IES Ribera del Tajo

EXAMEN UT1 A UT5

DESARROLLO DE INTERFACES

Alejandro Sánchez Gil
2ºDAM CURSO 2024/2025

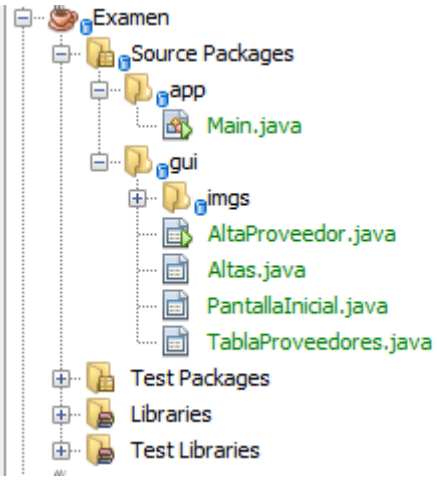
Castilla-La Mancha

ÍNDICE

Creación del proyecto

Se nos ha solicitado realizar una aplicación para una empresa llamada “Team Rocket” en la cual nos solicitan las siguientes pantallas, un login en el que el usuario debe introducir nombre y contraseña con la capacidad de mostrar la contraseña, una vez pulsado el botón de login accederíamos a un menú con cuatro opciones distintas, de momento el cliente solo nos solicita realizar la función de proveedores ya que le corre prisa, así que en esta pantalla simplemente al hacer clic en el botón de proveedores nos llevará a la siguiente pantalla solicitada de proveedores que es un formulario de inscripción de proveedor, y finalmente en otro de los botones se puede abrir la lista de proveedores, que cuenta con un botón para actualizar los proveedores.

Una vez establecido lo necesitado por el cliente comenzamos creando el proyecto, para esto estoy utilizando la herramienta NetBeans 8.2 y la interfaz de usuario la crearemos con elementos de tipo swing, esta es la organización del proyecto:



Aplicación de Look And Feel

Empezamos el proyecto de examen aplicando el estilo de look and feel a todo nuestro proyecto, para esto yo he creado un paquete app para manejar la aplicación de inicio del proyecto, dentro de este paquete existe el archivo Main donde inicializamos la pantalla inicial y establecemos el estilo para todo nuestro proyecto:

```
public class Main {
    public static void main (String [] args){
        JFrame.setDefaultLookAndFeelDecorated(true);
        //SubstanceLookAndFeel.setSkin("org.jvnet.substance.skin.BusinessBlackSteelSkin");
        PantallaInicial p = new PantallaInicial();
        p.setVisible(true);
    }
}
```

Tengo el estilo quitado por el error desconocido, pero está establecido el estilo de look and feel por defecto que es distinto del normal.

Pantalla inicial

Comenzamos creando la primera pantalla de la interfaz de usuario, como he comentado antes se trata de un login que solicita al usuario nombre de usuario y contraseña, yo he añadido una imagen descriptiva que se asocia con la empresa, y he añadido tres botones, el que inicia sesión, uno para limpiar campos y otro para salir, esta primera pantalla al igual que todas las de la interfaz no se puede cambiar de tamaño y aparece siempre en el centro:

```
 */
public PantallaInicial() {
    initComponents();
    jTextFieldClave.setVisible(false);
    setLocationRelativeTo(null);
    setResizable(false);
    this.setTitle("Logística");
    this.setIconImage(new ImageIcon(getClass().getResource("/gui/imgs/Logo.png")).getImage());
}
```

También le he puesto el título de logística a esta pantalla y el logo corporativo que nos solicitan más adelante.

Como parámetros de la clase de la pantalla he declarado dos elementos, una variable booleana para manejar un cambio de elemento y la de un elemento de la siguiente pantalla que es altas:

```
public class PantallaInicial extends javax.swing.JFrame {
    boolean clave = false;
    Altas a = new Altas(this);
    /**
     * Creates new form PantallaInicial
```

Con la variable booleana clave manejamos el cambio del icono y de mostrar contraseña al usuario al pulsar en el, yo he realizado este ajuste de manera que donde esta el campo de contraseña realmente lo que hay es un jPanel con un cardLayout que contiene ese campo de contraseña y otro campo jTextField normal, lo que sucede al pulsar el botón es que se oculta el campo de contraseña y aparece el de jTextField habiéndose pasado antes el texto claro.

```
private void jButtonMostrarClaveActionPerformed(java.awt.event.ActionEvent evt) {
    if(!clave){
        jTextFieldClave.setText(new String(jPasswordFieldClave.getPassword()));
        jPasswordFieldClave.setVisible(false);
        jTextFieldClave.setVisible(true);
        jButtonMostrarClave.setIcon(new ImageIcon(getClass().getResource("/gui/imgs/ojo.png")));
        clave = true;
    }
    else{
        jPasswordFieldClave.setText(jTextFieldClave.getText());
        jTextFieldClave.setVisible(false);
        jPasswordFieldClave.setVisible(true);
        jButtonMostrarClave.setIcon(new ImageIcon(getClass().getResource("/gui/imgs/ojo-abierto.png")));
        clave = false;

    }
}
```

También cambio el icono de mostrar por uno con un ojo cerrado.

El botón salir simplemente cuenta con una línea simple para salir de la aplicación:

```
private void jButtonSalirActionPerformed(java.awt.event.ActionEvent evt) {
    System.exit(0);
}
```

El botón limpiar simplemente nos establece los campos de texto de nuevo a vacío.

```
private void jButtonLimpiarActionPerformed(java.awt.event.ActionEvent evt) {
    jTextFieldClave.setText("");
    jTextFieldNomUsuario.setText("");
    jPasswordFieldClave.setText("");
}

/**
```

Y finalmente, tenemos el botón de login que nos abre la siguiente pestaña de altas, pero para poder abrir la pestaña de altas primero comprueba que el usuario haya introducido las entradas correctamente, en este caso comprueba que el nombre de usuario no tenga números y que ambos campos no estén en blanco.

```
private void jButtonLoginActionPerformed(java.awt.event.ActionEvent evt) {
    if(!clave){
        jTextFieldClave.setText(new String(jPasswordFieldClave.getPassword()));
    }else{
        jPasswordFieldClave.setText(jTextFieldClave.getText());
    }

    if(validarEntradas()){
        this.setVisible(false);
        a.setVisible(true);
    }
}
```

Aquí también le pasamos el texto de clave al textfield y viceversa para evitar que al pulsar mostrar contraseña y borrar el campo de texto y después hacer login no permita la opción.

Ahora vamos a ver el método para validar las entradas del usuario:

Lo primero que comprobamos es si alguno de los tres campos de texto esta vacío:

```
private boolean validarEntradas(){
    if(jPasswordFieldClave.getPassword().equals("") || (jTextFieldNomUsuario.getText().isEmpty()) || jTextFieldClave.getText().isEmpty()){
        JOptionPane.showMessageDialog(this, "No puede haber campos en blanco", "Error", JOptionPane.ERROR_MESSAGE);
        return false;
    }
```

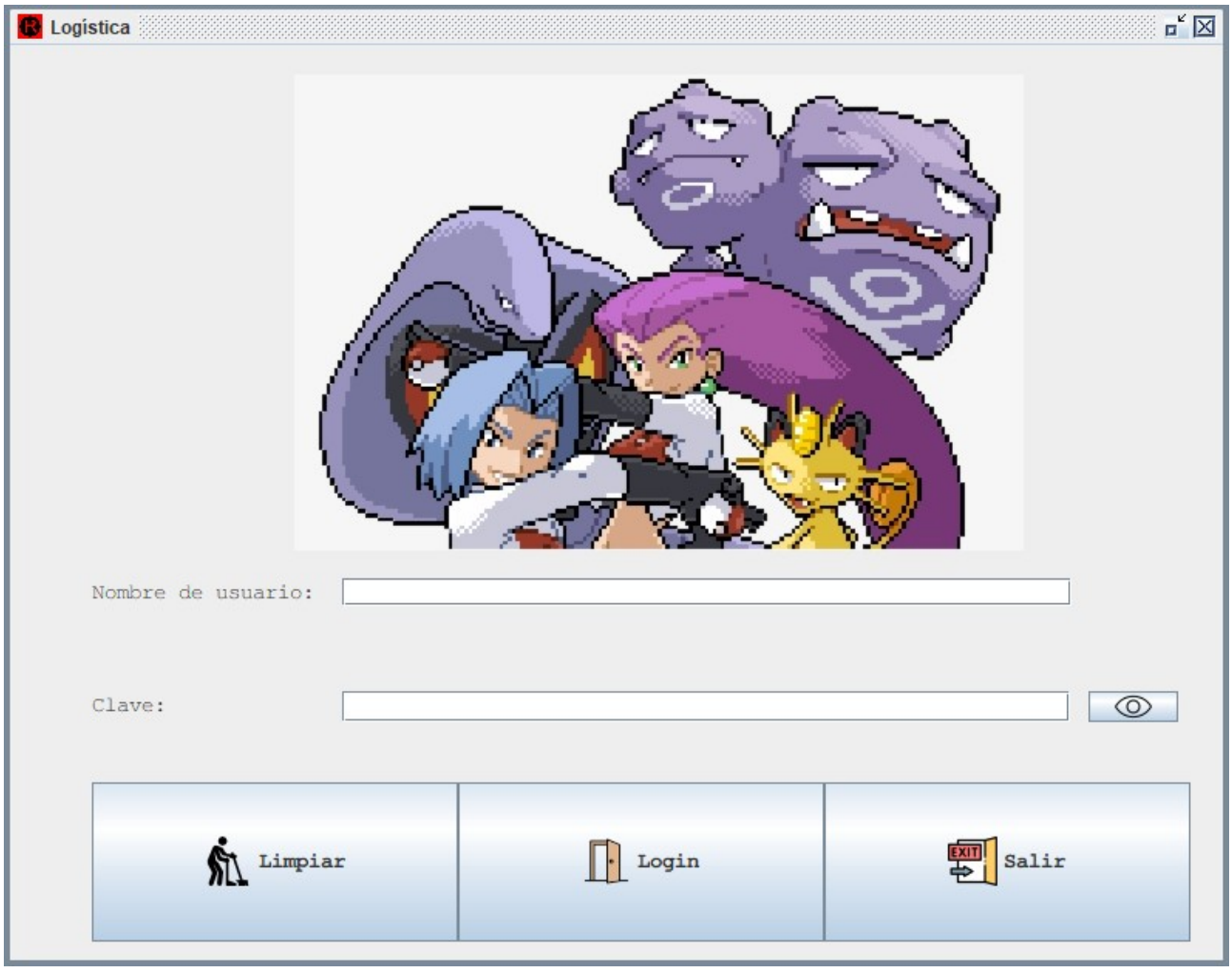
Si no es así continuamos, en caso de que alguno de los campos esté vacío le mandamos un mensaje al usuario y no permitimos el acceso estableciendo la devolución del método a false.

Ahora comprobamos si el nombre de usuario contiene números:

```
if(!jTextFieldNomUsuario.getText().matches("^[a-zA-Z]*$")){
    JOptionPane.showMessageDialog(this, "El nombre de usuario no puede contener números", "Error", JOptionPane.ERROR_MESSAGE);
    return false;
}else{
    return true;
}
```

Si el campo contiene números le lanzamos de nuevo un mensaje de error al usuario para que recuerde que no puede introducir números en el nombre de usuario, en caso de que no hayan números continuamos con normalidad y habríamos validad las dos entradas por lo que se nos abriría la siguiente ventana de altas.

Aquí una imagen del menú:



Pantalla de altas

Ahora veamos la pantalla de altas, como he comentado antes en esta parte se nos solicitaba que creásemos una pantalla con tres botones, uno para los proveedores, otro para los clientes y el último para los productos, yo he decidido crear uno más para la opción de salir y volver al login.

Veamos como hemos manejado esto:

```
import javax.swing.ImageIcon;

/**
 *
 * @author Tarde
 */
public class Altas extends javax.swing.JFrame {

    PantallaInicial parent;
    AltaProveedor altaP;
    /**
     * Creates new form Altas
     */
    public Altas(PantallaInicial parent) {
        this.parent = parent;
        initComponents();
        setLocationRelativeTo(null);
        setResizable(false);
        this.setTitle("Altas");
        this.altaP = new AltaProveedor(this);
        this.setIconImage(new ImageIcon(getClass().getResource("/gui/imgs/Logo.png")).getImage());
        jPanelImagenBasico1.setRutaImagen("/gui/imgs/Guarida.png");
    }
```

Como podemos ver esta pantalla tiene como padre a la PantallaInicial que hemos ejecutado antes, y declaramos un nuevo objeto de nombre “AltaProveedor” para iniciar la siguiente pantalla del formulario de alta, también podemos ver como establezco el logo corporativo como icono y además con un componente personalizado establezco el fondo de pantalla con una ruta relativa.

Aquí tenemos los botones que como he comentado solo nos han solicitado realizar la función de el de proveedores y el añadido del botón salir:

```
    private void jButtonSalirActionPerformed(java.awt.event.ActionEvent evt) {
        parent.setVisible(true);
        this.dispose();
    }

    private void jButtonProveedoresActionPerformed(java.awt.event.ActionEvent evt) {
        altaP.setVisible(true);
        this.setVisible(false);
    }
```

En el botón salir simplemente establecemos como visible la pestaña padre, es decir la pestaña inicial y desechamos esta pestaña y en el botón de Proveedores simplemente volvemos invisible esta pestaña y establecemos como visible la siguiente pestaña de altas de proveedores.

Aquí tenemos como se ve esta pestaña:

En esta pantalla al igual que en todas aquellas en las que se encuentre un botón he establecido una imagen para darle un toque más decorativo a todo el proyecto, estos iconos son todos extraídos de flaticon.

Formulario de alta de proveedores

A continuación la siguiente pantalla que nos sigue es la del formulario de alta de los proveedores, es decir, donde el usuario debe introducir los datos de los proveedores para poder añadir un nuevo proveedor, esta pantalla cuenta con múltiples campos para rellenar:

```
 */
public class AltaProveedor extends javax.swing.JFrame {
    Altas parent;
    TablaProveedores tab;
    /**
     * Creates new form AltaCliente
     */
    public AltaProveedor(Altas parent) {
        this.parent = parent;
        initComponents();
        setLocationRelativeTo(null);
        setResizable(false);
        this.setTitle("Alta Proveedor");
        loadComboBoxWithImages();
        this.tab = new TablaProveedores(this);
        this.setIconImage(new ImageIcon(getClass().getResource("/gui/imgs/Logo.png")).getImage());
        jPanelImagenBasico1.setRutaImagen("/gui/imgs/Logo.png");
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
```

De la misma manera que las otras pantallas en esta he declarado la pestaña padre de la cual proviene que es altas y un objeto del tipo “TablaProveedores” que es la última pestaña de la aplicación, es decir donde sale la información de los proveedores añadidos.

También podemos comprobar que contamos con el método “loadComboBoxWithImages()” para poder establecer imágenes en los elementos del combo box, también he establecido el logo corporativo como icono y también lo he puesto como elemento decorativo en la parte superior de la pantalla.

Esta pantalla cuenta con cuatro botones, de los cuales el botón salir y limpiar siguen la misma estructura que en la pantalla inicial:

```
private void jButtonSalirActionPerformed(java.awt.event.ActionEvent evt) {
    parent.setVisible(true);
    this.dispose();
}
```

```
private void jButtonLimpiarActionPerformed(java.awt.event.ActionEvent evt) {
    jTextFieldApellidos.setText("");
    jTextFieldCIF.setText("");
    jTextFieldDireccion.setText("");
    jTextFieldMailContacto.setText("");
    jTextFieldMailEmpresa.setText("");
    jTextFieldNombreContacto.setText("");
    jTextFieldNombreEmpresa.setText("");
    jTextFieldTlfContacto.setText("");
    jTextFieldTlfEmpresa.setText("");
}
```

Los otros dos botones serían el de añadir proveedor y el de mostrar la tabla.

En el de mostrar la tabla simplemente mostramos la siguiente pantalla con la tabla de proveedores y sus datos.

```
private void jButtonMostrarActionPerformed(java.awt.event.ActionEvent evt) {
    tab.setVisible(true);
}
```

Solo que en este caso al abrir la tabla no cerramos la pestaña de formulario.

Y la mayoría de la lógica la manejamos dentro del botón agregar proveedor que comprueba que las entradas del usuario sean válidas con el método “validarEntradas” y crea al proveedor con el método “crearProveedor”:

```
private void jButtonAgregarActionPerformed(java.awt.event.ActionEvent evt) {
    if(validarEntradas()){
        crearProveedor();
    }
}
```

El método validar entradas simplemente comprueba que todas las entradas del usuario no sean en blanco ni sean incorrectas, como nombres con números o emails incorrectos, también comprueba que no exista ya un CIF de una misma empresa, para evitar valores duplicados.

```
public boolean validarEntradas(){
    if(jTextFieldApellidos.getText().isEmpty() || jTextFieldCIF.getText().isEmpty() || jTextFieldDireccion.getText().isEmpty() || jTextFieldMailContacto.getText().isEmpty
            || jTextFieldMailEmpresa.getText().isEmpty() || jTextFieldNombreContacto.getText().isEmpty() || jTextFieldNombreEmpresa.getText().isEmpty() || jTextFieldTlfCo
            || jTextFieldTlfEmpresa.getText().isEmpty()){
        JOptionPane.showMessageDialog(this, "No puede haber campos en blanco", "Error", JOptionPane.ERROR_MESSAGE);
        return false;
    }

    if(jTextFieldNombreEmpresa.getText().matches(".*\\d.*")){
            JOptionPane.showMessageDialog(this, "El nombre de empresa introducido tiene números", "Error", JOptionPane.ERROR_MESSAGE);
            return false;
        }

    if(jTextFieldNombreContacto.getText().matches(".*\\d.*")){
            JOptionPane.showMessageDialog(this, "El nombre de contacto introducido tiene números", "Error", JOptionPane.ERROR_MESSAGE);
            return false;
        }

    if(jTextFieldApellidos.getText().matches(".*\\d.*")){
            JOptionPane.showMessageDialog(this, "Los apellidos del contacto introducido tienen números", "Error", JOptionPane.ERROR_MESSAGE);
            return false;
        }

    if(!jTextFieldMailContacto.getText().matches("^[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\\.[A-Za-z]{2,}$")){
            JOptionPane.showMessageDialog(this, "El email de contacto introducido es incorrecto", "Error", JOptionPane.ERROR_MESSAGE);
            return false;
        }

    if(!jTextFieldMailEmpresa.getText().matches("^[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\\.[A-Za-z]{2,}$")){
            JOptionPane.showMessageDialog(this, "El email de contacto introducido es incorrecto", "Error", JOptionPane.ERROR_MESSAGE);
            return false;
        }

    if(!jTextFieldTlfContacto.getText().matches(".*\\d.*")){
            JOptionPane.showMessageDialog(this, "El teléfono de contacto introducido contiene caracteres alfabéticos", "Error", JOptionPane.ERROR_MESSAGE);
            return false;
        }

    if(!jTextFieldTlfEmpresa.getText().matches(".*\\d.*")){
            JOptionPane.showMessageDialog(this, "El teléfono de empresa introducido contiene caracteres alfabéticos", "Error", JOptionPane.ERROR_MESSAGE);
            return false;
        }

    if(tab.CIFDuplicado(jTextFieldCIF.getText())){
        JOptionPane.showMessageDialog(this, "La empresa ya existe como proveedor", "Error", JOptionPane.ERROR_MESSAGE);
        return false;
```

Finalmente, el método “crearProveedor” haría lo siguiente:

```
] public String[] crearProveedor(){
      String[] proveedor = new String[9];
      proveedor[0] = jTextFieldCIF.getText();
      proveedor[1] = jTextFieldNombreEmpresa.getText();
      proveedor[2] = jTextFieldTlfEmpresa.getText();
      proveedor[3] = jTextFieldMailEmpresa.getText();
      proveedor[4] = jComboBoxLocalizacion.getSelectedItem().toString();
      proveedor[5] = jTextFieldDireccion.getText();
      proveedor[6] = jTextFieldNombreContacto.getText();
      proveedor[7] = jTextFieldTlfContacto.getText();
      proveedor[8] = jTextFieldMailContacto.getText();
      return proveedor;
-     }
```

Creamos un array de Strings de nombre proveedor y establecemos a cada uno de los valores el texto que haya introducido el usuario en ellos.

Aquí tenemos como se ve la pantalla:

Por último, llegamos a la pantalla de mostrar proveedores, que simplemente cuenta con una tabla y un botón para actualizar los proveedores

```
*/
public class TablaProveedores extends javax.swing.JFrame {
    String[] proveedor = new String[9];
    DefaultTableModel model = new DefaultTableModel();
    AltaProveedor parent;
    /**
     * Creates new form TablaProveedores
     */
    public TablaProveedores(AltaProveedor parent) {
        this.parent = parent;
        initComponents();
        setLocationRelativeTo(null);
        setResizable(false);
        model.setColumnIdentifiers(new String[]{"CIF","Empresa","Teléfono empresa","Email empresa","País empresa","Dirección empresa","Nombre contacto","Teléfono contact
        jTable1.setModel(model);
        setDefaultCloseOperation(DISPOSE_ON_CLOSE);
        this.setIconImage(new ImageIcon(getClass().getResource("/gui/imgs/Logo.png")).getImage());
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    Generated Code

    private void jButtonActualizarActionPerformed(java.awt.event.ActionEvent evt) {
        proveedor = parent.crearProveedor();
        if(!proveedor[0].isEmpty())
        model.addRow(proveedor);
        jTable1.setModel(model);
    }
```

Al igual que en el resto de clases tiene el logo corporativo establecido.

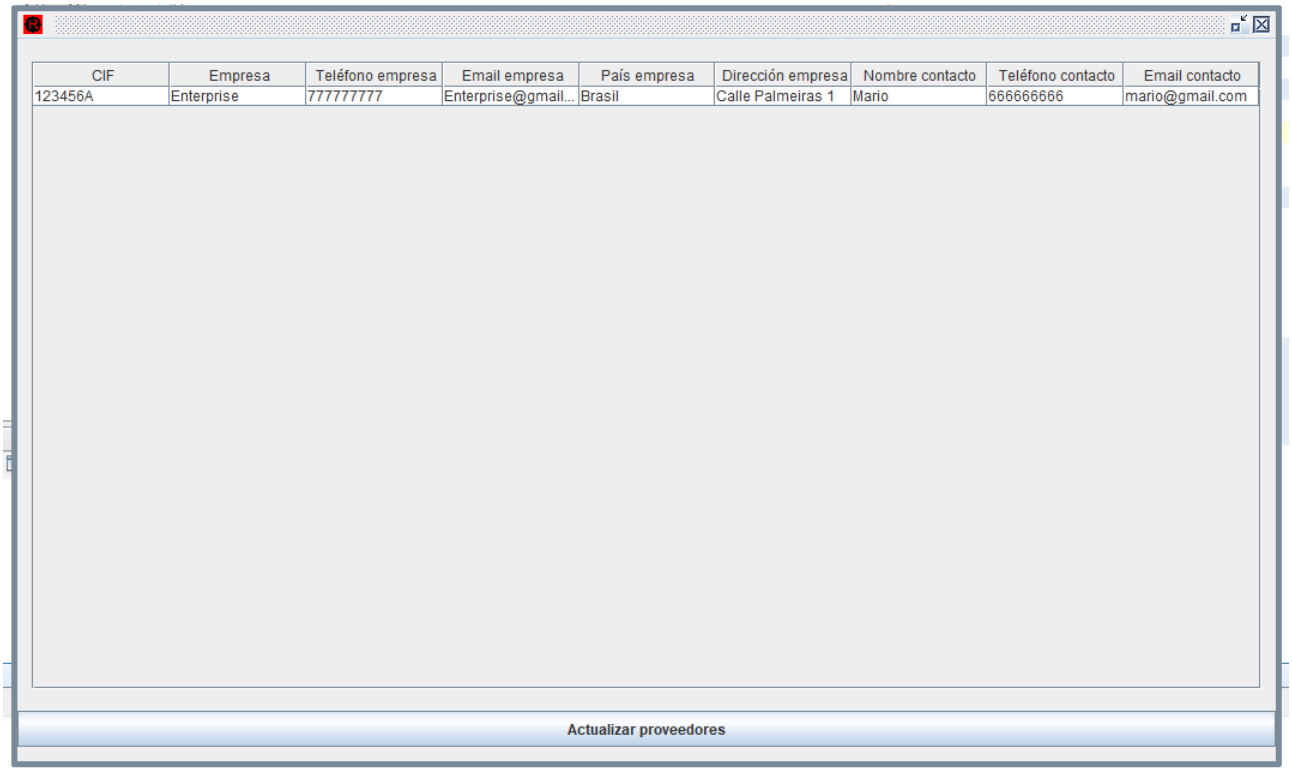
Como podemos ver creamos un modelo de tabla y le establecemos los campos que queremos introducir de los proveedores y en el botón de actualizar obtenemos el proveedor creado en la pantalla anterior, añadimos una fila al modelo de la tabla con el proveedor como dato y le ponemos el modelo a la tabla.

También contamos con el método para evitar CIF duplicados:

```
public boolean CIFDuplicado(String CIF) {
    for (int i = 0; i < jTable1.getRowCount(); i++) {
        if (jTable1.getValueAt(i, 0).toString().equalsIgnoreCase(CIF)) {
            return true;
        }
    }
    return false;
}
```

Que se ejecuta en la pantalla anterior al validar las entradas.

Aquí tenemos como se ve la pantalla:



Destacar en esta pantalla también que solo se puede actualizar una vez por cada cliente añadido, para evitar problemas de elementos duplicados al crear un proveedor, esto lo he realizado con el método

```
public void cambiarBoton(){
    jButtonActualizar.setEnabled(true);
}
```

Además de añadir la línea para establecer el botón como desactivado al actualizar la tabla:

```
private void jButtonActualizarActionPerformed(java.awt.event.ActionEvent evt) {
    proveedor = parent.crearProveedor();
    if(!proveedor[0].isEmpty())
    model.addRow(proveedor);
    jTable1.setModel(model);
    cont++;
    jButtonActualizar.setEnabled(false);
}
```

Preguntas sobre ISO

- **ISO 9241**
  1. **Eficiencia**: ¿La interfaz permite completar tareas rápidamente y con pocos errores?

     Si, la interfaz permite al usuario completar las tareas sin errores.
  2. **Efectividad**: ¿Los usuarios pueden realizar las tareas propuestas sin dificultad?

     Si, no hay dificultades para que los usuarios realicen ninguna tarea.
  3. **Satisfacción**: ¿Es agradable y clara la interacción con la interfaz?

     Sí, además tiene elementos de identidad del cliente y todo se ve claramente.

- **ISO 9126**
  1. **Facilidad de Aprendizaje**: ¿Qué tan fácil es para un nuevo usuario entender cómo funciona la interfaz?

     Sin ninguna dificultad, la interfaz es muy intuitiva.
  2. **Operabilidad**: ¿La interfaz es fácil de usar para cumplir con los objetivos del usuario?

     Sí.

Creación del logo corporativo

El jefe de la empresa Giovanni, nos ha solicitado que le creemos un logo corporativo utilizando las herramientas SVG para que sea de gran calidad, para ello lo primero que hacemos será crear un proyecto nuevo de tipo HTML5.

Una vez lo tengamos creado en el documento de nombre index crearemos el icono utilizando herramientas SVG:

```
<!DOCTYPE html>
<!--
To change this license header, choose License Headers in Project Properties.
To change this template file, choose Tools | Templates
and open the template in the editor.
-->
<html>
    <head>
        <title>LogoExamen</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>
    <body>
        <svg height="200" width="200">
        <rect x="0" y="0" height="200" width="200" fill="red"/>
        <circle cx="100" cy="100" r="80" stroke="black" stroke-width="6" fill="black"/>
        <text x="54"  y="144" fill="red" style="font-size: 140px" font-family="arial">R</text>
        </svg>
    </body>
</html>
```

Primero creamos un cuadrado rojo para el fondo del logo con la herramienta rect, después creamos el circulo con la herramienta circle y definimos su posición, su radio y lo rellenamos de negro.

Finalmente, la R la creamos con la herramienta text, establecemos su posición y tamaño y la rellenamos de rojo.

El logo se ve así: