Desarrollo

de

Aplicaciones

Multiplataforma

Título del proyecto : Software Terminal Cliente-hostelería

Alumna : Juliana Carla Freitas Solano

Tutor : Luis Velasco

Fecha de entrega : 09/06/2013

ÍNDICE

1.Justificación

Llevo varios años trabajando en el sector de la hostelería y éste software sería de gran ayuda en cualquier cafetería, la razón, porque habría menos equivocaciones de pedidos por parte de los camareros, con esto, conseguiríamos un buen servicio, porque mientras que los clientes hacen los pedidos por el terminal, el personal agilizará el proceso del pedido, habiendo menos retrasos en la entrega de los servicios a sus receptores y menos carga para los camareros ya que se ahorra en tiempo y malos entendidos entre cliente y personal.

2.Objetivos del proyecto

Dentro de los objetivos del proyecto podemos distinguir los objetivos generales de los objetivos funcionales.

Los generales se refiere a lo que en la totalidad la aplicación nos supone y los funcionales se refiere a que funciones realiza la aplicación.

Los objetivos generales del proyecto son los siguientes :

•**Agilizar el pedido** gracias a que la aplicación posee un diseño con iconos y descripciones necesarias, el cliente puede hacer su pedido sin tener problemas para encontrar los artículos necesarios.

•**Agilizar el servicio** gracias a que cuando un cliente haga un pedido se impriman ambos tickets tanto para clientes como para el personal, este último preparará los artículos y el camarero será el responsable de entregar dicho pedido a la mesa del cliente.

Los objetivos funcionales del proyecto son los siguientes :

•Ofrecer un óptimo servicio a los clientes rápido y eficaz.

3.Estado del arte

El estado del arte se analiza la situación actual del sector por el que vamos a mover, las tecnologías actuales, esto nos permite tener una visión global de las posibilidades que nos ofrece y su disponibilidad, de forma que podamos elegir en cada caso la solución que mejor se adapte a nuestras necesidades.

Primeramente nos vamos a centrar en la descripción del lenguaje en que hemos basado para hacer nuestra aplicación.

Finalmente, concluiremos el estado del arte explicando las herramientas de desarrollo que se ha utilizado para llevar a cabo el proyecto.

3.1Lenguaje de programación C#

Introducción

C# (pronunciado si sharp en ingles) es un lenguaje de programación orientado a objetos desarrollado y estandarizado por Microft como parte de su plataforma .NET, que después fue aprobado como estándar por la ECMA (ECMA-334) e ISO (ISO/IEC 23270). C# es uno de los lenguajes de programación diseñados para la infraestructura de lenguaje común.

Su sintaxis básica deriva de C/C++ y utiliza el modelo de objetos de la plataforma .NET, similar al de Java, aunque incluye mejoras derivadas de otros lenguajes.

El nombre C Sharp fue inspirado por la notación musical, donde '#' (sostenido, en ingles sharp) indica que la nota (C es la nota do en ingles) es un semitono más alta, sugiriendo que C# es superior a C/C++. Además, el signo '#' se compone de cuatro signos '+' pegados.

Aunque C# forma parte de la plataforma .NET, ésta es una API, mientras que C# es un lenguaje de programación independiente diseñado para generar programas sobre dicha plataforma. Ya existe un compilador implementado que provee el marco Mono -DotGNU, el cual genera programas para distintas plataformas como Windows, Unix, Android, iOS, Windows Phone, Mac OS y GNU/Linux.

Historia

Durante el desarrollo de la plataforma .NET, las bibliotecas de clases fueron escritas originalmente usando un sistema de código gestionado llamado Simple Managed C (SMC). En enero de 1999, Anders Hejlsberg formó un equipo con la misión de desarrollar un nuevo lenguaje de programación llamado Cool (C orientado a objetos). Este

nombre tuvo que ser cambiado debido a problemas de marca, pasando a llamarse C#. La biblioteca de clases de la plataforma .NET fue migrada entonces al nuevo lenguaje. Hejlsberg lideró el proyecto de desarrollo de C#.

3.2Herramientas para el desarrollo del proyecto

Vamos a dar paso a la explicación de todas las herramientas que hemos utilizado para desarrollar nuestro proyecto.

Herramientas como MonoDevelop para poder realizar nuestra aplicación con el lenguaje C# versión 2.8.6.3, la librería gráfica GTK versión 2.12.0.0, para desarrollar interfaces gráficas de usuario y PostgreSQL versión 1.14.0, para poder crear nuestra conexión de la Base de Datos con nuestra aplicación.

Biblioteca GTK

GTK “GIMP Tool Kit” es una conjunto de bibliotecas multiplataforma del equipo GTK+, la cual contiene los objetos y funciones para crear la interfaz gráfica de usuario (GUI). Maneja widgets como ventanas, botones, menús, etiquetas, deslizadores, pestañas, etc.

Inicialmente fueron creadas para desarrollar el programa de edición de imagen GIMP, sin embargo actualmente se usan bastante por muchos programas en los sistemas GNU/Linux. Junto a Qt es una de las bibliotecas más populares para X Window System.

4.Desarrollo

Definición

En concreto, mi aplicación consiste en el desarrollo de una software terminal para los clientes de una cafetería. El software resultante será una aplicación para terminal cliente que tendrá como objetivo que puedan hacer los pedidos de coca-colas, fantas, bocadillos etc... por la terminal, y cuando finalice el pedido se imprima un ticket para el cliente donde informará de la mesa que le corresponde, la fecha, hora, los artículos pedidos con sus precios, el precio total, etc...

Al mismo tiempo se imprimiría uno para el personal donde este llevará a cabo la preparación del pedido del cliente a la mesa identificada por el número que tendrá impreso en el ticket y se cobrará el importe en cuanto el camarero sirva el pedido en la mesa.

4.1Catálogo de usuarios

En esta tarea se identifican los usuarios Participantes y Finales.

Usuario Participantes

•Personal encargado de supervisar el sistema de información, tendrá como función, añadir nuevos artículos en las tablas de la base de datos, cambiar precios, etc...

Usuarios Finales

•Clientes: todas las personas que deseen realizar un pedido en la terminal cliente.
•Personal del local: son todas las personas que conforman el negocio, estos tendrán el objetivo de recoger los pedidos de cada cliente y ofrecérselo al cliente.

El objetivo es que la empresa pueda dar un mejor servicio a los clientes que frecuentan el establecimiento. Consiguiendo que los clientes se adapten a nuevas tecnologías y tengan un buen servicio.

4.2Diseño de Entidad-Relación

En este punto muestro como es el diseño de las tablas de la Base de Datos de la Aplicación. Donde sólo dos tablas tiene relación.

4.3Definición de clases

El objetivo de este punto es describir cada una de las clases que permite la funcionalidad de mi proyecto.

4.3.0Clase MainWindow

Esta la clase principal, es la encargada de que el cliente inicie el pedido y visualmente tendrá un panel de botones con todos los artículos que puede pedir en la cafetería**.**

En esta clase se encuentra la conexión a la base de datos y los eventos de los botones.

•Variables

Label total: es el label donde mostraremos la cantidad de dinero que llevamos acumulado durante el proceso de pedido.
Button buttonNuevoPedido: botón que utilizaremos para inicializar un pedido nuevo.
Gdk.Color fondo: color que le daremos de fondo al botón de Inicio Pedido.
IDbConnection dbConnection: la utilizo para hacer la conexión a la Base de Datos.
static bool HasIniciadoPedido : la utilizo para poder bloquear los botones para pedir los artículos, hasta que el cliente no haga clic en el botón de INICIAR PEDIDO, que cuando su valor cambia a true y se desbloquea los botones

•Constructor

public MainWindow () : Constructor de la clase, aquí estableceremos la conexión a la Base de Datos para extraer toda la información y así poder empezar a utilizar el software, además pondrá el color de fondo del botón “buttonNuevoPedido” y cargará el Label total con un estilo y tamaño que le hemos impuesto y nos mostrará el valor en euros del pedido que tengamos realizado en ese momento.

•Métodos (Eventos)

protected void OnDeleteEvent (object sender, DeleteEventArgs a) :

En esta función se crea por defecto cuando se crea un nuevo proyecto. Sirve para salir de la pantalla principal de MainWindow con el método Quit().
protected void OnBotonBebidasFriasClicked (object sender, System.EventArgs e) :

cuando hace clic el cliente en el botón correspondiente a las bebidas frías para realizar un pedido de esta salta el evento del botón donde se llamará a la función showTablaBebidas();

private void showTablaBebidas() :

en este método creamos un objeto de la clase BebidasView con dos parámetros: total es Label de la pantalla principal donde mostramos el total en euros de nuestro pedido y buttonNuevoPedido es el botón donde obligaremos al cliente primero de todo que inicialice un pedido nuevo, ambos serán pasados por parámetro en el constructor de la clase BebidasView() con el objetivo de que cuando realicemos un pedido de bebidas frías al aceptar dicho pedido el Label total se actualice a vista del cliente en la pantalla de Inicio el importe en euros que lleva el pedido actualmente y el buttonNuevoPedido desaparezca de la pantalla de Inicio para que el cliente no pueda iniciar un nuevo pedido mientras está en trámites de uno.

Ya explicado el paso de parámetros, el método en si nos mostrará información de todas las bebidas Frías como : refrescos, cervezas, batidos (chocolate y vainilla), etc...

protected void OnBotonBebidasCalientesClicked (object sender,System.EventArgs e) :

En este evento creamos un objeto de la clase BebidasCalientesView con dos parámetros: total es Label de la pantalla principal donde mostramos el total en euros de nuestro pedido y buttonNuevoPedido es el botón donde obligaremos al cliente primero de todo que inicialice un pedido nuevo, ambos serán pasados por parámetro en el constructor de la clase BebidasCalientesView(), con el objetivo de que cuando realicemos un pedido de bebidas calinetes al aceptar dicho pedido el Label total se actualice a vista del cliente en la pantalla de Inicio el importe en euros que lleva el pedido actualmente y el buttonNuevoPedido desaparezca de la pantalla de Inicio para que el cliente no pueda iniciar un nuevo pedido mientras está en trámites de uno.
Explicado el paso de parámetros, el método en si nos mostrará información de todas las bebidas calientes como : café, cortado, bombón, etc...

protected void OnBotonBocadillosClicked (object sender, System.EventArgs e):

En este evento creamos un objeto de la clase Bocadillos con dos parámetros: total es Label de la pantalla principal donde mostramos el total en euros de nuestro pedido y buttonNuevoPedido es el botón donde obligaremos al cliente primero de todo que inicialice un pedido nuevo, ambos serán pasados por parámetro en el constructor de la clase Bocadillos(), con el objetivo de que cuando realicemos un pedido de Bocadillos al aceptar dicho pedido el Label total se actualice a vista del cliente en la pantalla de Inicio el importe en euros que lleva el pedido actualmente y el buttonNuevoPedido desaparezca de la pantalla de Inicio para que el cliente no pueda iniciar un nuevo pedido mientras está en trámites de uno.

Explicado el paso de parámetros, el método en si nos mostrará información de todos los bocadillos disponible como : atún, jamón a la catalana, tortilla de patata, etc...

protected void OnBotonAlmuerzoCompletoClicked (object sender, System.EventArgs e) :

En este evento creamos un objeto de la clase AlmuerzoCompleto con dos parámetros: total es Label de la pantalla principal donde mostramos el total en euros de nuestro pedido y buttonNuevoPedido es el botón donde obligaremos al cliente primero de todo que inicialice un pedido nuevo, ambos serán pasados por parámetro en el constructor de la clase AlmuerzoCompleto(), con el objetivo de que cuando realicemos un pedido de AlmuerzoCompleto al aceptar dicho pedido el Label total se actualice a vista del cliente en la pantalla de Inicio el importe en euros que lleva el pedido actualmente y el buttonNuevoPedido desaparezca de la pantalla de Inicio para que el cliente no pueda iniciar un nuevo pedido mientras está en trámites de uno.

Explicado el paso de parámetros, el método en si nos mostrará información de todos los cafés, bocadillos y bebidas disponible como : cafés: cortado, bombón, bocadillos: atún, tortilla, bebidas: refrescos, cervezas etc...

protected void OnBotonAlmuerzoParteClicked (object sender, System.EventArgs e) :

En este evento creamos un objeto de la clase AlmuerzoParteView con dos parámetros: total es Label de la pantalla principal donde mostramos el total en euros de nuestro pedido y buttonNuevoPedido es el botón donde obligaremos al cliente primero de todo que inicialice un pedido nuevo, ambos serán pasados por parámetro en el constructor de la clase AlmuerzoParteView(), con el objetivo de que cuando realicemos un pedido de AlmuerzoCompleto al aceptar dicho pedido el Label total se actualice a vista del cliente en la pantalla de Inicio el importe en euros que lleva el pedido actualmente y el buttonNuevoPedido desaparezca de la pantalla de Inicio para que el cliente no pueda iniciar un nuevo pedido mientras está en trámites de uno.

Explicado el paso de parámetros, el método en si nos mostrará información de todos los cafés, bebidas disponible como : cafés: cortado, bombón, bebidas: refrescos, cervezas etc...

protected void OnBotonDesayunoPopularClicked (object sender, System.EventArgs e):

En este evento creamos un objeto de la clase DesayunoPopularView con dos parámetros: total es Label de la pantalla principal donde mostramos el total en euros de nuestro pedido y buttonNuevoPedido es el botón donde obligaremos al cliente primero de todo que inicialice un pedido nuevo, ambos serán pasados por parámetro en el constructor de la clase DesayunoPopularView(), con el objetivo de que cuando realicemos un pedido de DesayunoPopularView al aceptar dicho pedido el Label total se actualice a vista del cliente en la pantalla de Inicio el importe en euros que lleva el pedido actualmente y el buttonNuevoPedido desaparezca de la pantalla de Inicio para que el cliente no pueda iniciar un nuevo pedido mientras está en trámites de uno.
Explicado el paso de parámetros, el método en si nos mostrará información de la bebidas que por defecto sería café con leche, la comida que tendría que elegir entre: tostada (tomate o aceite) o croissant.

protected void OnBotonBolleriaDulceClicked (object sender, System.EventArgs e):

En este evento creamos un objeto de la clase BolleriaDulceView con dos parámetros: total es Label de la pantalla principal donde mostramos el total en euros de nuestro pedido y buttonNuevoPedido es el botón donde obligaremos al cliente primero de todo que inicialice un pedido nuevo, ambos serán pasados por parámetro en el constructor de la clase BolleriaDulceView(), con el objetivo de que cuando realicemos un pedido de BolleriaDulceView al aceptar dicho pedido el Label total se actualice a vista del cliente en la pantalla de Inicio el importe en euros que lleva el pedido actualmente y el buttonNuevoPedido desaparezca de la pantalla de Inicio para que el cliente no pueda iniciar un nuevo pedido mientras está en trámites de uno.
Explicado el paso de parámetros, el método en si nos mostrará información de todas las bollerías dulce disponible como : croissant normal, croissant de choco, magdalena, etc...

protected void OnBotonBolleriaSaladaClicked (object sender, System.EventArgs e):

En este evento creamos un objeto de la clase BolleriaSaladaView con dos parámetros: total es Label de la pantalla principal donde mostramos el total en euros de nuestro pedido y buttonNuevoPedido es el botón donde obligaremos al cliente primero de todo que inicialice un pedido nuevo, ambos serán pasados por parámetro en el constructor de la clase BolleriaSaladaView(), con el objetivo de que cuando realicemos un pedido de BolleriaSaladaView al aceptar dicho pedido el Label total se actualice a vista del cliente en la pantalla de Inicio el importe en euros que lleva el pedido actualmente y el buttonNuevoPedido desaparezca de la pantalla de Inicio para que el cliente no pueda iniciar un nuevo pedido mientras está en trámites de uno.
Explicado el paso de parámetros, el método en si nos mostrará información de todas las bollerías saladas disponible como : croissant vegetal, panines (york o queso), montadidos, etc...

protected void OnBotonticketClicked (object sender, System.EventArgs e) :

En este evento creamos un objeto de la clase TicketView y le pasamos el Label “total” y el Button “buttonNuevoPedido” por parámetro, total es Label de la pantalla principal donde mostramos el total en euros de nuestro pedido y buttonNuevoPedido es el botón donde obligaremos al cliente primero de todo que inicialice un pedido nuevo. Los recogemos en la clase TicketView para ser enviado por parámetro al objeto de la clase ImprimirTicket.
En la clase ImprimirTicket es donde los vamos a utilizar, cuando el cliente imprima el ticket, el botón de Iniciar Pedido vuelva ser visible en el MainWindow, con el método Visible=true; . Utilizo el Label “total” en la clase ImprimirTicket para que así pueda actualizar el Label de la clase de MainWindow poniendo el Label total a ZERO.
Finalmente con el método show(), mostramos la pantalla que corresponde a la clase TicketView.

protected void OnButtonNuevoPedidoClicked (object sender, System.EventArgs e) :

En este evento creo un objeto de la clase AñadirNumPersonas y muestro su pantalla corresponde con el método Show().

Hago una consulta para eliminar la tabla de pedidos, una vez que sea pulsado éste evento. Así no habrá problemas que coincida con pedidos antiguos.
El Label “total” se inicia a ZERO, con el método Markup(), cambio el tamaño y color de la fuente del Label.
Oculto el Button buttonNuevoPedido con el método Visible=false, para que hasta que no sea impreso el ticket, los clientes no tengan disponible el botón.

protected void OnBotonAyudaClicked (object sender, System.EventArgs e)

en este evento llamamos a la clase de la ayuda para los clientes de la cafetería.

4.3.1Clase BebidasView

- **Variables Global**

  IDbConnection dbConnection: la utilizo para hacer la conexión a la base de datos.

  ListStore listStore:

  Label total: es el label donde mostraremos la cantidad de dinero que llevamos acumulado durante el proceso de pedido.

  Button buttonNuevoPedido: botón que utilizaremos para inicializar un pedido nuevo.

- **Variable Local**

  IDbCommand dbCommand: es una interfaz donde se crea la conexión con una instancia del objeto dbConnection a la Base de Datos y hago la consulta sql, con la propiedad del objeto “CommandText”.

  IDataReader dataReader: es una interfaz donde recupero las filas de la consulta de la Base de Datos haciendo una instancia al objeto “dbCommand”.

- **Constructor**

public BebidasView (Label labelTotal,Button botonNP) :
aquí estableceremos la conexión a la Base de Datos para extraer toda la información y hago una consulta pidiendo todos los campos de la tabla de bebidasfrias para mostrarlos cuando el cliente haga el clic en este evento. Recogo el button y el label obtenido por el constructor de la clase para pasarlos a la clase CalculoLabelMain, que es donde hago el cálculo de los artículos que tengo en la tabla de pedidos y actualizo el label con dicho total en Euros, cambio el tamaño y el ancho del label con la propiedad Markup. Y con el button lo que hago es ocultarlo hasta que el cliente termine de hacer su pedido y haga un clic en imprimir ticket para que el cliente no inicie un nuevo pedido sin haber terminado el actual.
Hago un objeto de la clase RellenarTreeView para poder llamar al método de esta clase y pasarle el treeView y el dataReader. Recojo el listStore que uso en la clase RellenarTreeView, para que pueda usar en el evento del botón “Aceptar”.
Al finalizar el uso del dataReader en el constructor, cierro el dataReader con el método Close().

- **Métodos (eventos)**

protected void OnBotonAceptarClicked (object sender, System.EventArgs e):

Cuando el cliente hace clic en éste evento, se selecciona todas las filas sin ser vista esta selección por el cliente, para obtener la información de todas ellas. Pero sólo se guardará en la Base de Datos (en la tabla de pedidos)la fila donde la cantidad sea distinta a cero, si ello se cumple se insertará a continuación en la base de datos.
Hago un objeto de la clase CalculoLabelMain y llamo al método corresponde a esa clase pasandóle por parámetros el label y el button que los recogo del constructor.
Posteriormente, quito la pantalla con el método Detroy(), dejando sólo la principal a la vista.

protected void OnBotonInicioClicked (object sender, System.EventArgs e):

Cuando el cliente hace un clic en éste evento, lo que hago es llamar al método Destroy(), para que cierre la pantalla actual, dejando a la vista la pantalla principal.

protected void OnTreeViewRowActivated (object o, RowActivatedArgs args):

Utilizo éste evento para que cuando el cliente haga un doble clic en la fila del treeView nos muestre otra pantalla que será la encargada de recoger la cantidad que el usuario introduzca.
Hago un objeto de la clase SeleccionCantidad y le paso por parámetro el treeView, la cantidad, el tipo de dato(integer) y el label. Y llamo al método Show() para que muestre la pantalla de esta clase.

protected void OnBotonEliminarClicked (object sender, System.EventArgs e):

Cuando un cliente haga un clic en este evento, se pondrá a ZERO el valor del campo cantidad.

4.3.2Clase AñadirNumPersonas

- **Variables Global**

  IDbConnection dbConnection: la utilizo para hacer la conexión a la base de datos.

  Label totalMain: es el label donde mostraremos la cantidad de dinero que llevamos acumulado durante el proceso de pedido.

  Button botonNuevoPedido: botón que utilizaremos para inicializar un pedido nuevo.

- **Constructor**

public AñadirNumPersonas (Label labelTotalMainWindow,Button botonNP) :
este método lo único que hace es asignar las variables obtenida por parámetro y asignarlas a las variables globales.

- **Métodos**

protected void OnBotonAceptarClicked (object sender, System.EventArgs e)

cuando el cliente haga clic en este evento inicializamos sesión.
Hacemos una consulta a la base de datos para para obtener el máximo de la tabla mesa. Para así poder calcular las mesas disponibles para los números de personas introducido por el cliente en el método que hacemos la llamada y le pasamos dos parámetros, para el cálculo.

public void calculoMesa(double numPersonas,int max)

en este método miraríamos si hay mesas disponibles, no habiendo mostraríamos una información, para que el cliente intente otra vez más tarde y habiendo asignaríamos el identificador de la mesa al ticket y la mesa pasaría de estar disponible a no estarlo. El cliente ya podrá hacer su pedido.
protected void OnBotonCancelarClicked (object sender, System.EventArgs e)

en este evento actualizamos el label, ponemos a visible el botón de INICIAR SESIÓN y quitamos la pantalla.

4.3.3Clase ImprimirTicket

- **Variables Global**

IDbConnection dbConnection: la utilizo para hacer la conexión a la base de datos.

Label totalMain: es el label donde mostraremos la cantidad de dinero que llevamos acumulado durante el proceso de pedido.

Button botonNuevoPedido: botón que utilizaremos para inicializar un pedido nuevo.

- **Variables Local**

IDbCommand dbCommand: es una interfaz donde crearía la conexión con una instancia del objeto dbConnection a la Base de Datos y hago la consulta sql, con la propiedad del objeto “CommandText”.

IDataReader dataReader: es una interfaz donde recupero las filas de la consulta de la Base de Datos haciendo una instancia al objeto “dbCommand”.

- **Constructor**

public ImprimirTicket (Label labelTotal,Button botonNP) :
En el constructor obtengo el label y button pasados por parámetro y las asigno a las variables globales creadas para ser usadas en el evento del botón Aceptar.

Método (evento)

protected void OnBotonAceptarClicked (object sender, System.EventArgs e):

Cuando el cliente hace clic en este evento, si hará la conexión a la Base de Datos con la instancia a la clase ApplicationContext.
Se creará la conexión con la Base de Datos y se realizará la eliminación de los datos de la tabla pedidos, evitando que se mezcle con artículos de pedidos futuros, ya que no queremos guardar ningún historial de ésta tabla.
Se actualizará el label con el valor TOTAL en Euros a ZERO y se pondrá a visible el botón INICIAR PEDIDO.
Al finalizar lo anterior, se cerrará esta pantalla, dejando visible la principal.

4.3.4Clase SeleccionCantidad

•Variables Global

IDbConnection dbConnection: la utilizo para hacer la conexión a la base de datos.

Label labelPrecioTotal: es el label que utilizamos para mostrar el total.

Int pant: es la variable que utilizo para saber de que pantalla viene.
TreeView treeview: es el treeview que utilizamos para poder poner la cantidad.

•Variables Local

IDbCommand dbCommand: es una interfaz donde crearía la conexión con una instancia del objeto dbConnection a la Base de Datos y hago la consulta sql, con la propiedad del objeto “CommandText”.

•Constructor

public SeleccionCantidad (TreeView TV, string cantidad, int pantalla, Label labelPrecio) :
en el constructor hacemos la instancia a la conexión con la base de datos, asignamos las variables obtenida por parámetro a las variables globales y obtenemos el valor de la cantidad elegida por el cliente(spinbutton).

•Métodos

protected void OnBotonAceptarClicked (object sender, EventArgs e)

en este evento comprobamos de que pantalla viene y dependiendo de cual venga llamaremos un método u otro.
public void SeleccionCantidadNoTicket()

este es el método para recibir a las pantallas que no sea la de ticket.
En ella obtengo el valor de la lista de cantidad la convierto a string y muestro el nuevo valor elegido por el cliente.

public void SeleccionCantidadTicket()

este método es para recibir la pantalla que viene de ticket. Es donde obtiene los valores de las cantidades y actualiza la base de datos con los nuevos valores. Posteriormente llamaremos el método totalTicket(labelPrecioTotal) pasado por parámetro el label, para cálculo total del valor.

public void totalTicket(Label labelPrecioT)
en este método hago una consulta para obtener el precio y la cantidad de la tabla de pedidos, previamente actualizados, hacemos el cálculo de la cantidad de artículos por su precio y actualizamos el label, con el valor del precio total apagar.

protected void OnBotonCancelarClicked (object sender, EventArgs e)

en este evento salimos de la pantalla en la que estamos. Dejando visible la principal.

4.3.5Clases TicketView

•Variables Global

IDbConnection dbConnection: la utilizo para hacer la conexión a la base de datos.

ListStore liststore:

Label totalMain: es el label donde mostraremos la cantidad de dinero que llevamos acumulado durante el proceso de pedido.

Button botonNuevoPedido: botón que utilizaremos para inicializar un pedido nuevo.

•Variables Local

IDbCommand dbCommand: es una interfaz donde crearía la conexión con una instancia del objeto dbConnection a la Base de Datos y hago la consulta sql, con la propiedad del objeto “CommandText”.
IDataReader dataReader: es una interfaz donde recupero las filas de la consulta de la Base de Datos haciendo una instancia al objeto “dbCommand”.

•Constructor

public TicketView (Label labelMain, Button botonNP) :
En el constructor obtengo el label y button pasados por parámetro y las asigno a las variables globales creadas para ser usadas en el método totalTicket() y por el evento botón Aceptar. Establezco la conexión a la Base de Datos para extraer toda la información y hago una consulta pidiendo todos los campos de la tabla de pedidos para mostrarlos cuando el cliente esté en la pantalla de TicketView. Posteriormente llamo a la clase mostrarTablaPedidos y le paso por parámetro el treeview y el datareader. Al finalizar el uso del dataReader en el constructor, cierro el dataReader con el método Close() y llamo el método de totalTicket().

•Métodos (eventos)

protected void OnBotonAceptarClicked (object sender, System.EventArgs e):
En este evento creo un objeto de la clase Ticket y le paso por parámetro el label y button recogido en el constructor de la clase. Posteriormente llamo al método Show() para que me muestre la pantalla de imprimirTicket.

protected void OnBotonCancelarClicked (object sender, System.EventArgs e):
Cuando el cliente hace clic en este evento y haya cambiado el valor del campo cantidad siendo distinto de ZERO, éste volverá a su valor por defecto ZERO.

public void mostrarTablaPedidos(TreeView treeview,IDataReader datareader):

En este método lo que hago es llamar al método AppendColumns y pasarle por parámetro el treeview y el datareader. En un vector de tipo obtengo los tipos de los datos de los campos de la tabla pedidos, pasándolos a string.Posteriormente paso el vector de tipos al constructor del objeto liststore y después asigno el liststore al modelo del treeview, para que muestre los datos con los tipos obtenido del vector y que se vea como una lista. Y llamo el método Fill pasándole los tipos de los datos obtenido de la base de datos.

public void Fill(IDataReader datareader):

Con un bucle while leo los datos obtenido de la consulta de la Base de datos y en un bucle for adiciono los datos obtenido por la consulta en una lista de string y pongo los valores a la lista(liststore).

public void AppendColumns(TreeView treeView, IDataReader dataReader):
Este método recibe por parámetro el treeView y el dataReader para ser usados en este método. En este método añado las columnas a las celda del treeView y pongo oculto la columna del id. Porque es un campo no necesario para los clientes.
Posteriormente añado a mano la columna de la cantidad y el título de su columna, porque este campo no existe en la Base de datos.

public static Type[] GetTypes(Type type, int count):

método que devuelve el tipo a rellenar el treeview.

public void totalTicket():

En este método inicio una variable de tipo double a ZERO. Hago una consulta de la Base de Datos para obtener los campos de precio y cantidad de la tabla de pedidos y hago el cálculo cuando un cliente pone una cantidad en el artículo para que el label total actualice con su nuevo valor en Euros.

protected void OnTreeviewRowActivated (object o,RowActivatedArgs args):

Utilizo éste evento para que cuando el cliente haga un doble clic en la fila del treeView nos muestre una pantalla que será la encargada de recoger la cantidad que el usuario haya introducido. Hago un objeto de la clase SeleccionCantidad y le paso por parámetro el treeView, la cantidad, el tipo de dato(integer) y el label. Y llamo al método Show() para que muestre la pantalla de esta clase.

protected void OnBotonEliminarClicked (object sender, System.EventArgs e)

en este evento eliminaremos todos los datos de la tabla de pedidos y actualizamos el label a ZERO para que cuando vuelva a iniciar pedido el total esté a ZERO y el botón de INICIAR PEDIDO visible. La variable de iniciar sesión la ponemos a falso, para que el cliente vuelva a iniciar sesión.

4.3.6Clase ApplicationContext

- **Variables Global**

ApplicationContext instance:
IDbConnection dbConnection:

- **Métodos**

public static ApplicationContext Instance: en este método obtengo la instancia creada de la aplicación.

public IDbConnection DbConnection : Utilizo para crear una instancia a la Base de Datos.

4.3.7Clase DbCommandExtensions

- **Métodos**

public static void AddParameter(IDbCommand dbCommand, string name, object value):

Este método lo utilizo para cuando hago una sentencia de insert o update. Para poder pasar los valores que están en variables a las sentencias, tanto de insert, como la de update.

4.3.8Clase AlmuerzoCompleto

•Variables Global

IDbConnection dbConnection: la utilizo para hacer la conexión a la base de datos.

String valorComboBebida, valorComboBocadillo, valorComboCafe: guardo los valores introducidos por los clientes cuando elige una opción comboBox.

Label totalMainWindow: es el label donde mostraremos la cantidad de dinero que llevamos acumulado durante el proceso de pedido.

Button botonNuevoPedidoMainWindow: botón que utilizaremos para inicializar un pedido nuevo.

•Constructor

public AlmuerzoCompleto (Label labelTotal,Button botonNP) :

En el constructor cambio el tamaño y el ancho del label de la clase y le pongo un texto. Asigno los el label y el button recogido del constructor de la clase para ser pasados a la clase CalculoLabelMain. Posteriormente instancio la conexión a la Base de Datos y llamo los métodos:

cargarComboBebidas(),cargarComboBocadillo(),cargarComboPrecio(),cargarComboCafe( ).

•Métodos

public void cargarComboBebidas():

En este método hago una consulta a la Base de Datos para obtener los datos del campo nombre de la tabla de bebidasfrias y mostrar dichos valores por el comboBox, y pongo que siempre aparezca el mensaje de “Seleccione ...” por defecto en el combo.

public void cargarComboBocadillo():

En este método hago una consulta a la Base de Datos para obtener los datos del campo nombre de la tabla de bocadillos y mostrar dichos valores por el comboBox, y pongo que siempre aparezca el mensaje de “Seleccione ...” por defecto en el combo.

public void cargarComboCafe():

En este método hago una consulta a la Base de Datos para obtener los datos del campo nombre de la tabla de bebidascalientes y mostrar dichos valores por el comboBox, y pongo que siempre aparezca el mensaje de “Seleccione ...” por defecto en el combo.

public void cargarComboPrecio():

En este método hago una consulta a la Base de Datos para obtener el dato del campo precio de la tabla de almuerzocompleto y mostrar dicho valor por el comboBox, y pongo que siempre aparezca el mensaje de “Seleccione ...” por defecto en el combo.

protected void OnBotonAtrasClicked (object sender, System.EventArgs e)
Cuando el cliente hace clic en este evento, sale de él quedando sólo la pantalla principal.

protected void OnBotonAceptarClicked (object sender,System.EventArgs e):

Cuando el cliente hace clic en este evento, se guarda los valores obtenido por los comboBox sólo se dichos valores son distintos del valor por defecto y si no sale un mensaje de alerta, informando al cliente que tiene que elegir todos los campos. Posteriormente hago un objeto de la clase calculoLabel y le paso el label y el total para que el label actualice su valor .

protected void OnBotonCancelarClicked (object sender,System.EventArgs e):

Cuando el cliente hace clic en este evento, se cancela el pedido, poniendo los valores de los comboBox con su valor por defecto.

protected void OnComboboxBebidaChanged (object sender,System.EventArg e):

Evento del comboBox de Bebida, para obtener el valor elegido por el cliente.

protected void OnComboboxBocadilloChanged (object sender,System.EventArgs e)

Evento del comboBox de Bocadillos, para obtener el valor elegido por el cliente.

protected void OnComboboxCafeChanged (object sender, System.EventArgs e)

Evento del comboBox de Cafés, para obtener el valor elegido por el cliente.

4.3.9Clase AlmuerzoParteView

- **Variable Global**

IDbConnection dbConnection: la utilizo para hacer la conexión a la base de datos.

String valorComboBebida, valorComboCafe: guardo los valores introducidos por los clientes cuando elige una opción comboBox.

Label totalMainWindow: es el label donde mostraremos la cantidad de dinero que llevamos acumulado durante el proceso de pedido.

Button botonNuevoPedidoMainWindow: botón que utilizaremos para inicializar un pedido nuevo.

- **Constructor**

public AlmuerzoParteView (Label labelTotalMainWindow,Button botonNP) :

En el constructor cambio el tamaño y el ancho del label de la clase y le pongo un texto. Asigno los el label y el button recogido del constructor de la clase para ser pasados a la clase CalculoLabelMain. Posteriormente instancio la conexión a la Base de Datos y llamo los métodos: cargarComboBebidas(),cargarComboPrecio(),cargarComboCafe().

- **Métodos**

  public void cargarComboBebidas():

  En este método hago una consulta a la Base de Datos para obtener los datos del campo nombre de la tabla de bebidasfrias y mostrar dichos valores por el comboBox, y pongo que siempre aparezca el mensaje de “Seleccione ...” por defecto en el combo.

  public void cargarComboCafe():

  En este método hago una consulta a la Base de Datos para obtener los datos del campo nombre de la tabla de bebidascalientes y mostrar dichos valores por el comboBox, y pongo que siempre aparezca el mensaje de “Seleccione ...” por defecto en el combo.

public void cargarComboPrecio():

En este método hago una consulta a la Base de Datos para obtener el dato del campo precio de la tabla de almuerzocompleto y mostrar dicho valor por el comboBox, y pongo que siempre aparezca el mensaje de “Seleccione ...” por defecto en el combo.

protected void OnBotonAtrasClicked (object sender, System.EventArgs e)

Cuando el cliente hace clic en este evento, sale de él quedando sólo la pantalla principal.

protected void OnBotonAceptarClicked (object sender, System.EventArgs e):

Cuando el cliente hace clic en este evento, se guarda los valores obtenido por los comboBox sólo se dichos valores son distintos del valor por defecto y si no sale un mensaje de alerta, informando al cliente que tiene que elegir todos los campos. Posteriormente hago un objeto de la clase calculoLabel y le paso el label y el total para que el label actualice su valor .

protected void OnBotonCancelarClicked (object sender, System.EventArgs e):

Cuando el cliente hace clic en este evento, se cancela el pedido, poniendo los valores de los comboBox con su valor por defecto.

protected void OnComboboxBebidaChanged (object sender, System.EventArg e):

Evento del comboBox de Bebida, para obtener el valor elegido por el cliente.

protected void OnComboboxCafeChanged (object sender, System.EventArgs e)

Evento del comboBox de Cafés, para obtener el valor elegido por el cliente.

4.3.10Clase BebidasCalientes

- **Variables Global**

  IDbConnection dbConnection: la utilizo para hacer la conexión a la base de datos.
  Label totalMainWindow: es el label donde mostraremos la cantidad de dinero que llevamos acumulado durante el proceso de pedido.

  Button botonNuevoPedidoMainWindow: botón que utilizaremos para inicializar un pedido nuevo.

- **Variables Local**

  IDbCommand dbCommand: es una interfaz donde se crea la conexión con una instancia del objeto dbConnection a la Base de Datos y hago la consulta sql, con la propiedad del objeto “CommandText”.

  IDataReader dataReader: es una interfaz donde recupero las filas de la consulta de la Base de Datos haciendo una instancia al objeto “dbCommand”.

- **Constructor**

public BebidasCalientesView (Label labelTotalMainWindow,Button botonNP) :

aquí estableceremos la conexión a la Base de Datos para extraer toda la información y hago una consulta pidiendo todos los campos de la tabla de bebidascalientes para mostrarlos cuando el cliente haga el clic en este evento.
Recogo el button y el label obtenido por el constructor de la clase para pasarlos a la clase CalculoLabelMain, que es donde hago el cálculo de los artículos que tengo en la tabla de pedidos y actualizo el label con dicho total en Euros, cambio el tamaño y el ancho del label con la propiedad Markup. Y con el button lo que hago es ocultarlo hasta que el cliente termine de hacer su pedido y haga un clic en imprimir ticket para que el cliente no inicie un nuevo pedido sin haber terminado el actual.

Hago un objeto de la clase RellenarTreeView para poder llamar al método de esta clase y pasarle el treeView y el dataReader. Recojo el listStore que uso en la clase RellenarTreeView, para que pueda
usar en el evento del botón “Aceptar”.
Al finalizar el uso del dataReader en el constructor, cierro el dataReader con el método Close().

- **Métodos (eventos)**

protected void OnBotonAceptarClicked (object sender, System.EventArgs e):

Cuando el cliente hace clic en éste evento, se selecciona todas las filas sin ser vista esta selección por el cliente, para obtener la información de todas ellas. Pero sólo se guardará en la Base de Datos (en la tabla de pedidos)la fila donde la cantidad sea distinta a cero, si ello se cumple se insertará a continuación en la base de datos. Hago un objeto de la clase CalculoLabelMain y llamo al método corresponde a esa clase pasandóle por parámetros el label y el button que los recogo del constructor.
Posteriormente, quito la pantalla con el método Detroy(), dejando sólo la principal a la vista.

protected void OnBotonInicioClicked (object sender, System.EventArgs e):

Cuando el cliente hace un clic en éste evento, lo que hago es llamar al método Destroy(), para que cierre la pantalla actual, dejando a la vista la pantalla principal.

protected void OnTreeViewRowActivated (object o, RowActivatedArgs args):

Utilizo éste evento para que cuando el cliente haga un doble clic en la fila del treeView nos muestre otra pantalla que será la encargada de recoger la cantidad que el usuario introduzca.
Hago un objeto de la clase SeleccionCantidad y le paso por parámetro el treeView, la cantidad, el tipo de dato(integer) y el label. Y llamo al método Show() para que muestre la pantalla de esta clase.

protected void OnBotonEliminarClicked (object sender, System.EventArgs e):

Cuando un cliente haga un clic en este evento, se pondrá a ZERO el valor del campo cantidad.

4.3.11Clase Bocadillos

- **Variables Global**

  IDbConnection dbConnection: la utilizo para hacer la conexión a la base de datos.

  ListStore listStore:

  Label totalMainWindow: es el label donde mostraremos la cantidad de dinero que llevamos acumulado durante el proceso de pedido.

  Button botonNuevoPedidoMainWindow: botón que utilizaremos para inicializar un pedido nuevo.

- **Variables Local**

  IDbCommand dbCommand: es una interfaz donde se crea la conexión con una instancia del objeto dbConnection a la Base de Datos y hago la consulta sql, con la propiedad del objeto “CommandText”.

  IDataReader dataReader: es una interfaz donde recupero las filas de la consulta de la Base de Datos haciendo una instancia al objeto “dbCommand”.

- **Constructor**

public Bocadillos (Label labelTotalMainWindow,Button botonNP) :

aquí estableceremos la conexión a la Base de Datos para extraer toda la información y hago una consulta pidiendo todos los campos de la tabla de bocadillos para mostrarlos cuando el cliente haga el clic en este evento.

Recogo el button y el label obtenido por el constructor de la clase para pasarlos a la clase CalculoLabelMain,que es donde hago el cálculo de los artículos que tengo en la tabla de pedidos y actualizo el label con dicho total en Euros, cambio el tamaño y el ancho del label con la propiedad Markup. Y con el button lo que hago es ocultarlo hasta que el cliente termine de hacer su pedido y haga un clic en imprimir ticket para que el cliente no inicie un nuevo pedido sin haber terminado el actual.
Hago un objeto de la clase RellenarTreeView para poder llamar al método de esta clase y pasarle el treeView y el dataReader.
Recojo el listStore que uso en la clase RellenarTreeView, para que pueda usar en el evento del botón “Aceptar”. Al finalizar el uso del dataReader en el constructor, cierro el dataReader con el método Close().

- **Métodos (eventos)**

protected void OnBotonAtrasClicked (object sender, System.EventArgs e):

Cuando el cliente hace un clic en éste evento, lo que hago es llamar al método Destroy(), para que cierre la pantalla actual, dejando a la vista la pantalla principal.

protected void OnBotonAceptarClicked (object sender, System.EventArgs e):

Cuando el cliente hace clic en éste evento, se selecciona todas las filas sin ser vista esta selección por el cliente, para obtener la información de todas ellas. Pero sólo se guardará en la Base de Datos (en la tabla de pedidos)la fila donde la cantidad sea distinta a cero, si ello se cumple se insertará a continuación en la base de datos. Hago un objeto de la clase CalculoLabelMain y llamo al método corresponde a esa clase pasandóle por parámetros el label y el button que los recogo del constructor.
Posteriormente, quito la pantalla con el método Detroy(), dejando sólo la principal a la vista.

protected void OnBotonEliminarClicked (object sender, System.EventArgs e):

Cuando un cliente haga un clic en este evento, se pondrá a ZERO el valor del campo cantidad.

protected void OnTreeViewRowActivated (object o, Gtk.RowActivatedArgs args):

Utilizo éste evento para que cuando el cliente haga un doble clic en la fila del treeView nos muestre otra pantalla que será la encargada de recoger la cantidad que el usuario introduzca. Hago un objeto de la clase SeleccionCantidad y le paso por parámetro el treeView, la cantidad, el tipo de dato(integer) y el label. Y llamo al método Show() para que muestre la pantalla de esta clase.

4.3.12Clase BolleriaDulceView

- **Variables Global**

IDbConnection dbConnection: la utilizo para hacer la conexión a la base de datos.

Label totalMainWindo: es el label donde mostraremos la cantidad de dinero que llevamos acumulado durante el proceso de pedido.

Button botonNuevoPedidoMainWindow: botón que utilizaremos para inicializar un pedido nuevo.

- **Variables Local**

IDbCommand dbCommand: es una interfaz donde se crea la conexión con una instancia del objeto dbConnection a la Base de Datos y hago la consulta sql, con la propiedad del objeto “CommandText”.

IDataReader dataReader: es una interfaz donde recupero las filas de la consulta de la Base de Datos haciendo una instancia al objeto “dbCommand”.

- **Constructor**

public BolleriaDulceView (Label labelTotalMainWindow,Button botonNP) :

aquí estableceremos la conexión a la Base de Datos para extraer toda la información y hago una consulta pidiendo todos los campos de la tabla de bolleriadulce para mostrarlos cuando el cliente haga el clic en este evento. Recogo el button y el label obtenido por el constructor de la clase para pasarlos a la clase CalculoLabelMain, que es donde hago el cálculo de los artículos que tengo en la tabla de pedidos y actualizo el label con dicho total en Euros, cambio el tamaño y el ancho del label con la propiedad Markup. Y con el button lo que hago es ocultarlo hasta que el cliente termine de hacer su pedido y haga un clic en imprimir ticket para que el cliente no inicie un nuevo pedido sin haber terminado el actual. Hago un objeto de la clase RellenarTreeView para poder llamar al método de esta clase y pasarle el treeView y el dataReader. Recojo el listStore que uso en la clase RellenarTreeView, para que pueda usar en el evento del botón “Aceptar”. Al finalizar el uso del dataReader en el constructor, cierro el dataReader con el método Close().

- **Métodos (eventos)**

protected void OnBotonAtrasClicked (object sender, System.EventArgs e):

Cuando el cliente hace un clic en éste evento, lo que hago es llamar al método Destroy(), para que cierre la pantalla actual, dejando a la vista la pantalla principal.

protected void OnBotonAceptarClicked (object sender, System.EventArgs e):

Cuando el cliente hace clic en éste evento, se selecciona todas las filas sin ser vista esta selección por el cliente, para obtener la información de todas ellas. Pero sólo se guardará en la Base de Datos (en la tabla de pedidos)la fila donde la cantidad sea distinta a cero, si ello se cumple se insertará a continuación en la base de datos.

Hago un objeto de la clase CalculoLabelMain y llamo al método corresponde a esa clase pasándole por parámetros el label y el button que los recojo del constructor. Posteriormente, quito la pantalla con el método Detroy(), dejando sólo la principal a la vista.

protected void OnBotonEliminarClicked (object sender, System.EventArgs e)

Cuando un cliente haga un clic en este evento, se pondrá a ZERO el valor del campo cantidad.

protected void OnTreeViewRowActivated (object o, RowActivatedArgs args):

Utilizo éste evento para que cuando el cliente haga un doble clic en la fila del treeView nos muestre otra pantalla que será la encargada de recoger la cantidad que el usuario introduzca. Hago un objeto de la clase SeleccionCantidad y le paso por parámetro el treeView, la cantidad, el tipo de dato(integer) y el label. Y llamo al método Show() para que muestre la pantalla de esta clase.

4.3.13Clase BolleriaSaladaView

- **Variables Global**

IDbConnection dbConnection: la utilizo para hacer la conexión a la base de datos.

ListStore listStore:

Label totalMainWindow: es el label donde mostraremos la cantidad de dinero que llevamos acumulado durante el proceso de pedido.

Button botonNuevoPedidoMainWindow: botón que utilizaremos para inicializar un pedido nuevo.

- **Variables Local**

  IDbCommand dbCommand: es una interfaz donde se crea la conexión con una instancia del objeto dbConnection a la Base de Datos y hago la consulta sql, con la propiedad del objeto “CommandText”.
  IDataReader dataReader: es una interfaz donde recupero las filas de la consulta de la Base de Datos haciendo una instancia al objeto “dbCommand”.

- **Constructor**

  public BolleriaSaladaView (Label labelTotalMainWindow,Button botonNP) :

  aquí estableceremos la conexión a la Base de Datos para extraer toda la información y hago una consulta pidiendo todos los campos de la tabla de bolleriasalada para mostrarlos cuando el cliente haga el clic en este evento. Recogo el button y el label obtenido por el constructor de la clase para pasarlos a la clase CalculoLabelMain, que es donde hago el cálculo de los artículos que tengo en la tabla de pedidos y actualizo el label con dicho total en Euros, cambio el tamaño y el ancho del label con la propiedad Markup. Y con el button lo que hago es ocultarlo hasta que el cliente termine de hacer su pedido y haga un clic en imprimir ticket para que el cliente no inicie un nuevo pedido sin haber terminado el actual. Hago un objeto de la clase RellenarTreeView para poder llamar al método de esta clase y pasarle el treeView y el dataReader.Recojo el listStore que uso en la clase RellenarTreeView, para que pueda usar en el evento del botón “Aceptar”. Al finalizar el uso del dataReader en el constructor, cierro el dataReader con el método Close().

- **Métodos (eventos)**
  protected void OnBotonAtrasClicked (object sender, System.EventArgs e):

  Cuando el cliente hace un clic en éste evento, lo que hago es llamar al método Destroy(), para que cierre la pantalla actual, dejando a la vista la pantalla principal.

protected void OnBotonAceptarClicked (object sender, System.EventArgs e):

Cuando el cliente hace clic en éste evento, se selecciona todas las filas sin ser vista esta selección por el cliente, para obtener la información de todas ellas. Pero sólo se guardará en la Base de Datos (en la tabla de pedidos)la fila donde la cantidad sea distinta a cero, si ello se cumple se insertará a continuación en la base de datos. Hago un objeto de la clase CalculoLabelMain y llamo al método corresponde a esa clase pasándole por parámetros el label y el button que los recogo del constructor.
Posteriormente, quito la pantalla con el método Detroy(), dejando sólo la principal a la vista.

protected void OnBotonEliminarClicked (object sender, System.EventArgs e)

Cuando un cliente haga un clic en este evento, se pondrá a ZERO el valor del campo cantidad.

protected void OnTreeViewRowActivated (object o, Gtk.RowActivatedArgs args):

Utilizo éste evento para que cuando el cliente haga un doble clic en la fila del treeView nos muestre otra pantalla que será la encargada de recoger la cantidad que el usuario introduzca. Hago un objeto de la clase SeleccionCantidad y le paso por parámetro el treeView, la cantidad, el tipo de dato(integer) y el label. Y llamo al método Show() para que muestre la pantalla de esta clase.

4.3.14Clase CalculoLabelMain

- **Variable Global**

IDbConnection dbConnection: la utilizo para hacer la conexión a la base de datos.

- **Variables Local**

precio: es el dato que obtengo de la base de datos de los precios de los artículos.

Cantidad: es dato que obtengo de la Base de Datos, para saber la cantidad de los artículos elegidos por el cliente.

PrecioTotal: es el valor del precio total a pagar el cliente por su pedido.

Método

public void calculoLabelTotal(Label total, Button botonNuevoPedido):

En este método obtengo por parámetros un label y un button, para ser usados en este mismo método para actualizar el label con el valor del precio del pedido realizado por el cliente. El button lo utilizo para ocultar el mismo, para que los clientes no tenga acceso a él hasta que finalice su pedido y quiera realizar un nuevo pedido.Posteriormente hago una instancia para realizar la conexión a la Base de Datos. Creo la conexión y realizo una consulta para obtener los registros precio y cantidad de la tabla pedidos. Para finalmente hacer el cálculo del precio de los artículos por la cantidad del mismo.

4.3.15Clase DesayunoPopularView

- **Variables Global**
  IDbConnection dbConnection: la utilizo para hacer la conexión a la base de datos.
  Label totalMainWindow: es el label donde mostraremos la cantidad de dinero que llevamos acumulado durante el proceso de pedido.
  Button botonNuevoPedidoMainWindow: botón que utilizaremos para inicializar un pedido nuevo.
- **Variables Local**

  IDbCommand dbCommand: es una interfaz donde se crea la conexión con una instancia del objeto dbConnection a la Base de Datos y hago la consulta sql, con la propiedad del objeto “CommandText”.

IDataReader dataReader: es una interfaz donde recupero las filas de la consulta de la Base de Datos haciendo una instancia al objeto “dbCommand”.

- **Constructor**

  public DesayunoPopularView (Label labelTotalMainWindow,Button botonNP) :
  en el constructo cambiamos el tamaño de la fuente del label. Asignamos los parámetros recibidos por el mismo y asignamos a las variables global.
  Llamamos a los métodos para obtener los valores de los combos.

- **Métodos**

public void cargarComboCafe()

hacemos una consulta a la tabla de bebidascalientes y mostramos los nombres de las bebidas por el combo. Poniendo por defecto la frase de : “Seleccione ….”
public void cargarComboComida()

en este método mostramos los valores que los facilitamos nosotros sin ayuda de una consulta.
public void cargarComboPrecio()

en este método hacemos una consulta a la tabla de desayunopopular y obtenemos el precio para mostrarlo por defecto en el combo, como la única opción.

protected void OnComboboxCafeChanged (object sender, System.EventArgs e)
evento del combo para obtener el valor del Café elegido por el cliente.

protected void OnComboboxComidaChanged (object sender, System.EventArgs e)
evento del combo para obtener el valor de la comida elegida por el cliente.

protected void OnBotonAtrasClicked (object sender, System.EventArgs e)
en este evento cerramos la pantalla actual, dejando visible la principal.

protected void OnBotonAceptarClicked (object sender, System.EventArgs e)

en este evento comprobamos si todos los combos tienen un valor que no sea el de por defecto, porque siendo así se mostraría un mensaje de alerta informando a los clientes que tiene que elegir un valor, en caso contrario se guardaría los valores en la base de datos.
protected void OnBotonCancelarClicked (object sender, System.EventArgs e)

en este evento ponemos los valores de los combos por defecto, para que los clientes vuelvan a elegir.

4.3,16Clase MensajeComboInfo

- **Constructor**
  public MensajeComboInfo () :

  en el constructor llamo a la función Markup para llevar a cabo el cambio del tamaño y ancho de la fuente del label, que utilizo para mostrar un mensaje informativo, para que aparezca en las pantallas en las cuales los valores del combobox no han sido cambiados.

  - **Método (evento)**

  protected void OnBotonAceptarClicked (object sender, System.EventArgs e):

  Cuando el cliente haga clic en este evento, la pantalla se cerrará.

4.3.17Clase RellenarTreeView

- **Métodos (eventos)**

public void llenarTreeView(TreeView treeView, IDataReader dataReader)

En este método lo que hago es llamar al método AppendColumns y pasarle por parámetro el treeview y el datareader.En un vector de tipo obtengo los tipos de los datos de los campos de la tabla pedidos, pasándolos a string. Posteriormente paso el vector de tipos al constructor del objeto liststore y después asigno el liststore al modelo del treeview, para que muestre los datos con los tipos obtenido del vector y que se vea como una lista. Y llamo el método Fill pasándole los tipos de los datos obtenido de la base de datos.

public void Fill(IDataReader datareader):

Con un bucle while leo los datos obtenido de la consulta de la Base de datos y en un bucle for adiciono los datos obtenido por la consulta en una lista de string y pongo los valores a la lista(liststore).

public void AppendColumns(TreeView treeView, IDataReader dataReader):

Este método recibe por parámetro el treeView y el dataReader para ser usados en este método. En este método añado las columnas a las celda del treeView y pongo oculto la columna del id. Porque es un campo no necesario para los clientes.Posteriormente añado a mano la columna de la cantidad y el título de su columna, porque este campo no existe en la Base de datos.

public static Type[] GetTypes(Type type, int count):

método que devuelve el tipo a rellenar el treeview.

public ListStore get_ListStore():

este método devuelve el listStore. Para ser recogido y utilizado en otras clases.

5.Primera Aproximación del planteamiento

Software para que los clientes de una cafetería pueda hacer su pedido a través de una terminal y que se imprima un ticket tanto para el cliente como para el personal.

La información fluye por mi sistema a través de una base de datos donde se conecta con la aplicación para mostrar y guardar información de los artículos pedidos por los clientes.
La aplicación es intuitiva, fácil de usar , muestra una interfaz sin complicaciones para que cualquier persona sin tener conocimientos informáticos sepa usarla.

6.Resultados y Pruebas de la Aplicación

Las pruebas de la aplicación son actividades que conducen a la verificación del software. La verificación del software consiste en construir el software de forma correcta.

La realización de las pruebas de la aplicación ha tenido un resultado exitoso. A continuación definimos tres tipos de pruebas realizadas:

- Pruebas unitarias: Se realizan sobre cada una de las clases independientemente.
- Pruebas de integración: Son aquellas que se realizan para comprobar el correcto ensamblaje entre las clases.
- Pruebas de sistema: Son las pruebas de carga, rendimiento, tiempo de respuesta, etc. que se realizan sobre el sistema completo, ya ensamblado de forma completa.

Pruebas Unitarias

En esta actividad se realizan las pruebas unitarias de cada una de las clases del sistema de información, una vez codificadas, con el objetivo de comprobar que sus estructuras son correctas y que se ajusta a la funcionalidad establecida.

Las pruebas unitarias en nuestra aplicación empezaron por comprobar las interfaces del usuario, visualizando los resultados de esta y comprobando que todo pueda funcionar bien.

Pruebas de Integración

El objetivo de las pruebas de integración es verificar si las clases interactúan correctamente a través de sus interfaces, tanto internas como externas, cubren la funcionalidad, y se ajustan a los requisitos especificados.

Las comprobaciones del correcto funcionamiento de las interfaces, se realizó mediante la definición de secuencias de acciones de usuarios comunes. Para este fin se requirió la ayuda de conocidos, que intentara hacer un pedido de varios artículos, probando así todas las interfaces de usuario, botones, datos establecidos etc. para descubrir fallos no sospechados.

Pruebas del sistema

El objetivo de las pruebas del sistema es comprobar la integración del sistema de información globalmente, verificando el funcionamiento correcto de las interfaces entre distintas las clases y con el sistema de la Base de Datos.

Para la realización de estas pruebas se añadieron en todas las clases de la aplicación mensajes de salida para que el programador, cuando estuviera trabajando y ejecutando la aplicación en MonoDevelop, pudiera depurar los posibles errores o mejoras que podía hacer a la aplicación.

7.Requisitos para la Instalación de la aplicación

- **Entorno de ejecución:** Mono (mono runtime). Se puede encontrar en centro de software de ubuntu o en el siguiente enlace : http://mono-project.com/Main_Page
  Es una implementación opensource independiente del framework .NET de Microsoft.
  La gran ventaja de Mono radica en que no sólo es capaz de ejecutarse sobre sistemas Windows, sino también un gran rango de entornos *nix : Linux, MacOS X y Solaris.

- **Servidor:** PostgreSQL . Cliente gráfico pgAdmin III. Es un potente motor de bases de datos, que tiene prestaciones y funcionalidades equivalentes a muchos gestores de bases de datos comercial. Podéis encontrar su paquete en el centro de software de ubuntu o en el siguiente enlace : http://www.postgresql.org.es/principal

- **Pequeño Guía de instalación PostgreSQL:** http://www.guia-ubuntu.com/index.php?title=PostgreSQL

- **Descarga ejecutable, estructura y datos sql**

  git://github.com/JulianaCFS/Proyecto.git

Observación: Para probar la aplicación la base de datos ha sido configurada sólo para ser probada en local. Para poder conectar la base de datos con la aplicación tendréis que crear la base de datos con :
Nombre de la Base de Datos dbcafeteria, **usuario** y **contraseña** dbcafeteria.

8.Justificación Económica

Coste recursos materiales

Hardware y Software:

Tablet con cerradura + soporte de pared fijo + cable.

- Sistema operativo
- 4G RAM
- 500G disco duro
- wifi

Precio : 200,00 €

Impresoras para imprimir ticket

- Funcionamiento en Linux y Windows
- wifi

Precio : 115 * 2 = 230,00 €

Mueble para adaptar la impresora (clientes)

Precio : 150,00€

Precio total materiales = 580,00 €
pero a este total se le tiene que dar un 20% de amortización, con lo que se obtiene un total de:

Total coste de HW y SW = 116 Euros.

Conclusión

A este total del software se le debe añadir unos costes indirectos, como por ejemplo, los gastos del mantenimiento de hardware y software, gastos de comunicaciones , gasto de material, etc de un 20% sobre el total, se nos queda en un coste de **139,20 Euros.**

Finalmente, añadimos un 15% de beneficios, el coste final estimado del proyecto es: **154,20 Euros.**

9.Prevención de Riesgo Laboral

Para la prevención del riesgo laboral, hay algunos requisitos que es recomendable seguirlo.

Equipos de trabajo

Pantallas

Deberá ser orientable e inclinable a voluntad, se recomienda situarla a una distancia superior a 40 cm. De los ojos del usuario y a una altura que el campo de visión esté comprendido entre la línea de visión horizontal y la trazada a 60º bajo la horizontal.

Los caracteres deberán estar bien definidos y configurados de forma clara. La imagen será estable, sin destellos ni centelleos, etc.

Teclado

Móvil e independiente respecto al resto del equipo para poder ubicarlo conforme a los cambios de postura del usuario. El diseño debe cumplir los requisitos ergonómicos de altura, inclinación, espacio entre caracteres, baja reflectancia de su superficie, etc.

Ratón

Su diseño deberá resultar confortable y adaptado a la curvatura de la mano, sin bordes agudos y de fácil accionamiento sin necesidad de adoptar posturas poco naturales. El movimiento del ratón deberá adecuarse a la actuación del mismo en la pantalla.

Puesto de trabajo

Mesa o superficie de trabajo

Deberá ser poco reflectante, con el fin de minimizar los reflejos y su color no debería ser demasiado claro u oscuro. De dimensiones suficientes para colocar los elementos de trabajo, especialmente el teclado de forma que exista un espacio suficiente para apoyar las manos y los brazos.

Atril o portadocumento

Permite la colocación del documento a una altura y distancia similares a la de la pantalla, reduciendo los esfuerzo visuales y los movimientos de giro de cabeza.

Debe ser ajustable en altura, inclinación y distancia, de baja reflectancia y resistencia suficiente para soportar el peso de los documentos sin oscilaciones.
Se recomienda su utilización cuando se deba trabajar con documentos impresos.

Silla de trabajo

Dotada de dispositivos de ajuste en altura del asiento y en inclinación del respaldo, fácilmente manejables en la posición de sentado. Con una suave prominencia para dar apoyo a la zona lumbar y profundidad en el asiento de tal forma que el border no presione las piernas.

El uso de reposapiés cuando no se pueda regular la altura de la mesa y la altura del asiento no permita descansar los pies sobre el suelo.

Entorno

Iluminación

La iluminación general debe garantizar un nivel de iluminación adecuado para la tarea y una distribución del brillo (luminancia) entre la pantalla y su entorno, que evite una constante adaptación visual, ya que no es la adaptación la causa de riesgo sino la frecuencia con la que se experimenta.

Para evitar reflejos y deslumbramiento directos los puestos de trabajo deberán instalarse de tal forma que las fuentes de luz, artificiales y naturales, no incidan directamente a la pantalla o al trabajador. Además las ventanas deberán ir equipadas con dispositivo de cobertura adecuado y regulable para atenuar la luz. Un exceso de luz puede disminuir el contraste necesario entre los caracteres y el fondo, ya que los objetos son vistos gracias precisamente al contraste que ofrecen con el fondo.

Ambiente térmico

La pantalla de visualización normalmente no despide mucho calor, si bien se recomienda unos valores para temperatura y humedad considerados de confort de:

Temperatura seca: 19 a 24º C.
Humedad de 40 a 70 %.

Radiaciones

Los terminales con pantalla catódica no produce radiaciones cuantificable, ya que la mayor parte son absorbidas por el propio vidrio de la pantalla, por lo que las radiaciones que llegan al usuario, raramente puede superar la radiación de fondo a la que están expuestos todas las personas.

Ruido

Se recomienda que el nivel sonoro entorno al puesto de trabajo se encuentre entre los 70 y 65 dB (A) para un nivel de concentración normal, para tareas más difíciles o de mayor concentración como programación y diseño, se recomienda un nivel de ruido de 55 dB (A).

10.Conclusión

En todo proyecto de fin de curso, es una tarea interesante revisar los objetivos que nos marcamos al principio del proyecto, y comprobar en qué grado los hemos alcanzado, y si no es así, cuáles han sido las razones de ello, con propósito de aprender de la experiencia.

Los objetivos que se han alcanzado en este proyecto final de curso son los siguientes:

- Ofrecer una aplicación para realizar pedidos de clientes de una cafetería por terminal. Esto supone la visualización de los artículos disponible de cada grupo diferente, con la opción de elegir una cantidad a ese artículo y solicitarlo para que el personal te lo sirva en la mesa que este disponible por el número de persona que has indicado.
- Proporciona un software totalmente funcional, que sirve de punto de partida para el desarrollo de versiones más completas de esta aplicación.
- Permite el usuario elegir entre bebidas como refrescos, cervezas, bocadillos, etc. con los mensajes informativos necesarios para mantener el cliente informado, tantos si existe mesa o no, en el momento que introduzca el número de personas.

Los objetivos que no han podido cumplir

- La realización del software para el personal de la cafetería, por cuestión de tiempo no se ha podido implementar, pero al ser una versión para los clientes, se puede añadir en un futuro sin ningún problema.

11.Ampliaciones futuras

El objetivo principal del presente proyecto es el desarrollo de dos software para una cafetería, una para los clientes y la otra para el personal. La idea es que cuando el cliente de una cafetería indique el número de persona, le asignamos mesa (si hay disponibles), hago su pedido y se imprima dos tickets uno para el cliente y otro para el personal, especificando detalladamente los artículos pedidos con sus precios, cantidades y el valor total en Euros, la fecha, el identificador del ticket, los identificadores de las mesas asignadas . Y cuando la mesa este vuelva esta disponible, que sea el mismo personal en cual actualice introduzca el código del ticket para que vuelva a disponer de la mesa, aunque no incluya funcionalidades no prioritarias, que alargarían su desarrollo.

En resume, la línea de trabajo futuro considerada desde el análisis del proyecto es la siguiente:

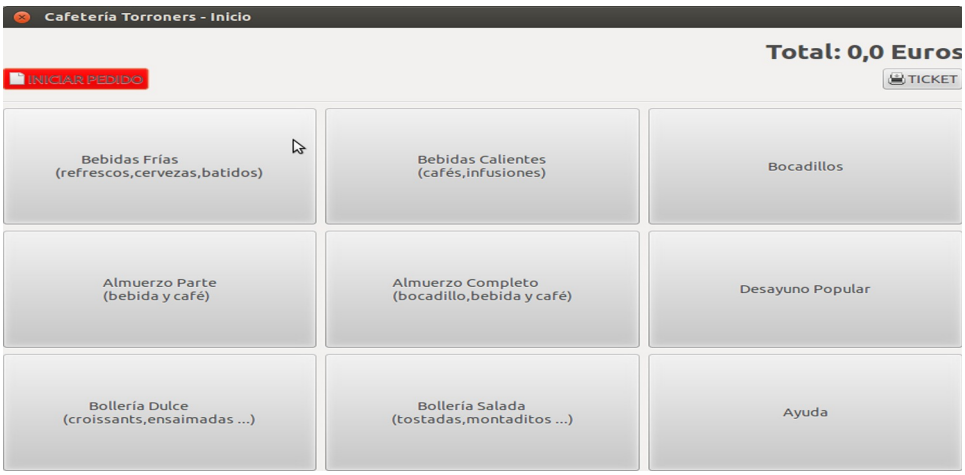
- Proporcionar un software para la terminal del personal, que incluya la opción, en que sea el propio personal que ponga en disponibilidad la mesa que está ocupada a disponible, con la introducción del identificador del ticket.

12.Anexos

Manual de usuario

En el manual de usuario explicaremos los pasos a seguir para hacer un pedido en este software.

Pantalla Principal - INICIAR PEDIDO



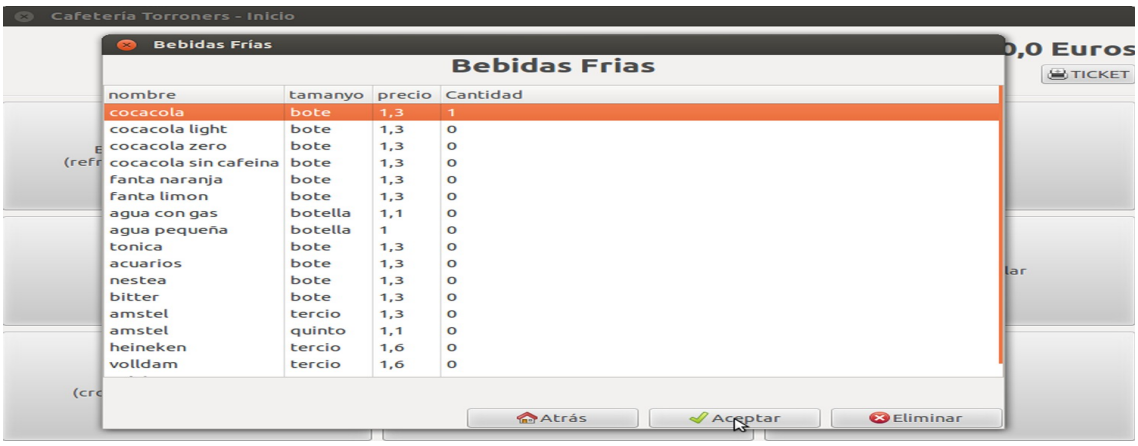
Para iniciar un pedido en la terminal de clientes debemos ir en el botón de INICIAR PEDIDO (color rojo).

Una vez hemos iniciado pedido nos mostrará una nueva pantalla, donde nos pedirá el número de persona, para que nos pueda asignar una mesa o varias mesas.

No habiendo mesa nos informará con un mensaje y en algunos minutos volvemos a intentarlo.
Habiendo mesa, podemos elegir los artículos a pedir en los botones inferiores.

Para elegir una cantidad

Debemos dar dos clic en el artículo a elegir, donde nos saldrá una nueva pantalla para que podamos poner la cantidad y se ACEPTA.



Pantallas con Botones de Cancelar o Eliminar

El de CANCELAR, para que canceles el valor elegido y la de ELIMINAR se elimina el pedido.

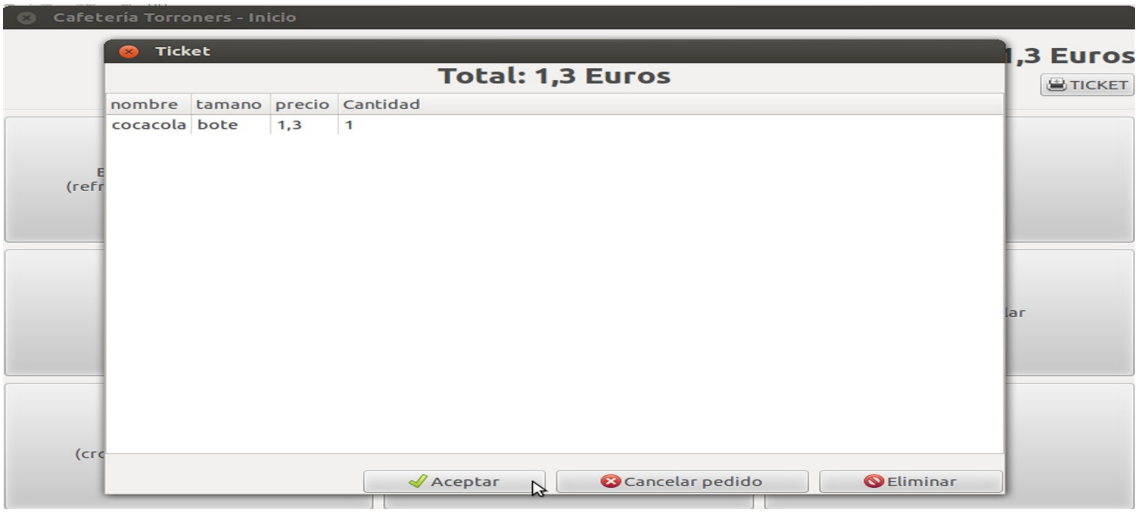
Para CANCELAR un valor elegido, tienes que haber seleccionado un artículo para que puedas cancelarlo.

Pantallas con listas de artículos a seleccionar

en las pantallas como indicamos arriba, debes seleccionar algún artículo antes de ACEPTAR, porque en caso que no lo hagas no te guardará el pedido.

Imprimir ticket

Para finalizar con su pedido y imprimir su ticket debes ir a la **PANTALLA PRINCIPAL** y hacer un clic en el botón de ticket, que lo encontrarás arriba a la derecha, abajo del valor total a pagar.



13.Bibliografía

Referencias Web

- Página oficial Mono
  URL: **http://www.mono-project.com/Main_Page**
- Página oficial GTKSharp: Color widgets

  URL: http://www.mono-project.com/GtkSharp:_Widget_Colours
- Tutorial GTK ListView

  URL: http://silentorbit.com/notes/2011/05/gtk-sharp-listview
- GTK Buttons

  URL: http://www.mono-project.com/GtkSharp:_Buttons
- Proyecto fin de carrera

  URL : http://www.proyectosfindecarrera.com/
- Guardar imágenes PostgreSQL

  URL: http://www.taringa.net/posts/info/15017020/Imagenes-en-Postgres-con-net.html
- Estimación de costes

  URL: http://www.dccia.ua.es/dccia/inf/asignaturas/GPS/archivos/Tema2.PDF
- Tutorial GTK TreeView

  URL: http://www.mono-project.com/GtkSharp_TreeView_Tutorial