MEDAC
Instituto Oficial de Formación Profesional

Documentación
Proyecto Final

Pablo Rodríguez Segura
Ignacio Vidal Mendoza

Índice

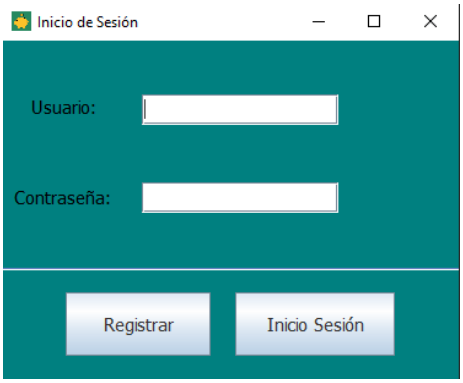
Documentación Proyecto Final

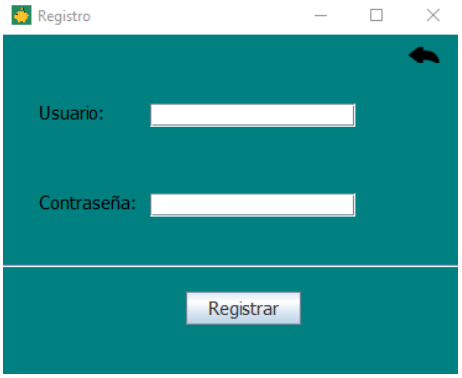
1. Introducción a la aplicación

Nuestra aplicación estará basada en un sistema de gestión de gastos (no se trata de una aplicación de banco, sino personal), dónde podremos añadir los ingresos que entran a nuestra cuenta y los gastos que realizamos para así llevar un control más ordenado del dinero que entra en nuestra cuenta.

2. Instrucciones de uso.

En primer lugar comenzaremos hablando sobre la página de inicio de sesión la cual no tiene mucho misterio. Si ya tienes un usuario registrado, debes iniciar sesión, pero en el caso contrario pulsando el botón de registrarse, abre otra pestaña dónde puedes registrar tu usuario y contraseña que se quedarán guardados en la base de datos para que la próxima vez puedas entrar con ese usuario.





En el registro o el inicio de sesión será donde estableceremos la conexión con la base de datos:

```
private boolean ConexionEstablecida = false; //Creamos una variable booleana para saber si la conexion ya ha sido establecida
```

Dentro del actionPerformed crearemos una estructura if en la cuál dependiendo si entramos a través del inicio de sesión o el registro establecerá la conexión o uno u otro.

Si pulsamos el botón de registro:

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

    if(e.getSource()==botonregistro) {

        //Establece conexión con la base de datos para registro si no se ha establecido previamente
        if(ConexionEstablecida ==false) {
            conexion = new ConexionMySQL("proyectofinal", "proyectofinal", "proyectofinal");
            try {
                conexion.conectar();
                ConexionEstablecida=true;//Marcamos que la conexión se ha establecido

            } catch (SQLException e1) {
                e1.printStackTrace();
            }
        }
```

Si pulsamos el de inicio de sesión:

```
else {


// Recoge el nombre de usuario y contraseña de la interfaz

String usuario = UsuarioValor.getText();
String contraseña = ContraseñaValor.getText();// Pendiente

System.out.println("Controlador: " + "Usuario: " + usuario + " " + "Contraseña: " + contraseña);

// Establece conexión con la base de datos desde el inicio de sesión
if(ConexionEstablecida ==false) {
    conexion = new ConexionMySQL("proyectofinal", "proyectofinal", "proyectofinal");
    try {
        conexion.conectar();
        ConexionEstablecida=true;//Marcamos que la conexión se ha establecido

    } catch (SQLException e1) {
        e1.printStackTrace();
    }
}
```

Para rescatar los datos de la base de datos utilizaremos dos SELECT uno para el usuario y otro para la contraseña, estos métodos están localizados en nuestro paquete *Repositorio* el cual llamaremos desde la clase que estemos usando.

```
public static boolean login(String usuario, String contraseña, ConexionMySQL conexion) throws SQLException  {
      String resultado = "";
      String resultado2 = "";

        System.out.println("Repositorio: consultando a la base de datos. usuario: " + usuario + " contraseña: " + contraseña);


// comprobar si el usuario y la contraseña existen en la base de datos String
String sentenciaNombre = "SELECT usuario FROM Usuarios  WHERE usuario = '" + usuario + "'";

String sentenciaContraseña ="SELECT contraseña  FROM Usuarios    WHERE contraseña = '" + contraseña + "'";
```

Para comprobar que el usuario y la contraseña coinciden con la de la base de datos usaremos la estructura ResultSet:

```
ResultSet datos;
//Comprobación  de nombre
datos = conexion.ejecutarSelect(sentenciaNombre);
while(datos.next()) {
    // Consulta del nombre
    resultado = datos.getString("usuario");//nombre del campo en la base de datos
    System.out.println("RESULTADO USUARIO: " + resultado);
}
//Comprobación de contraseñaDO
datos = conexion.ejecutarSelect(sentenciaContraseña);
while(datos.next()) {
    // Consulta del nombre
  resultado2 = datos.getString("contraseña");
    System.out.println("RESULTADO CONTRASEÑA: " + resultado2);
}

if(!resultado.equals("") && (!resultado2.equals(""))) {
    System.out.println("El usuario existe");
    return true;
}
  return false;
```

Para el registro usaremos un Insert pasandole los campos que queramos añadir a nuestra base de datos ,en este caso usuario y contraseña

```
public class FuncionesRegistro {

    public static int registro(String usuario, String contraseña, ConexionMySQL conexion) throws SQLException  {


      // comprobar si el usuario y la contraseña existen en la base de datos String
      String sentenciaRegistro = "INSERT INTO Usuarios( usuario, contraseña) " + " VALUES ('" + usuario + "','" + contraseña + "');";

       int numfilas = conexion.ejecutarInsertDeleteUpdate(sentenciaRegistro);
       return numfilas;
```

Una vez hayamos iniciado sesión entraremos en la Ventana Principal. Esta ventana está compuesta por un Menú en el cuál encontraremos las secciones de Categoría, Historial y Salir, en las que podremos añadir nuevas categorías las cuales veremos más adelante, consultar las transacciones del usuario registrado y cerrar la sesión donde nos desconectamos de la base de datos.

Esta ventana está formada principalmente por dos botones vistosos con los cuales podremos añadir un ingreso o un gasto, esto modificará el saldo que tengamos en ese momento:

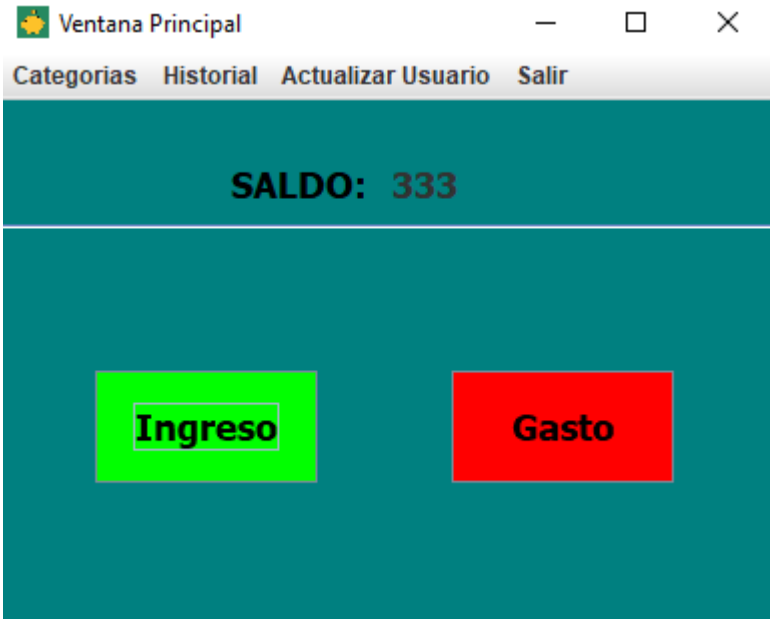
Para actualizar nuestro saldo haremos dos consultas a nuestra base de datos en las que pediremos que nos den los ingresos y gastos correspondientes al usuario registrado.

```
//Obtener saldo
    public static int obtenerSaldo(String usuario, ConexionMySQL conexion) throws SQLException {
        String resultadoGastos="", resultadoIngresos="";


        ////////////////////////   Sentencias SQL para obtener gastos e ingresos del usuario  ////////////////////

        String sentenciaGasto = "SELECT sum(importe) AS SumaGastos "
        + "FROM `Operaciones` WHERE movimiento='Gasto' AND usuario ='" + usuario + "';" ;

        String sentenciaIngreso ="SELECT sum(importe) AS SumaIngresos "
        + "FROM `Operaciones` WHERE movimiento='Ingreso' AND usuario = '" + usuario + "';" ;
```

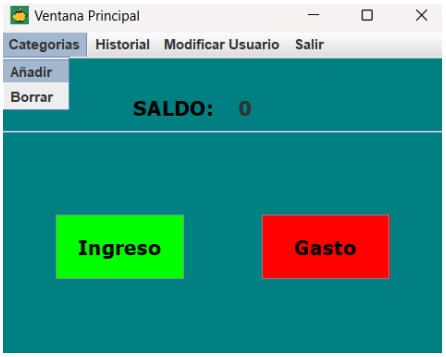
Después de comprobar que existen estos datos, pasaremos nuestras variables creadas de enteros a String para posteriormente poder restarlas y que el método nos devuelva el saldo exacto, el cual llamaremos en nuestra clase Principal

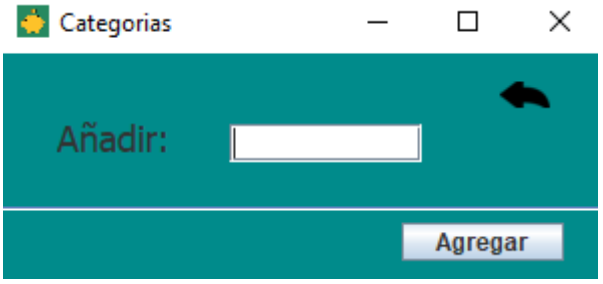
```
//Pasamos las variables String a entero para poder operar con ellas y obtener el saldo final
            int TotalIngresos=0;
            int TotalGastos=0;

            if(resultadoIngresos!=null) {
                TotalIngresos= Integer.parseInt(resultadoIngresos);
            }
            if(resultadoGastos!=null) {
                TotalGastos = Integer.parseInt(resultadoGastos);

            }
            return TotalIngresos - TotalGastos;
        }
```

Saldo actualizado:



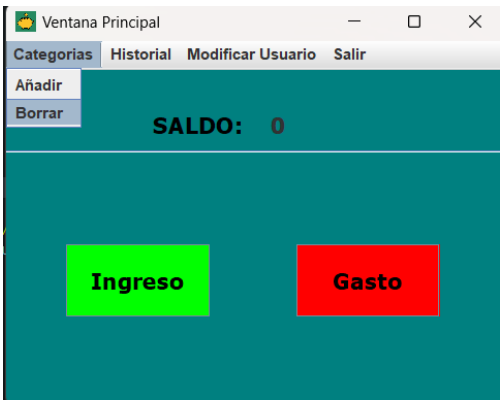
En la barra horizontal,si pinchamos en **Añadir** dentro de **Categorías** nos llevará a una ventana en la cuál podremos añadir nuevas categorías en el menú desplegable que veremos más adelante situado en la siguiente ventana.

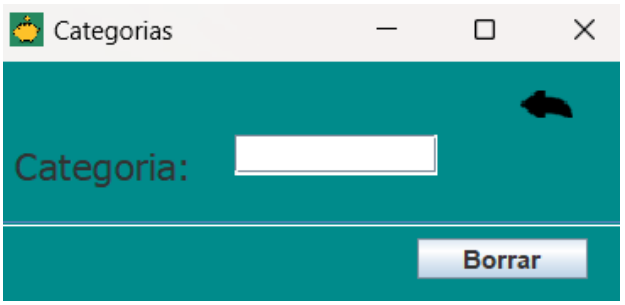





```java
package Repositorio;

import java.sql.SQLException;

public class FuncionesCategorias {

    public static int añadirCategorias(String nombre , ConexionMySQL conexion) throws SQLException {
        String sentenciaCategorias = "INSERT INTO Categorias(nombre) "
                + " VALUES ('" +nombre+"');";

        int numfilas = conexion.ejecutarInsertDeleteUpdate(sentenciaCategorias);
         return numfilas;


    }
}
```

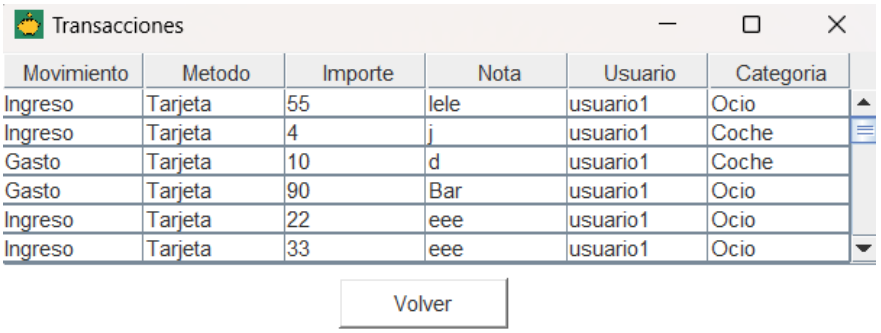
Si pinchamos en **Borrar** dentro de **Categorías** nos llevará a una ventana en la cuál podremos borrar categorías ya existentes , en el caso de que el nombre que introducimos no coincida con uno ya existente nos saltará un mensaje de fallo.





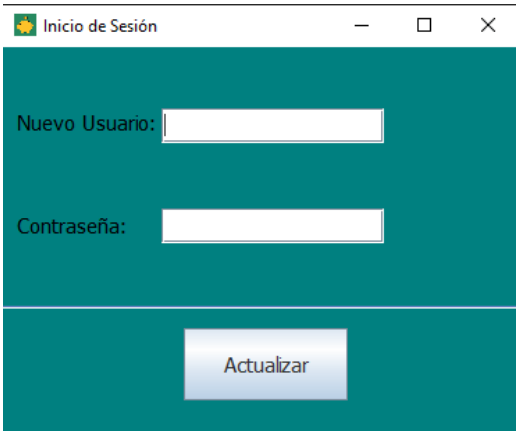
```java
public static int BorrarCategoria(String categoria, ConexionMySQL conexion) throws SQLException {

    String sentenciaBorrar = "DELETE FROM Categorias WHERE Nombre = '" + categoria + "';";
    String sentenciaConsulta = "SELECT Nombre FROM Categorias WHERE Nombre= '" + categoria + "';";
    String resultado = "";


    /////// Consulta de la categoría a borrar

    ResultSet borrar;
      //Comprobación  de nombre
    borrar = conexion.ejecutarSelect(sentenciaConsulta);
      while(borrar.next()) {
          // Consulta del nombre
          resultado = borrar.getString("Nombre");//nombre del campo en la base de datos
      }
      //
      if(resultado.equals(categoria)) {
          int fila= conexion.ejecutarInsertDeleteUpdate(sentenciaBorrar);
      }
      else {
          JOptionPane.showMessageDialog(null, "La categoria no existe",
                  "Error", JOptionPane.ERROR_MESSAGE);
      }
        return 0;
```

Si pinchamos en **Transacciones** nos mostrará un historial de los movimientos realizados por el usuario logueado



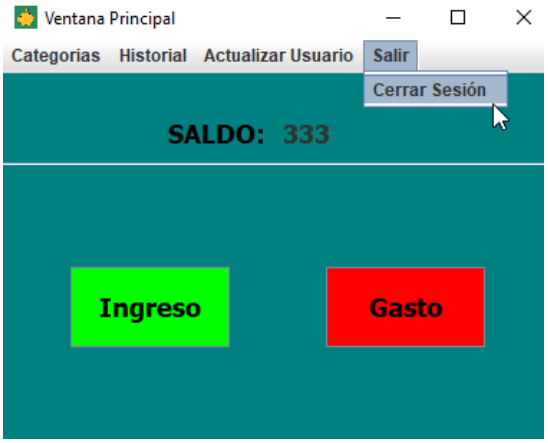
```java
public static ArrayList<Transaccion> Historial(String usuario,ConexionMySQL conexion) throws SQLException {
    String sentenciaImporte = "SELECT movimiento, metodopago, importe, notas, usuario, Categoria FROM Operaciones where usuario="+"'"+usuario+"';";

    ArrayList<Transaccion> resultadoHistorial= new ArrayList<>();

    // campos de cada transacción
    String movimiento,metodopago,importe, notas, Categoria;
    Transaccion transaccion;


    ResultSet Transacciones;
    //Obtención de Transacciones del usuario
    Transacciones = conexion.ejecutarSelect(sentenciaImporte);
      while(Transacciones.next()) {
          // Consulta de ls Transacciones

          movimiento = Transacciones.getString("movimiento");
          metodopago = Transacciones.getString("metodopago");
          importe = Transacciones.getString("importe");
          notas = Transacciones.getString("notas");
          Categoria = Transacciones.getString("Categoria");

          transaccion =  new Transaccion(movimiento,metodopago,importe, notas, usuario,Categoria);
          resultadoHistorial.add(transaccion);

          }
    return resultadoHistorial;
```

En Nuevo Usuario podremos cambiar el nombre del usuario con el que nos hayamos registrado

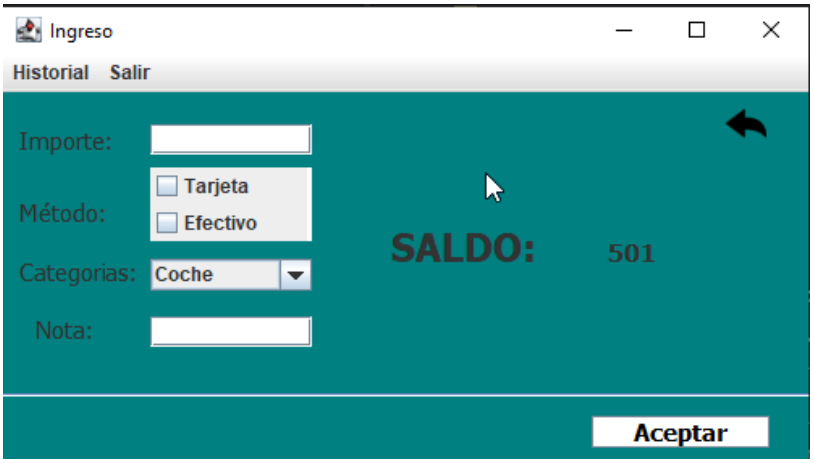


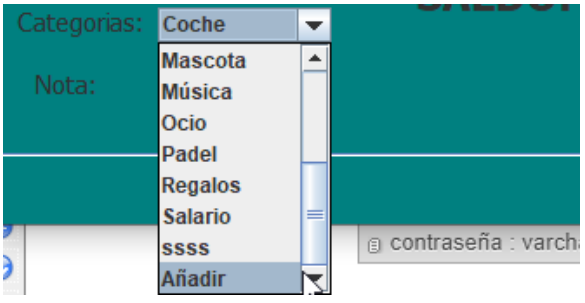

```java
//Actualizar nombre usuario
public static void actualizar(String usuario,String usuarionuevo,ConexionMySQL conexion) throws SQLException {
    String CambiarUsuario = "UPDATE Usuarios  SET usuario = '" + usuarionuevo+ "' WHERE usuario = '"
+ usuario + "';";
    System.out.println("usuario nuevo: " + usuarionuevo +  "usuario: " + usuario);


    int filas = conexion.ejecutarInsertDeleteUpdate(CambiarUsuario);
    System.out.println("fila: " + filas);
    if(filas==1)//Significa que he hecho
    {
        JOptionPane.showMessageDialog(null, "Usuario cambiado con éxito",
                "Cambio de nombre de usuario", JOptionPane.YES_NO_CANCEL_OPTION);}
    }

}
```

Cerramos la sesión y desconectamos con la base de datos



Para finalizar con las instrucciones de uso, si en la pestaña principal le hemos dado a **"gasto"** o **"ingreso"** se nos abrirá una ventana en la cual podremos de añadir el importe del gasto o del ingreso, el método de pago, ya sea en efectivo o con tarjeta, filtrar por categorías (inicialmente la aplicación ofrece unas categorías predeterminadas, y una opción de **añadir** que nos redirigirá a la ventana de **"Categorías"**, dónde podremos añadir nuevas categorías y guardarlas en nuestro menú desplegable. Y por último la opción de dejar una nota a modo de descripción del gasto o del ingreso.





Cuando pulsemos el botón de aceptar el nos redirigirá a la Ventana Principal y se actualizará el Saldo y el Historial.

```java
package Repositorio;

import java.sql.SQLException;

public class FuncionesCategorias {

    public static int añadirCategorias(String nombre , ConexionMySQL conexion) throws SQLException {
        String sentenciaCategorias = "INSERT INTO Categorias(nombre) "
                + " VALUES ('" +nombre+"');";

        int numfilas = conexion.ejecutarInsertDeleteUpdate(sentenciaCategorias);
         return numfilas;


    }
}
```

Por último, si te equivocas y quieres volver al menú principal, en vez de cerrar sesión, puedes pinchar en el icono de la flecha hacia atrás situado arriba a la derecha.

3. Descripción de las funcionalidades

Las funcionalidades principales de la aplicación son:

- **Ingresos de datos:** Permiten al usuario añadir datos sobre sus transacciones, ya sea un ingreso o un gasto.
- **Categorización:** La aplicación permite dividir en categorías las posibles opciones de las que suponen el ingreso o el gasto.
- **Historial de transacciones:** Da la opción a ver el historial completo de todas las transacciones que ha realizado el usuario desde que se registró por primera vez y filtrar y buscar datos
- **Balance:** La aplicación es capaz de hacer un balance entre los ingresos y los gastos para mostrar el saldo actual que tiene el usuario.

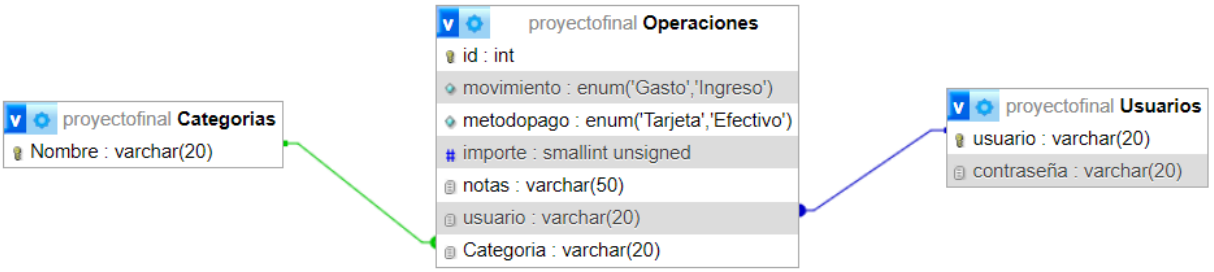
4. Explicación de la estructura del código

La codificación está hecha en eclipse, y está dividida en 5 packages (paquetes). Los paquetes son los siguientes:

- **Modelo:** El cual tiene una clase dónde están los métodos de la ventana de transacciones.
- **Imágenes:** Paquete en el cual guardaremos las imágenes que usemos en la aplicación.
- **Principal:** Aquí encontramos la clase main, la cual es la que ejecuta el programa y se abre el inicio de sesión.
- **Repositorio:** En el cual están situados las conexiones con la base de datos y las funciones de cada ventana.
- **Vista:** Por último en este paquete tenemos las clases en las que se encuentran las interfaces, las variables y los métodos necesarios para que la aplicación funcione como es debido.

5. Explicación del diseño de la base de datos

Para la creación de la base de datos hemos creado 3 tablas: **Operaciones**, **Usuarios** y **Categorías**.



La PrimaryKey de **Operaciones** es el id, tiene otros atributos como movimiento, metodopago, importe, notas, usuario y categoría. Estos dos últimos están conectados por ForeingKey a otras tablas. El atributo categoría está unida a la primary key de la tabla **Categorías**, que es el nombre, mientras que el usuario de la tabla operaciones está unido a la primary key de la tabla **Usuarios** que es el Id. Además Usuarios tiene otros atributos como usuario y contraseña.

Para concretar las condiciones de los atributos procederemos a explicarlas:

- **Varchar:** Permite almacenar tantos caracteres como se indique entre paréntesis.
- **Enum:** Significa que solo se podrá escoger uno de los dos valores que aparecen.

- **Tinyint unsigned:** Permite escribir números enteros hasta 8 bits, el unsigned prohíbe usar números negativos.
- **Smallint unsigned:** Permite escribir números enteros hasta 16 bits, al igual que antes debido al unsigned, solo se pueden escribir números positivos o cero.