



Instituto Profesional Los Leones  
Escuela de Diseño y Tecnologías de la Información

# **Documentación Actividad Integrativa**

## **Programación I**

### **“Tienda de Abarrotes”**

Alumno: Jose González Riquelme

Profesor: Gonzalo Vidal Peña

Santiago, Diciembre 2019



## Indice

Introducción.....	4
Requerimientos.....	5
Herramientas.....	6
Lenguaje.....	6
Entorno de Desarrollo.....	6
Base de datos.....	6
Desarrollo.....	7
Ventana principal.....	7
Agregar producto.....	8
Consultar producto.....	8
Modificar producto.....	8
Venta.....	8
Eliminar producto.....	8
Salir.....	8
Agregar producto.....	9
Diseño.....	9
Código.....	10
Librerías.....	10
Clase de la ventana.....	10
Boton Cancelar.....	11
Boton Agregar.....	11
Consultar Producto.....	14
Diseño.....	14
Código.....	14
Librerías.....	14
Botón Buscar.....	15
Botón Mostrar Todo.....	16
Busqueda por nombre.....	17
Mostrar todo.....	18
Modificar Producto.....	19
Diseño.....	19
Codigo.....	20
Boton Modificar.....	20
Boton Buscar.....	21
Boton cerrar.....	21
Venta.....	22
Diseño.....	22
Código.....	22
Botón Buscar.....	23
Botón Agregar.....	23
Botón salir.....	24
Botón venta.....	24



Eliminar producto.....	25
Diseño.....	25
Codigo.....	25
Botón Salir.....	25
Botón eliminar.....	26
Conclusion.....	27
Diagrama aplicación.....	28



## Introducción

En el mundo de los negocios, muchas veces es difícil llevar el día a día en la contingencia tanto nacional como internacional, los negocios tradicionales muchas veces van quedando atrás y es menester encontrar nuevas formas de realizar las transacciones en este siglo XXI que nos engulle día a día.

Es el caso de “El bazar de Josesin” local de barrio arraigado a sus costumbres, costumbres que serán actualizadas con el siguiente proyecto. Se solicita crear una aplicación para gestionar las ventas y el inventario de este local, la aplicación como tal debe ser simple para que las transacciones sean ejecutadas rápidamente sin estorbar la atención fluida de la clientela que frecuenta el local.

Muchos comercios ya han adoptado este nuevo cambio y modalidad, dado que abunda la desorganización al llevar las cuentas en papel, los requisitos para esta aplicación han de mantenerse simples y expeditas para el uso de cualquier empleado del local y su dueño quien ya goza de mayor edad.



## Requerimientos

Se debe desarrollar de manera rápida una aplicación para satisfacer las necesidades de registro de ventas del local.

Los requerimientos para esta serán:

- Poseer una pequeña base de datos para almacenar el stock del local.
- Tener interfaz gráfica intuitiva.
- Fácil uso.
- Fácil implementación.
- Procesos rápidos.
- Cumplir con las demandas del cliente.



## Herramientas

Para el desarrollo de la aplicación se necesitara una gama de herramientas tecnologicas.

## Lenguaje

El lenguaje escogido para este fin, dado su requerimiento y capacidad de multiplataforma sera **C#**.

*C# es un lenguaje de programación multiparadigma desarrollado y estandarizado por Microsoft como parte de su plataforma .NET, que después fue aprobado como un estándar por la ECMA e ISO. Fuente: [https://es.wikipedia.org/wiki/C\\_Sharp](https://es.wikipedia.org/wiki/C_Sharp)*

Fue finalmente seleccionado por que permite la creacion de una interfaz grafica de manera rapida.

## Entorno de Desarrollo

Para el desarrollo de la aplicación en si se hara uso de el entorno de desarrollo MonoDevelop, esto permite combinar el lenguaje **C#** con un creador de interfaz de usuario gráfico lo que permitirá un desarrollo más ágil de la aplicación.

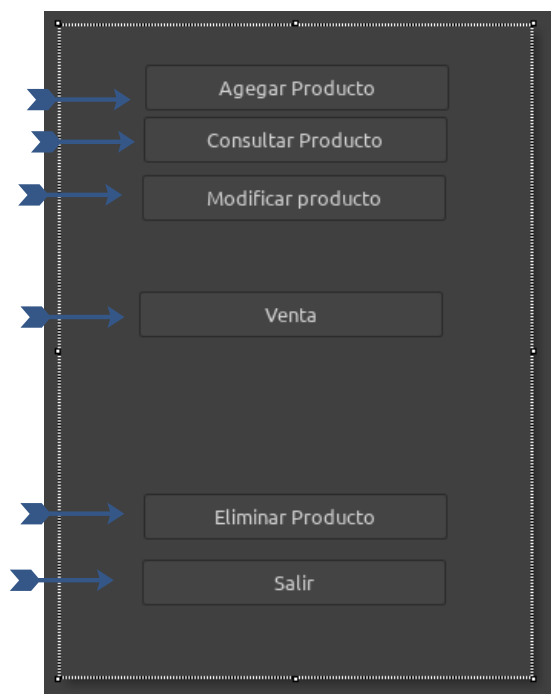
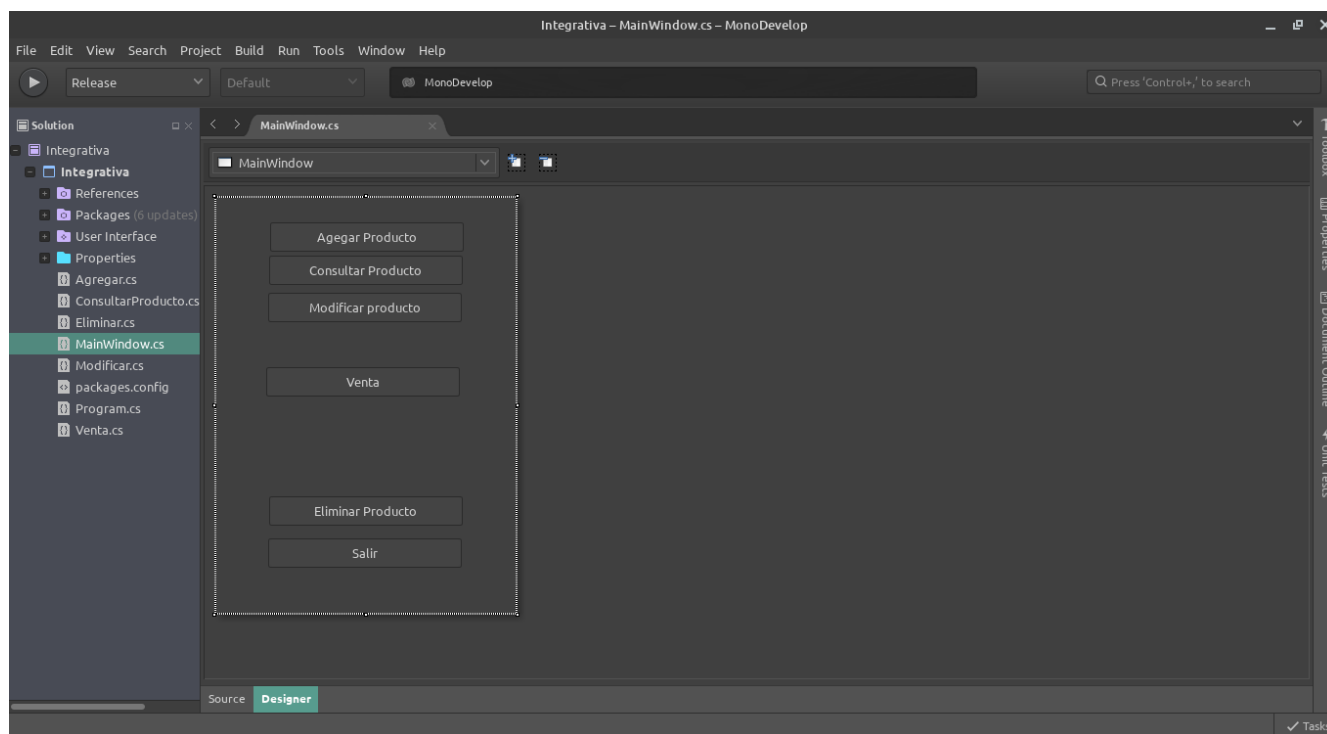
## Base de datos

Para el motor de base de datos se usara el sistema relacional MySQL.

*MySQL es un sistema de gestión de bases de datos relacional desarrollado bajo licencia dual: Licencia pública general/Licencia comercial por Oracle Corporation y está considerada como la base de datos de código abierto más popular del mundo. Fuente: <https://es.wikipedia.org/wiki/MySQL>*

# Desarrollo

## Ventana principal



La ventana principal es la primera vista que el usuario tiene de la aplicación, por lo tanto en ella se ubican los botones que nos permiten navegar a cada una de las ventanas presentes en la aplicación.

- Agregar producto.
- Consultar producto.
- Modificar producto.
- Venta.
- Eliminar producto.
- Salir.



## **Agregar producto**

La opción agregar producto permite agregar un nuevo producto a la base de datos.

## **Consultar producto**

La opción consultar producto permite consultar 1 o más productos en stock realizando una consulta a la base de datos.

## **Modificar producto**

La opción modificar producto, permite modificar un producto presente actualmente en la base de datos de la aplicación.

## **Venta**

La opción venta permite realizar una venta de producto con su eventual descuento en el área de la cantidad de productos que quedan en la tienda.

## **Eliminar producto**

La opción eliminar producto permite eliminar un registro de la tabla, ya sea que ese producto no se venderá más en el local o hubo un error en el ingreso del mismo.

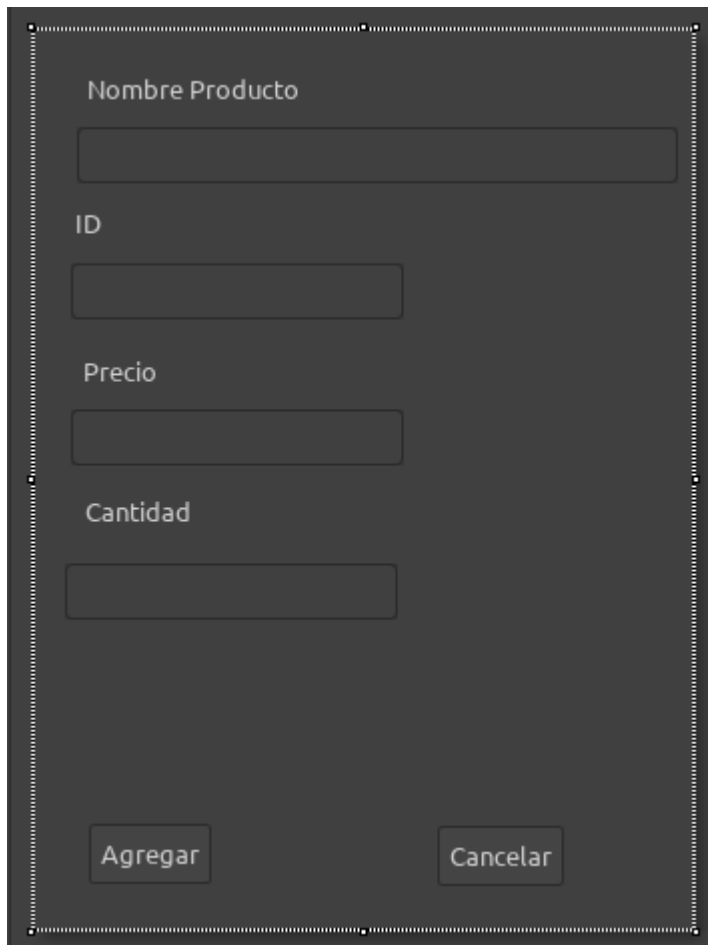
## **Salir**

Salir de la aplicación.



## Agregar producto

### Diseño



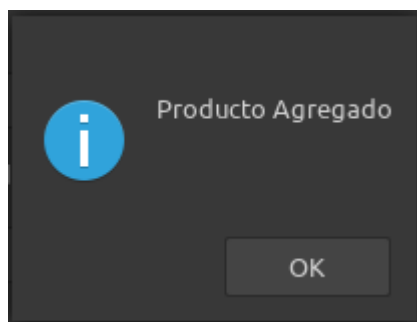
The form is titled 'Agregar producto' and contains four input fields with labels: 'Nombre Producto', 'ID', 'Precio', and 'Cantidad'. At the bottom, there are two buttons: 'Agregar' and 'Cancelar'.

La ventana presenta 4 entrybox, donde se ingresaran los datos del producto a agregar, cada una de estas entry box posee su respectiva label la que indica que tipo de dato capturaran para su almacenamiento en la base de datos.

En su parte inferior tiene los respectivos botones para indicar la accion que se desea realizar, ya sea salir o cancelar la operación.

**Agregar:** Agrega el elemento con los datos obtenidos en las entrybox y realiza el insert en la base de datos.

**Cancelar:** Cierra la ventana.



**MessageDialog:** un message dialog alertara al usuario que el producto ha sido agregado al inventario del local.

## Código

### Librerías

```
using System;  
using Gtk;  
using MySql.Data.MySqlClient;
```

Las librerías que se usaran son:

- System
- Gtk
- MySql.Data.MySqlClient

System y Gtk son nativas del sistema mientras que MySql.Data.MySqlClient fue agregada al proyecto desde internet

### Clase de la ventana

```
public partial class Agregar : Gtk.Window  
{  
    public Agregar() :  
        base(Gtk.WindowType.Toplevel) => this.Build();  
}
```

Este código lo genera automáticamente monoDevelop al crear la ventana nueva, en el constructor llama a la función Build() que crea la ventana cuando la clase es invocada desde la ventana principal.

Public partial class Agregar : Gtk.Window  
desciende desde la clase Gtk.Window.

Indica que la clase de la ventana agregar

## Boton Cancelar

```
protected void OnButton4Clicked(object sender, EventArgs e)
{
    this.Destroy();
}
```

La funcion OnButton4Clicked controla el evento del boton Cancelar, la funcion de esta funcion es cerrar la ventana lo que realiza en la instrucción this.Destroy(); donde this es el objeto en el cual se esta trabajando y Destroy() es el metodo Gtk para cerrar la ventana en la cual se refiere this anteriormente.

## Boton Agregar

```
protected void OnButton2Clicked(object sender, EventArgs e)
{
    string query = null;
    string connectionString = null;
    MySqlConnection connection1;
    connectionString = "server=localhost;database=TIENDA_ABARROTES;uid=root;pwd=root;";
    connection1 = new MySqlConnection(connectionString);
    connection1.Open();
    query = "insert into tbProductos (ID, Nombre, Precio, Cantidad)values('"+Int32.Parse(this.entry2.Text)
    + "','"+ this.entry1.Text + "','"+ Int32.Parse(this.entry4.Text)
    + "','"+ Int32.Parse(this.entry3.Text) + "')";
    MySqlDataReader MyReader2;
    MySqlCommand MyCommand2 = new MySqlCommand(query, connection1);
    MyReader2 = MyCommand2.ExecuteReader();
    connection1.Close();

    MessageDialog md = new MessageDialog(this, DialogFlags.DestroyWithParent,
        MessageType.Info,
        ButtonsType.Ok, "Producto Agregado");
    md.Run();
    md.Destroy();
}
```

La funcion OnButton2Clicked se encarga de agregar el producto cuando es presionado.

```
string query = null;  
string connectionString = null;  
MySqlConnection connection1;  
connectionString = "server=localhost;database=TIENDA_ABARROTES;uid=root;pwd=root;";  
connection1 = new MySqlConnection(connectionString);  
connection1.Open();
```

La primera parte de la función es la encargada de crear la conexión con la base de datos donde se almacenan los productos del local. Para ello es creada la variable `query` que es igualada a `null` y `connectionString` también es igualada a `null`; `query` se encarga de almacenar la query que será enviada a la base de datos y `connectionString` enviará los datos necesarios para que la aplicación obtenga acceso a la base de datos en sí.

Se crea un objeto del tipo `MySqlConnection` cuyo nombre será `connection1` luego se crea una nueva conexión con la base de datos y se le envía como parámetro `connectionString` previamente creado.

Finalmente se abre la conexión para que la aplicación tenga acceso a los datos y a la base de datos con `connection1.Open()`.

```
query = "insert into tbProductos (ID, Nombre, Precio, Cantidad) values ('"+Int32.Parse(this.entry2.Text)  
+"', '"+ this.entry1.Text + "', '"+ Int32.Parse(this.entry4.Text)  
+ "', '"+ Int32.Parse(this.entry3.Text) + "')";
```

En la variable `query` se crea la consulta que se realizará a la base de datos en este caso se trata de un insert por lo que se usa la instrucción `INSERT INTO` seguido del nombre de la tabla y como parámetros se le envía a las columnas a las cuales se insertarán datos y los correspondientes valores encerrados en comillas.

```
MySqlDataReader MyReader2;  
MySqlCommand MyCommand2 = new MySqlCommand(query, connection1);  
MyReader2 = MyCommand2.ExecuteReader();  
connection1.Close();
```

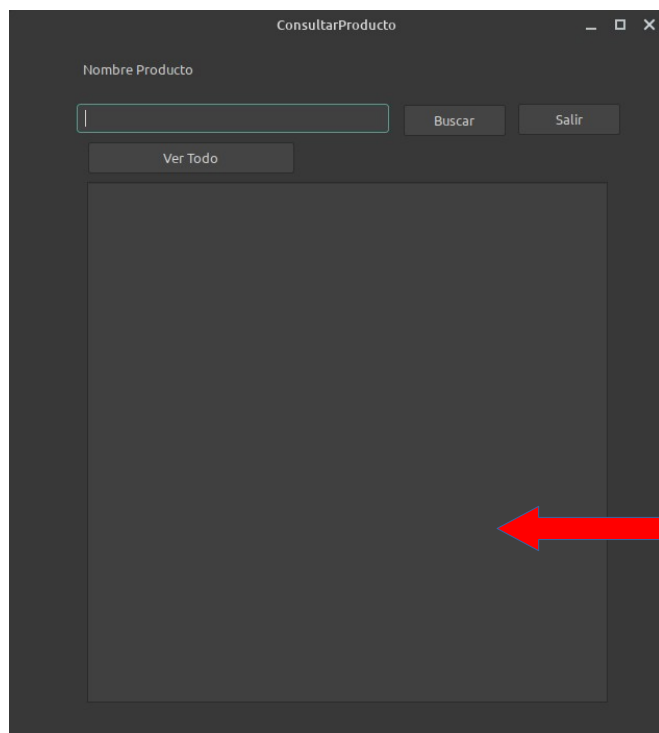
Se crea un objeto del tipo `MySqlDataReader` que tendrá por nombre `MyReader2` que se encargará de ejecutar la query en la base de datos, seguidamente se crea un objeto del tipo `MySqlCommand` con el nombre `MyCommand2` al cual se le asigna la query y la conexión realizada como parámetro. Habiendo asignado estos valores se le envía a `MyReader2` la instrucción de ejecutar el comando o la query en la base de datos, y luego se termina por cerrar la conexión, de lo contrario el programa quedará trabado en esta pantalla.

```
AlertDialog md = new AlertDialog(this, DialogFlags.DestroyWithParent,  
    MessageType.Info,  
    ButtonType.Ok, "Producto Agregado");  
md.Run();  
md.Destroy();
```

Habiendo ingresado el dato correctamente en la base de datos, el programa retornara un mensaje indicando que el producto ha sido agregado.

## Consultar Producto

### Diseño



La ventana consultar producto posee diversas características distintas a las demás. En ella es donde se localizan los productos que se quieren consultar o vender a los clientes.

En ella se encuentran como en las demás instancias de la aplicación un botón para realizar la acción y otro para salir de la ventana actual.

Esta area se denomina nodeview y permite mostrar elementos como los que se recuperan desde una base de datos.

### Código

#### Librerías

```
using System;  
using MySql.Data.MySqlClient;  
using Gtk;  
using System.Data;
```

Las librerías usadas en el programa son globales por lo tanto se importan en todas las instancias, sin embargo aquí se agrega System.Data, la cual nos permitirá el manejo de tablas como las traídas desde la base de datos.

## Botón Buscar

```
protected void OnBuscarBotonClicked(object sender, EventArgs e)
{
    string query = null;
    string connectionString = null;
    MySqlConnection connection1;
    connectionString = "server=localhost;database=TIENDA_ABARROTES;uid=root;pwd=root;";
    connection1 = new MySqlConnection(connectionString);
    connection1.Open();
    query = "select * from tbProductos where Nombre like '%" + this.entry1.Text + "%' order by Precio asc;";
    MySqlCommand MyCommand2 = new MySqlCommand(query, connection1);
    MySqlDataReader MyReader2;
    MyReader2 = MyCommand2.ExecuteReader();
    ListStore tipoListado;
    tipoListado = new ListStore(typeof(string), typeof(string), typeof(string), typeof(string));
    nodeview2.AppendColumn("ID", new CellRendererText(), "text", 0);
    nodeview2.AppendColumn("Nombre", new CellRendererText(), "text", 1);
    nodeview2.AppendColumn("Precio", new CellRendererText(), "text", 2);
    nodeview2.AppendColumn("Cantidad", new CellRendererText(), "text", 3);
    nodeview2.Model = tipoListado;
}
```

La función que se ejecuta la momento de hacer click en el botón Buscar es OnBuscarBotonClicked, como anteriormente, es creada la conexión a la base de datos para la consulta de los datos, esta vez la query es:

**query = "select \* from tbProductos where Nombre like '%" + this.entry1.Text + "%'**  
**order by Precio asc;";**

Como el fin del boton es consultar productos se traen los resultados ordenados de forma ascendente de esta forma siempre mostrara el mas barato para recomendar a los clientes. Los operadores % que encierran el texto ingresado permiten traer resultados con nombres similares al buscado.

Mas adelante se encuentra donde se arma el listado con los registros traídos desde la base de datos, a este listado se le agregan las mismas columnas que posee la base de datos para que los datos que se mostraran y los registros de la base de datos coincidan. Se crea el modelo con la instrucción nodeview2.Model = tipoListado asignandoselo a la nodeview presente en la ventana.

```
while (MyReader2.Read())
{
    string id = (string)MyReader2["ID"];
    string nombre = (string)MyReader2["Nombre"];
    string precio = (string)MyReader2["Precio"];
    string cant = (string)MyReader2["Cantidad"];
    tipoListado.AppendValues(id,nombre,precio,cant);

    nodeview2.EnableGridLines = TreeViewGridLines.Horizontal;
}
connection1.Close();
}
```

Con el loop while se recorren los registros que se trajeron desde la base de datos y cada columna se guarda en una variable para su posterior exhibición en el área del nodeview luego se le activa el modo grid al nodeview dándole la instrucción de volcado horizontal para luego cerrar la conexión una vez salido del loop while.

## Botón Mostrar Todo

Botón mostrar todo nos permite realizar la misma acción anterior pero mostrando todos los registros de la tabla, por ende todo el stock de la tienda.

```
protected void OnButton3Clicked(object sender, EventArgs e)
{
    string query = null;
    string connectionString = null;
    MySqlConnection connection1;
    connectionString = "server=localhost;database=TIENDA_ABARROTES;uid=root;pwd=root;";
    connection1 = new MySqlConnection(connectionString);
    connection1.Open();
    query = "select * from tbProductos;";
    MySqlCommand MyCommand2 = new MySqlCommand(query, connection1);
    MySqlDataReader MyReader2;
    MyReader2 = MyCommand2.ExecuteReader();
    ListStore tipoListado;
    tipoListado = new ListStore(typeof(string), typeof(string), typeof(string), typeof(string));
    nodeview2.AppendColumn("ID", new CellRendererText(), "text", 0);
    nodeview2.AppendColumn("Nombre", new CellRendererText(), "text", 1);
    nodeview2.AppendColumn("Precio", new CellRendererText(), "text", 2);
    nodeview2.AppendColumn("Cantidad", new CellRendererText(), "text", 3);
    nodeview2.Model = tipoListado;
}
```



Se realizan, por lo tanto, las mismas acciones que en la función anterior, solo que en esta ocasión la query es cambiada:

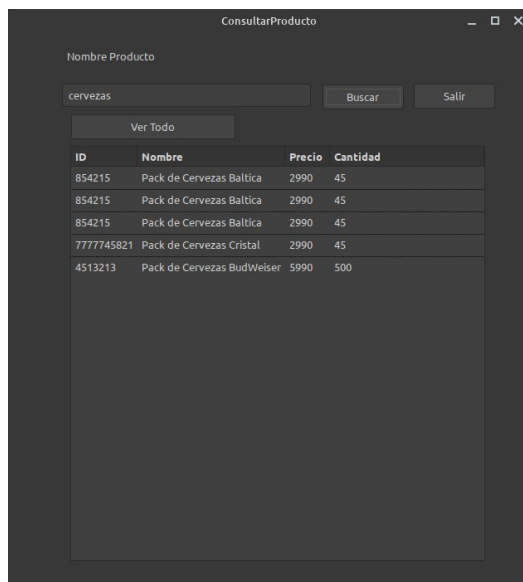
```
query = "select * from tbProductos;";
```

Esta consulta permite traer todos los datos presentes en la tabla tbProductos.

```
while (MyReader2.Read())  
{  
  
    string id = (string)MyReader2["ID"];  
    string nombre = (string)MyReader2["Nombre"];  
    string precio = (string)MyReader2["Precio"];  
    string cant = (string)MyReader2["Cantidad"];  
    tipoListado.AppendValues(id, nombre, "$ "+precio, cant);  
  
    nodeview2.EnableGridLines = TreeViewGridLines.Horizontal;  
  
}  
connection1.Close();
```

Como la vez anterior, los datos son mostrados una en el nodeview.

## Busqueda por nombre.



ID	Nombre	Precio	Cantidad
854215	Pack de Cervezas Baltica	2990	45
854215	Pack de Cervezas Baltica	2990	45
854215	Pack de Cervezas Baltica	2990	45
7777745821	Pack de Cervezas Cristal	2990	45
4513213	Pack de Cervezas BudWeiser	5990	500



## Mostrar todo

ConsultarProducto

Nombre Producto

Buscar Salir

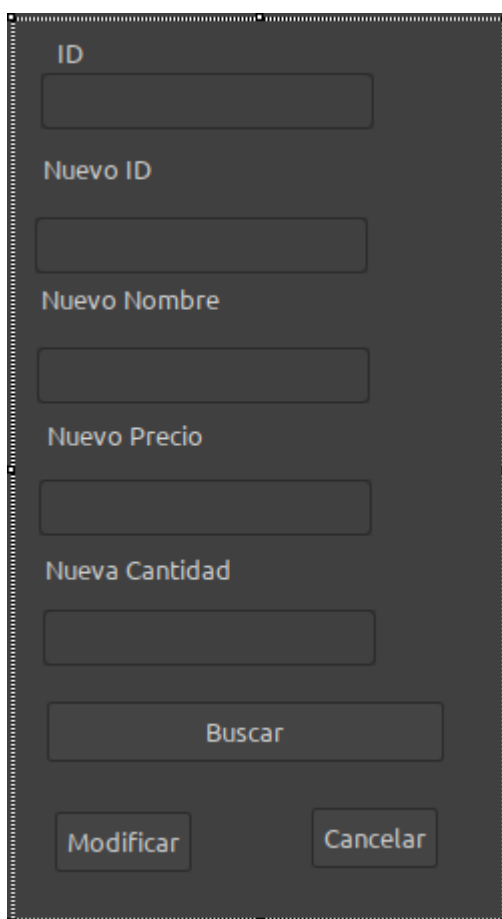
Ver Todo

ID	Nombre	Precio	Cantidad
854215	Pack de Cervezas Baltica	\$ 2990	45
854215	Pack de Cervezas Baltica	\$ 2990	45
777757813485	Hallulla Corriente	\$ 1290	500
5489615	Bebida Gatorade	\$ 990	123
777774548313	Pañales Babysec	\$ 9990	100
854762	Vino Alto Del Carmen	\$ 3990	40
77777123453	Marraqueta Corriente	\$ 1290	600
154567	Arroz PreGraneado Tucapel 1K	\$ 790	500
854215	Pack de Cervezas Baltica	\$ 2990	45
777757813485	Hallulla Corriente	\$ 1290	500
5489615	Bebida Gatorade	\$ 990	123
777774548313	Pañales Babysec	\$ 9990	100
854762	Vino Alto Del Carmen	\$ 3990	40
77777123453	Marraqueta Corriente	\$ 1290	600
154567	Arroz PreGraneado Tucapel 1K	\$ 790	500
7777745685132	Queso Gauda	\$ 3990	100
7777745821	Pack de Cervezas Cristal	\$ 2990	45
77777451212	Tallipes Minionias	\$ 1000	50

## Modificar Producto

La ventana de modificar producto, permite modificar los datos a un producto presente en los registros de productos.

### Diseño

The image shows a dark-themed user interface for modifying a product. It features a vertical stack of input fields. The first field is labeled 'ID'. Below it is a field labeled 'Nuevo ID'. This is followed by a field labeled 'Nuevo Nombre', then 'Nuevo Precio', and finally 'Nueva Cantidad'. At the bottom of the form is a wide button labeled 'Buscar'. Below the 'Buscar' button are two smaller buttons: 'Modificar' on the left and 'Cancelar' on the right. The entire form is enclosed in a thin border.

El diseño de modificar producto es presentado con una interfaz lo mas intuitiva posible, en ella se indica el ID actual del producto a modificar.

Posteriormente se solicita el ingreso del nuevo ID si requiere ingresar los demás datos del producto.

Para facilidad del usuario se incorpora nuevamente el botón buscar que trae la ventana de consultar producto para buscar el producto que se quiere modificar.

## Codigo

### Boton Modificar

```
protected void OnBotonModificarClicked(object sender, EventArgs e)
{
    string query = null;
    string connectionString = null;
    MySqlConnection connection1;
    connectionString = "server=localhost;database=TIENDA_ABARROTES;uid=root;pwd=root;";
    connection1 = new MySqlConnection(connectionString);
    connection1.Open();
    query = "update tbProductos set ID='" + this.entry5.Text + "', Nombre='" + this.entry2.Text + "', Precio='" + this.entry3.Text + "', Cantidad = '" + this.entry4.Text + "' where ID = '" + this.entry1.Text + "'";
    MySqlDataReader MyReader2;
    MySqlCommand MyCommand2 = new MySqlCommand(query, connection1);
    MyReader2 = MyCommand2.ExecuteReader();
    connection1.Close();

    MessageBox md = new MessageBox(this, DialogFlags.DestroyWithParent,
        MessageType.Info,
        ButtonsType.Ok, "Producto Actualizado");
    md.Run();
    md.Destroy();
}
```

La función que se ejecuta al presionar el botón modificar es OnBotonModificarClicked en ella se abre la conexión a la base de datos y se realiza la query:

```
query = "update tbProductos set ID='" + this.entry5.Text + "', Nombre='" + this.entry2.Text + "',
Precio='" + this.entry3.Text
        + "', Cantidad = '" + this.entry4.Text + "' where ID = '" + this.entry1.Text + "'";
```

en ella se toman los valores ingresados por el usuario y se realiza un update a la tabla con los nuevos datos.

Ejecutado el proceso se abrirá una ventana emergente que nos alertara que el cambio ha sido efectuado.

## Boton Buscar

```
protected void OnBotonBuscaClicked(object sender, EventArgs e)
{
    Integrativa.ConsultarProducto consultar = new Integrativa.ConsultarProducto();
    consultar.Show();
}
```

El botón buscar trae la pantalla de consultar elemento para consultar el ID del registro que se desea modificar.

## Boton cerrar

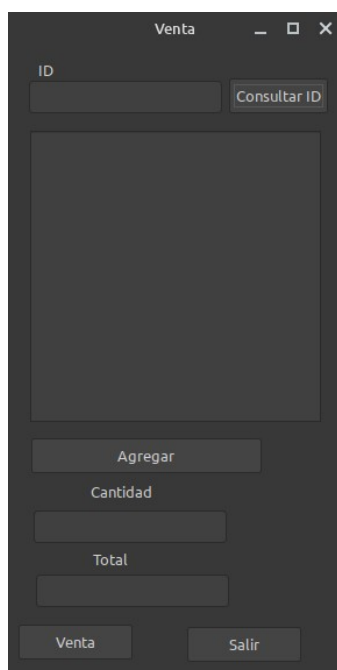
```
protected void OnBotonCancelarClicked(object sender, EventArgs e)
{
    this.Destroy();
}
```

Botón cerrar cierra la ventana en la que se encuentra actualmente.

## Venta

### Diseño

El botón venta permite realizar el propósito del programa al fin y al cabo, vender los productos, también realiza el correspondiente resta de la cantidad del producto vendido a la base de datos para mantener actualizado el stock del local. Se agrega nuevamente el botón buscar para poder tener al alcance los datos de los demás productos.



La venta por temas de asegurar que el producto vendido sea el correcto, se presenta pidiendo el ID del producto a vender, se incluye el botón ConsultarID que permite consultar los datos de los productos en el caso de que no se recuerde, el botón agregar agrega los productos a una lista para visualizar el total de la compra, se requiere indicar la cantidad del producto y al finalizar el Total nos muestra el total de la compra.

### Código

```
int preciofinal;  
int nuevoStock;  
int stock, cantidad;
```

Se declaran 4 variables globales en la clase de la ventana, las cuales permitirán llevar los registros que se quieren modificar para realizar la venta y los correspondientes cálculos.

## Botón Buscar.

El botón buscar nuevamente es incluido para poder acceder de manera rapida a la información de los productos.

```
protected void OnButton4Clicked(object sender, EventArgs e)
{
    Integrativa.ConsultarProducto consulta = new Integrativa.ConsultarProducto();
    consulta.Show();
}
```

## Botón Agregar.

```
protected void OnButtonAgregarClicked(object sender, EventArgs e)
{
    string query = null;
    string connectionString = null;
    MySqlConnection connection1;
    connectionString = "server=localhost;database=TIENDA_ABARROTES;uid=root;pwd=root;";
    connection1 = new MySqlConnection(connectionString);
    connection1.Open();
    query = "select ID, Nombre,Precio,Cantidad from tbProductos where ID='"+ this.entry1.Text+"'";
    MySqlCommand MyCommand2 = new MySqlCommand(query, connection1);
    MySqlDataReader MyReader2;
    MyReader2 = MyCommand2.ExecuteReader();
    ListStore tipoListado;
    tipoListado = new ListStore(typeof(string), typeof(string), typeof(string), typeof(string));
    nodeview1.AppendColumn("ID", new CellRenderText(), "text", 0);
    nodeview1.AppendColumn("Nombre", new CellRenderText(), "text", 1);
    nodeview1.AppendColumn("Precio", new CellRenderText(), "text", 2);
    nodeview1.Model = tipoListado;
```

La principal función del botón agregar recae en, usando los métodos usados en las funciones anteriores, agrega los productos a la lista de compra para después calcular el total.

En el proceso son convertidos a int los datos que se usaran en el calculo, como lo son el precio y la cantidad. La cantidad es capturada a través de un entry en la ventana.

```
while (MyReader2.Read())
{
    string id = (string)MyReader2["ID"];
    string nombre = (string)MyReader2["Nombre"];
    stock = Convert.ToInt32(MyReader2["Cantidad"]);
    int precio = Convert.ToInt32(MyReader2["Precio"]);
    cantidad = Convert.ToInt32(entry6.Text);
    tipoListado.AppendValues(id, nombre, precio.ToString());
    preciofinal = precio * cantidad;

    nodeview1.EnableGridLines = TreeViewGridLines.Horizontal;
}

connection1.Close();
}
```

## Botón salir

```
protected void OnButton3Clicked(object sender, EventArgs e)
{
    this.Destroy();
}
```

El botón salir cierra la ventana.

## Botón venta

```
protected void OnButton2Clicked(object sender, EventArgs e)
{
    string connectionString = null;
    MySqlConnection connection1;
    connectionString = "server=localhost;database=TIENDA_ABARROTES;uid=root;pwd=root;";
    connection1 = new MySqlConnection(connectionString);
    connection1.Open();
    nuevoStock = stock - cantidad;
    string query2 = "update tbProductos set Cantidad='" + nuevoStock.ToString() + "' where ID = '" + this.entry1.Text + "'";
    MySqlCommand MyCommand3 = new MySqlCommand(query2, connection1);
    MySqlDataReader MyReader2;
    MyReader2 = MyCommand3.ExecuteReader();
    this.entry7.Text = preciofinal.ToString();
    connection1.Close();
}
```

Botón venta es el encargado de calcular el total de los productos agregados, y lo muestra en pantalla a través de un entry en la parte inferior de la ventana. Luego es realizada la query para hacer el update correspondiente a la tabla con el nuevo stock con la cantidad descontada.



## Eliminar producto

### Diseño



El diseño de la ultima ventana, se mantuvo acorde a la estética de la aplicación, es por esto que en ella se ve que se puede eliminar ya sea por el nombre o por el ID del producto.

### Codigo

#### Botón Salir

```
protected void OnButton1Clicked(object sender, EventArgs e)
{
    this.Destroy();
}
```

El botón salir cierra la ventana actual.

## Botón eliminar

```
protected void OnButton3Clicked(object sender, EventArgs e)
{
    string query = null;
    string connectionString = null;
    MySqlConnection connection1;
    connectionString = "server=localhost;database=TIENDA_ABARROTES;uid=root;pwd=root;";
    connection1 = new MySqlConnection(connectionString);
    connection1.Open();
    query = "delete from tbProductos where ID like = '" + this.entry1.Text + "'";
    query = "delete from tbProductos where Nombre = '" + this.entry2.Text + "'";
    MySqlDataReader MyReader2;
    MySqlCommand MyCommand2 = new MySqlCommand(query, connection1);
    MyReader2 = MyCommand2.ExecuteReader();
    connection1.Close();

    MessageDialog md = new MessageDialog(this, DialogFlags.DestroyWithParent,
        MessageType.Info,
        ButtonsType.Ok, "Producto Eliminado");
    md.Run();
    md.Destroy();
}
```

Botón eliminar es un remanente de las funciones anteriores, se cambio la query a la necesaria y el mensaje emergente.



## Conclusion

El proyecto a pesar de que una de las herramientas (MonoDevelop) posee una documentación que en algunos casos podría considerarse nula, presenta una aplicación intuitiva y sencilla de uso para un local comercial con necesidades humildes y acotadas.

El proyecto, sin embargo, posee amplias opciones de escalado debido a su conexión con base de datos y su uso de C# el cual se podría transformar y servir como lenguaje de lado a una posible pagina web, con estas tecnologías daría la oportunidad de agregar nuevo hardware y funcionalidades al programa.

## Diagrama aplicación

