establece la conexión entre los componentes principales del programa y se inicia el flujo de ejecución.

Código: Main

- Descripción: *La clase **'Main'** es la clase principal del programa, siendo el punto base del código.*
- Función: *El método **'main()'** se encarga de iniciar la aplicación y coordinar el flujo principal del programa. Se crean las instancias iniciales de la interfaz gráfica de usuario y se llaman a los métodos para cargar los datos de libros y usuarios desde archivos de texto.*
- Relación: *La clase **'Main'** interactúa con otras clases del sistema, como **'Ventana'**, **'BuscarLibro'**, **'MenuInicial'**, **'Usuario'**, **'Libro'** y otras, para iniciar la aplicación y establecer la base de datos inicial del sistema.*

```java
public static void main(String[] args) {
    leerArchivoUsuarios();
    leerArchivoLibros();

    /**
     * Crea e inicia la ventana
     */
    claudialovera *
    SwingUtilities.invokeLater(new Runnable() {
        no usages   claudialovera *
        @Override
        public void run() {
            //Crea la ventana enviandole los arreglos para su uso
            JFrame ventana = new Ventana(libroList,usuarioList);
            //Establece el tamaño de la ventana
            ventana.setSize( width: 400, height: 400);
            //Pone como visible la ventana
            ventana.setVisible(true);

            //Inicia el menu principal
            MenuInicial menuInicial = new MenuInicial(libroList);
            menuInicial.setVisible(true);

        }
    });

}
```

6) Inicio Sesión

Proporciona la ventana de inicio de sesión y gestiona la autenticación de los usuarios en el sistema de biblioteca. Además, se encarga de crear un archivo de datos al cerrar el programa.

- Descripción: *La clase **'Ventana'** representa la ventana de inicio de sesión del sistema. Utiliza **'JFrame'** para crear una ventana.*

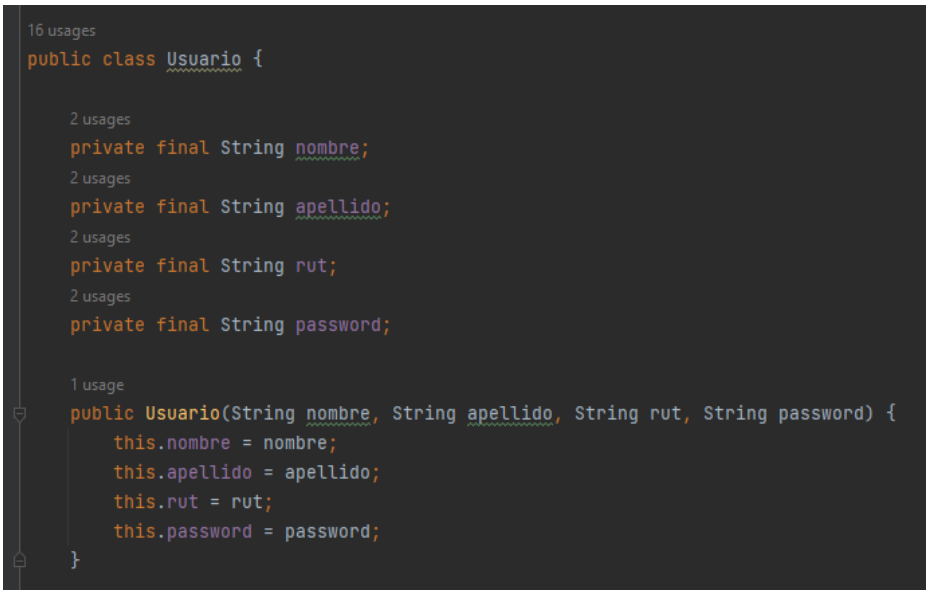
- Atributos: ***'libroList'*** *y* ***'usuariosList'*** *son listas que almacenan los datos de libros y usuarios respectivamente.*
- Función: *En el método* ***'actionPerformed'*** *del botón* ***"Iniciar Sesión"****, se verifica el RUT y la contraseña ingresados por el usuario, si coinciden con un usuario registrado, se muestra un mensaje de ingreso válido y se crea una instancia de la clase* ***'MenuInicial'****. Además, el método* ***'crearArchivoDatos()'*** *se llama al cerrar el programa para crear un archivo con los datos de los libros prestados por los usuarios.*
- Relación: *La clase* ***'Ventana'*** *interactúa con las clases* ***'Usuario'****,* ***'MenuInicial'*** *y otras clases para gestionar el inicio de sesión y crear el archivo de datos.*

```
public class Ventana extends JFrame {
    3 usages
    private static List<Libro> libroList;
    3 usages
    private static List<Usuario> usuariosList;
    3 usages
    private MenuInicial menuInicial;
    no usages
    private AgregarLibro agregarLibro;
    2 usages
    private JTextField rutField;
    2 usages
    private JTextField ContraseniaField;
    2 usages
    private JButton iniciarSesionButton;
    2 usages
    private JButton cerrarProgramaButton;
    2 usages
    private JPanel Panel;

    /**
     * Constructor de la clase
     * @param listaLibros ; Lista de libros
     * @param listaUsuarios ; Lista de Usuarios
     */
    claudialovera *
    public Ventana(List<Libro> listaLibros, List<Usuario> listaUsuarios) {
        super( title: "Iniciar Sesion");
        setContentPane(Panel);
        setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);

        libroList = listaLibros;
        usuariosList = listaUsuarios;
```

7) Usuario

Representa a los usuarios y proporciona métodos para gestionar su información personal y los libros que han sido prestados. A través de esta clase, se puede acceder, modificar y utilizar los datos de los usuarios.

- Descripción: *La clase **'Usuario'** representa a un usuario en el sistema de biblioteca.*
- Atributos: *rut, nombre, apellido, password representan los datos del usuario.*
- Función: *La clase **'Usuario'** permite acceder y modificar los atributos del usuario, como el RUT, nombre, apellido y contraseña. Además, la clase proporciona métodos para obtener los libros prestados por el usuario.*
- Relación: *La clase **'Usuario'** interactúa con las clases **'Ventana'** y otras clases para gestionar los datos del usuario.*



```
16 usages
public class Usuario {

    2 usages
    private final String nombre;
    2 usages
    private final String apellido;
    2 usages
    private final String rut;
    2 usages
    private final String password;

    1 usage
    public Usuario(String nombre, String apellido, String rut, String password) {
        this.nombre = nombre;
        this.apellido = apellido;
        this.rut = rut;
        this.password = password;
    }
```

8) Libros

Representa un libro en el sistema y almacena información relevante sobre el mismo, como el ISBN, título, autor, categoría, cantidad de copias y precio. Permite acceder y modificar los atributos de un libro, lo que brinda la posibilidad de realizar operaciones relacionadas con los libros, como la búsqueda, préstamo y gestión de copias disponibles.

- Descripción: *La clase **'Libro'** representa a un libro en el sistema de biblioteca.*
- Atributos: *isbn, titulo, autor, categoria, copias y precio representan los datos del libro.*

- Función: *La clase **'Libro'** permite acceder y modificar los atributos del libro, como el ISBN, título, autor, categoría, cantidad de copias y precio. También proporciona métodos para realizar operaciones relacionadas con los libros, como la búsqueda y préstamo de libros.*
- Relación: *La clase **'Libro'** interactúa con las clases **'BuscarLibro'**, **'AgregarLibro'**, **'PrestarLibro'** y otras clases para gestionar los datos de los libros y realizar operaciones relacionadas con ellos.*

```
public class Libro {
    3 usages
    private String isbn;
    3 usages
    private String title;
    3 usages
    private String author;
    3 usages
    private String category;
    3 usages
    private int copies;
    3 usages
    private int price;

    2 usages
    public Libro(String isbn, String title, String author, String category, int copies, int price) {
        this.isbn = isbn;
        this.title = title;
        this.author = author;
        this.category = category;
        this.copies = copies;
        this.price = price;
    }
```

Recursos de Ayuda Externos

↬ Videos explicativos sobre Interfaz Gráfica de Usuario para JAVA.
Re: https://youtu.be/Fc4uFeMXBS8
Re: https://youtu.be/opL10REyVn4

Estos videos fueron utilizados como apoyo durante la creación de nuestro código para mejorar nuestra implementación y comprender de mejor manera el proceso de la creación de una interfaz gráfica.

Tiempo estimado en el aporte a la solución.

Fecha inicio de desarrollo de solución: 26 Junio

Fecha termino de desarrollo de solución: 9 Julio
Horas estimadas : 45 horas

Horas estimadas por estudiante:

- **-Antonia Flores = 22,5**
- **-Claudia Lovera = 22,5**