

# ACTIVIDAD 8 - PROYECTO FINAL



Grupo:

Lorenzo Cruz Fernández  
David González Valderrama

Curso: 1º Desarrollo de aplicaciones multiplataformas

Prof.ª: Soraya Galisteo Rego

Fecha: 23/05/2025

# **1. Introducción**

Se trata de una aplicación móvil en el cual sirve para llevar el stock y ventas de una tienda el cual cada trabajador tendrá su propio usuario de ventas donde podrá registrar clientes, ver juegos disponibles con todos sus datos y realizar ventas.

## **2. Motivación / Justificación**

El proyecto lo pensamos de forma que queríamos que fuese un caso real dentro de lo posible donde se pudiera registrar clientes listar los juegos y poder venderlos la motivación principal es nuestro afán de los videojuegos y queríamos hacer algo relacionado con esto.

## **3. Objetivos Propuestos**

- Entrar con un login
- Tener una parte de administración de la app
- Tener una parte de trabajadores
- En la parte de admin crear y borrar usuarios
- En la parte de trabajadores poder vender, listar y cambiar precios de videojuegos

## 4. Metodología Utilizada

Para el desarrollo de este trabajo se utilizó la metodología ágil Scrum, adaptada a clase. Scrum nos permitió organizar el trabajo en etapas cortas e iterativas, denominadas sprints, lo cual facilitó una planificación flexible, una ejecución ordenada y una mejora continua del proyecto.

Lo aplicamos a la idea de hacer cada uno partes cortas cada día y hacer reuniones diarias en clases las sprint que hicimos fue dividir el trabajo para trabajar solos principalmente pero sí que trabajamos juntos a la hora de problemas o juntar código.

### **Roles definidos:**

Realmente no definimos roles como tal ya que eramos solo 2 personas tampoco nos beneficiaba poner roles como tal ya que la comunicación la llevamos muy bien y nos entendemos .

### **Sprints:**

Dividimos el trabajo en sprints diarios. Cada día teníamos tareas específicas: análisis, diseño de base de datos, desarrollo de funcionalidades, pruebas, interfaz gráfica, documentación, etc.

### **Reuniones (simuladas):**

Realizamos una planificación al inicio de cada sprint y una revisión al final, debatiendo avances, dificultades y ajustes. Estas reuniones fueron clave para mejorar la gestión del tiempo.

## 6. Tecnologías y Herramientas Utilizadas

- Java 23.0.2
- MySQL 8.0.40 community
- IntelliJ IDEA
- MySQL Workbench
- JDBC

```
<dependency>  
  <groupId>mysql</groupId>  
  <artifactId>mysql-connector-java</artifactId>  
  <version>8.0.33</version>  
</dependency>
```

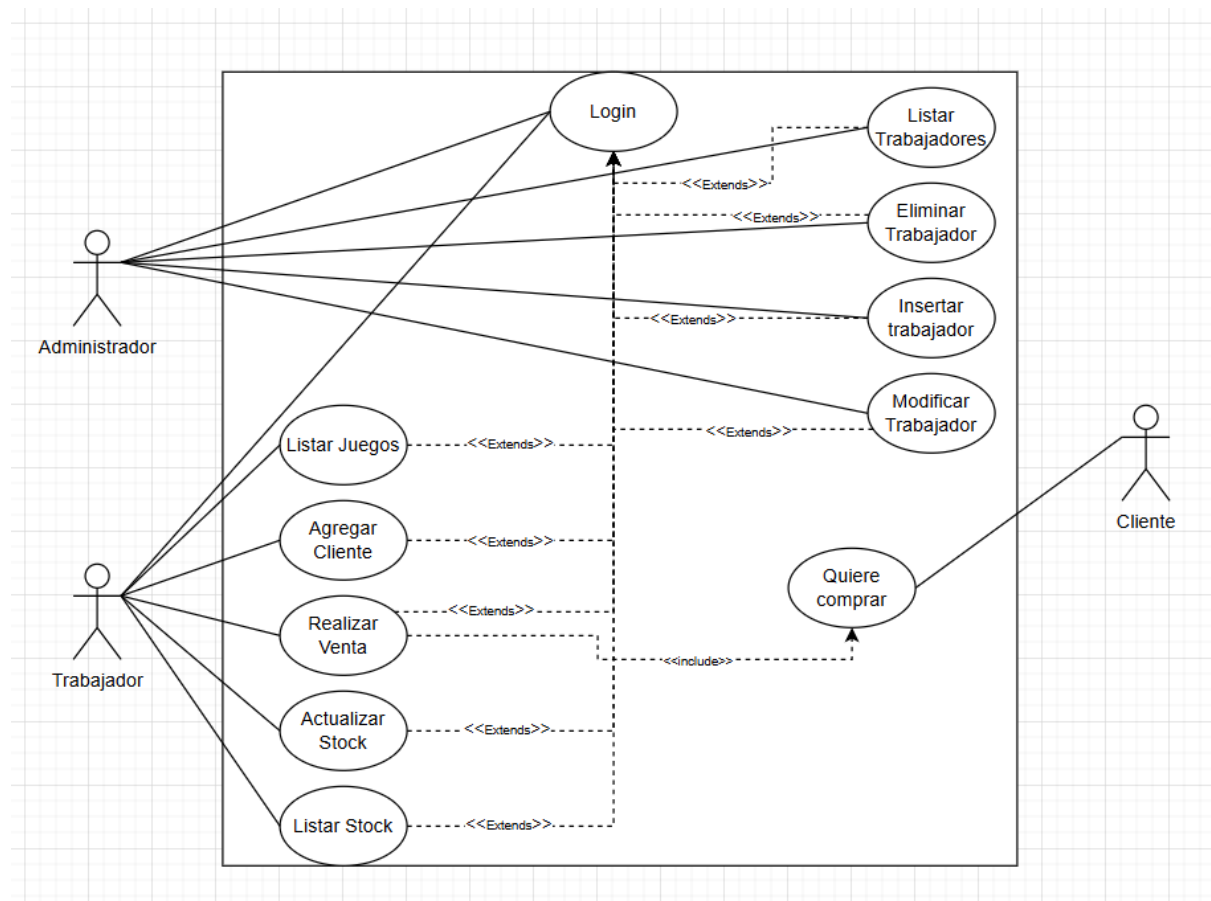
- GitHub: <https://github.com/Dagova/TrabajoFinalProgramacion.git>

## 7. Análisis

### a. Requisitos Funcionales

- Registrar empleados.
- Login de la aplicación
- Editar sus datos.
- Ver la lista completa.
- Eliminar empleados.
- Agregar cliente
- Realizar venta
- Actualizar precio
- Actualizar stock
- Listar juegos
- Insertar juegos
- Eliminar juego

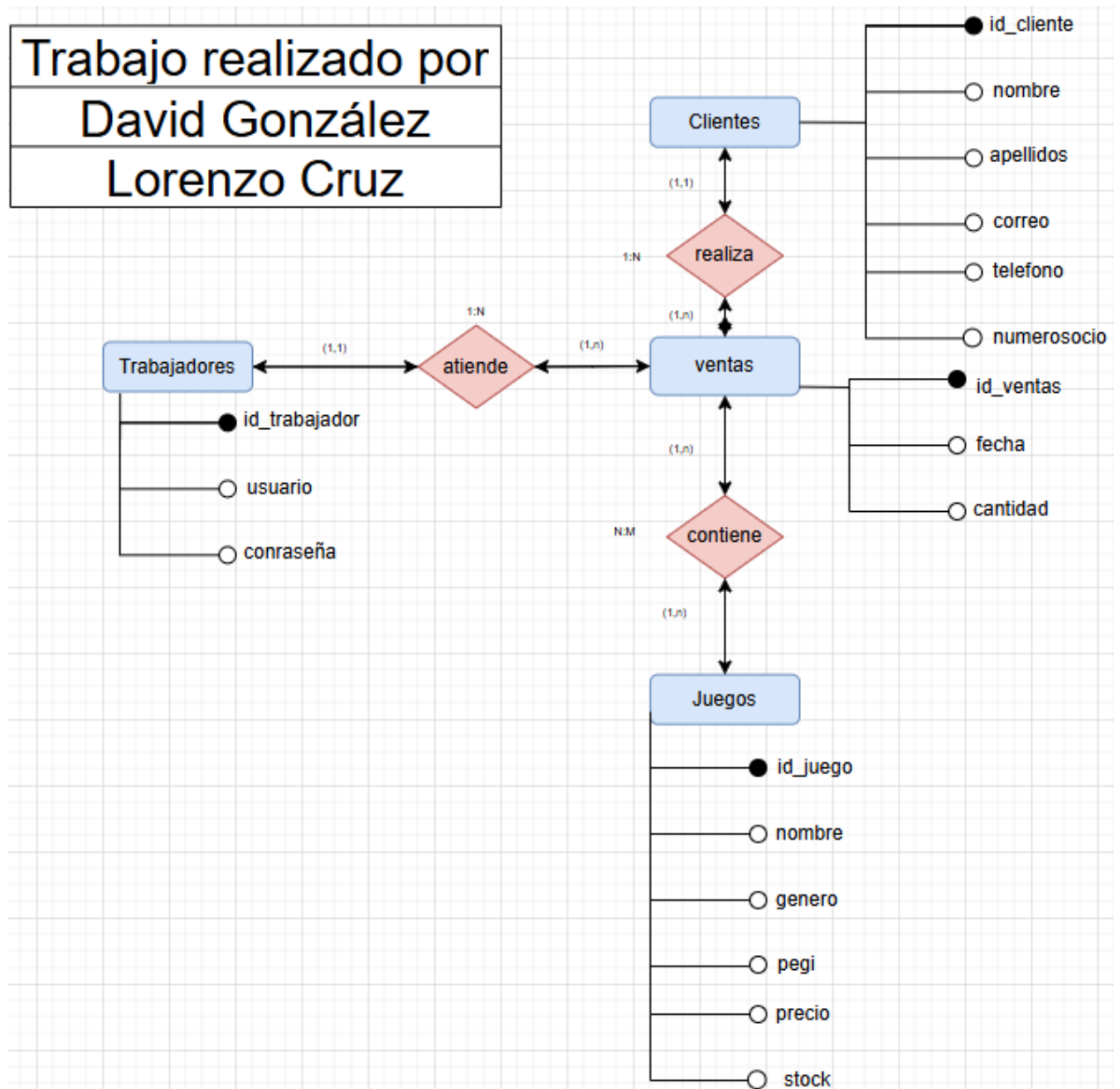
Diagrama de Casos de uso

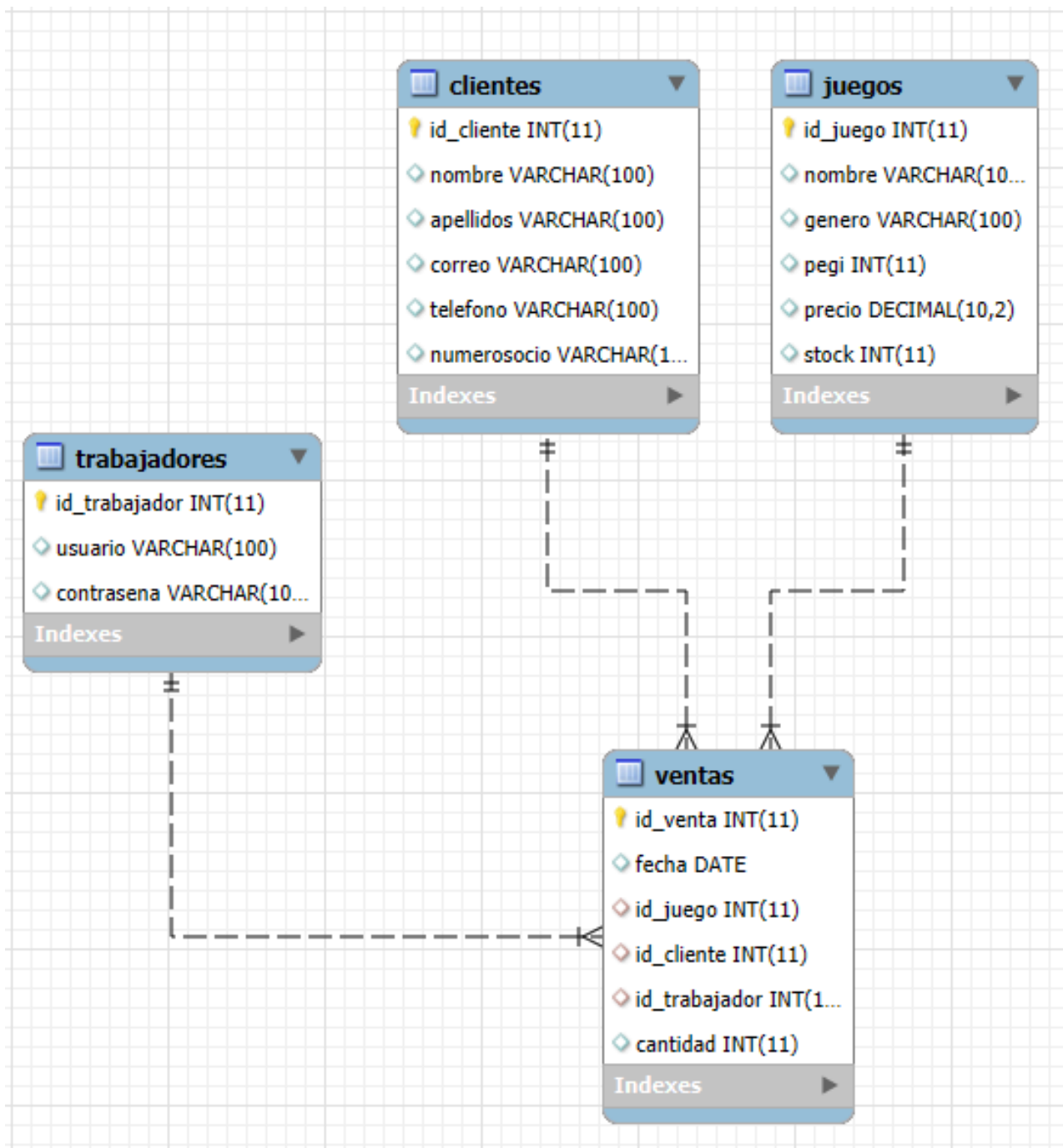


## b. Requisitos No Funcionales

- Interfaz intuitiva.
- Facilidad de uso
- Implementación de interfaz
- diseño tipo móvil

## c. Diagrama Entidad-Relación (E-R)

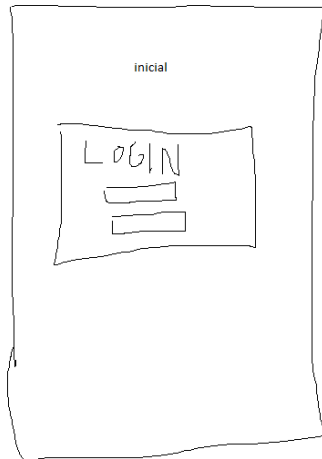




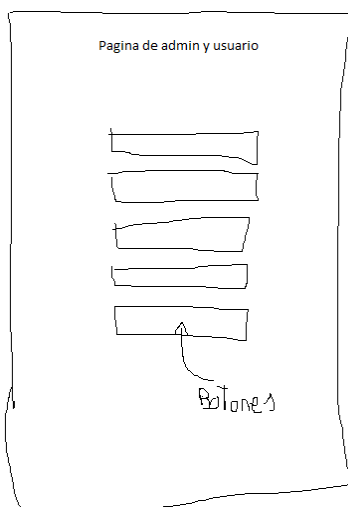
## 8. Diseño

### a. Mock-up

Página de login



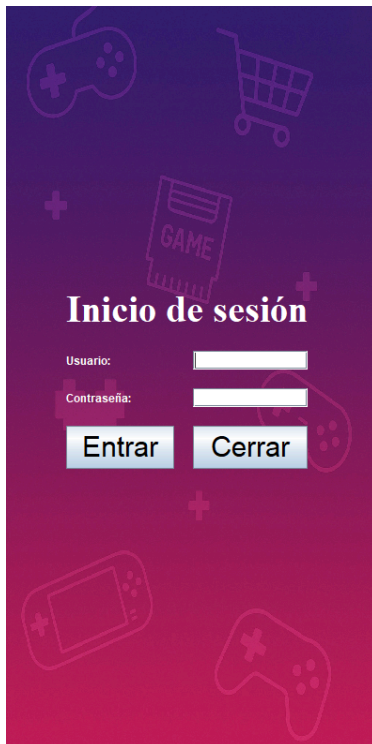
Página de administrador y usuario





## b. Resultado Final

Página de login



The login page features a vertical gradient background transitioning from dark purple at the top to a vibrant red at the bottom. Scattered across the background are faint, white line-art icons: a game controller, a shopping cart, a game cartridge labeled 'GAME', and a handheld console. The title 'Inicio de sesión' is centered in a bold, white, sans-serif font. Below the title, the labels 'Usuario:' and 'Contraseña:' are positioned to the left of two white rectangular input fields. At the bottom of the form, there are two white buttons with black text: 'Entrar' on the left and 'Cerrar' on the right.

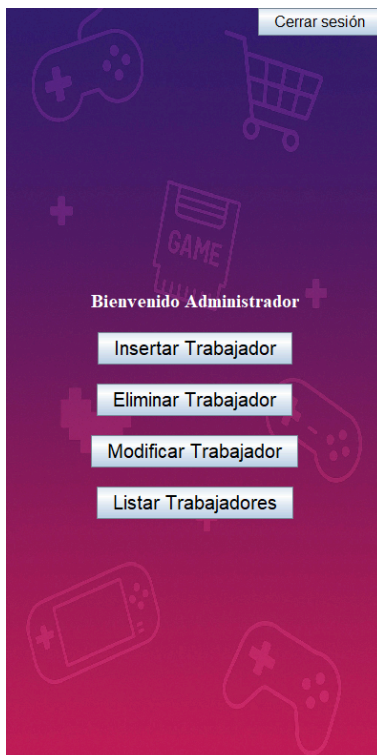
**Inicio de sesión**

Usuario:

Contraseña:

**Entrar** **Cerrar**

Página de administrador y usuario



The administrator page shares the same purple-to-red gradient background and gaming icons as the login page. In the top right corner, there is a small white button with the text 'Cerrar sesión'. The main heading 'Bienvenido Administrador' is centered in a bold, white, sans-serif font. Below this heading, four white buttons with black text are stacked vertically: 'Insertar Trabajador', 'Eliminar Trabajador', 'Modificar Trabajador', and 'Listar Trabajadores'.

**Cerrar sesión**

**Bienvenido Administrador**

**Insertar Trabajador**

**Eliminar Trabajador**

**Modificar Trabajador**

**Listar Trabajadores**

## 9. Partes Resaltables del Código / Explicación

InicioPanel: es el primer panel que se pinta en mi programa y trata de un pequeño login donde se pide Usuario y contraseña

```
public InicioPanel(GameStoreFrame frame) { 1 usage new *
    this.frame = frame;
    imgFondo = new ImageIcon( filename: "src/main/java/org/example/imagenes/fondo.jpg").getImage();
    setLayout(new GridBagLayout());

    GridBagConstraints gbc = new GridBagConstraints();
    gbc.insets = new Insets( top: 10, left: 10, bottom: 10, right: 10);
    gbc.fill = GridBagConstraints.HORIZONTAL;

    JLabel label = new JLabel( text: "Inicio de sesión");
    label.setForeground(Color.WHITE);
    label.setFont(new Font( name: "Serif", Font.BOLD, size: 40));
    label.setHorizontalAlignment(SwingConstants.CENTER);

    gbc.gridx = 0;
    gbc.gridy = 0;
    gbc.gridwidth = 2;
    add(label, gbc);

    // Usuario
    JLabel labelUsuario = new JLabel( text: "Usuario:");
    labelUsuario.setForeground(Color.WHITE);
    gbc.gridx = 0;
    gbc.gridy = 1;
    gbc.gridwidth = 1;
    add(labelUsuario, gbc);

    campoUsuario = new JTextField();
    gbc.gridx = 1;
    gbc.gridy = 1;
    add(campoUsuario, gbc);

    // Contraseña
    JLabel labelContraseña = new JLabel( text: "Contraseña:");
    labelContraseña.setForeground(Color.WHITE);
    gbc.gridx = 0;
    gbc.gridy = 2;
    add(labelContraseña, gbc);
}
```

aparte tenemos 2 botones

```
// Botones
btnInicio = new JButton( text: "Entrar");
btnInicio.setForeground(Color.black);
btnInicio.setFont(new Font( name: "Arial", Font.PLAIN, size: 30));
btnInicio.setFocusPainted(false);

btnCerrar = new JButton( text: "Cerrar");
btnCerrar.setForeground(Color.black);
btnCerrar.setFont(new Font( name: "Arial", Font.PLAIN, size: 30));
btnCerrar.setFocusPainted(false);

gbc.gridx = 0;
gbc.gridy = 3;
add(btnInicio, gbc);

gbc.gridx = 1;
gbc.gridy = 3;
add(btnCerrar, gbc);

setOpaque(false);
```

Entrar: pilla los datos dados de usuario, contraseña rellenados en el apartado de antes y manda esos datos a gestor.login función que explicaremos más tarde.

dependiendo de lo que pongas en usuario si pones admin y su contraseña entrarás en el apartado de admin, si pones otro usuario creado por el admin entrarás en la parte de trabajador.

```
// ACCIÓN DEL BOTÓN ENTRAR
btnInicio.addActionListener(new ActionListener() { new *
    @Override new *
    public void actionPerformed(ActionEvent e) {
        String usuario = campoUsuario.getText();
        String contraseña = campoContraseña.getText();
        System.out.println(usuario);
        Gestor gestor = new Gestor();
        boolean loginCorrecto = gestor.login(usuario, contraseña);

        if (loginCorrecto) {
            JOptionPane.showMessageDialog( parentComponent: null, message: "Inicio de sesión correcto. ¡Bienvenido!" + usuario);

            if (usuario.equalsIgnoreCase( anotherString: "admin")) {
                frame.mostrarAdmin();
            } else if (!usuario.equalsIgnoreCase( anotherString: "admin")) {
                frame.mostrarTrabajador(usuario);
            }
            campoUsuario.setText("");
            campoContraseña.setText("");
        } else {
            JOptionPane.showMessageDialog( parentComponent: null, message: "Usuario o contraseña incorrectos", title: "Error", JOptionPane.ERROR_
        }
    }
});
```

Cerrar: es el botón usado para cerrar la aplicación usamos `system.exit(0)` para llevarlo a cabo.

```
btnCerrar.addActionListener(new ActionListener() { new *
    @Override new *
    public void actionPerformed(ActionEvent e) { System.exit( status: 0); }
});
}
```

TrabajadorPanel: panel en el que se entra cuando el login es correcto y pertenece a un trabajador esta formada por un fondo y por botones bastante basico no tiene nada mas

```
public class TrabajadorPanel extends JPanel { 2 usages new *
    private Image imgFondo; 3 usages
    public JButton btnRealizarVenta; 7 usages
    public JButton btnActualizarPrecio; 9 usages
    public JButton btnActualizarStock; 6 usages
    public JButton btnAgregarCliente; 8 usages
    public JButton btnListarJuegos; 6 usages
    public JButton btnInsertarJuego; 6 usages
    public JButton btnBorrarJuego; 6 usages

    public JButton btnCerrarSesion; 6 usages
    private GameStoreFrame frame; 1 usage
    Gestor gestor = new Gestor(); 10 usages
    private JLabel labelNombre; 6 usages
```

El botón cerrar sesion que lo que hace es volver a la parte del login

```
// Acción del botón "Cerrar sesión"
btnCerrarSesion.addActionListener(new ActionListener() { new *
    @Override new *
    public void actionPerformed(ActionEvent e) { frame.mostrarInicio(); // Vuelve al panel de inicio }
});
```

Las funciones de los botones de el programa funcionan de tal manera que cuando van a realizar una acción rellenas un formulario, estos datos se llevan a otro método de la clase gestor que se conectara con la base datos y haciendo el update, insert and delete dependiendo de lo que queramos en ese botón ahora explicaremos un botón ya que todo funciona más o menos igual.

## Botón de agregar cliente

en esta parte se definen todos los componentes de la interfaz que saldrá a darle al botón

```
btnAgregarCliente.addActionListener( ActionEvent e -> {  
    JDialog dialogo = new JDialog((Frame) SwingUtilities.getWindowAncestor(btnAgregarCliente), title: "Agregar Cliente", modal: true);  
    dialogo.setSize( width: 400, height: 300);  
    dialogo.setLayout(new GridBagLayout());  
    dialogo.setLocationRelativeTo(null);  
  
    GridBagConstraints gbc2 = new GridBagConstraints();  
    gbc2.insets = new Insets( top: 5, left: 5, bottom: 5, right: 5);  
    gbc2.fill = GridBagConstraints.HORIZONTAL;  
  
    JLabel lblNombre = new JLabel( text: "Nombre:");  
    JTextField campoNombre = new JTextField( columns: 15);  
  
    JLabel lblApellidos = new JLabel( text: "Apellidos:");  
    JTextField campoApellidos = new JTextField( columns: 15);  
  
    JLabel lblCorreo = new JLabel( text: "Correo:");  
    JTextField campoCorreo = new JTextField( columns: 15);  
  
    JLabel lblTelefono = new JLabel( text: "Teléfono:");  
    JTextField campoTelefono = new JTextField( columns: 15);  
  
    JLabel lblSocio = new JLabel( text: "Número de socio:");  
    JTextField campoSocio = new JTextField( columns: 10);  
  
    JButton btnInsertar = new JButton( text: "Insertar");
```

Aquí se define el grid que le está dando forma a mi interfaz lo hacemos así ya que lo tendríamos más dividido y mejor organizado ya que es bastante sencillo darle una posición “x” e “y” para organizar

```
gbc2.gridx = 0; gbc2.gridy = 0;
dialogo.add(lblNombre, gbc2);
gbc2.gridx = 1;
dialogo.add(campoNombre, gbc2);

gbc2.gridx = 0; gbc2.gridy = 1;
dialogo.add(lblApellidos, gbc2);
gbc2.gridx = 1;
dialogo.add(campoApellidos, gbc2);

gbc2.gridx = 0; gbc2.gridy = 2;
dialogo.add(lblCorreo, gbc2);
gbc2.gridx = 1;
dialogo.add(campoCorreo, gbc2);

gbc2.gridx = 0; gbc2.gridy = 3;
dialogo.add(lblTelefono, gbc2);
gbc2.gridx = 1;
dialogo.add(campoTelefono, gbc2);

gbc2.gridx = 0; gbc2.gridy = 4;
dialogo.add(lblSocio, gbc2);
gbc2.gridx = 1;
dialogo.add(campoSocio, gbc2);

gbc2.gridx = 0; gbc2.gridy = 5; gbc2.gridwidth = 2;
dialogo.add(btnInsertar, gbc2);
```

Aquí tenemos que poner el boton de de insertar datos en el que ingresamos los textareas en variables y después depuramos con un primer if por si alguna variable no la tuviéramos rellena para que no nos de null, si está todo correcto se mandaran los datos a gestor.insertar cliente, si dicha función nos devuelve un false dará un mensaje de error por no conectarse en la BBDD

```
btnInsertar.addActionListener( new ActionListener() {
    public void actionPerformed( ActionEvent ae ) {
        String nombre = campoNombre.getText().trim();
        String apellidos = campoApellidos.getText().trim();
        String correo = campoCorreo.getText().trim();
        String telefono = campoTelefono.getText().trim();
        String socio = campoSocio.getText().trim();

        if (nombre.isEmpty() || apellidos.isEmpty() || correo.isEmpty() || telefono.isEmpty() || socio.isEmpty()) {
            JOptionPane.showMessageDialog(dialogo, "Rellena todos los campos", "Error", JOptionPane.ERROR_MESSAGE);
        } else {
            Gestor gestor = new Gestor();
            boolean exito = gestor.insertarCliente(nombre, apellidos, correo, telefono, socio);
            if (exito) {
                JOptionPane.showMessageDialog(dialogo, "Cliente insertado correctamente.");
                dialogo.dispose();
            } else {
                JOptionPane.showMessageDialog(dialogo, "Error al insertar cliente", "Error", JOptionPane.ERROR_MESSAGE);
            }
        }
    }
});
```

La siguiente parte seria gestor.insertarCliente

Clase Gestor

Antes de seguir con la funcionalidad explicaremos brevemente cómo está conectado esa clase a la BBDD

Se define la conexión a la BBDD y se crea un método para conectar. Método que se usara en cada función de dicha clase para poder conectar

```
public class Gestor { 25 usages new *  
  
    private static final String URL = "jdbc:mysql://localhost:3306/GameStop"; 1 usage  
    private static final String USUARIO = "root"; 1 usage  
    private static final String PASSWORD = ""; 1 usage  
    public static String nombre; 1 usage  
  
    public static Connection conectar() { 17 usages new *  
        try {  
            return getConnection(URL, USUARIO, PASSWORD);  
        } catch (SQLException e) {  
            System.out.println("Error al conectar con la base de datos");  
            e.printStackTrace();  
            return null;  
        }  
    }  
}
```

Se define atributo que se insertará en la BBDD que este caso sería un insert después realizamos la conexión a la BBDD con el método dicho antes Conectar()

se define el preparedStatement que es el conversor para insertar los datos en la base de datos al lenguaje de mysql

cada ? es un dato en el que se introduce según la posición abajo de eso poniendo posición 1,2,3 tantas como ? encontremos para rellenarlo y que no de null importe que estén todos los ? rellenos y en el orden que lo tenemos en la base de datos para que no de error

```
public boolean insertarCliente(String nombre, String apellidos, String correo, String telefono, String numeroSocio) {  
    String sql = "INSERT INTO clientes (nombre, apellidos, correo, telefono, numerosocio) VALUES (?, ?, ?, ?, ?)";  
  
    try (Connection con = conectar();  
        PreparedStatement ps = con.prepareStatement(sql)) {  
  
        ps.setString(1, nombre);  
        ps.setString(2, apellidos);  
        ps.setString(3, correo);  
        ps.setString(4, telefono);  
        ps.setString(5, numeroSocio);  
  
        ps.executeUpdate();  
        return true;  
    } catch (SQLException e) {  
        e.printStackTrace();  
        return false;  
    }  
}
```

## 10. Conclusión del Trabajo y del Curso

- Avances en el uso de la parte de interfaz gráfica añadiendo cosas nuevas y adquiriendo conocimientos de cosas nuevas que hemos querido implementar en nuestra parte de interfaz.
- Uso bastante avanzado de la implementación de la BBDD en nuestro proyecto.
- En resumidas cuentas es un trabajo en el que hemos sabido trabajar en equipo y aprender el uno del otro, nos hemos nutrido bastante del uso de BBDD ya que sería la novedad y hemos querido hacer lo máximo en este trabajo dentro del tiempo que hemos dispuesto para ello. muchas gracias por tu tiempo y dedicación, y buen verano.