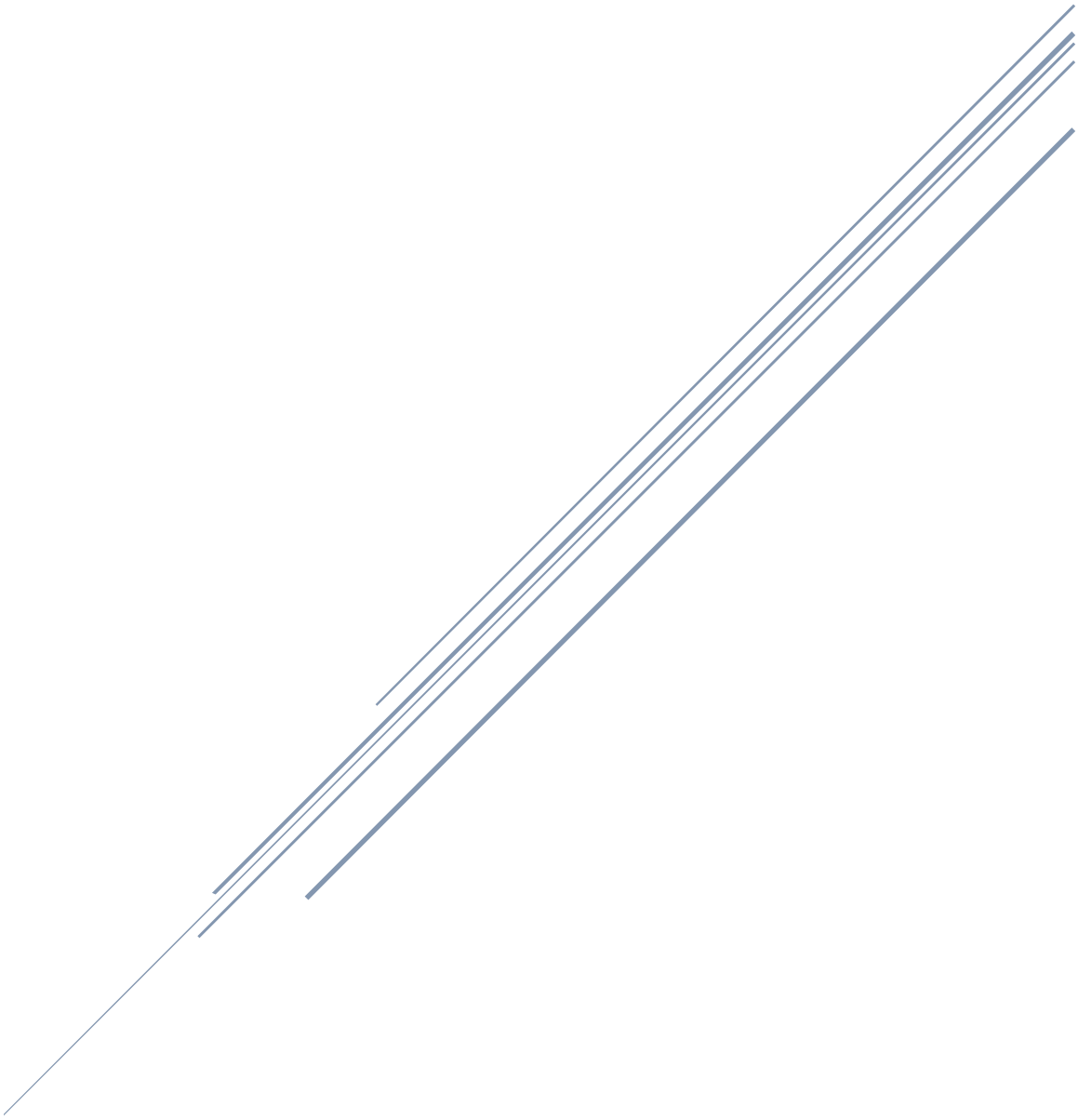
MANUAL TECNICO

BIBLIOTECA FACULTAD DE INGENIERÍA USAC



Proyecto 1 - IPC1
Damian Ignacio Peña Afre - 202110568

Contenido

Paquete Controlador

Clase Biblioteca

```
package Controlador;

import Vista.Inicio;

public class Biblioteca {

    public static void main(String[] args) {
        Usuario administrador = new Usuario("1","administrador", "administrador",
"administrador", "1", "password");
        administrador.crearUsuario();

        Inicio inicio = new Inicio();
        inicio.setVisible(true);
    }
}
```

Creación del usuario administrador

Instancia el JFrame de la página inicial

Clase Usuario

```
package Controlador;

public class Usuario {

    //datos de los usuarios
    public static String datos = "";

    private final static int NUMERO_CAMPOS=6;

    //propiedades de cada usuario
    private String id;
    private String nombre;
    private String apellido;
    private String nombreUsuario;
    private String rol;
    private String contrasena;

    public Usuario(String id, String nombre, String apellido, String nombreUsuario,
String rol, String contrasena){
        this.id = id;
        this.nombre = nombre;
        this.apellido = apellido;
        this.nombreUsuario = nombreUsuario;
        this.rol = rol;
        this.contrasena = contrasena;

        //crearUsuario();
    }
```

Cadena que guarda datos separados por ;

Numero de propiedades

Propiedades de cada usuario

Propiedades inicializadas en el constructor

```
    public boolean crearUsuario(){

        //validacion
        if (!existeUsuario(id)) {
            datos = datos + this.id
+";"+this.nombre+";"+this.apellido+";"+this.nombreUsuario+";"+this.rol+";"
+this.contrasena+"\n";
            return true;
        }else{
            return false;
        }
    }

    public static String[][] datosUsuario(){
        String[] arregloDatos = datos.split("\n");

        String [][] datosFormateados = new String[arregloDatos.length][6];

        if (datos.length()>0) {
            for (int i = 0; i < arregloDatos.length; i++) {
                for (int j = 0; j < NUMERO_CAMPOS; j++) {
                    datosFormateados[i][j]=arregloDatos[i].split(";")[j];
                }
            }
        }


        return datosFormateados;
    }

    public static boolean actualizarUsuario(String id, String nombre, String apellido,
String nombreUsuario, String rol, String contrasena){
            if (existeUsuario(id)) {
                //validación
                String nuevosDatos="";
                for (int i = 0; i < datosUsuario().length; i++) {
                    if (!(datosUsuario()[i][0].equals(id))) {
                        for (int j = 0; j < NUMERO_CAMPOS; j++) {
                            nuevosDatos=nuevosDatos+ datosUsuario()[i][j] +";";
                        }
                        nuevosDatos=nuevosDatos+"\n";
                    }else{
                        nuevosDatos=nuevosDatos+id
+";"+nombre+";"+apellido+";"+nombreUsuario+";"+rol+";"+contrasena+"\n";
                    }
                }

                datos=nuevosDatos;

                return true;
            }else{
                return false;
            }
        }
```

Registro de usuarios en la cadena general

Generación de matriz con los datos de usuarios en la cadena general

Sobrescribe los datos guardados previamente en la cadena general si coincide el ID a modificar, reescribiendo toda la cadena general

```java
public static void eliminarUsuario(String id){

    //validación
    String nuevosDatos="";
    for (int i = 0; i < datosUsuario().length; i++) {
        if (!(datosUsuario()[i][0].equals(id))) {

            for (int j = 0; j < NUMERO_CAMPOS; j++) {
                nuevosDatos=nuevosDatos+ datosUsuario()[i][j] +";";
            }
            nuevosDatos=nuevosDatos+"\n";
        }
    }
    datos=nuevosDatos;
}

public static String[] buscarUsuario(String id) {

    String [] datosUsuarioBusqueda = null;

    for (int i = 0; i < datosUsuario().length; i++) {
        if ((datosUsuario()[i][0].equals(id))) {
            datosUsuarioBusqueda = datosUsuario()[i];
        }
    }

    return datosUsuarioBusqueda;

}

private static boolean existeUsuario(String id){

    //siempre que tenga datos
    if (datos.length()>0) {
        boolean existe =false;
        for (int i = 0; i < datosUsuario().length; i++) {
            if ((datosUsuario()[i][0].equals(id))) {
                existe=true;
            }
        }
        return existe;
    }else{
        return false;
    }
}
```

Reescribe nuevamente la cadena general omitiendo el registro con la ID especificada, eliminado así.

Retorna un arreglo con los datos del usuario con ID especificada

Retorna verdadero en caso encuentre registro de algún usuario con la ID especificada

```java
    public static String[] login(String nombreUsuario, String contrasena){
        String [] mensaje = new String[4];

        for (int i = 0; i < datosUsuario().length; i++) {
            if (datosUsuario()[i][3].equals(nombreUsuario)) {
                if (datosUsuario()[i][5].equals(contrasena)) {
                    mensaje[0]="1";
                    mensaje[1]="Bienvenido al sistema " + datosUsuario()[i][3];
                    mensaje[2]=datosUsuario()[i][0];
                    mensaje[3]=datosUsuario()[i][4];

                }else{
                    mensaje[0]="0";
                    mensaje[1]="Credenciales invalidas para el usuario " +
datosUsuario()[i][3];
                    mensaje[2]=null;
                    mensaje[3]=null;

                }
                break;
            }else{
                //despues de comprobar a todos
                if (i == datosUsuario().length-1) {
                    mensaje[0]="0";
                    mensaje[1]="El usuario no existe, por favor pongase
en contácto con el administrador del sistema para efectuar el registro";
                    mensaje[2]=null;
                    mensaje[3]=null;

                }
            }

        }

        return mensaje;
    }

}
```

Busca entre todos los

Si encuentra un usuario con ese nombre en específico regresa un mensaje, así como el ID y el Rol del usuario validado

Si no coincide la contraseña pero si el usuario regresa un mensaje solamente

Si no coincide ni la contraseña ni el usuario regresa otro tipo de mensaje

NOTA: Los métodos para crear, eliminar, buscar y listar se reutilizan en el resto clases con pequeñas variaciones.

Clase Bibliografía

```
package Controlador;

import java.util.Locale;

public class Bibliografia {

    //Datos requeridos
    //Tipo; Autor; Título; Descripción; Edición; Temas; Frecuencia Actual; Ejemplares;
Área; Copias; Disponibles

    //datos de las bibliografias
    private static String datos = "";

    private final static int NUMERO_CAMPOS=11;

    //propiedades de cada bibliografia
    private String tipo;
    private String autor;
    private String titulo;
    private String descripcion;
    private String edicion;
    private String[] temas;
    private String frecuenciaActual;
    private String ejemplares;
    private String area;
    private String copias;
    private String disponibles;

    public Bibliografia(String tipo, String autor, String titulo, String descripcion,
String edicion, String[] temas, String frecuenciaActual, String ejemplares,String
area,String copias, String disponibles){
        this.tipo = tipo;
        this.autor = autor.trim();
        this.titulo = titulo.trim(); //pequeño formato
        this.descripcion = descripcion.trim();
        this.edicion = edicion.trim();
        this.temas = temas;
        this.frecuenciaActual = frecuenciaActual.trim();
        this.ejemplares = ejemplares.trim();
        this.area = area.trim();
        this.copias = copias.trim();
```

Cadena general, guarda los datos separados por ";" y saltos de linea

Propiedades de cada bibliografía

Establece las propiedades esenciales de cada bibliografía quitando posibles espacios antes de la primera palabra y después de la última.

```
        this.disponibles = disponibles.trim();

    }

    private static boolean existeBibliografia(String titulo){
        //siempre que tenga datos
        if (datos.length()>0) {
            boolean existe =false;
            for (int i = 0; i < datosBibiliografia().length; i++) {
                if ((datosBibiliografia()[i][2].equals(titulo))) {
                    existe=true;
                }
            }
            return existe;
        }else{
            return false;
        }
    }

    private static String formatearTemas(String[] temas){
        String temasFormateados = "";
        if (temas !=null) {
            for (int i = 0; i < temas.length; i++) {
                if (i< temas.length-1) {
                    temasFormateados = temasFormateados + temas[i] + ",";
                }else{
                    temasFormateados = temasFormateados + temas[i];
                }
            }
        }
        return temasFormateados;
    }


    public boolean crearBibliografiaIndividual(){

        if (!existeBibliografia(this.titulo)) {
            //validacion existe bibliografia ?
            String temasFormateados = formatearTemas(this.temas);
            datos = datos + this.tipo
+";"+this.autor+";"+this.titulo+";"+this.descripcion+";"
+this.edicion+";"+temasFormateados+";"+this.frecuenciaActual+
";"+this.ejemplares+";"+this.area+";"+this.copias+";"+this.disponibles+"\n";
            return true;
        }else{
            return false;
        }

    }

    public static boolean verificarFormato(String datos) {
        return true;
    }

    public static boolean crearBibliografiaMasiva(String datosFormateados){
        if (verificarFormato(datosFormateados)) {
            //validacion existe bibliografia ? ->

            String[] datosIndividuales = datosFormateados.split("\n");
```

Convierte un arreglo de temas a una String con cada tema separado por ","

Guarda la información de la bibliografía en la cadena general

Recibe una string con datos separados por ";" y crea un objeto tipo Bibliografía

```
            for (int i = 0; i < datosIndividuales.length; i++) {
                String[] datosFila = datosIndividuales[i].split(";");

                String[] temasFormateados = datosFila[5].split(",");
                Bibliografia bibliografia =
new Bibliografia(datosFila[0],datosFila[1],
datosFila[2],datosFila[3],datosFila[4],
temasFormateados,datosFila[6], datosFila[7],
datosFila[8], datosFila[9],datosFila[10]);
                bibliografia.crearBibliografiaIndividual();
            }
            return true;
        }else {
            return false;
        }
    }







    public static String[][] datosBibiliografia(){
        String[] arregloDatos = datos.split("\n");

        String [][] datosFormateados = new String[arregloDatos.length][NUMERO_CAMPOS];

        if (datos.length()>0) {
            for (int i = 0; i < arregloDatos.length; i++) {
                for (int j = 0; j < NUMERO_CAMPOS; j++) {
                    datosFormateados[i][j]=arregloDatos[i].split(";")[j];
                }
            }
        }


        return datosFormateados;
    }

    public static boolean actualizarBibliografia(String tipo, String autor, String
titulo, String descripcion, String edicion, String[] temas, String frecuenciaActual,
String ejemplares,String area,String copias, String disponibles){
            if (existeBibliografia(titulo)) {
                //validación
                String nuevosDatos="";
                for (int i = 0; i < datosBibiliografia().length; i++) {
                    if (!(datosBibiliografia()[i][2].equalsIgnoreCase(titulo))) {
                        for (int j = 0; j < NUMERO_CAMPOS; j++) {
                            nuevosDatos=nuevosDatos+ datosBibiliografia()[i][j] +";";
                        }
                        nuevosDatos=nuevosDatos+"\n";
                    }else{
                        String temasFormateados = formatearTemas(temas);
                        nuevosDatos=nuevosDatos+tipo
+";"+autor+";"+titulo+";"+descripcion+";"+edicion+";"+temasFormateados+";"+frecuenciaActu
al+";"+ejemplares+";"+area+";"+copias+";"+disponibles+"\n";
                    }
                }
```

Retorna una matriz con los datos de cada bibliografía a partir de la cadena general

*Similar al método explicado anteriormente para la clase usuario

```
            datos=nuevosDatos;

            return true;
        }else{
            return false;
        }
    }

public static void eliminarBibliografia(String titulo){

    //validación
    String nuevosDatos="";
    for (int i = 0; i < datosBibiliografia().length; i++) {
        if (!(datosBibiliografia()[i][2].equals(titulo))) {
            for (int j = 0; j < NUMERO_CAMPOS; j++) {
                nuevosDatos=nuevosDatos+ datosBibiliografia()[i][j] +";";
            }
            nuevosDatos=nuevosDatos+"\n";
        }
    }
    datos=nuevosDatos;
}



public static String[] buscarBibliografia(String titulo) {

    String [] datosUsuarioBusqueda = null;

    if (datosBibiliografia()[0][0] == null) {
        return null;
    }

    for (int i = 0; i < datosBibiliografia().length; i++) {
        if ((datosBibiliografia()[i][2].trim().equalsIgnoreCase(titulo))) {
            datosUsuarioBusqueda = datosBibiliografia()[i];
        }
    }

    return datosUsuarioBusqueda;

}

public static String disponibilidadBibliografia(String titulo) {

    String datosUsuarioBusqueda=null;

    for (int i = 0; i < datosBibiliografia().length; i++) {
        if ((datosBibiliografia()[i][2].trim().equalsIgnoreCase(titulo))) {
            datosUsuarioBusqueda = datosBibiliografia()[i][10];

        }
    }

    return datosUsuarioBusqueda;

}
```

***Similar al método explicado anteriormente para la clase usuario**

Validación para valores nulos

Regresa una cadena con la disponibilidad de la bibliografía con el tilo especificado

Cuenta el numero de registros

```
    public static String[][] buscarCoincidenciasBibliografia(String
coincidenciaDeBusqueda) {

        //Contar coincidencias
        int coincidencias = 0;
        for (int i = 0; i < datosBibiliografia().length; i++) {
            //verificar si contiene el tema
            String [] temas = datosBibiliografia()[i][5].split(",");
            for (int j = 0; j < temas.length; j++) {
                if (temas[j].trim().equalsIgnoreCase(coincidenciaDeBusqueda)) {
                    coincidencias++;
                }
            }
        }

        if (coincidencias>0) {
            String [][] datosUsuarioBusqueda = new String[coincidencias][NUMERO_CAMPOS];
            int contadorAux =0;

            for (int i = 0; i < datosBibiliografia().length; i++) {
                String [] temas = datosBibiliografia()[i][5].split(",");
                for (int j = 0; j < temas.length; j++) {
                    if (temas[j].trim().equalsIgnoreCase(coincidenciaDeBusqueda)) {
                        datosUsuarioBusqueda[contadorAux] =datosBibiliografia()[i];
                        contadorAux++;
                    }
                }

            }

            return datosUsuarioBusqueda;
        }else{
            return null;
        }

    }

}
```

Clase Préstamo

```
package Controlador;

import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.Date;

public class Prestamo {

    //datos de los usuarios
    public static String datos = "";
    public static int numeroPrestamo =-1;

    private final static int NUMERO_CAMPOS=5;

    //propiedades de cada prestamo
    private String id;
    private String titulo;
    private String fecha;
    private String devuelto; //0 no devuelto, 1 devuelto
    private String idUsuario;

    public Prestamo(String titulo, String idUsuario){
        DateTimeFormatter formatoDeFecha = DateTimeFormatter.ofPattern("yyyy/MM/dd
```

Cadena general que contiene los datos de prestamos separados por ";" y el numero de correlativo de cada préstamo

Propiedades del préstamo

Aumenta el correlativo en 1 y almacena las propiedades

```
HH:mm:ss");

        numeroPrestamo+=1;
        this.id = (numeroPrestamo) +"";
        this.titulo = titulo;
        this.fecha = formatoDeFecha.format(LocalDateTime.now());
        this.devuelto = "0";
        this.idUsuario = idUsuario;
    }

    private static boolean modificarDisponiblidad(String tituloBuscar, boolean aumentar)
{

        System.out.println(tituloBuscar);
        String[] datosBibliografia = Bibliografia.buscarBibliografia(tituloBuscar);

            String tipo = datosBibliografia[0];
            String autor = datosBibliografia[1];
            String titulo = datosBibliografia[2];
            String descripcion = datosBibliografia[3];
            String edicion = datosBibliografia[4];
            String[] temas = datosBibliografia[5].split(",");
            String frecuenciaActual = datosBibliografia[6];
            String ejemplares = datosBibliografia[7];
            String areas = datosBibliografia[8];
            String copias = datosBibliografia[9];
            int disponibles = Integer.parseInt(datosBibliografia[10].trim());

            int nuevaDisponibilidad;
            if (aumentar) {
                nuevaDisponibilidad = disponibles + 1;
            } else {
                nuevaDisponibilidad = disponibles - 1;

            }
            return Bibliografia.actualizarBibliografia(tipo, autor, titulo, descripcion,
edicion, temas, frecuenciaActual, ejemplares, areas, copias, nuevaDisponibilidad+"");
    }

    public String crearPrestamo(){

        String respuesta;

        //validar si existe dispo
        if (existeDisponibilidad(this.titulo)) {

            //modificar disponibilidad
                if(modificarDisponiblidad(this.titulo, false)){
                    datos = datos + this.id
+";"+this.titulo+";"+this.fecha+";"+this.devuelto+";"+this.idUsuario+"\n";
                    respuesta ="Prestamo creado correctamente";
                }else{
                    respuesta ="Hubo algun error";
                }

        }else{
            respuesta ="No existe disponibilidad de esta bibliografía";
        }
        return respuesta;
```

Guarda la información del préstamo en caso haya disponibilidad

En caso de no haber disponibilidad devuelve un mensaje

```
    }

    public static String[][] listarPrestamoNoDevueltos(String idUsuario){
        //String[] datosUsuario = Usuario.buscarUsuario(idUsuario);
        int datosDisponibles = 0;

        if (datosPrestamo()[0][0] != null) {
            for (int i = 0; i < datosPrestamo().length; i++) {
                if ((datosPrestamo()[i][4].equals(idUsuario)) &&
(datosPrestamo()[i][3].equals("0"))) {
                    datosDisponibles++;
                }
            }
        }


        if (datosDisponibles>0) {
            String [][] datosPrestamoBusqueda = new String [datosDisponibles][4];
            int contadorAux = 0;
            for (int i = 0; i < datosPrestamo().length; i++) {
                if ((datosPrestamo()[i][4].equals(idUsuario)) &&
(datosPrestamo()[i][3].equals("0"))) {
                    datosPrestamoBusqueda[contadorAux]=datosPrestamo()[i];
                }
            }

            //regresar tipo ?
            return datosPrestamoBusqueda;
        }else{
            return null;
        }
    }
```

Cuenta el número de préstamos no devueltos por el usuario

```
    public static String[][] listarPrestamos(String idUsuario) {

        if (datosPrestamo()[0][0] == null) {
            return null;
        }

        //Contar prestamos asociados
        int coincidencias = 0;
        for (int i = 0; i < datosPrestamo().length; i++) {
            //verificar si contiene el tema
            if (datosPrestamo()[i][4].equalsIgnoreCase(idUsuario)) {
                coincidencias++;
            }
        }

        if (coincidencias>0) {
            String [][] datosPrestamoBusqueda = new String[coincidencias][NUMERO_CAMPOS];
            int contadorAux =0;
```

Devuelve una matriz con todos los prestamos no devueltos por el usuario especificado

Cuenta el numero de prestamos asociados (devueltos o no) por el usuario

Devuelve una matriz con todos los prestamos independientemente si ya han sido devuelto o no por usuario especificado

```
            for (int i = 0; i < datosPrestamo().length; i++) {
                if (datosPrestamo()[i][4].equalsIgnoreCase(idUsuario)) {
                    datosPrestamoBusqueda[contadorAux] =datosPrestamo()[i];
                    contadorAux++;
                }
            }

            return datosPrestamoBusqueda;
        }else{
            return null;
        }

    }


    public static String devolverPrestamo(String idPrestamo){
        String [] datosUsuarioBusqueda = null;

        for (int i = 0; i < datosPrestamo().length; i++) {
            if ((datosPrestamo()[i][0].equals(idPrestamo)) &&
(datosPrestamo()[i][3].equals("0"))) {
                datosUsuarioBusqueda = datosPrestamo()[i];
            }
        }

        //sobreescribe unicamente la propiedad devuelto
        if (datosUsuarioBusqueda != null){
            String nuevosDatos="";
            for (int i = 0; i < datosPrestamo().length; i++) {
                if (!(datosPrestamo()[i][0].equals(idPrestamo))) {
                    for (int j = 0; j < NUMERO_CAMPOS; j++) {
                        nuevosDatos=nuevosDatos+ datosPrestamo()[i][j] +";";
                    }
                    nuevosDatos=nuevosDatos+"\n";
                }else{
                    nuevosDatos=nuevosDatos+datosUsuarioBusqueda[0]
+";"+datosUsuarioBusqueda[1]+";"+datosUsuarioBusqueda[2]+
";"+"1"+";"+datosUsuarioBusqueda[4]+"\n";
                    modificarDisponiblidad(datosUsuarioBusqueda[1],true);
                }
            }
            datos=nuevosDatos;
            return "Biliografia devuelta correctamente";
        }else{
            return "Biliografia devuelta previamente";
        }
    }

    private boolean existeDisponibilidad(String titulo){

        int disposibles =
Integer.parseInt(Bibliografia.disponibilidadBibliografia(titulo));
        if (disposibles>=1) {
            return true;
        }else{
            return false;
        }
    }
```

Devuelve una matriz con todos los prestamos independientemente si ya han sido devuelto o no por usuario especificado

Si no encontró nada la búsqueda regresa un mensaje

Llama al método creado con anterioridad en la clase bibliografía para determinar si hay disponibilidad de bibliografía.

```
    public static String[][] datosPrestamo(){
        String[] arregloDatos = datos.split("\n");

        String [][] datosFormateados = new String[arregloDatos.length][NUMERO_CAMPOS];

        if (datos.length()>0) {
            for (int i = 0; i < arregloDatos.length; i++) {
                for (int j = 0; j < NUMERO_CAMPOS; j++) {
                    datosFormateados[i][j]=arregloDatos[i].split(";")[j];
                }
            }
        }


        return datosFormateados;
    }



}
```

Retorna una matriz con todos lo prestamos realizados hasta el momento a partir de la cadena general

Clase Reporte

```
package Controlador;

public class Reporte {

    private String encabezado = "";
    private String pie = "\n</body>\n" +
                         "</html>";
```

Inicio (puede ser modificado en función del préstamo) y final del documento HTML

Crea un encabezado con el título de Reporte usuarios

Encabezados de tabla e información

```
    public String reporteUsuarios(){
        String reporte="";

        crearEncabezado("Reporte de usuarios");
        reporte = reporte + encabezado;

        String [] encabezados = {"ID del usuario", "Nombre", "Apellido", "Nombre de
usuario", "Rol", "Bibliografias prestadas"};
        String[][] datosUsuarios = Usuario.datosUsuario();
        String[][] datosReporteFormateados = new String[datosUsuarios.length][6];

        for (int i = 0; i < datosUsuarios.length; i++) {
            datosReporteFormateados[i][0]=datosUsuarios[i][0];
            datosReporteFormateados[i][1]=datosUsuarios[i][1];
            datosReporteFormateados[i][2]=datosUsuarios[i][2];
            datosReporteFormateados[i][3]=datosUsuarios[i][3];
            String numeroRol =datosUsuarios[i][4];

            if (numeroRol.equals("1")) {
                datosReporteFormateados[i][4]="Administrador";
            }else if (numeroRol.equals("2")) {
                datosReporteFormateados[i][4]="Estudiante";
            }else if (numeroRol.equals("3")) {
                datosReporteFormateados[i][4]="Catedratico";
            }

            //Contar bibliografias
            String bibliografiasPrestadas = "0";
            String[][] datosPrestamoPorUsuario =
Prestamo.listarPrestamoNoDevueltos(datosUsuarios[i][0]);

            if (datosPrestamoPorUsuario != null) {
                if (datosPrestamoPorUsuario[0][0] != null) {
                    bibliografiasPrestadas = datosPrestamoPorUsuario.length +"";
                }
            }
            datosReporteFormateados[i][5]=bibliografiasPrestadas;
        }

        String tabla = crearTabla(encabezados,datosReporteFormateados);

        reporte =reporte+tabla+pie;

        return reporte;
    }




    public String reportePrestamos(){
        String reporte="";

        crearEncabezado("Reporte de prestamos");
        reporte = reporte + encabezado;

        String tabla ="";
```

Crea un encabezado con el titulo para el reporte de prestamos

Encabezados de tabla

```
        String [] encabezados = {"ID del prestamo",
"ID del usuario", "Titulo", "Tipo", "Fecha", "Devuelto"};

        String[][] datosPrestamo = Prestamo.datosPrestamo();
        String[][] datosFormateados = new String[datosPrestamo.length][6];

        //Del mas reciente al mas antiguo
        int contadorAux = 0;
        for (int i = datosFormateados.length-1; i >= 0; i--) {
            datosFormateados[contadorAux][0]=datosPrestamo[i][0];
            datosFormateados[contadorAux][1]=datosPrestamo[i][4];
            datosFormateados[contadorAux][2]=datosPrestamo[i][1];

            if (Bibliografia.buscarBibliografia(datosPrestamo[i][1]) == null) {
                return "";
            }

            String numeroTipo =
Bibliografia.buscarBibliografia(datosPrestamo[i][1])[0];

            if (numeroTipo.equals("0")) {
                datosFormateados[i][3]="Libro";
            }else if (numeroTipo.equals("1")) {
                datosFormateados[i][3]="Revista";
            }else{
                datosFormateados[i][3]="Tesis";
            }

            datosFormateados[contadorAux][4]=datosPrestamo[i][2];

            String devuelto = "No";
            if (datosPrestamo[i][3].equals("1")) {
                devuelto="Si";
            }
            datosFormateados[contadorAux][5]=devuelto;

            contadorAux++;
        }

        tabla= crearTabla(encabezados,datosFormateados);

        reporte =reporte+tabla+pie;
        return reporte;
    }









    public String reporteBibliografias() {

        String reporte="";
```

Reordena la información para que aparezca del mas reciente al mas antiguo

Recopila la información de cada bibliografía

Crea la tabla con la información recopilada y el fin del documento HTML

Crea un encabezado con el titulo para el reporte de bibliografías

Establece el encabezado y recopila los datos de todas las bibliografías

```
        crearEncabezado("Reporte de bibliografia");
        reporte = reporte + encabezado;

        String [] encabezados = {"Tema","Bibliografias asociadas"};
        String[][] datosBibliografia = Bibliografia.datosBibiliografia();

        //Temas disponibles/registrados

        if (datosBibliografia[0][0] == null) {
            return "";
        }

        String temasSeparadosPorComas = "";
        for (int i = 0; i < datosBibliografia.length; i++) {
            String[] temasBibliografiaIndividual = datosBibliografia[i][5].split(",");

            for (int j = 0; j < temasBibliografiaIndividual.length; j++) {

                String temaIndividual = temasBibliografiaIndividual[j];

                if (i == 0 && j ==0) {
                    temasSeparadosPorComas+= temaIndividual + " ;";
                }else{
                    int verificaciones = 1;
                    String [] temasVerificados = temasSeparadosPorComas.split(";");
                    boolean noAparecio =true;
                    for (int k = 0; k < temasVerificados.length; k++) {

                        if (temasVerificados[k].trim().equals(temaIndividual.trim())) {
                            noAparecio = false;
                        }
                        //Si ya reviso todo y no aparecio
                        if (verificaciones == temasVerificados.length && noAparecio) {
                            temasSeparadosPorComas+= " " + temaIndividual + " ;";
                        }

                        verificaciones++;
                    }
                }
            }
        }

        //quita el ultimo caracter que es un ;
        temasSeparadosPorComas = temasSeparadosPorComas.substring(0,
temasSeparadosPorComas.length()-1);

        String [] temas = temasSeparadosPorComas.split(";");
        String[][] datosReporteFormateados = new String[temas.length][2];









        for (int i = 0; i < datosReporteFormateados.length; i++) {
```

Establece el tema en la primera columna

```
            datosReporteFormateados[i][0] =temas[i].trim();

            int bibliografiasRegistradas = 0;
            //Contar temas
            for (int j = 0; j < datosBibliografia.length; j++) {
                String[] temasBibliografiaIndividual =
datosBibliografia[j][5].split(",");
                for (int k = 0; k < temasBibliografiaIndividual.length; k++) {
                    if (temasBibliografiaIndividual[k].trim().equals(temas[i].trim())) {
                        bibliografiasRegistradas++;

                    }
                }
            }


            datosReporteFormateados[i][1] =bibliografiasRegistradas+"";

        }

        String tabla = crearTabla(encabezados,datosReporteFormateados);

        reporte =reporte+tabla+pie;
        return reporte;

    }

        private String crearTabla(String [] encabezados, String[][] datosCuerpo){
        String tabla ="\n<table>";


        //Encabezados
        String encabezadoTabla = "\n\t<thead>\n\t\t<tr>";
        for (String encabezado:encabezados) {
            encabezadoTabla = encabezadoTabla + "\n\t\t\t<th>"+encabezado+"</th>";
        }
        encabezadoTabla=encabezadoTabla+"\n\t\t</tr>\n\t</thead>";

        //cuerpo de tabla
        String cuerpoTabla = "\n\t<tbody>";
        for (int i = 0; i < datosCuerpo.length; i++) {
            String fila = "\n\t\t<tr>";
            for (int j = 0; j < datosCuerpo[i].length; j++) {
                fila=fila+"\n\t\t\t<td>"+datosCuerpo[i][j]+"</td>";
            }
            fila=fila+"\n\t\t</tr>";
            cuerpoTabla=cuerpoTabla+fila;
        }
        cuerpoTabla=cuerpoTabla+"\n\t</tbody>";

        tabla=tabla+encabezadoTabla+cuerpoTabla+"\n</table>";

        return tabla;
    }
```

```
    private void crearEncabezado (String titulo){
        encabezado =    "<!DOCTYPE html>\n" +
                        "<html lang=\"es\">\n" +
                        "<head>\n" +
                        "\t<meta charset=\"UTF-8\">\n" +
                        "\t<meta http-equiv=\"X-UA-Compatible\" content=\"IE=edge\">\n" +
                        "\t<meta name=\"viewport\" content=\"width=device-width, initial-
scale=1.0\">\n" +
                        "\t<title>"+titulo+"</title>\n" +
                        "\t<!-- Fuentes de google -->\n" +
                        "\t<link rel=\"preconnect\"
href=\"https://fonts.googleapis.com\">\n" +
                        "\t<link rel=\"preconnect\" href=\"https://fonts.gstatic.com\"
crossorigin>\n" +
                        "\t<link
href=\"https://fonts.googleapis.com/css2?family=Fredoka:wght@300&display=swap\"
rel=\"stylesheet\">\n" +
                        "\t<style>\n" +
                        "\t\tbody {\n" +
                        "\t\t\tdisplay: flex;\n" +
                        "\t\t\tflex-direction: column;\n" +
                        "\t\t\tbackground-color: rgb(32, 39, 58);\n" +
                        "\t\t\tmin-height: 100vh;\n" +
                        "\t\t\tfont-family: 'Fredoka', sans-serif;\n" +
                        "\t\t}\n" +
                        "\n" +
                        "\t\th1 {\n" +
                        "\t\t\tcolor: rgba(236, 238, 71);\n" +
                        "\t\t\ttext-align: center;\n" +
                        "\t\t}\n" +
                        "\n" +
                        "\t\ttable {\n" +
                        "\t\t\tbackground-color: rgb(32, 39, 58);\n" +
                        "\t\t\tborder-collapse: collapse;\n" +
                        "\t\t\tcolor: rgb(165, 165, 165);\n" +
                        "\t\t}\n" +
                        "\n" +
                        "\t\ttd {\n" +
                        "\t\t\tbackground-color: rgb(50, 61, 79);\n" +
                        "\t\t\tborder: 2px solid rgb(32, 39, 58);\n" +
                        "\t\t\tmargin: 0;\n" +
                        "\t\t\tpadding: 1rem;\n" +
                        "\t\t\ttext-align: center;\n" +
                        "\t\t}\n" +
                        "\n" +
                        "\t\tth {\n" +
                        "\t\t\tborder-bottom: 2px solid rgb(30, 46, 112);\n" +
                        "\t\t\tcolor: rgb(63, 156, 193);\n" +
                        "\t\t\tpadding-bottom: 0.5rem;\n" +
                        "\t\t}\n" +
                        "\n" +
                        "\n" +
                        "\t\ttr:nth-child(odd) td {\n" +
                        "\t\t\tbackground-color: rgb(45, 52, 71);\n" +
                        "\t\t}\n" +
                        "\t</style>"+
```

Inicio y estilos del documento HTML

```
                        "</head>\n" +
                        "<body>\n"+
                        "<h1>"+titulo+"</h1>";
    }

}
```

Paquete Vista

About

```
package Vista;

import javax.swing.*;
import java.awt.*;

public class About extends JFrame {

    public About(){
        this.setTitle("ABOUT");
        this.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);   No finaliza la ejecución del programa
        this.setResizable(false);

        Toolkit pantalla = Toolkit.getDefaultToolkit();
        Dimension tamanoPantalla = pantalla.getScreenSize();

        this.setSize(500,200);
        this.setLocation(tamanoPantalla.width/8, tamanoPantalla.height/8);

        ImageIcon logo = new
ImageIcon(getClass().getClassLoader().getResource("logo.png"));
        this.setIconImage(logo.getImage());

        this.getContentPane().setBackground(Color.white);

        JLabel datos = new JLabel("Creado por Damián Ignacio Peña  - 202110568");
        datos.setBounds(100,100,100,30);

        this.add(datos);
    }
}
```

Estilos Base

```
package Vista;

import javax.swing.*;
import java.awt.*;

public class EstilosBase extends JFrame {

    public EstilosBase(JFrame frame, String titulo){
        frame.setTitle(titulo);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setResizable(false);

        Toolkit pantalla = Toolkit.getDefaultToolkit();
        Dimension tamanoPantalla = pantalla.getScreenSize();

        frame.setSize(tamanoPantalla.width *3/4,tamanoPantalla.height*3/4);
        frame.setLocation(tamanoPantalla.width/8, tamanoPantalla.height/8);

        ImageIcon logo = new
ImageIcon(getClass().getClassLoader().getResource("logo.png"));
        frame.setIconImage(logo.getImage());

        Dimension tamanoVentana = this.getSize();
        int altoVentana = tamanoVentana.height;
        int anchoVentana = tamanoVentana.width;

        frame.getContentPane().setBackground(Color.white);

    }

}
```

Establece todos los atributos que tendrán los JFrames de la aplicación

Nota: Todos los formularios copiarán los atributos de “Estilos Base” por lo que únicamente se explicará la lógica interna de los formularios principales.

Inicio

Crea una instancia del “About”

```
btnAbout.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        About login = new About();
        login.setVisible(true);
    }
});

panel13.add(btnAbout);
```

Crea una instancia del inicio de sesión

```
JButton btnLogin = new JButton("Login");
btnLogin.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        Inicio.this.setVisible(false);
        Login login = new Login();
        login.setVisible(true);
    }
});
```

Login

Captura los datos de los JTextFields y los pasa como parámetro al método login de la clase Usuario para posteriormente mostrar en un JOptionpane el mensaje de respuesta.

```
@Override
public void actionPerformed(ActionEvent e) {

    if (e.getSource() == btnLogin) {
        String usuario = textFieldUsuario.getText();
```

```
        String contrasena = textFieldpassword.getText();


        // 0->dejar pasar, 1-> Mensaje, 2->idUsuario, 3->rol

        if (usuario.equals("") || contrasena.equals("")) {
            JOptionPane.showMessageDialog(this, "Debe de rellenar los campos", "Error",
JOptionPane.ERROR_MESSAGE);

        } else {
            String[] respuesta = Usuario.login(usuario, contrasena);
            if (respuesta[0].equals("0")) {
                JOptionPane.showMessageDialog(this, respuesta[1], "Error",
JOptionPane.WARNING_MESSAGE);
            } else {
                Login.this.setVisible(false);

                //0 -> administrador 1->regular maestro/alumno
                if (respuesta[3].equals("1")) {
                    PanelAdministrador panel = new PanelAdministrador(respuesta[2]);
                    panel.setVisible(true);
                    JOptionPane.showMessageDialog(panel, respuesta[1], "Bienvenido",
JOptionPane.INFORMATION_MESSAGE);

                }
                if (respuesta[3].equals("3") || respuesta[3].equals("2")) {
                    PanelUsuario panel = new PanelUsuario(respuesta[2]);
                    panel.setVisible(true);
                    JOptionPane.showMessageDialog(panel, respuesta[1], "Bienvenido",
JOptionPane.INFORMATION_MESSAGE);

                }

            }
        }

    }


}
```

Panel Administrador

Redirige a el formulario pertinente y muestra el nombre del usuario que es pasado por referencia cuando se inicia sesión. Ejemplo:

```
public PanelAdministrador(String id)
```

```
if (e.getSource() == btnLogout) {
    PanelAdministrador.this.setVisible(false);
    Login login = new Login();
    login.setVisible(true);
    JOptionPane.showMessageDialog(login, "Hasta pronto",
"LogOut",JOptionPane.INFORMATION_MESSAGE);
}
```

Crear usuario

Captura los datos de los JTextFields y los pasa como parámetro al constructor de la clase Usuario, luego de crear un objeto, utiliza el método crearUsuario y muestra en un JOptionpane un mensaje de respuesta.

```
String id = textFieldId.getText();
    String nombre = textFieldNombre.getText();
    String apellido = textFieldApellido.getText();
    String usuario = textFieldUsuario.getText();
    String rol = (String) comboRol.getSelectedItem();
    String contrasena = textFieldContrasena.getText();
    String confirmar = textFieldConfirmar.getText();

    if (id.equals("") || nombre.equals("") || apellido.equals("") || usuario.equals("")
|| rol.equals("") || contrasena.equals("") || confirmar.equals("")) {
        JOptionPane.showMessageDialog(this, "Debe rellenar todos los campos", "Error",
JOptionPane.WARNING_MESSAGE);
    }else{
        String tipoRol = "2";
        if (rol.equals("Catedratico")) {
            tipoRol="3";
        }

        if (contrasena.equals(confirmar)) {
            Usuario usuarioNuevo = new
Usuario(id,nombre,apellido,usuario,tipoRol,contrasena);

            if (usuarioNuevo.crearUsuario()) {
                JOptionPane.showMessageDialog(this, "Usuario creado correctamente",
"Exito", JOptionPane.INFORMATION_MESSAGE);
            }else{
                JOptionPane.showMessageDialog(this, "Hubo algun error, es posible que ya
exista un usuario con dicha ID", "Error", JOptionPane.ERROR_MESSAGE);
            }
        }else{
            JOptionPane.showMessageDialog(this, "Las contraseñas no coinciden", "Error",
JOptionPane.WARNING_MESSAGE);
        }
    }
}
```

Crear Bibliografía individual

Captura los datos de los JTextFields y los pasa como parámetro al constructor de la clase bibliografía, para luego utilizar el método crearBibliografiaIndividual.

```
Bibliografia bibliografia = new Bibliografia(tipo, autor, titulo,
descripcion,edicion,temasFormateados,frecuencia,ejemplares,area,copias,disponibles);

if(bibliografia.crearBibliografiaIndividual()){
    JOptionPane.showMessageDialog(this, "Bibliografia creada correctamente", "Aviso",
JOptionPane.INFORMATION_MESSAGE);
}else{
    JOptionPane.showMessageDialog(this, "Hubo algun problema", "Error",
JOptionPane.WARNING_MESSAGE);
}
```

Crear Bibliografía Masiva

Captura el texto de un JTextArea y llama al método crearCargaMasiva

```
Bibliografia.crearBibliografiaMasiva(areaTexto.getText())
```

Modificar Bibliografía

Carga en los JTexfields los datos de la búsqueda

```
String[] datosBusqueda = Bibliografia.buscarBibliografia(textFieldBuscar.getText());
```

Llama al método actualizarBibliografia pasando como parámetro los datos capturados de los JTexfields

```
Bibliografia.actualizarBibliografia(tipo, autor, titulo, edicion,
descripcion,temasFormateados,frecuencia,ejemplares,area,copias,disponibles);
```

Modificar Usuario

Carga los datos de la búsqueda en los JTextfields

```
String[] datosBusqueda =Usuario.buscarUsuario(id);
```

Pasa como parámetro los datos capturados de los JTextfields

```
Usuario.actualizarUsuario(id,nombre,apellido,usuario,tipoRol,contrasena)
```

Eliminar Bibliografía

Carga en los JTexfields los datos de la búsqueda

```
String[] datosBusqueda = Bibliografia.buscarBibliografia(textFieldBuscar.getText());
```

Llama al método eliminarBibliografia pasando como parámetro el titulo que se captura en JTextfield.

```
Bibliografia.eliminarBibliografia(titulo);
```

Eliminar Usuario

Carga los datos de la búsqueda en los JTextfields

```
String[] datosBusqueda =Usuario.buscarUsuario(id);
```

Llama al método eliminarUsuario pasando como parámetro el ID que se captura en JTextfield.

```
Usuario.eliminarUsuario(id);
```

Listar Bibliografía

Crea un modelo de tabla, añadiendo los encabezados y creando un objeto tipo Object que se añade a modo de fila a partir del método datosBibliografia.

```
DefaultTableModel modeloTabla = new DefaultTableModel();

modeloTabla.addColumn("Tipo");
modeloTabla.addColumn("Autor");
modeloTabla.addColumn("Titulo");
modeloTabla.addColumn("Descripcion");
modeloTabla.addColumn("Edicion");
modeloTabla.addColumn("Temas");
modeloTabla.addColumn("Frecuencia Actual");
modeloTabla.addColumn("Ejemplares");
modeloTabla.addColumn("Area");
modeloTabla.addColumn("Copias");
modeloTabla.addColumn("Disponibles");

if (Bibliografia.datosBibiliografia()[0][0]!=null) {
    for (String[] dato : Bibliografia.datosBibiliografia()) {
        String tipo = "";
        if (dato[0].equals("0")) {
            tipo = "Libro";
        } else if (dato[0].equals("1")) {
            tipo = "Revista";
        } else if (dato[0].equals("2")) {
            tipo = "Tesis";
        }
        modeloTabla.addRow(new Object[]{tipo, dato[1], dato[2], dato[3], dato[4],
dato[5], dato[6], dato[7], dato[8], dato[9], dato[10]});
    }
}

tablaUsuarios.setModel(modeloTabla);
```

Listar Usuarios

Crea un modelo de tabla, añadiendo los encabezados y creando un objeto tipo Object que se añade a modo de fila a partir del método datosUsuario.

```
DefaultTableModel modeloTabla = new DefaultTableModel();

modeloTabla.addColumn("Id");
modeloTabla.addColumn("Nombre");
modeloTabla.addColumn("Apellido");
modeloTabla.addColumn("Usuario");
modeloTabla.addColumn("Rol");
modeloTabla.addColumn("Password");

for (String[] dato: Usuario.datosUsuario()) {
    String tipoRol = "Administrador";
    if (dato[4].equals("2")) {
        tipoRol = "Estudiante";
    }else if (dato[4].equals("3")) {
        tipoRol = "Catedratico";
    }
    modeloTabla.addRow(new Object[]{dato[0],dato[1], dato[2], dato[3],tipoRol,dato[5]});
}

tablaUsuarios.setModel(modeloTabla);
```

Generar reporte

Panel usuario

1. Referencia a la respectiva ventana.
2. Captura eventos en el JTable

Carga de datos en JTable:

```
//reinicia el modelo
tablaBibliografias.setModel(new DefaultTableModel());
DefaultTableModel modeloTabla = new DefaultTableModel();

modeloTabla.setRowCount(0);
modeloTabla.addColumn("Tipo");
modeloTabla.addColumn("Autor");
modeloTabla.addColumn("Titulo");
modeloTabla.addColumn("Descripcion");
modeloTabla.addColumn("Edicion");
modeloTabla.addColumn("Temas");
modeloTabla.addColumn("Frecuencia Actual");
modeloTabla.addColumn("Ejemplares");
modeloTabla.addColumn("Area");
modeloTabla.addColumn("Copias");
modeloTabla.addColumn("Disponibles");

if (!textFieldBuscar.getText().equals("")) {

generarModelo(modeloTabla,Bibliografia.buscarCoincidenciasBibliografia(textFieldBuscar.ge
tText()));
}else{
    generarModelo(modeloTabla,Bibliografia.datosBibiliografia());
}

tablaBibliografias.setModel(modeloTabla);
```

Creación del modelo de tabla en función de si es una búsqueda o si debe listar en su totalidad los registros:

```
if (datosCargar != null) {
    if (datosCargar[0][0] != null) {

        for (String[] dato : datosCargar) {
            String tipo = "";
            if (dato[0].equals("0")) {
                tipo = "Libro";
            } else if (dato[0].equals("1")) {
                tipo = "Revista";
```

```
            } else if (dato[0].equals("2")) {
                tipo = "Tesis";
            }
            modeloTabla.addRow(new Object[]{tipo, dato[1], dato[2], dato[3], dato[4],
dato[5], dato[6], dato[7], dato[8], dato[9], dato[10]});
        }
    }
}else {
    JOptionPane.showMessageDialog(this, "Información no encontrada", "Exito",
JOptionPane.INFORMATION_MESSAGE);
}
```

Captura del evento click en la tabla

```
if (e.getSource() == tablaBibliografias) {
    final int fila = tablaBibliografias.getSelectedRow();
    final String titulo = (String)tablaBibliografias.getValueAt(fila, 2);

    labelDatosSeleccionados.setText(titulo);
}
```

Al realizar click sobre la tabla, encuentra el número de fila, así como el título, para que se cargue en JLabel.

Si el usuario decide prestar la bibliografía y confirma el JOptionPane que se encuentra a continuación, se decide por crear un objeto tipo préstamo de la siguiente manera:

```
Prestamo prestamo = new Prestamo(labelDatosSeleccionados.getText(),datosUsuario[0]);
JOptionPane.showMessageDialog(this, prestamo.crearPrestamo(), "Aviso",
JOptionPane.INFORMATION_MESSAGE);
```

Listar Prestamos

Carga en un JTable todos los prestamos del usuario:

```
DefaultTableModel modeloTabla = new DefaultTableModel();

modeloTabla.addColumn("Id del prestamo");
modeloTabla.addColumn("Titulo prestado");
modeloTabla.addColumn("Fecha");
modeloTabla.addColumn("Devuelto");

String [][] datosPrestamo = Prestamo.listarPrestamos(id);


if (datosPrestamo != null) {
    if (datosPrestamo[0] != null) {
        for (String[] dato: datosPrestamo) {
            String devuelto = "No";
            if (dato[3].equals("1")) {
                devuelto ="Si";
            }
            modeloTabla.addRow(new Object[]{dato[0],dato[1], dato[2], devuelto});
        }
    }
}
```

```
tablaPrestamos.setModel(modeloTabla);
```

De manera similar al Panel de Usuario, si se decide por hacer click en una fila, se cargará en un JLabel el ID del préstamo, para de esta manera utilizar el método devolverPrestamo de la siguiente manera:

```
String mensaje = Prestamo.devolverPrestamo(labelIdPrestamo.getText());
```