



**INSTITUTO POLITECNICO NACIONAL  
ESCUELA SUPERIOR DE CÓMPUTO**

**“Practica 9-JDBC Gameplanet”**

**-Tinoco Videgaray Sergio Ernesto**

Grupo: 3BV1

Materia: Paradigmas de programación

INSTITUTO POLITÉCNICO NACIONAL



ESCOM®

05/12/21

En esta práctica se va a desarrollar una GUI que va a permitir al usuario realizar consultas a una base de datos local, así como realizar altas, bajas y modificaciones por medio de la interfaz.

Primeramente se va a implementar la clase Videojuego para la gestión de los datos capturados, la cual va a funcionar como intermediario entre la base de datos y la interfaz de usuario.

```
public class Videojuego {  
  
    String upc,  
        descripcion,  
        desarrollador,  
        plataforma,  
        clasificacion,  
        generos,  
        precio,  
        imagen;  
}
```

Se declaran las variables dentro de la clase Videojuego las cuales corresponden con los atributos del videojuego.

Constructor de la clase Videojuego

```
public Videojuego(String upc, String desc, String dev, String plat,  
                  String clasif, String gen, String precio) {  
    this.upc=upc;  
    this.descripcion=desc;  
    this.desarrollador=dev;  
    this.plataforma=plat;  
    this.clasificacion=clasif;  
    this.generos=gen;  
    this.precio=precio;  
}
```

```
public String getUpc() {
    return upc;
}

public void setUpc(String upc) {
    this.upc = upc;
}

public String getDescripcion() {
    return descripcion;
}

public void setDescripcion(String descripcion) {
    this.descripcion = descripcion;
}

public String getDesarrollador() {
    return desarrollador;
}

public void setDesarrollador(String desarrollador) {
    this.desarrollador = desarrollador;
}

public String getPlataforma() {
    return plataforma;
}

public void setPlataforma(String plataforma) {
    this.plataforma = plataforma;
}

public String getClassificacion() {
    return clasificacion;
}

public void setClassificacion(String clasificacion) {
    this.clasificacion = clasificacion;
}

public String getGeneros() {
    return generos;
}

public void setGeneros(String generos) {
    this.generos = generos;
}
```

Métodos sets y gets para establecer y recuperar los valores de los atributos del videojuego actual.

```

public static Videojuego getVideojuegoFromBD(String UPC, Properties prop) {
    Videojuego vid = new Videojuego(); // Nuevo libro en blanco

    try
    {
        String driver = prop.getProperty("dbdriver");
        String host    = prop.getProperty("dbhost");
        String user    = prop.getProperty("dbuser");
        String password = prop.getProperty("dbpassword");
        String name     = prop.getProperty("dbname");
        String url = host + name + "?user=" + user + "&password=" + password;
        System.out.println("Conexion a la BD: " + url);
    }
}

```

Se declara un método que va a realizar la consulta a la base de datos y va a almacenar los valores en un objeto de la clase videojuego.

Se obtienen los parámetros del archivo properties para realizar la conexión a la base de datos.

Se realiza la conexión a la base de datos por medio del driver.

Posteriormente se declara un objeto de tipo PreparedStatement que va a realizar la consulta por medio de la sentencia Select \* from videojuegos, mandándole como parámetro el UPC del juego.

```

Class.forName(driver); // Carga el driver

Connection con = DriverManager.getConnection(url); // Crea una conexion a la BD

PreparedStatement ps = con.prepareStatement("SELECT * FROM VIDEOJUEGOS WHERE UPC = "+UPC);
System.out.println(ps.toString());
ps.executeQuery();
ResultSet rs = ps.getResultSet();

```

```

if(rs!=null && rs.next())
{
    String upc    = rs.getString("upc");
    String descripcion = rs.getString("descripcion");
    String desarrollador = rs.getString("desarrollador");
    String plataforma = rs.getString("plataforma");
    String clasificacion = rs.getString("clasificacion");
    String generos = rs.getString("generos");
    String precio = rs.getString("precio");
    String imagen = rs.getString("imagen");

    vid.setUpc(upc);
    vid.setDescripcion(descripcion);
    vid.setDesarrollador(desarrollador);
    vid.setPlataforma(plataforma);
    vid.setClasificacion(clasificacion);
    vid.setGeneros(generos);
    vid.setPrecio(precio);
    vid.setImagen(imagen);
    con.close();
    return vid;
}

```

Se evalúa si la consulta existe y en dado caso se almacenan los valores de cada columna en una variable.

Posteriormente se almacenan los valores de las variables que contienen los atributos del juego en el objeto de la clase videojuego declarado previamente.

En caso de que la consulta no se ejecute correctamente se manda un mensaje de error en la consola.

```
catch (Exception ex)
{
    ex.printStackTrace();
}
return null;
```

De manera análoga se declara el método cambiar, que va a realizar una actualización de los datos almacenados. Por lo que vuelven a declarar las variables para obtener los parámetros del archivo properties y se cargan al controlador para realizar la conexión.

```
public boolean cambiar(Properties prop)
{
    boolean exito = false;

    try
    {
        String driver = prop.getProperty("dbdriver");
        String host    = prop.getProperty("dbhost");
        String user    = prop.getProperty("dbuser");
        String password = prop.getProperty("dbpassword");
        String name     = prop.getProperty("dbname");
        String url = host + name + "?user=" + user + "&password=" + password;
        System.out.println("Conexion a la BD: " + url);

        Class.forName(driver);        // Carga el driver
    }
}
```

Se realiza la conexión a la base de datos por medio del driver y se crea un objeto con la sentencia requerida para actualizar los datos de la BD, en este caso se utiliza la sentencia Update videojuegos para modificar los valores de cada columna. Mandándole como parámetro las variables del objeto juego y el UPC del juego para identificar el registro correspondiente.

Posteriormente se cierra la conexión a la BD.

```
PreparedStatement ps = con.prepareStatement("UPDATE VIDEOJUEGOS SET "
    + "DESCRIPCION = ?,DESARROLLADOR = ?,PLATAFORMA = ?,CLASIFICACION = ?,"
    + "GENEROS = ?,PRECIO = ?,IMAGEN = ? WHERE UPC = "+this.upc);

ps.setString(1, this.descripcion);
ps.setString(2, this.desarrollador);
ps.setString(3, this.plataforma);
ps.setString(4, this.clasificacion);
ps.setString(5, this.generos);
ps.setString(6, this.precio);
ps.setString(7, this.imagen);

System.out.println(ps.toString());
exito = ps.executeUpdate() > 0;
con.close();
```

En caso de que la consulta no se ejecute correctamente se manda un mensaje de error en la consola.

```
catch (Exception ex)
{
    ex.printStackTrace();
}
return null;
```

De igual forma se declara el método borrar, que va a eliminar un registro de la base de datos. Por lo que vuelven a declarar las variables para obtener los parámetros del archivo properties y se cargan al controlador para realizar la conexión.

```
public boolean borrar(Properties prop)
{
    boolean exito = false;

    try
    {
        String driver = prop.getProperty("dbdriver");
        String host    = prop.getProperty("dbhost");
        String user    = prop.getProperty("dbuser");
        String password = prop.getProperty("dbpassword");
        String name     = prop.getProperty("dbname");
        String url = host + name + "?user=" + user + "&password=" + password;
        System.out.println("Conexion a la BD: " + url);

        Class.forName(driver); // Carga el driver
    }
}
```

En este caso se va a utilizar únicamente el UPC para ubicar el registro del juego que se quiere eliminar.

```
Connection con = DriverManager.getConnection(url); // Crea una conexion a la BD

PreparedStatement ps = con.prepareStatement("DELETE FROM VIDEOJUEGOS WHERE UPC = "+this.upc);
System.out.println(ps.toString());
exito = ps.executeUpdate() > 0;
con.close();
```

En caso de que la conexión no se realice correctamente se muestra un mensaje de excepción en consola.

```
catch (Exception ex)
{
    ex.printStackTrace();
}
return exito;
```

Por último se declara el método alta, para crear un nuevo registro en la base de datos.

De igual manera se van a cargar los valores del archivo properties en un controlador que va a realizar la conexión con la base de datos.

```
public boolean alta(Properties prop)
{
    boolean exito = false;

    try
    {
        String driver = prop.getProperty("dbdriver");
        String host    = prop.getProperty("dbhost");
        String user    = prop.getProperty("dbuser");
        String password = prop.getProperty("dbpassword");
        String name     = prop.getProperty("dbname");
        String url = host + name + "?user=" + user + "&password=" + password;
        System.out.println("Conexion a la BD: " + url);

        Class.forName(driver); // Carga el driver
    }
}
```

Para este caso se va a utilizar la sentencia "Insert into Videojuegos values".

Por lo que se van a mandar como parámetros los valores del objeto videojuego en el orden que se muestra a continuación:

```
Connection con = DriverManager.getConnection(url); // Crea una conexion a la BD

PreparedStatement ps = con.prepareStatement("INSERT INTO VIDEOJUEGOS VALUES (?, ?, ?, ?, ?, ?, ?, ?)");
ps.setString(1, this.upc);
ps.setString(2, this.descripcion);
ps.setString(3, this.desarrollador);
ps.setString(4, this.plataforma);
ps.setString(5, this.clasificacion);
ps.setString(6, this.generos);
ps.setString(7, this.precio);
ps.setString(8, this.imagen);

System.out.println(ps.toString());
exito = ps.executeUpdate() > 0;
con.close();
```

En caso de que la conexión no se realice correctamente se muestra un mensaje de excepción en consola.

```
catch (Exception ex)
{
    ex.printStackTrace();
}
return exito;
```



Clase Vista:

Se define la clase Vista que hereda de la clase JFrame.

Se declaran los componentes necesarios para generar la interfaz de usuario.

```
public class Vista extends JFrame{

    // DECLARACION DE CAMPOS Y BOTONES

    private final JTextField UPC = new JTextField();
    private final JTextField Descripcion = new JTextField();
    private final JTextField Desarrollador = new JTextField();
    private final JTextField Plataforma = new JTextField();
    private final JTextField Clasificacion = new JTextField();
    private final JTextField Generos = new JTextField();
    private final JTextField Precio = new JTextField();
    private final JTextField Imagen = new JTextField();

    private final JLabel Portada = new JLabel("Portada");

    private final JButton btnBuscar = new JButton("Buscar");
    private final JButton btnAgregar = new JButton("Agregar");
    private final JButton btnActualizar = new JButton("Actualizar");
    private final JButton btnEliminar = new JButton("Eliminar");
    private final JButton btnLimpiar = new JButton("Limpiar");

    // Variables auxiliares
    String mensajeError="";
```

Se declara el constructor de la clase vista que recibe como parámetro un objeto de tipo properties y va a generar el nombre de la ventana, las coordenadas y los tamaños del Frame.

Se cargan los valores del archivo properties en la variable prop.

En caso de que se genere una excepción se muestra un error en consola.

```
public Vista(Properties prop) // Constructor
{
    this.properties=prop;
    this.setTitle("Componentes SWING");
    this.setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
    this.setBounds(50,50,580,400);

    initComponents();

    this.setVisible(true);

    try
    {
        prop.load(new FileInputStream("config.properties"));
    }
    catch (Exception ex)
    {
        ex.printStackTrace();
    }
}
```

Método initComponents:

Se declaran objetos de tipo JLabel para las etiquetas de la interfaz.

```
public void initComponents()
{
    // Etiquetas

    JLabel et1 = new JLabel("UPC:");
    JLabel et2 = new JLabel("Descripcion:");
    JLabel et3 = new JLabel("Desarrollador:");
    JLabel et4 = new JLabel("Plataforma:");
    JLabel et5 = new JLabel("Clasificacion:");
    JLabel et6 = new JLabel("Generos:");
    JLabel et7 = new JLabel("Precio: ");
    JLabel et8 = new JLabel("Imagen: ");
}
```

Se generan los objetos para la barra de menú del frame.

```
// Diseña el menu

JMenuBar barraMenus = new JMenuBar();
JMenu menuArchivo    = new JMenu("Archivo");
JMenuItem opSalir    = new JMenuItem("Salir");
this.setJMenuBar(barraMenus);

barraMenus.add(menuArchivo);
menuArchivo.add(opSalir);
```

Se establecen los tamaños y las coordenadas de las etiquetas y las cajas de texto.

```
et1.setBounds(40, 30, 150, 20);
UPC.setBounds(200, 30, 150, 20);

et2.setBounds(40, 60, 150, 20);
Descripcion.setBounds(200, 60, 150, 20);

et3.setBounds(40, 90, 150, 20);
Desarrollador.setBounds(200, 90, 150, 20);

et4.setBounds(40, 120, 150, 20);
Plataforma.setBounds(200, 120, 150, 20);

et5.setBounds(40, 150, 150, 20);
Clasificacion.setBounds(200, 150, 150, 20);

et6.setBounds(40, 180, 150, 20);
Generos.setBounds(200, 180, 150, 20);

et7.setBounds(40, 210, 150, 20);
Precio.setBounds(200, 210, 150, 20);

et8.setBounds(40, 240, 150, 20);
Imagen.setBounds(200, 240, 150, 20);
```

Se establecen los tamaños y las coordenadas de los botones.

```
btnBuscar.setBounds(380, 30, 80, 30);  
btnAgregar.setBounds(20, 280, 80, 30);  
btnActualizar.setBounds(125, 280, 100, 30);  
btnEliminar.setBounds(250, 280, 80, 30);  
btnLimpiar.setBounds(350, 280, 80, 30);
```

```
this.add(et1);  
this.add(et2);  
this.add(et3);  
this.add(et4);  
this.add(et5);  
this.add(et6);  
this.add(et7);  
this.add(et8);  
  
this.add(UPC);  
this.add(Descripcion);  
this.add(Desarrollador);  
this.add(Plataforma);  
this.add(Clasificacion);  
this.add(Generos);  
this.add(Precio);  
this.add(Imagen);  
  
this.add(btnBuscar);  
this.add(btnAgregar);  
this.add(btnActualizar);  
this.add(btnEliminar);  
this.add(btnLimpiar);  
  
this.add(Portada);
```

Se añaden los componentes previamente establecidos al frame actual.

Se añaden los listeners correspondientes a los botones del frame.

```
// Eventos de botones  
btnBuscar.addActionListener(evt -> gestionaBuscar(evt));  
btnAgregar.addActionListener(evt -> gestionaAgregar(evt));  
btnActualizar.addActionListener(evt -> gestionaActualizar(evt));  
btnEliminar.addActionListener(evt -> gestionaEliminar(evt));  
btnLimpiar.addActionListener(evt -> gestionaLimpiar(evt));
```

Métodos para el botón salir, en caso de que el usuario presione el botón salir se va a mostrar en pantalla un mensaje de aviso preguntándole al usuario si desea salir del programa, en caso de ser así se termina la ejecución del programa y de la máquina virtual de java.

```
public void gestionaSalir(java.awt.event.ActionEvent evt)
{
    exit();
}

public void exit()
{
    int respuesta = JOptionPane.showConfirmDialog(rootPane, "Desea salir?", "Aviso", JOptionPane.YES_NO_OPTION);
    if(respuesta==JOptionPane.YES_OPTION) System.exit(0);
}
```

Método para buscar un registro dentro de la base de datos por medio del UPC.

En caso de que la caja de texto del UPC este vacía, se mostrara un mensaje de advertencia al usuario ya que no se puede realizar la consulta sin un UPC.

```
public void gestionaBuscar(java.awt.event.ActionEvent evt)
{
    if(UPC.getText().isBlank())
    {
        JOptionPane.showMessageDialog(this, "Para localizar un videojuego se requiere el UPC",
            "Aviso!", JOptionPane.ERROR_MESSAGE);
    }
}
```

En caso contrario, se realiza la consulta por medio de un objeto tipo Videojuego.

Una vez se carguen los datos del juego en el objeto videojuego se van a mostrar en las cajas de texto correspondientes así como la imagen del juego.

```
else
{
    Videojuego videojuego = Videojuego.getVideojuegoFromBD(UPC.getText(), properties);
    if(videojuego != null) // Si hubo éxito
    {
        String nomImg=videojuego.getImagen();
        Descripcion.setText(videojuego.getDescripcion());
        Desarrollador.setText(videojuego.getDesarrollador());
        Plataforma.setText(videojuego.getPlataforma());
        Clasificacion.setText(videojuego.getClasificacion());
        Generos.setText(videojuego.getGeneros());
        Precio.setText(videojuego.getPrecio());
        Imagen.setText(nomImg.substring(9));

        String nombreArchivoImagen = videojuego.getImagen();
        MostrarPortada(nombreArchivoImagen);
    }
}
```

Método para redimensionar la imagen de la portada del videojuego, para posteriormente mostrarla en el Frame.

```
public void MostrarPortada(String archivoImg){

    ImageIcon imagenCargada = new ImageIcon(archivoImg);
    double w = imagenCargada.getIconWidth();
    double h = imagenCargada.getIconHeight();

    // Redimensionar la imagen
    if(h > 160 || w > 160)
    {
        double r;
        if( h > w)
            r = 160.0/h;
        else
            r = 160.0/w;
        w = w*r;
        h = h*r;
        Image imagenOriginal = imagenCargada.getImage();
        Image imagenRedimensionada = imagenOriginal.getScaledInstance((int)w, (int)h, java.awt.Image.SCALE_SMOOTH);
        imagenCargada = new ImageIcon(imagenRedimensionada);
    }

    Portada.setBounds(380,85,imagenCargada.getIconWidth(), imagenCargada.getIconHeight());
    Portada.setIcon(imagenCargada);

}
```

Método para actualizar los datos del juego previamente consultado por medio del UPC.

En caso de que la caja de texto del UPC este vacía, se mostrara un mensaje de advertencia al usuario ya que no se puede realizar la actualización sin un UPC.

```
public void gestionaActualizar(java.awt.event.ActionEvent evt)
{
    if(UPC.getText().isBlank())
    {
        JOptionPane.showMessageDialog(this, "Para localizar el juego que se va \na actualizar se requiere el UPC",
            "Aviso!", JOptionPane.ERROR_MESSAGE);
    }
}
```

En caso contrario, se evalúa si las casillas tienen valores capturados y de ser el caso se realiza la consulta.

En caso de que se presente un error de consulta se mostrara un mensaje de error en pantalla.

```
if(!invalido()) // Se intenta realizar el UPDATE solo si no hay error de captura
{
    Videojuego vid = Videojuego.getVideojuegoFromBD(UPC.getText(),properties); // Método estático para obtener un libro desde la BD
    if(vid != null)
    {
        vid.setDescripcion(Descripcion.getText());
        vid.setDesarrollador(Desarrollador.getText());
        vid.setPlataforma(Plataforma.getText());
        vid.setClasificacion(Clasificacion.getText());
        vid.setGeneros(Generos.getText());
        vid.setPrecio(Precio.getText());
        vid.setImagen(Imagen.getText());

        if(vid.cambiar(properties)) // Si hubo éxito
            JOptionPane.showMessageDialog(this, "Registro actualizado: " + UPC.getText(), "Aviso!",JOptionPane.INFORMATION_MESSAGE);
        else
            JOptionPane.showMessageDialog(this, "Acción no realizada!!", "Aviso!",JOptionPane.ERROR_MESSAGE);
    }
    else JOptionPane.showMessageDialog(this, "El juego con el UPC indicado no fue localizado", "Aviso!", JOptionPane.ERROR_MESSAGE);
}
else JOptionPane.showMessageDialog(this, mensajeError, "Aviso!", JOptionPane.ERROR_MESSAGE);
```

Se define un método para realizar una alta en la base de datos.

Se evalúa si los campos de los datos están llenos y de ser así se instancia un objeto de tipo videojuego, y se comprueba que el UPC del videojuego no esté ya registrado en la BD.

Posteriormente se almacenan los datos en el objeto de la clase videojuego.

```
public void gestionaAgregar(java.awt.event.ActionEvent evt)
{
    if(!invalido()) // Se intenta realizar el INSERT solo si no hay error de captura
    {
        // Primero investigamos si no hay otro registro con el mismo ISBN
        Videojuego vid = Videojuego.getVideojuegoFromBD(UPC.getText(), properties);

        if(vid == null) // Solo si el ISBN no está registrado
        {
            // Adquirimos los datos de la vista
            vid = new Videojuego();
            vid.setUpc(UPC.getText());
            vid.setDescripcion(Descripcion.getText());
            vid.setDesarrollador(Desarrollador.getText());
            vid.setPlataforma(Plataforma.getText());
            vid.setClasificacion(Clasificacion.getText());
            vid.setGeneros(Generos.getText());
            vid.setPrecio(Precio.getText());
            vid.setImagen(Imagen.getText());
        }
    }
}
```

Se invoca al método “alta” de la clase videojuego y se realiza la consulta correspondiente, en caso de que se presente algún error de consulta se mostrará un mensaje de error.

```
if(vid.alta(properties)) // Si la alta fue exitosa
    JOptionPane.showMessageDialog(this, "Registro agregado: " + UPC.getText(), "Aviso!", JOptionPane.INFORMATION_MESSAGE);
else
    JOptionPane.showMessageDialog(this, "Acción no realizada!!", "Aviso!", JOptionPane.ERROR_MESSAGE);
}
else JOptionPane.showMessageDialog(this, "El UPC ya está registrado", "Aviso!", JOptionPane.ERROR_MESSAGE);
}
else JOptionPane.showMessageDialog(this, mensajeError, "Aviso!", JOptionPane.ERROR_MESSAGE);
```

Método para borrar un registro de la BD.

Se comprueba si el campo del UPC tiene texto y de no ser así se mostrará un mensaje de error al usuario.

```
public void gestionaEliminar(java.awt.event.ActionEvent evt)
{
    if(UPC.getText().isBlank())
    {
        JOptionPane.showMessageDialog(this, "Para localizar el videojuego que se va a eliminar se requiere el UPC",
            "Aviso!", JOptionPane.ERROR_MESSAGE);
    }
}
```

En caso contrario se mostrará un mensaje de advertencia al usuario para preguntarle si desea eliminar el registro indicado.

En caso de que el usuario elija la opción “YES” se va a instanciar un objeto de la clase videojuego.

Se evalúa si el registro existe y en dado caso se manda a llamar al método borrar de la clase videojuego, si la sentencia se ejecutó correctamente se muestra un mensaje de Registro eliminado y se invoca al método limpiarCampos.

En caso de que se produzca algún error al realizar la baja se mostrara un mensaje de error al usuario.

```
// Solicitamos confirmación
int respuesta = JOptionPane.showConfirmDialog(this, "Desea borrar este registro?", "Atención!!!",
    JOptionPane.YES_NO_OPTION, JOptionPane.QUESTION_MESSAGE);

if(respuesta==JOptionPane.YES_OPTION) // Si el usuario confirma
{
    Videojuego vid = Videojuego.getVideojuegoFromBD(UPC.getText(),properties); // Trata de recuperar el libro de la BD

    if(vid != null) // Si lo encuentra
    {
        // Intenta eliminar el registro
        if(vid.borrar(properties)) // Si hubo éxito
        {
            JOptionPane.showMessageDialog(this, "Registro eliminado: " + UPC.getText(), "Aviso!",JOptionPane.WARNING_MESSAGE);
            Limpiar();
        }
        else JOptionPane.showMessageDialog(this, "Acción no realizada!!","Aviso!",JOptionPane.ERROR_MESSAGE);
    }
    else JOptionPane.showMessageDialog(this, "El juego con el UPC indicado no fue localizado", "Aviso!", JOptionPane.ERROR_MESSAGE);
}
```

Método para vaciar las cajas de texto y la imagen del juego.

```
public void Limpiar() {
    UPC.setText("");
    Descripcion.setText("");
    Desarrollador.setText("");
    Clasificacion.setText("");
    Generos.setText("");
    Plataforma.setText("");
    Imagen.setText("");
    Precio.setText("");

    Portada.setIcon(null);
}
```



Método para comprobar si todas las cajas de texto están llenas, en caso de no ser así mostrara un mensaje de error al usuario.

```
// Validación de datos
private boolean invalido()
{
    boolean hayError = false;
    mensajeError = "";

    if(UPC.getText().isBlank())
    {
        hayError = true;
        mensajeError = mensajeError.concat("No debe dejar el UPC en blanco\n");
    }

    if(Descripcion.getText().isBlank())
    {
        hayError = true;
        mensajeError = mensajeError.concat("No debe dejar la descripcion en blanco\n");
    }

    if(Desarrollador.getText().isBlank())
    {
        hayError = true;
        mensajeError = mensajeError.concat("No debe dejar el desarrollador en blanco\n");
    }

    if(Plataforma.getText().isBlank())
    {
        hayError = true;
        mensajeError = mensajeError.concat("No debe dejar la plataforma en blanco\n");
    }
}
```

Clase Main:

Se declara un objeto de tipo Properties para recuperar los parámetros del archivo config.properties el cual almacena los valores necesarios para realizar la conexión a la base de datos gameplanet.

En caso de que se presente alguna excepción se va a mostrar un error en la ventana del usuario y se imprime en consola el código del error.

De otro modo se cargan los valores recuperados a las variables correspondientes para el driver.

```

public static void main(String[] args)
{
    // Testing Database

    boolean todoBien = true;
    Properties prop = new Properties(); // Para guardar la conf de la base de datos

    // Carga las propiedades desde el archivo
    try
    {
        prop.load(new FileInputStream("config.properties"));
    }
    catch (Exception ex)
    {
        todoBien = false;
        JOptionPane.showMessageDialog(null, "Problema con el archivo de propiedades", "Aviso!", JOptionPane.ERROR_MESSAGE);
        ex.printStackTrace();
    }

    if(todoBien)
    {
        String driver = prop.getProperty("dbdriver");
        String host = prop.getProperty("dbhost");
        String user = prop.getProperty("dbuser");
        String password = prop.getProperty("dbpassword");
        String name = prop.getProperty("dbname");
        String url = host + name + "?user=" + user + "&password=" + password;
        System.out.println("Conexion a la BD: " + url);
    }
}

```

Finalmente se realiza la conexión a la BD y se crea la vista del usuario.

```

    try
    {
        Class.forName(driver); // Carga el driver
        Connection con = DriverManager.getConnection(url); // Crea una conexion a la BD
        con.close();
    }
    catch (ClassNotFoundException ex)
    {
        todoBien = false;
        JOptionPane.showMessageDialog(null, "Problema al cargar el driver", "Aviso!!!", JOptionPane.ERROR_MESSAGE);
    }
    catch (SQLException ex)
    {
        todoBien = false;
        JOptionPane.showMessageDialog(null, "Problema al tratar de hacer la conexion", "Aviso!!!", JOptionPane.ERROR_MESSAGE);
    }
}

if(todoBien) // Solo si todo está bien con la BD
{
    Vista v = new Vista(prop);
    v.setVisible(true);
}
}

```

## Pruebas de funcionamiento:

Componentes SWING

Archivo

UPC: 1

Descripcion: Fortnite

Desarrollador: Epic Games

Plataforma: PC

Clasificacion: T

Generos: BattleRoyale

Precio: 59

Imagen: fortnite.jpg



Componentes SWING

Archivo

UPC: 2

Descripcion: Halo

Desarrollador: Bungie

Plataforma: XBOX

Clasificacion: T

Generos: Disparos

Precio: 200

Imagen: halo.jpg



Componentes SWING

Archivo

UPC: 4

Descripcion: Genshin Impact

Desarrollador: miHoyo

Plataforma: PlayStation


Clasificacion: T

Generos: MMO, Aventura, RPG

Precio:

Imagen: genshin.jpg

**Aviso!**  
No debe dejar el precio en blanco



Componentes SWING

UPC: 4

Descripcion: Genshin Impact

Desarrollador: miHoyo

Plataforma: PlayStation

Clasificacion: T

Generos: MMO, Aventura, RPG

Precio: 260

Imagen: genshin.jpg

**Aviso!**  
Registro agregado: 4

UPC: 4

Descripcion: Genshin Impact

Desarrollador: miHoyo

Plataforma: PlayStation

Clasificacion: T

Generos: MMO, Aventura, RPG

Precio: 260

Imagen: genshin.jpg



UPC: 3

Descripcion: Banjo-Kazooie

Desarrollador: Rareware

Plataforma: Nintendo 64

Clasificacion: A+

Generos: Aventura, Puzzle

Precio: 45

Imagen: bk.jpg



UPC:

Descripcion:

Desarrollador:

Plataforma:

Clasificacion:


Generos:

Precio:

Imagen:

Aviso!

Registro actualizado: 3



UPC:

Descripcion:

Desarrollador:

Plataforma:

Clasificacion:

Generos:

Precio:

Imagen:



UPC:

Descripcion:

Desarrollador:

Plataforma:

Clasificacion:


Generos:

Precio:

Imagen:

Atención!!!

Desea borrar este registro?



UPC:

Descripcion:

Desarrollador:

Plataforma:

Clasificacion:

Generos:

Precio:

Imagen:

Aviso!

Registro eliminado: 4



UPC:

Descripcion:

Desarrollador:

Plataforma:

Clasificacion:

Generos:

Precio:

Imagen:

Aviso!

El videojuego con el UPC indicado no fue localizado

UPC:

Descripcion:

Desarrollador:

Plataforma:

Clasificacion:

Generos:

Precio:

Imagen:

Aviso

Desea salir?

## Observaciones y conclusiones:

En el desarrollo de esta práctica se presentaron algunos errores a la hora de realizar la conexión debido al archivo properties, el cual resulta ser de nuevo un elemento importante a la hora de realizar la conexión a la base de datos.

De igual forma resalta la parte del uso de clases para obtener y gestionar la información almacenada en una base de datos ya que como había mencionado en prácticas anteriores, una clase me permite almacenar información de algún elemento en tiempo de ejecución, y una base de datos me permite hacer lo mismo pero con la diferencia de que la información almacenada no se borra cuando cierro el programa, por lo que el uso de una clase me permite cargar mis datos en memoria para posteriormente ser mostrados en el programa y si es necesario realizar modificaciones o eliminaciones en la base de datos, por lo que es un recurso muy sustancial en este tipo de programas de usuario.