

2022

ALVAODOO

ALVAODOO, BASE DE DATOS GRÁFICA
ADRIÁN DE LA FUENTE

ÍNDICE

ÍNDICE.....	1
DIFINICIÓN DE PROYECTO	2
INTRODUCCIÓN Y JUSTIFICACION	2
OBJETIVOS DE LA APLICACIÓN	2
DESCRIPCION DE PROCESOS	2
REQUISITOS DEL PROGRAMA.....	2
TECNOLOGIAS, FRAMEWORKS, ESTANDARES, PAQUETES Y LIBRERIAS UTILIZADAS	3
DIAGRAMAS, BOCETOS Y ESQUEMAS	4
BOCETOS	4
DIAGRAMA DE E-R.....	4
ORGANIZACIÓN DEL PROYECTO	5
CARPETAS	5
VENTANAS.....	6
FICHEROS.....	8
CODIGO: CLASES Y CODIGO DE LA APLICACIÓN.....	8
BD UTILS.....	8
LOGIN	9
REGISTRO.....	11
AÑADIR PROVEEDOR.....	11
AÑADIR PRODUCTO.....	12
MOSTRAR PRODUCTO	13
MOSTRAR PROVEEDOR.....	15
METODOS COMUNES.....	15

DIFINICIÓN DE PROYECTO

INTRODUCCIÓN Y JUSTIFICACION

Alvaodoo es una pequeña aplicación que permite el almacenado de datos de productos y sus respectivos proveedores de una forma cómoda, rápida e intuitiva. Además, también permite visualizar cómodamente los productos con sus fotografías, además de listarlos y eliminarlos fácilmente.

OBJETIVOS DE LA APLICACIÓN

El objetivo de esta aplicación es hacer que las personas que tengan que recurrir a las bases de datos tengan una mayor facilidad para trabajar sobre ellas. No todas las personas son capaces de introducir datos en una base de datos rápidamente y sin equivocarse, y esto es lo que consigue Alvaodoo.

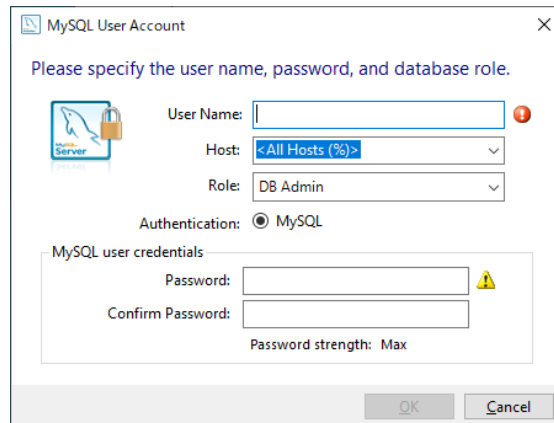
DESCRIPCION DE PROCESOS

REQUISITOS DEL PROGRAMA

El programa necesita una base de datos MySQL instalada además de los correspondientes conectores para .NET

MySQL Server	8.0.28	X64
Connector/.NET	8.0.28	X86

Esta base de datos debe tener un usuario llamado "dela" y con contraseña "abc123." para poder conectarse. Este deberá ser creado en el proceso de instalación de la base de datos o una vez creada la base de datos, aunque para evitar problemas se recomienda mientras instalamos la base de datos en nuestro equipo.



Para la creación de la base de datos es dado un script que contiene la creación de la base de datos, así como el añadido de datos de prueba para tener como referencia la primera vez que se abre la aplicación.

En caso de no querer los usuarios por defecto, deberemos borrarlos del script o no ejecutarlo. En caso de no añadirlos, será indispensable registrar un nuevo usuario nada más abrir la aplicación, ya que sin él no podremos acceder a ella.

TECNOLOGIAS, FRAMEWORKS, ESTANDARES, PAQUETES Y LIBRERIAS UTILIZADAS

La aplicación esta creada sobre el entorno de desarrollo Microsoft Visual Studio 2022 y 2019.

Para la creación de la aplicación se ha necesitado instalar la extensión de "MySql.Data.MySqlClient" para VS2019. Con ella tenemos acceso a crear conexiones con la base de datos de MySQL. Además de esta, hemos tenido que instalar también la extensión "Microsoft RDLC Report Designer" para la creación de los informes de la aplicación. Estas son las dos principales extensiones usadas en la base de datos.

El lenguaje de la aplicación está basado en C#, XAML y MySQL. Son los 3 lenguajes usados en la aplicación para el código (C#), la interfaz (XAML) y la base de datos (MySQL).

DIAGRAMAS, BOCETOS Y ESQUEMAS

BOCETOS

Primeros bocetos e ideas de cómo podría ser la aplicación.

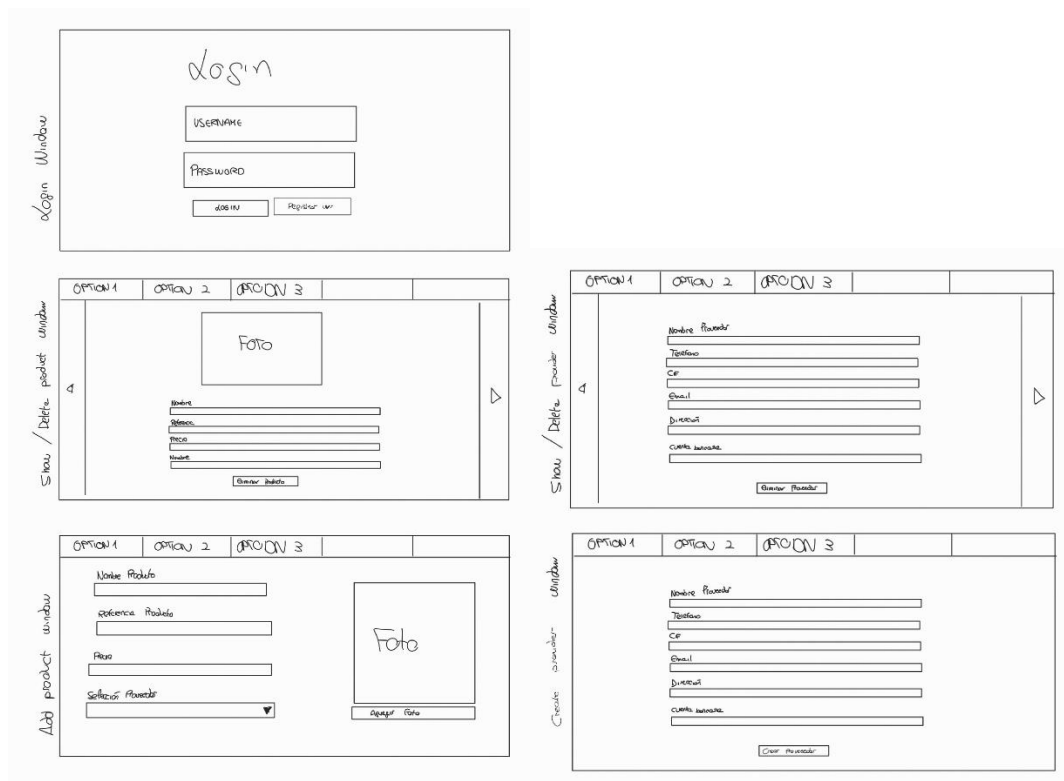
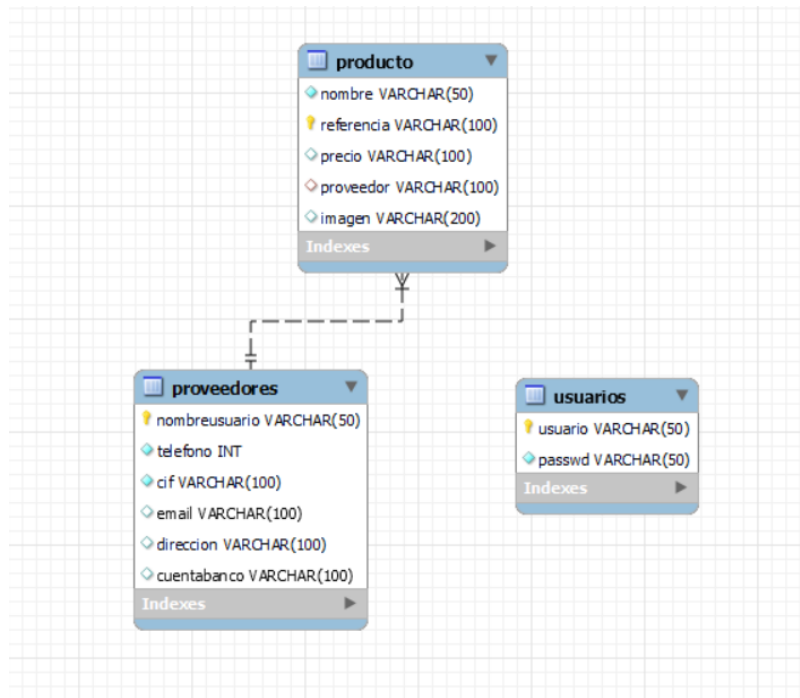


DIAGRAMA DE E-R

La base de datos de la aplicación tiene el siguiente modelo entidad – relación:



ORGANIZACIÓN DEL PROYECTO

CARPETAS

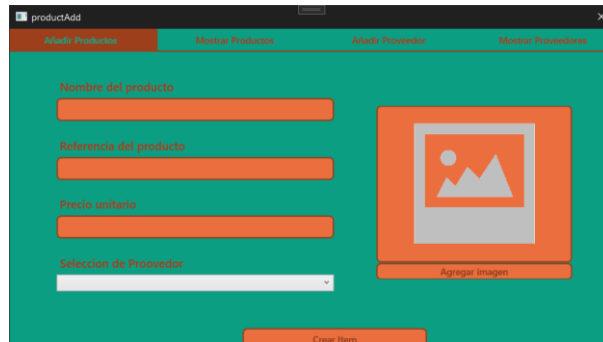
La aplicación en cuestión tiene varias carpetas que se utilizan para almacenar distintos tipos de datos. Las carpetas que hay en el proyecto son las siguientes:

- Ayuda:
 - La carpeta donde se almacenan la ayuda de la aplicación. En este caso, un PDF que se abrirá cuando pulsemos F1.
- ImageProduct:
 - La carpeta donde se almacenan las fotos de los ítems que agreguemos a la base de datos.
- Resources:
 - La carpeta donde se almacenarán recursos que se utilizan varias veces, como sería la imagen por defecto de los ítems que no tienen una imagen asignada

VENTANAS

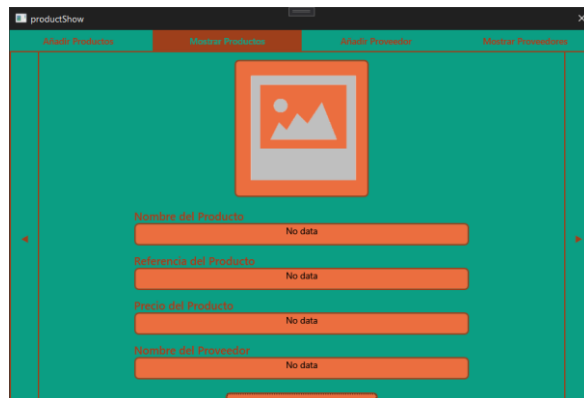
La aplicación consta de 7 ventanas, entre las cuales se encuentran las de funcionalidad de la aplicación, la de registro y login y el formulario que podemos ver cuando pulsamos F2.

- ProductAdd



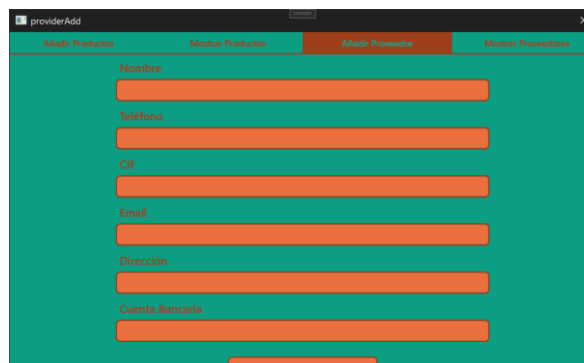
The screenshot shows the 'productAdd' window with a teal background and a dark header. The header contains four tabs: 'Añadir Productos', 'Mostrar Productos', 'Añadir Proveedores', and 'Mostrar Proveedores'. The 'Añadir Productos' tab is active. The main area contains four orange input fields labeled 'Nombre del producto', 'Referencia del producto', 'Precio unitario', and 'Selección de Proveedor'. To the right of these fields is a placeholder for an image with a mountain icon and a button labeled 'Agregar imagen'. At the bottom center is a button labeled 'Crear Item'.

- ProductShow



The screenshot shows the 'productShow' window with a teal background and a dark header. The header contains four tabs: 'Añadir Productos', 'Mostrar Productos', 'Añadir Proveedores', and 'Mostrar Proveedores'. The 'Mostrar Productos' tab is active. The main area features a large image placeholder with a mountain icon at the top. Below it are four orange input fields labeled 'Nombre del Producto', 'Referencia del Producto', 'Precio del Producto', and 'Nombre del Proveedor'. Each field contains the text 'No data'. At the bottom center is a button labeled 'Actualizar Item'.

- ProviderAdd



The screenshot shows the 'providerAdd' window with a teal background and a dark header. The header contains four tabs: 'Añadir Productos', 'Mostrar Productos', 'Añadir Proveedores', and 'Mostrar Proveedores'. The 'Añadir Proveedores' tab is active. The main area contains seven orange input fields labeled 'Nombre', 'Telefono', 'C/P', 'Email', 'Direccion', and 'Cuenta Bancaria'. At the bottom center is a button labeled 'Crear proveedor'.

- ProviderShow

- Login

- Register

- Form

FICHEROS

Además de las ventanas y de las carpetas ya mostradas, también hay más archivos que se utilizan en la aplicación:

- alvaodooDataSet:
 - Este DataSet es el encargado de recopilar los datos desde la BD para poder asignarlos a las tablas que están dentro del Report, y así poder mostrar todos los datos.
- Report:
 - Parte grafica de la aplicación donde asignamos los valores del DataSet, y a su vez este Report se integra dentro del informe, donde podremos visualizar todos los datos de la BD
- Styles:
 - Parte de la aplicación donde se configuran los estilos de esta. Aquí se almacena todo lo que tenga relación con los estilos.
- BDUtils:
 - Clase donde se realiza la conexión a la base de datos, además de donde se recuperan los datos de la BD para almacenarlos en el DataTable correspondiente.

CÓDIGO: CLASES Y CÓDIGO DE LA APLICACIÓN

BD UTILS

Método que sirve para crear una conexión con la BD y que devuelve un parámetro de tipo conexión

```
public MySqlConnection ConexionDB()
{
    String server = "127.0.0.1";
    String database = "alvaodoo";
    String uid = "dela";
    String password = "abc123.";
    //string connectionString;
    connectionString = "SERVER=" + server + "; PORT = 3306 ;" + "DATABASE=" + database + ";" + "UID=" + uid + ";" + "PASSWORD=" + password + ";";
    MySqlConnection mycon = new MySqlConnection(connectionString);
    return mycon;
}
```

Método que sirve para hacer un llenado de un DataTable mediante una sentencia SQL que le pasamos anteriormente

```
public DataTable RellenarDt(string sentenciaSql, MySqlConnection con)
{
    using (MySQLCommand cmd = new MySQLCommand(sentenciaSql, con))
    {
        cmd.CommandType = CommandType.Text;
        using (MySQLDataAdapter sda = new MySQLDataAdapter(cmd))
        {
            using (DataTable dt = new DataTable())
            {
                sda.Fill(dt);
                return dt;
            }
        }
    }
}
```

LOGIN

Este método crea una conexión con la BD para recuperar la contraseña del usuario que está intentando iniciar sesión. Si el valor coincide, el usuario podrá iniciar sesión y acceder a la aplicación, en caso de que sea incorrecto, devolverá un mensaje de error.

```
private void Login(object sender, RoutedEventArgs e)
{
    var cmd = new MySQLCommand();
    cmd.Connection = conn;

    String query = "SELECT passwd FROM USUARIOS WHERE usuario = '\" + usernameTxt.Text.ToString() + '\"";

    MySQLCommand mycommand = new MySQLCommand(query, conn);
    MySQLDataReader myreader = mycommand.ExecuteReader();
    myreader.Read();

    String strnombbre = myreader["passwd"].ToString();

    if (passwdTxt.Password == strnombbre)
    {
        productShow win2 = new productShow();
        win2.Show();
        this.Close();
    }
    else
    {
        MessageBox.Show("Error en el Login");
    }

    myreader.Close();
}
```

Este método manda abrir una nueva ventana (Registro) desde la que poder registrar un nuevo usuario en la BD

```
1 referencia
private void Register(object sender, RoutedEventArgs e)
{
    register reg = new register();
    reg.Show();
    this.Close();
}
```

REGISTRO

Este método permite registrar un nuevo usuario en la base de datos. Para ello primero compara los dos campos de contraseña para ver que tengan el mismo valor. Si lo tiene, entonces intentará hacer el añadido a la BD, siempre y cuando el usuario no exista. En caso de existir, emitirá un error.

```
private void RegistrarUsuario(object sender, RoutedEventArgs e)
{
    if (passw1.Password != passw2.Password)
    {
        MessageBox.Show("Las contraseñas no coinciden");
    }

    else
    {
        conn = bd.ConexionDB();

        conn.Open();

        var cmd = new MySqlCommand();
        cmd.Connection = conn;

        MySqlCommand comm = conn.CreateCommand();
        cmd.CommandText = "INSERT INTO USUARIOS (usuario, passwd) VALUES (?usuario, ?passwd)";

        cmd.Parameters.Add("?usuario", MySqlDbType.VarChar).Value = username.Text.ToString();
        cmd.Parameters.Add("?passwd", MySqlDbType.VarChar).Value = passw1.Password;

        try
        {
            cmd.ExecuteNonQuery();

            MessageBox.Show("El usuario " + username.Text.ToString() + " se agregó correctamente.", "Acción Completada");
        }
        catch (Exception)
        {
            MessageBox.Show("El usuario ya existe, intentelo con otro nombre");
        }
    }
}
```

AÑADIR PROVEEDOR

Este método permite añadir proveedores a la BD. Primero, comprueba que los campos requeridos están cubiertos. En el caso de no estarlos dará un error. En caso de estar todo bien, realizará una conexión con la BD y añadirá el proveedor con los valores de los campos de la ventana. En caso de error, devolverá un mensaje.

```

private void CreateProvider(object sender, RoutedEventArgs e)
{
    if (nombreproveedor.Text == "" || telefonoproveedor.Text == "" || cifproveedor.Text == "")
    {
        MessageBox.Show("El campo Nombre, Telefono y CIF son OBLIGATORIOS", "ERROR");
    }
    else
    {
        try
        {
            conn = bd.ConexionDB();
            conn.Open();

            var cmd = new MySqlCommand();
            cmd.Connection = conn;

            MySqlCommand comm = conn.CreateCommand();
            cmd.CommandText = "INSERT INTO PROVEEDORES VALUES (?nombre, ?telefono, ?cif, ?email, ?direccion, ?cuenta)";

            cmd.Parameters.Add("?nombre", MySqlDbType.VarChar).Value = nombreproveedor.Text.ToString();
            cmd.Parameters.Add("?telefono", MySqlDbType.Int32).Value = telefonoproveedor.Text.ToString();
            cmd.Parameters.Add("?cif", MySqlDbType.VarChar).Value = cifproveedor.Text.ToString();
            cmd.Parameters.Add("?email", MySqlDbType.VarChar).Value = emailproveedor.Text.ToString();
            cmd.Parameters.Add("?direccion", MySqlDbType.VarChar).Value = direccionproveedor.Text.ToString();
            cmd.Parameters.Add("?cuenta", MySqlDbType.VarChar).Value = bancoproveedor.Text.ToString();

            cmd.ExecuteNonQuery();

            MessageBox.Show("El usuario " + nombreproveedor.Text.ToString() + " se agregó correctamente", "Acción Completada");

            nombreproveedor.Text = "";
            telefonoproveedor.Text = "";
            cifproveedor.Text = "";
            emailproveedor.Text = "";
            direccionproveedor.Text = "";
            bancoproveedor.Text = "";
        }
        catch (MySqlException ex)
        {
            MessageBox.Show(ex.ToString(), "Error");
        }
    }
}

```

AÑADIR PRODUCTO

Este método se ejecuta cuando intentas crear un producto. En caso de que los campos requeridos estén cubiertos, el programa seguirá adelante.

Lo que hará será recopilar los datos de los campos de la ventana de añadido de producto y hacer una consulta y añadir el ítem.

Una vez se haya añadido, los valores volverán al valor por defecto.

```

private void createItem_Click(object sender, RoutedEventArgs e)
{
    if (nombreproducto.Text == "" || referenciaproducto.Text == "" || proveedorCB.Text == "")
    {
        MessageBox.Show("El campo Nombre, Referencia y Proveedor son OBLIGATORIOS", "ERROR");
    }
    else
    {
        try
        {
            conn = bd.ConexionDB();
            conn.Open();

            var cmd = new MySqlCommand();
            cmd.Connection = conn;

            MySqlCommand comm = conn.CreateCommand();
            cmd.CommandText = "INSERT INTO PRODUCTO (nombre, referencia, precio, proveedor, imagen) VALUES (?nombre, ?referencia, ?precio, ?proveedor, ?imagen)";

            cmd.Parameters.Add("?nombre", MySqlDbType.VarChar).Value = nombreproducto.Text.ToString();
            cmd.Parameters.Add("?referencia", MySqlDbType.VarChar).Value = referenciaproducto.Text.ToString();
            cmd.Parameters.Add("?precio", MySqlDbType.VarChar).Value = precioproducto.Text.ToString();
            cmd.Parameters.Add("?proveedor", MySqlDbType.VarChar).Value = proveedorCB.Text.ToString();
            cmd.Parameters.Add("?imagen", MySqlDbType.VarChar).Value = nombre;

            MessageBox.Show("El producto " + nombreproducto.Text.ToString() + " se agregó correctamente.", "Acción Completada");

            nombreproducto.Text = "";
            referenciaproducto.Text = "";
            precioproducto.Text = "";
            proveedorCB.Text = "";

            String baseImage = System.IO.Path.Combine(System.IO.Path.Combine(System.AppDomain.CurrentDomain.BaseDirectory, "..\\..\\resources"), "base.png");

            BitmapImage myBitmapImage = new BitmapImage();
            myBitmapImage.BeginInit();
            myBitmapImage.UriSource = new Uri(baseImage);
            myBitmapImage.DecodePixelWidth = 288;
            myBitmapImage.EndInit();

```

Este método permite generar los ítems que aparecerán en la lista de proveedores. Para ello realiza una conexión a la base de datos y lista todos los proveedores disponibles.

```
private void generarItems()
{
    var cmd = new MySqlCommand();
    cmd.Connection = conn;

    MySqlCommand comm = conn.CreateCommand();
    cmd.CommandText = "SELECT nombreusuario FROM proveedores";

    cmd.ExecuteNonQuery();

    MySqlDataReader reader = cmd.ExecuteReader();
    while (reader.Read())
    {
        proveedorCB.Items.Add(reader["nombreusuario"].ToString());
    }
}
```

Este método es el que permite añadir imágenes y que se puedan visualizar en la aplicación. Genera una ventana desde la que seleccionar la imagen y luego la mostrará.

```
private void addImage(object sender, RoutedEventArgs e)
{
    dialog = new OpenFileDialog();
    dialog.Filter = "JPG images (*.jpg)|*.jpg|PNG images (*.png)|*.png|JPEG images (*.jpeg)|*.jpeg";

    if (dialog.ShowDialog() == true)
    {
        filePath = dialog.FileName;
        nombre = System.IO.Path.GetFileName(filePath);

        BitmapImage myBitmapImage = new BitmapImage();
        myBitmapImage.BeginInit();
        myBitmapImage.UriSource = new Uri(filePath);
        myBitmapImage.DecodePixelWidth = 200;
        myBitmapImage.EndInit();

        imagenProducto.Source = myBitmapImage;

        var fileStream = dialog.OpenFile();
    }
}
```

MOSTRAR PRODUCTO

Este método carga el contenido que se mostrará en la ventana de muestra de productos. En el caso de que no hubiese ningún producto, esta mostrar "No data" en los campos.

```

private void CargarContenido(DataTable dt)
{
    if (dt.Rows.Count > 0) {
        itemName.Text = dt.Rows[0].Field<String>("nombre");
        refProduct.Text = dt.Rows[0].Field<String>("referencia");
        precioProduct.Text = dt.Rows[0].Field<String>("precio");
        proveName.Text = dt.Rows[0].Field<String>("proveedor");

        String imageName = dt.Rows[0].Field<String>("imagen");
        String baseImage;

        if (imageName != null)
        {
            baseImage = System.IO.Path.Combine(System.IO.Path.Combine(System.AppDomain.CurrentDomain.BaseDirectory, "..\\..\\ImageProduct"), imageName);
        }
        else
        {
            baseImage = System.IO.Path.Combine(System.IO.Path.Combine(System.AppDomain.CurrentDomain.BaseDirectory, "..\\..\\resources"), "base.png");
        }

        BitmapImage myBitmapImage = new BitmapImage();
        myBitmapImage.BeginInit();
        myBitmapImage.CacheOption = BitmapCacheOption.OnLoad;
        myBitmapImage.UriSource = new Uri(baseImage);
        myBitmapImage.DecodePixelWidth = 200;
        myBitmapImage.EndInit();

        imageProduct.Source = myBitmapImage;
    }
    else
    {
        itemName.Text = "No data";
        refProduct.Text = "No data";
        precioProduct.Text = "No data";
        proveName.Text = "No data";
    }
}

```

Este método permite eliminar un elemento desde la ventana de listado de ítems. Si clicamos en el botón nos mostrará un mensaje pidiéndonos confirmación. Si lo hacemos, entonces el elemento se eliminará. Después se recargará la lista de ítems.

```

private void deleteItemShow(object sender, RoutedEventArgs e)
{
    if (dt.Rows.Count > 0)
    {
        MessageBoxResult result = MessageBox.Show("¿Desea realmente eliminar el producto mostrado?", "Eliminación de producto", MessageBoxButton.YesNo);
        switch (result)
        {
            case MessageBoxResult.Yes:
                try
                {
                    String imagenProducto = "";
                    String imageName = dt.Rows[0].Field<String>("imagen");
                    if (imageName != null)
                    {
                        imagenProducto = System.IO.Path.Combine(System.IO.Path.Combine(System.AppDomain.CurrentDomain.BaseDirectory, "..\\..\\ImageProduct"), imageName);
                    }

                    String itemDelete = (dt.Rows[0].Field<String>("referencia"));

                    conn.Open();
                    var cmd = new MySqlCommand();
                    cmd.Connection = conn;

                    MySqlCommand comm = conn.CreateCommand();

                    cmd.CommandText = "DELETE FROM PRODUCTO WHERE referencia = ?nombre";
                    cmd.Parameters.Add("nombre", MySqlDbType.VarChar).Value = itemDelete;

                    try
                    {
                        cmd.ExecuteNonQuery();

                        itemNumber = 0;
                        dt.Clear();
                        RellenarTabla();
                        CargarContenido(dt);
                    }
                    catch { }
                }
                catch { }
            case MessageBoxResult.No:
                // No action
                break;
        }
    }
}

```

MOSTRAR PROVEEDOR

Este método carga el contenido que se mostrará en la ventana de muestra de proveedores. En el caso de que no hubiese ningún proveedor, esta mostrar "No data" en los campos.

```
private void CargarContenido(DataTable dt)
{
    if (dt.Rows.Count > 0)
    {
        providerName.Text = dt.Rows[itemNumber].Field<string>("nombreusuario");
        phoneNumber.Text = dt.Rows[itemNumber].Field<int32>("telefono").ToString();
        cif.Text = dt.Rows[itemNumber].Field<string>("cif");
        email.Text = dt.Rows[itemNumber].Field<string>("email");
        address.Text = dt.Rows[itemNumber].Field<string>("direccion");
        backaccount.Text = dt.Rows[itemNumber].Field<string>("cuentabanco");
    }
    else
    {
        providerName.Text = "No data";
        phoneNumber.Text = "No data";
        cif.Text = "No data";
        email.Text = "No data";
        address.Text = "No data";
        backaccount.Text = "No data";
    }
}
```

Este método permite eliminar un proveedor desde la ventana de listado de proveedores. Si clicamos en el botón nos mostrará un mensaje pidiéndonos confirmación. Si lo hacemos, entonces el elemento se eliminará. Después se recargará la lista de proveedores.

```
private void deleteItemInShow(object sender, RoutedEventArgs e)
{
    if (dt.Rows.Count > 0)
    {
        MessageBoxResult result = MessageBox.Show("¿Desea realmente eliminar el proveedor mostrado?", "Eliminación del proveedor", MessageBoxButton.YesNo);
        switch (result)
        {
            case MessageBoxResult.Yes:
            {
                try
                {
                    string proveedorDelete = dt.Rows[itemNumber].Field<string>("cif");

                    conn.Open();

                    var cmd = new MySqlCommand();
                    cmd.Connection = conn;

                    MySqlCommand comm = conn.CreateCommand();

                    cmd.CommandText = "DELETE FROM PROVEEDORES WHERE cif = ?cif";
                    cmd.Parameters.Add("?cif", MySqlDbType.VarChar).Value = proveedorDelete;

                    try
                    {
                        cmd.ExecuteNonQuery();

                        itemNumber = 0;

                        dt.Clear();
                        ReloadarTabla();
                        CargarContenido(dt);

                        conn.Close();
                    }
                    catch (MySqlException ex)
                    {
                        MessageBox.Show("No se puede eliminar un proveedor que tenga productos relacionados. Borre primero los productos relacionados.");
                    }
                }
            }
        }
    }
}
```

METODOS COMUNES

Métodos encargados de generar las ventanas para los botones del menú.


```

private void openProviderAdd(object sender, RoutedEventArgs e)
{
    providerAdd providerAdd = new providerAdd();
    providerAdd.Show();
    this.Close();
}

1 referencia
private void openProviderShow(object sender, RoutedEventArgs e)
{
    providerShow providerShow = new providerShow();
    providerShow.Show();
    this.Close();
}

1 referencia
private void openProductAdd(object sender, RoutedEventArgs e)
{
    productAdd productAdd = new productAdd();
    productAdd.Show();
    this.Close();
}

1 referencia
private void openProductShow(object sender, RoutedEventArgs e)
{
    productShow productShow = new productShow();
    productShow.Show();
    this.Close();
}

```

Método encargado de hacer que se genere el informe o la ayuda, dependiendo de la tecla clicada.

```

1 referencia
private void Window_KeyUp(object sender, KeyEventArgs e)
{
    if (e.Key == Key.F2)
    {
        Form1 form = new Form1();
        form.Show();
    }
}

```