****

**Área del Conocimiento Ciencia, Tecnología y Ambiente**

**Ingeniería en Sistemas de Información**

**Administración de base de datos**

# **Documentación del Proyecto**

**Equipo de Desarrollo:**

Octavio Guevara

Martin Majewsky

Carlos Mendoza

Jan Carlos Prado

**Profesora:**

Irene Martinez

**Abril 2024**

## ***Introduccion***

El proyecto de desarrollo de un Sistema de Gestión de Inventario para una Tienda de Ropa tiene como objetivo principal proporcionar una aplicación en Escritorio eficiente y fácil de usar para realizar un seguimiento detallado de los productos, proveedores, compras y ventas. El sistema permitirá una gestión óptima del inventario, facilitando la toma de decisiones y mejorando la eficiencia operativa de la tienda.

Para garantizar una experiencia de usuario satisfactoria, se ha desarrollado una interfaz intuitiva y amigable que permitirá al personal de la tienda interactuar de manera efectiva con la base de datos. Esto facilitará la inserción, modificación y consulta de datos, optimizando así los procesos de gestión de inventario.

## 

## 

## 

## 

## 

## 

## 

## 

## 

## ***Tecnología Usada***

**Lenguaje de Programación**: C#

**Entorno de Desarrollo Integrado (IDE)**: Visual Studio 2020

**Base de Datos**: Microsoft SQL Server 2017

**Framework**: .NET Framework 4.7.2

**Librería Gráfica**: Windows Presentation Foundation (WPF)

**Control de Versionamiento**: Git

## 

## 

## 

## 

## 

## 

## ***Arquitectura***

El Patrón de Arquitectura usado fue el modelo de “N capas”, también conocido como arquitectura de capas, que es un diseño comúnmente utilizado en el desarrollo de software para organizar y separar las diferentes partes de una aplicación. Este modelo divide la aplicación en capas distintas, cada una con una responsabilidad específica. Las tres capas fundamentales de este modelo son:

* **Capa de Entidades**
* **Capa de Datos**
* **Capa de Presentación**

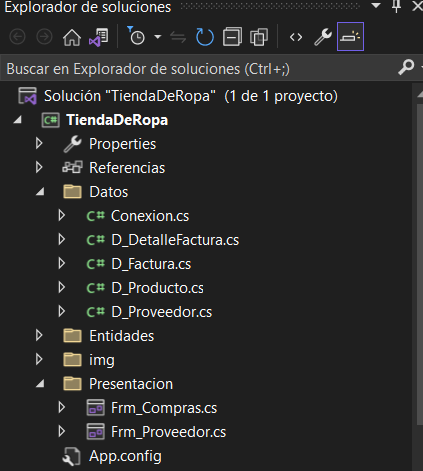


Diagrama 1. Estructura del proyecto visualizado en el IDE.

**Capa de Entidades**

Dentro de la capa de entidades se encuentran declarados los objetos a usar dentro de las transacciones del programa.

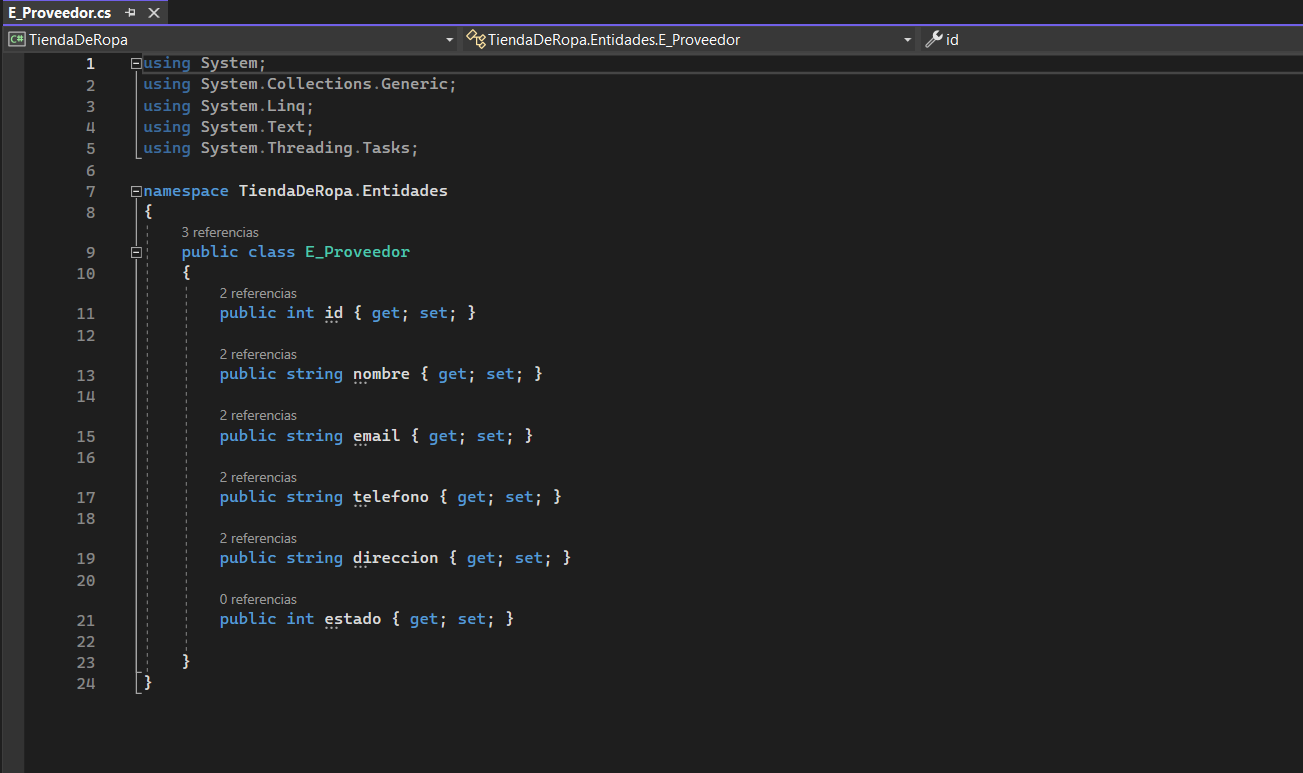


Diagrama 2. Ejemplo de Clase Proveedores en la capa de entidades, encargada del manejo de la tabla Proveedor.

**Capa de Datos**

La capa de datos se encarga del intercambio de datos entre la aplicación y la base de datos.

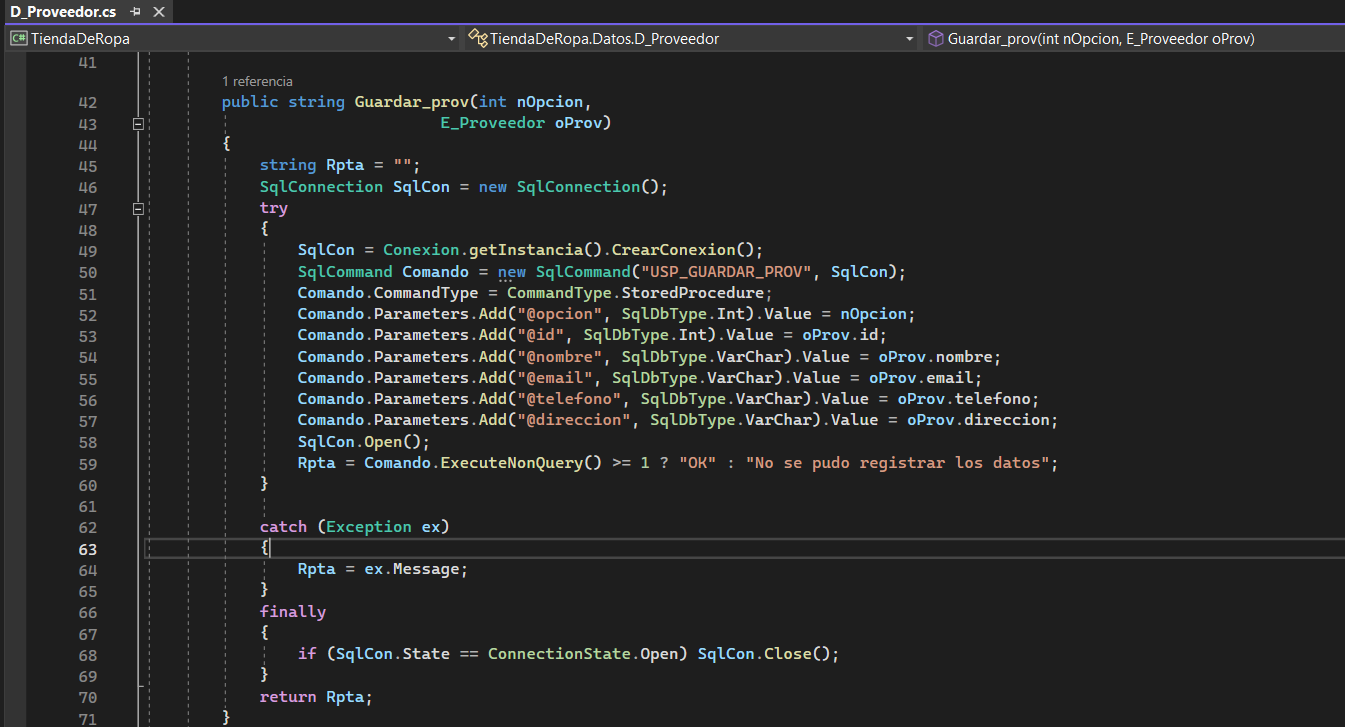


Diagrama 3. Clase en capa de datos encargada de interactuar con los datos de la tabla Proveedor.

Esta capa realiza los métodos para las operaciones CRUD (Crear, Leer, Actualizar, Eliminar) usando procedimientos (Stored Procedures) en la base de datos. Las clases en esta capa encapsulan toda la lógica relacionada a la manipulación y transmisión de datos, permitiendo que la aplicación pueda acceder a los datos guardados.

**Capa de Presentación**

Esta capa se encarga de la interacción con el usuario y de la presentación de la información. Maneja la interfaz de usuario donde se le presenta la información al usuario y funciona como el punto de interacción entre el usuario y el sistema.

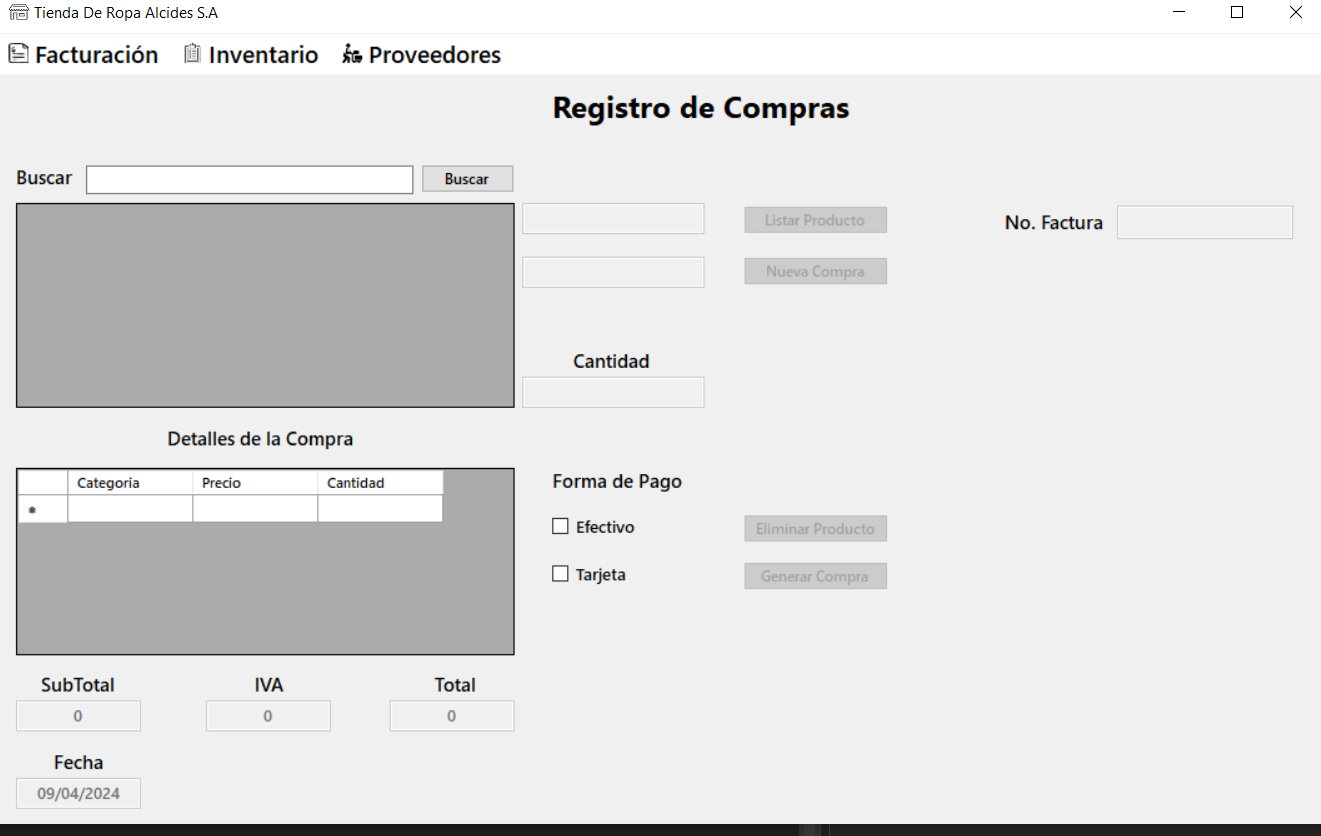


Diagrama 4. Interfaz de Usuario del apartado Proveedores.

**Interacciones entre Componentes**

Cuando un empleado de la tienda utiliza la interfaz de usuario para agregar, modificar, leer o eliminar, por ejemplo, un producto del inventario, el formulario de la interfaz de usuario (en la Capa de Presentación) captura los detalles del producto y llama a un método en la capa de Datos correspondiente

Los productos son validados, luego se realiza cualquier procesamiento adicional necesario y llama a los métodos de acceso a datos en la capa de Datos (Datos) para guardar (dependiendo del caso) el producto en la base de datos.

Los diferentes atributos como el nombre o estado provienen de una clase entidad de la capa Entidad, son usadas por la capa de Datos.

Después de que los datos se guardan en la base de datos, la interfaz de usuario se actualiza para reflejar los cambios y mostrar al usuario el resultado de la operación.

## 

## ***Base de Datos***

### **Diagrama de la Base de Datos**

La base de datos permite gestionar las compras, ventas, el inventario, los detalles de los productos, los proveedores, las facturas en la tienda de ropa. Cada tabla tiene un propósito específico y está interconectada para mantener la integridad de los datos.

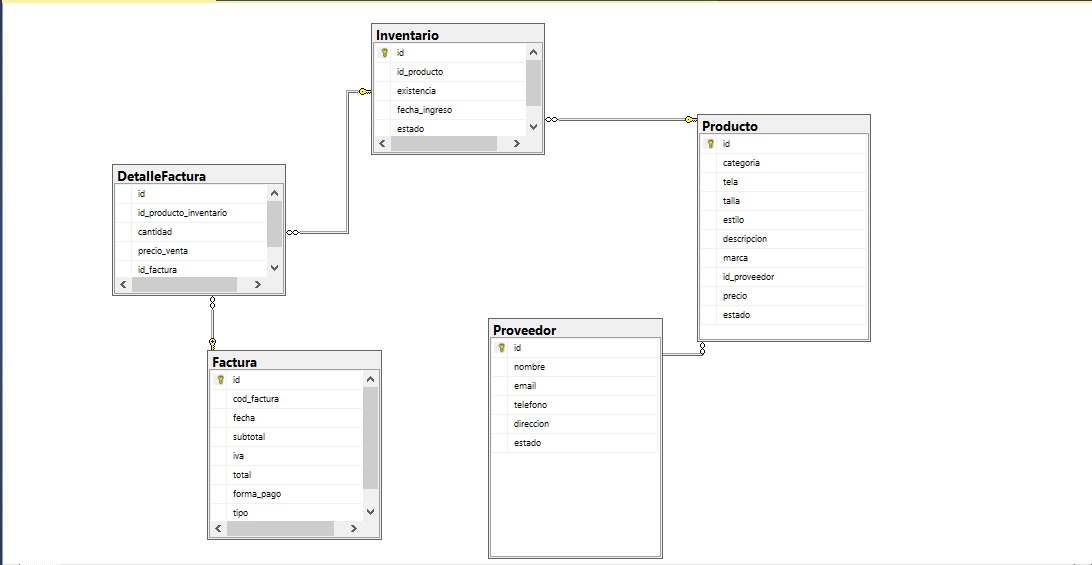


Diagrama 5. Diagrama de Base de Datos

### **Esquema de la Base de Datos**

| ***Tabla*** | Proveedor |
| --- | --- |
| ***Descripción*** | Almacena los distintos proveedores de la tienda |

| **Parametro** | **Descripción** |
| --- | --- |
| id | Identificador único del proveedor. |
| nombre | Nombre de la compañía o de la persona que provee. |
| email | Correo del proveedor. |
| telefono | Teléfono del proveedor. |
| direccion | Dirección del proveedor. |
| estado | Estado del proveedor (activo o inactivo). |

### 

| ***Tabla*** | Inventario |
| --- | --- |
| ***Descripción*** | Mantiene un registro de los productos disponibles en el inventario. |

| **Parametro** | **Descripción** |
| --- | --- |
| id | Identificador único del registro en inventario. |
| id\_producto | Referencia al producto específico. |
| existencia | Cantidad actual de unidades en stock. |
| fecha\_ingreso | Fecha en que se ingresó el producto al inventario. |
| estado | Estado del producto (disponible, agotado). |

### 

| ***Tabla*** | Producto |
| --- | --- |
| ***Descripción*** | Describe los productos (en inventario o no) en la tienda. |

| **Parametro** | **Descripción** |
| --- | --- |
| id | Identificador único del registro del producto. |
| categoría | Atributos del producto. |
| tela | Atributos del producto. |
| talla | Atributos del producto. |
| estilo | Atributos del producto. |
| descripción | Descripción adicional del producto (opcional). |
| marca | Marca del producto. |
| id\_proveedor | Referencia al proveedor del producto. |
| precio | Precio del producto unitario. |
| estado | Estado del producto (por ejemplo, disponible, agotado). |

### 

| ***Tabla*** | Factura |
| --- | --- |
| ***Descripción*** | Registra información general sobre las facturas generadas. |

| **Parametro** | **Descripción** |
| --- | --- |
| id | Identificador único de la factura. |
| cod\_factura | Código único de la factura. |
| fecha | Fecha de emisión de la factura. |
| subtotal | Montos relacionados con la factura. |
| iva | Montos relacionados con la factura. |
| total | Montos relacionados con la factura. |
| forma\_pago | Método de pago utilizado. |
| tipo | Tipo de factura (por ejemplo, venta o devolución). |

| ***Tabla*** | DetalleFactura |
| --- | --- |
| ***Descripción*** | Almacena los detalles de cada producto en una factura. |

| **Parametro** | **Descripción** |
| --- | --- |
| id | Identificador único para cada detalle. |
| id\_producto\_inventario | Referencia al producto en el inventario. |
| cantidad | Cantidad de unidades del producto vendidas. |
| precio\_venta | Precio de venta del producto. |
| id\_factura | Referencia a la factura a la que pertenece este detalle. |

**Procedimientos Almacenados**

Los procedimientos almacenados desarrollados siguen una estructura en común. Se encargan de realizar el CRUD dentro de la base de datos. Los siguientes ejemplos se muestran para la entidad Proveedor.

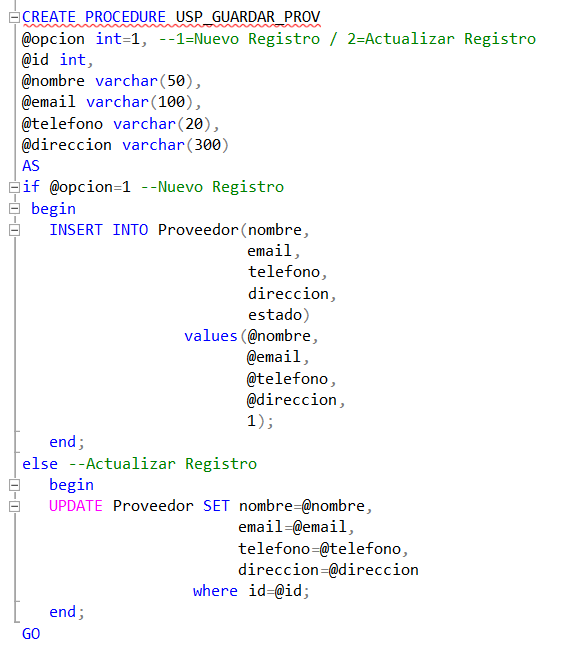


USP\_LISTADO\_{ENTIDAD}

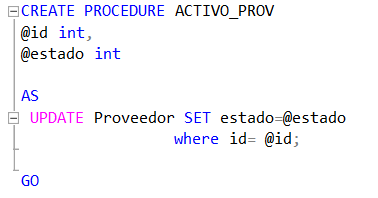
Los procedimientos almacenados LISTADO devuelven un listado de su entidad correspondiente, filtrado por un texto de búsqueda proporcionado como parámetro (@cTexto**). Si no se proporciona ningún texto de búsqueda, se devolverán todos los proveedores activos.**

**Utiliza una cláusula** WHERE **para filtrar las entidades activas (**estado = 1**) cuyo cualquiera de sus parámetros coinciden con el texto de búsqueda proporcionado.**

**Los resultados se ordenan por el ID de la entidad (**id**).**

****

* **USP\_GUARDAR\_PROV:**
  + **Este procedimiento almacenado se utiliza para guardar o actualizar un proveedor en la base de datos, dependiendo del valor del parámetro** @opcion**.**
  + **Si** @opcion **es igual a 1, se trata de un nuevo registro y se realiza una inserción en la tabla** Proveedor **con los datos proporcionados (**nombre**,** email**,** telefono**,** direccion**). El estado del nuevo proveedor se establece en 1, lo que indica que está activo.**
  + **Si** @opcion **es diferente de 1, se trata de una actualización de registro. Se actualizan los datos del proveedor correspondiente al** id **proporcionado.**

****

* **ACTIVO\_PROV:**
  + **Este procedimiento almacenado se utiliza para cambiar el estado de un proveedor (activo o inactivo) en la base de datos.**
  + **Recibe como parámetros el** id **del proveedor y el nuevo** estado **que se desea establecer.**
  + **Utiliza una instrucción** UPDATE **para cambiar el estado del proveedor en la tabla** Proveedor **según el** id **proporcionado.**

## 

## 

## 

## 

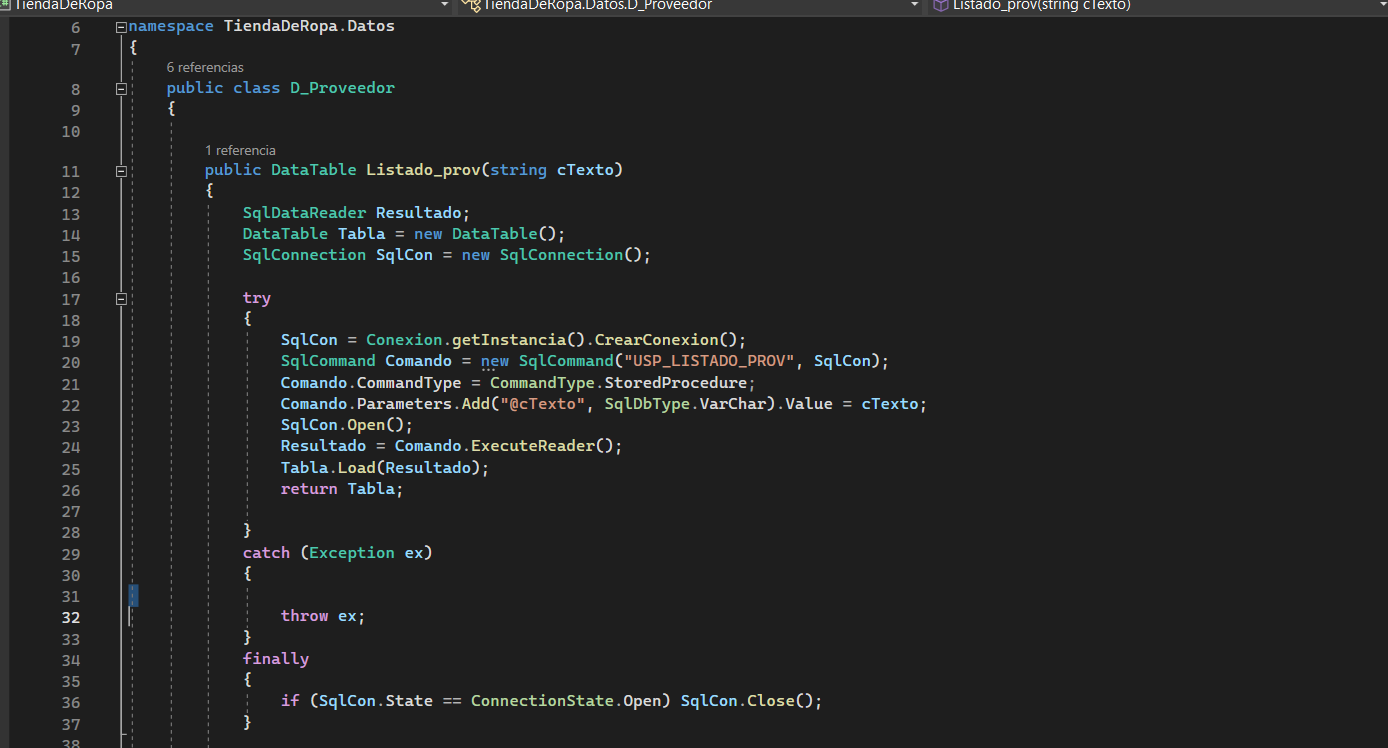
## 

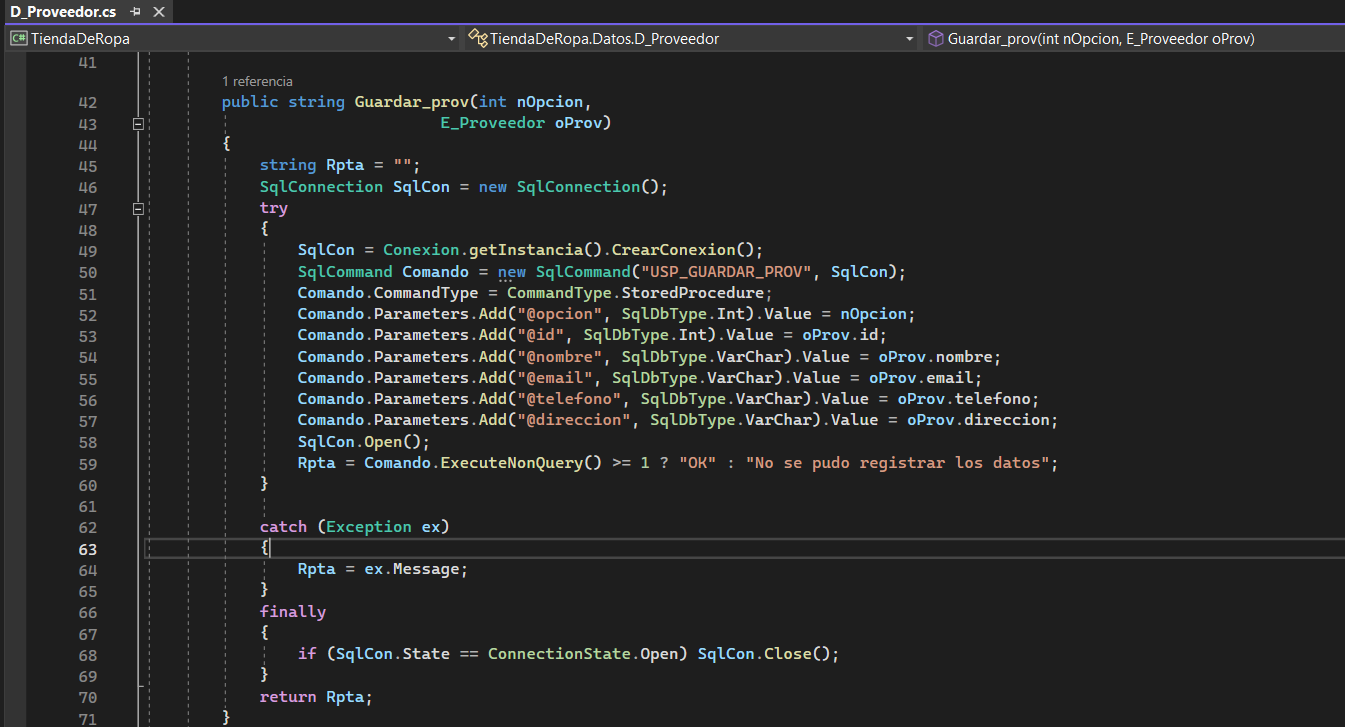
## 

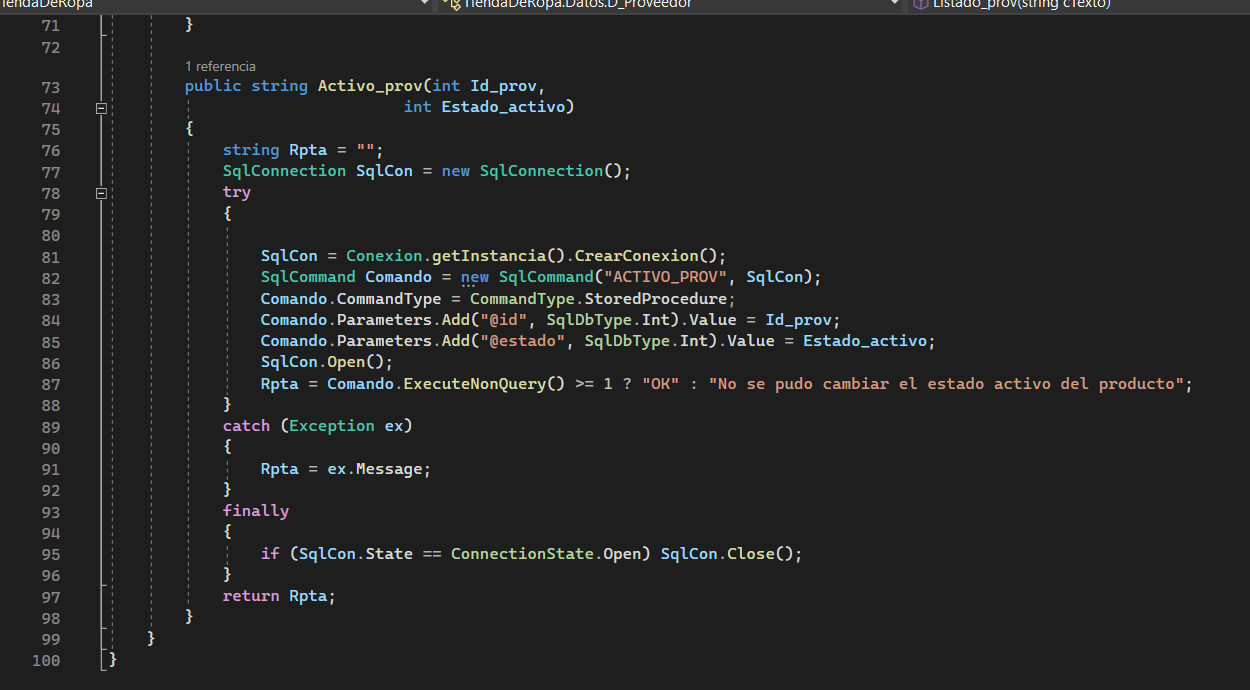
## ***Desarrollo de Software***

### **Estructura del Código Fuente**

* Nota: El código mostrado aquí, es la misma estructura que sigue toda la aplicación,por ello, solo se explica con Proveedor.

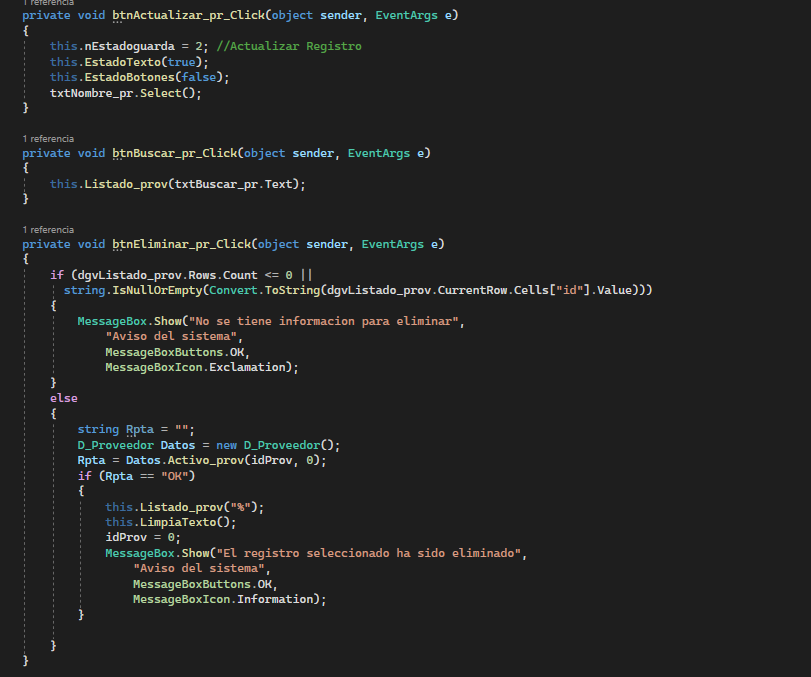


* **(Capa de Datos)**
* Listado\_prov(string cTexto):
  + Este método recibe un parámetro cTexto **que se utilizará para filtrar la búsqueda de proveedores.**
  + **Se establece una conexión a la base de datos utilizando la clase** Conexion**, que devuelve una instancia de** SqlConnection**.**
  + **Se ejecuta un procedimiento almacenado llamado** USP\_LISTADO\_PROV **que recibe como parámetro el texto de búsqueda.**
  + **Se carga el resultado de la consulta en un objeto** DataTable **y se retorna este** DataTable**.**
* 
* Guardar\_prov(int nOpcion, E\_Proveedor oProv):
  + Este método recibe dos parámetros: nOpcion**, que indica la operación a realizar (por ejemplo, agregar, actualizar, eliminar), y** oProv**, que es un objeto de tipo** E\_Proveedor **que contiene la información del proveedor a guardar.**
  + **Se establece una conexión a la base de datos utilizando la clase** Conexion**.**
  + **Se ejecuta un procedimiento almacenado llamado** USP\_GUARDAR\_PROV **que recibe como parámetros la opción, el ID, nombre, email, teléfono y dirección del proveedor.**
  + **Se retorna un mensaje indicando si la operación fue exitosa o no.**

****

* Activo\_prov(int Id\_prov, int Estado\_activo):
  + Este método recibe dos parámetros: Id\_prov**, que es el ID del proveedor cuyo estado se desea cambiar, y** Estado\_activo**, que indica el nuevo estado activo del proveedor.**
  + **Se establece una conexión a la base de datos utilizando la clase** Conexion**.**
  + **Se ejecuta un procedimiento almacenado llamado** ACTIVO\_PROV **que recibe como parámetros el ID del proveedor y su nuevo estado activo.**
  + **Se retorna un mensaje indicando si la operación fue exitosa o no.**
* **(Capa de Presentacion)**

### 



### **Clase** Frm\_Proveedor:

### Esta clase representa un formulario de Windows Forms que permite al usuario administrar la información de los proveedores.

### El formulario contiene varios controles, como cajas de texto (TextBox**), botones (**Button**), y un control de tabla (**DataGridView**) para mostrar la lista de proveedores.**

### **Métodos y Eventos:**

### Frm\_Proveedor()**: Es el constructor del formulario. Se inicializan los componentes del formulario.**

### LimpiaTexto()**: Método que limpia el texto de las cajas de texto del formulario.**

### EstadoTexto()**: Método que habilita o deshabilita la edición de las cajas de texto del formulario.**

### EstadoBotones()**: Método que habilita o deshabilita los botones del formulario.**

### Formato\_prov()**: Método que define el formato de las columnas en el control de tabla que muestra la lista de proveedores.**

### Listado\_prov(string cTexto)**: Método que carga la lista de proveedores en el control de tabla. Recibe un parámetro de texto que se utiliza para filtrar la búsqueda de proveedores.**

### Selecciona\_Item\_prov()**: Método que se ejecuta cuando se hace clic en una fila del control de tabla de proveedores. Permite seleccionar un proveedor para ver o editar su información en las cajas de texto del formulario.**

### **Eventos de Botones:**

### btnNuevo\_pr\_Click()**,** btnCancelar\_pr\_Click()**,** btnGuardar\_pr\_Click()**,** btnActualizar\_pr\_Click()**,** btnBuscar\_pr\_Click()**,** btnEliminar\_pr\_Click()**,** btnSalir\_pr\_Click()**: Son los eventos asociados a los botones del formulario. Cada evento realiza una acción específica, como agregar un nuevo proveedor, cancelar una operación, guardar cambios, actualizar información, buscar proveedores, eliminar un proveedor o salir del formulario.**

### **Validación de Datos:**

### **Dentro del evento** btnGuardar\_pr\_Click()**, se realizan validaciones de los datos ingresados en las cajas de texto, como verificar si se han ingresado datos obligatorios**

## ***Implementación***

**Instalación y Configuración**

* **Configuración de Hardware:**
  + No requiere de muchos recursos. Las especificaciones minimas serian: 4 GB de memoria RAM, un espacio de almacenamiento de almenos 20 gb en disco duro y el tipo de procesador con al menos 2 núcleos.
  + El software se desarrolló en un entorno Windows, los equipos locales necesitarán tener instalado un sistema operativo Windows compatible, como Windows 10 o el 11.
* **Base de Datos:**
  + El software utiliza como base de datos el SQL Server. Por lo que hay que asegurarse de que los equipos locales lo tengan instalado.
  + Una vez esté instalado, se puede crear la base de datos. Esto se hace mediante la ejecución de comandos SQL o mediante interfaces gráficas proporcionadas por el software.
* **Instalador:**
* El propósito principal del instalador es configurar automáticamente todos los componentes necesarios del software en el sistema del usuario, lo que incluye el software de la aplicación, las bibliotecas auxiliares, y cualquier otro recurso necesario para su funcionamiento.

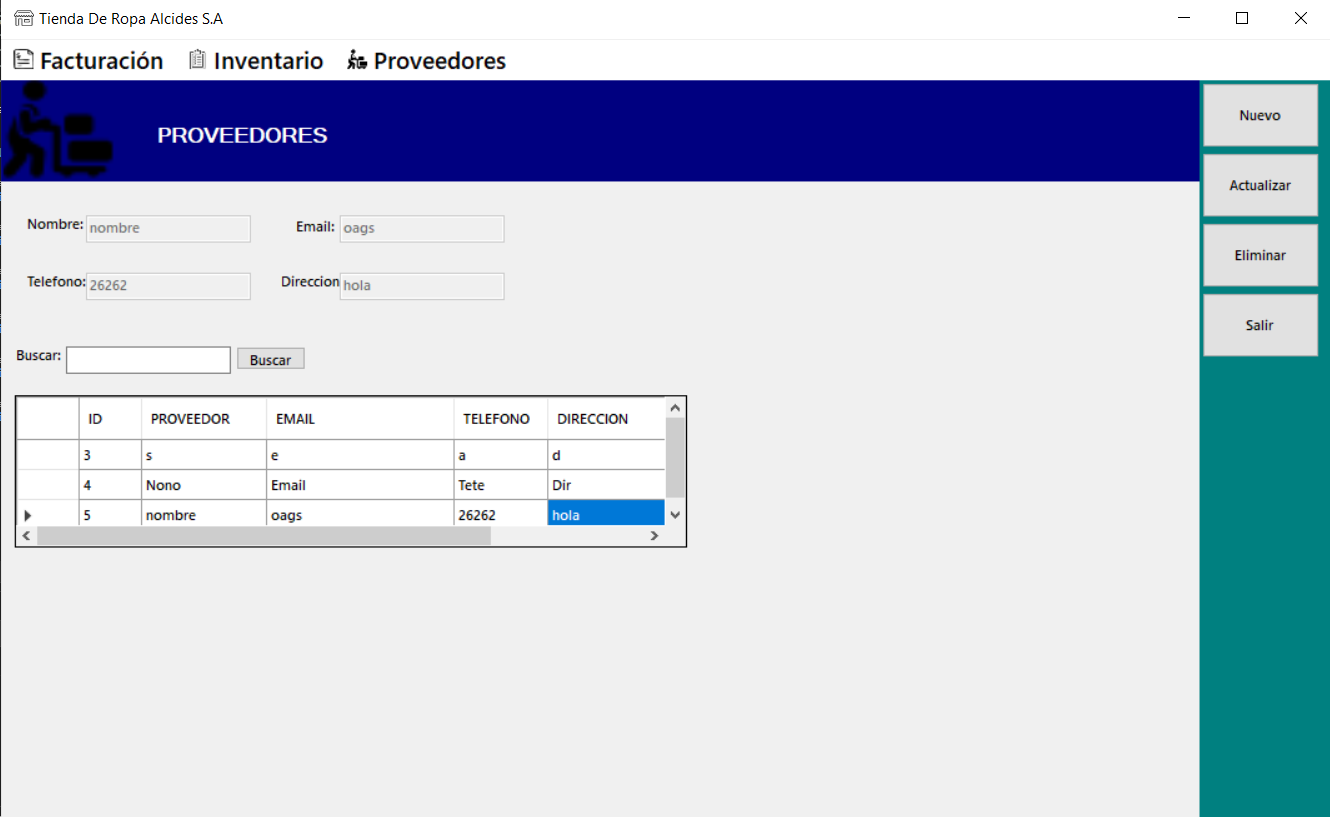
**- \*\*Repositorio de Código:\*\*** https://github.com/JanCarlosPM/TiendaDeRopaBD.git

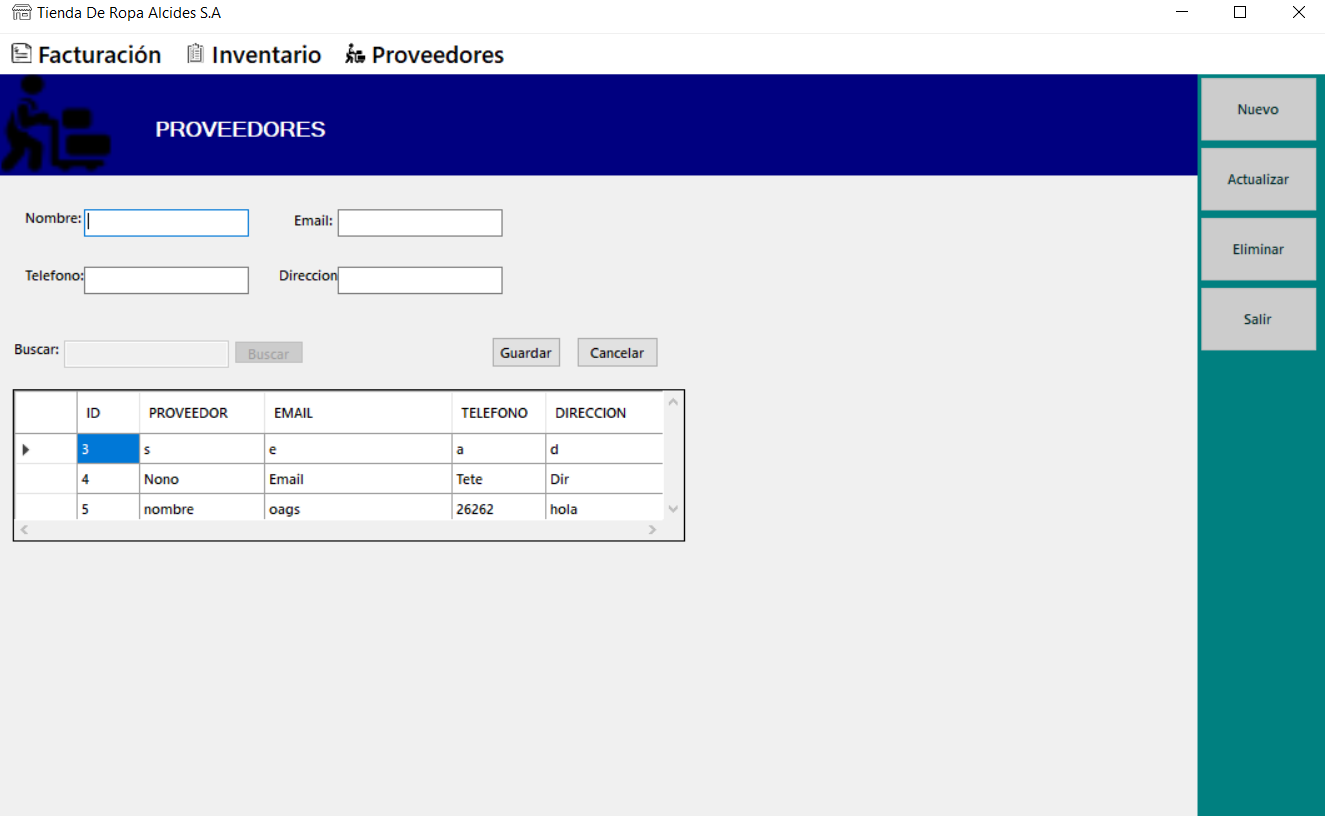
### ***Pruebas Realizadas***

* Pruebas Unitarias:
  + Se llevaron a cabo pruebas unitarias para evaluar el funcionamiento individual de cada componente del software, como clases, métodos y funciones.
  + Se crearon casos de prueba para cada unidad de código, y se verificó que cada una funcionará correctamente de acuerdo con sus especificaciones.
* Pruebas de Integración:
  + Se realizaron pruebas de integración para evaluar la interoperabilidad entre los diferentes módulos y componentes del software.
  + Se verificó que los distintos módulos del software se integran correctamente y que los datos se transfirieron correctamente entre ellos.

## 

***Instrucciones para el Uso***

****

****

Tenemos un menú principal que permite navegar a los usuarios a través de las distintas ventanas de la interfaz con un solo click: “Facturación”, “Inventario” y “Proveedores”.

Cada interfaz de las distintas ventanas siguen una misma estructura la cual seria:

La visualización de las cajas de texto y a la par tiene una etiqueta que indica que clase de información necesita llenarse. El usuario si se llega a confundir al ingresar un dato, este cuenta con validaciones que le lanzan un mensaje de advertencia.

La interfaz cuenta con una barra de búsqueda, en donde filtra los diferentes datos, dependiendo de la necesidad del usuario.

Posee una tabla de datos en donde se puede visualizar la información. Al momento de seleccionar una tabla, ésta lleva los datos a la caja de texto.

A la par de cada tabla, se pueden ver varios botones. El botón “Nuevo” que limpia los campos. Luego se activan dos botones: “Cancelar” y “Guardar” en donde bien el usuario ingresa los nuevos datos y los guarda o bien cancela la acción.

El botón “Actualizar” que funciona de esta manera: Se tiene que seleccionar una de las filas de la tabla, en donde lleva la información a las cajas de texto. Para editar esa información, solo se cambia los datos que fueron traídos a la caja de texto para posteriormente guardar esa información .

El botón “Eliminar” que solamente, hay que seleccionar una fila de la tabla y posteriormente darle click al botón para que este se elimine de la tabla.

El botón “Salir” que regresa cierra la ventana actual.

## ***Conclusiones***

**Resultados del Proyecto:**

* Se logró desarrollar con éxito un Sistema de Gestión de Inventario para una Tienda de Ropa, cumpliendo con los requisitos y expectativas del cliente.
* El software proporciona funcionalidades clave, como seguimiento de productos, gestión de proveedores y control de inventario, que mejoran la eficiencia y la productividad de la tienda.

**Lecciones Aprendidas:**

* Durante el desarrollo del proyecto, se adquirieron conocimientos sobre el diseño de bases de datos, la programación en C#, el uso de Visual Studio 2020 y la implementación de patrones de arquitectura como el Modelo de N Capas.
* Se comprendió la importancia de una comunicación efectiva con el cliente para garantizar la alineación de las expectativas y la entrega de un producto final satisfactorio.

**Áreas de Mejora:**

* En futuras versiones del software, se podría explorar la implementación de funcionalidades adicionales, como análisis de ventas, generación de informes y seguimiento de clientes, para mejorar aún más la utilidad y la versatilidad del sistema.
* Se podría mejorar la documentación del proyecto, incluyendo tutoriales para el usuario final sobre cómo utilizar todas las funcionalidades ofrecidas.