



SOLUSOFT

Manual Técnico del Navegador

Manual Técnico del Navegador

MT-NAV-HSC-01



Universidad Mariano Gálvez de Guatemala
Carrera: Ingeniería en sistemas de información y ciencias de la computación
Curso: Análisis de sistemas II
Catedrático: Ing. Esduardo del Águila
Sección: "A"

Manual Técnico del Navegador

Integrantes:

Kevin Rolando González Ramírez	0901-18-1387
Wilber Enrique Segura Ramirez	0901-18-13952
Liam Patrick Bernard García	0901-18-10092
Jaime Noel López Daniel	0901-18-735
Melissa Odily Aldana Mejía	0901-18-335
Gabriel Hugo Alejandro Coyoy Zacarías	0901-18-20630
Daniel Enrique Navas Hernandez	0901-18-15032
Josué Zapata	9959-18-4829
Jorge Castañeda	9959-18-4964
Geovanni Mendoza	9959-18-15407
Brayan Cifuentes	9959-18-11113
Wilmer Torres	9959-18-9131

Manual Técnico del Navegador

MT-NAV-HSC-01

Índice

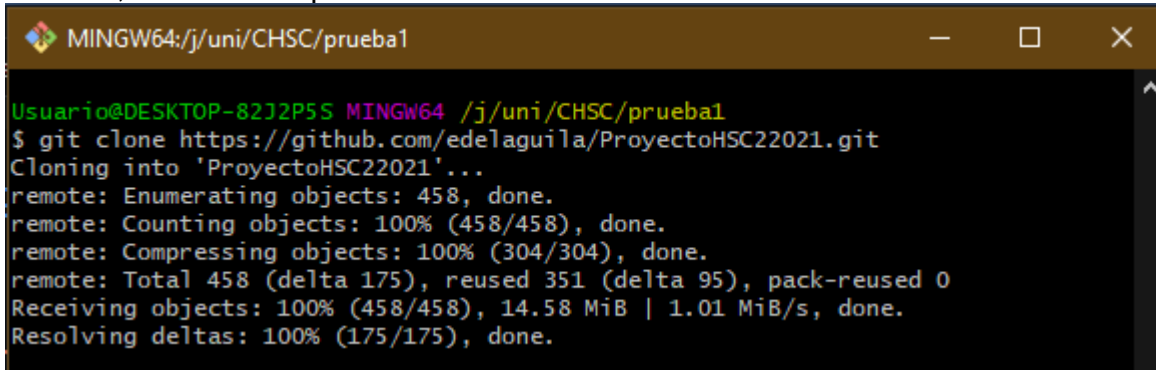
Paso 0: Compilación	4
Paso 1: Incorporación del navegador en los componentes de usuario y configuraciones de su formulario	9
Paso 2: Atribución de Tags	18
Paso 3: Parametrización del navegador	19
Paso Extra: Ejemplo de funcionamiento del navegador	24
Control de Versiones	28

Uso del navegador en sus formularios (Implementación)

Paso 0: Compilación

Para poder utilizar el navegador, debe compilar primero el proyecto que contiene al mismo, para ellos, deberá seguir las siguientes instrucciones:

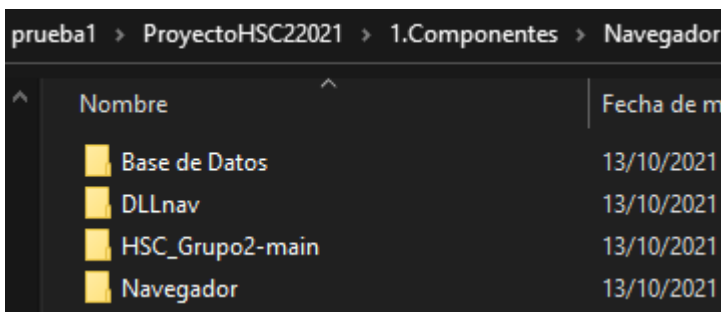
Primero, clonará el repositorio oficial:



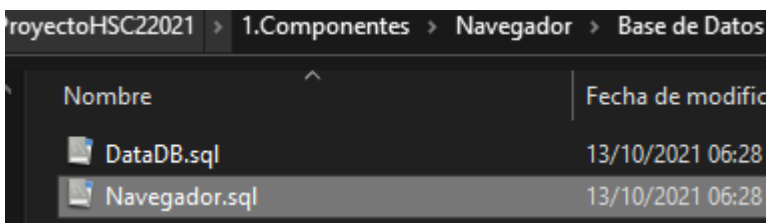
```
MINGW64:/j/uni/CHSC/prueba1

Usuario@DESKTOP-82J2P5S MINGW64 /j/uni/CHSC/prueba1
$ git clone https://github.com/edelaguila/ProyectoHSC22021.git
Cloning into 'ProyectoHSC22021'...
remote: Enumerating objects: 458, done.
remote: Counting objects: 100% (458/458), done.
remote: Compressing objects: 100% (304/304), done.
remote: Total 458 (delta 175), reused 351 (delta 95), pack-reused 0
Receiving objects: 100% (458/458), 14.58 MiB | 1.01 MiB/s, done.
Resolving deltas: 100% (175/175), done.
```

Segundo, deberá abrir el proyecto del navegador, para ello, debe irse a la carpeta de componentes, navegador, y allí dentro encontrará estas carpetas:

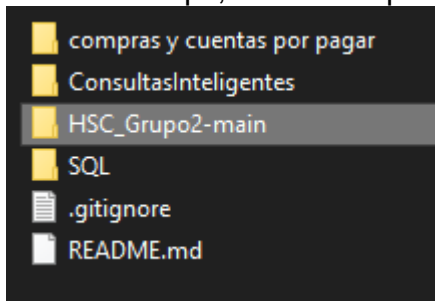


Antes de continuar, asegúrese de crear la base de datos que esta dentro de la carpeta “Base de Datos”, el archivo con la base de datos necesario se llama “Navegador.sql”

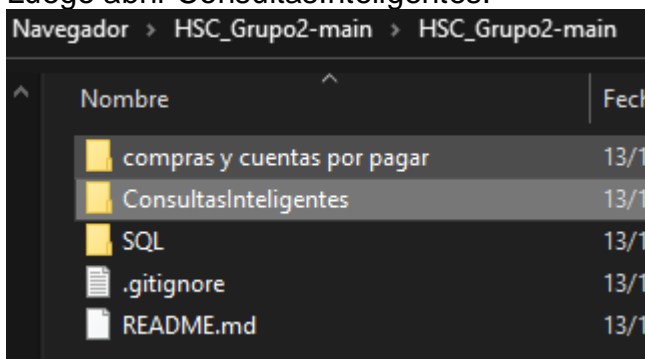


Luego de ejecutar esta base de datos, dirijase a la carpeta “HSC_Grupo2-main” para compilar la versión ajustada al navegador del objeto de consultas inteligentes.

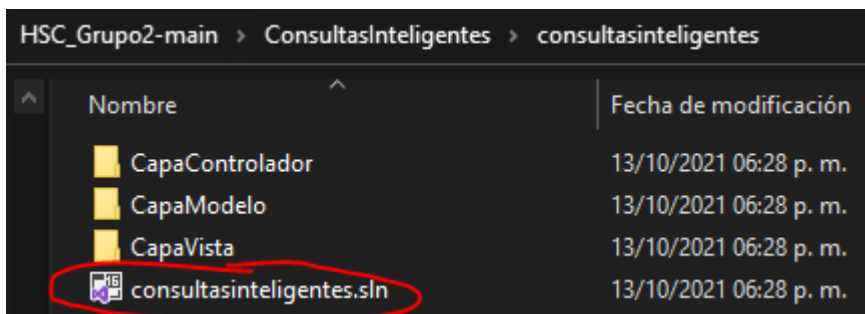
Dentro de aquí, abrir la carpeta HSC-Grupo2-main:



Luego abrir ConsultasInteligentes:

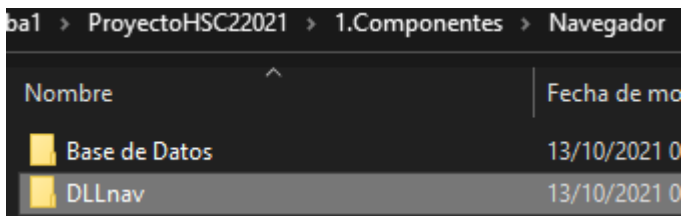


Luego, abrir otra vez consultasinteligentes y abrimos la solución:



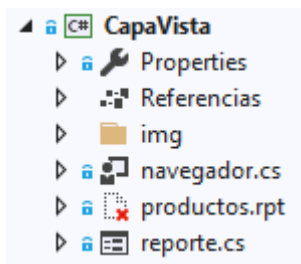
Una vez abierta la solución, realizamos el proceso de siempre, compilar hasta que queden los 3 actualizados.

Una vez hecho este paso, regrese hasta la carpeta del navegador, y abra la carpeta DLLNav:

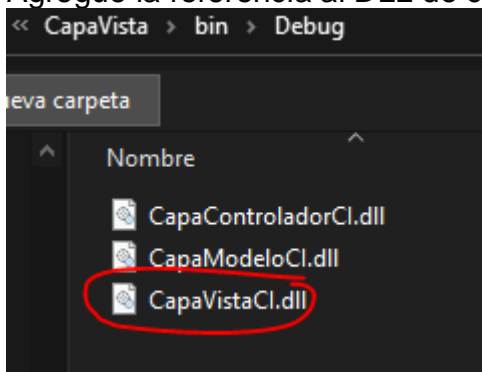


Allí dentro, abra la solución DLLNav.sln

Una vez abierto, diríjase a la capa vista, deberemos agregar el DLL de consultas inteligentes:



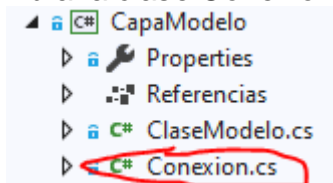
Agregue la referencia al DLL de consultas inteligentes que acabamos de compilar:



Asegúrese de verificar que el DLL tenga de nombre CapaVistaCI.dll, de lo contrario tendrá problemas con el nombre de las librerías.

A continuación, vaya a la capa modelo, nos hizo falta eliminar unas referencias al dll de mysql, por lo que tendrá que eliminarlas manualmente, nos disculpamos por las molestias. Esto se arreglará en una futura actualización que esta en proceso en el momento en que se escribe esta parte del manual (13/10/2021 – 7PM)

Abra la clase Conexion.cs



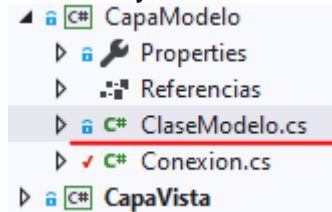
Y borre la línea remarcada:

```

1  using System;
2  using System.Collections.Generic;
3  using System.Data.Odbc;
4  using System.Linq;
5  using System.Text;
6  using System.Threading.Tasks;
7  using System.Windows.Forms;
8  using MySql.Data.MySqlClient;
9

```

Ahora vayamos a la clase ClaseModelo.cs



Y eliminemos las líneas marcadas:

```

1  using MySql.Data.MySqlClient;
2  using System;
3  using System.Collections.Generic;
4  using System.Linq;
5  using System.Text;
6  using System.Threading.Tasks;
7  using System.Windows.Forms;
8  using System.Data.Odbc;
9  using System.Collections;
10 using System.Data;
11

```

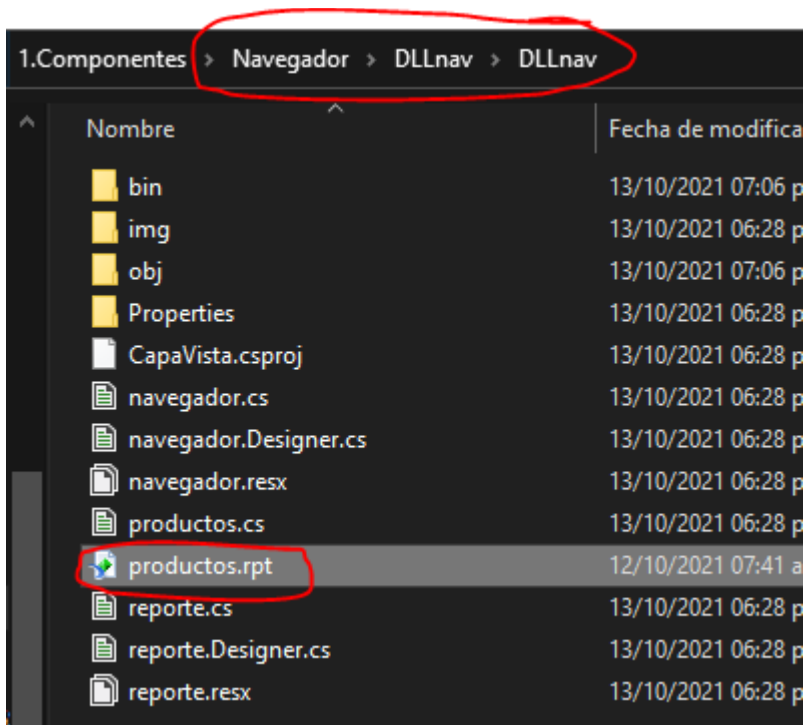
Luego de estos pasos, compile de nuevo el proyecto

Al hacerlo, le aparece el siguiente error:



Deberá incluir el archivo de reportes que se le compartirá por otro medio, lamentamos nuevamente las molestias, se está trabajando en una solución.

Cuando tenga el archivo, por favor colóquelo en la siguiente ruta:



(Es dentro de la capa Vista)

Luego de colocarlo, compile nuevamente hasta obtener 3 actualizados.

Buen trabajo, para este punto ya tenemos las DLL's del navegador compiladas, puede seguir con la incorporación de la misma.

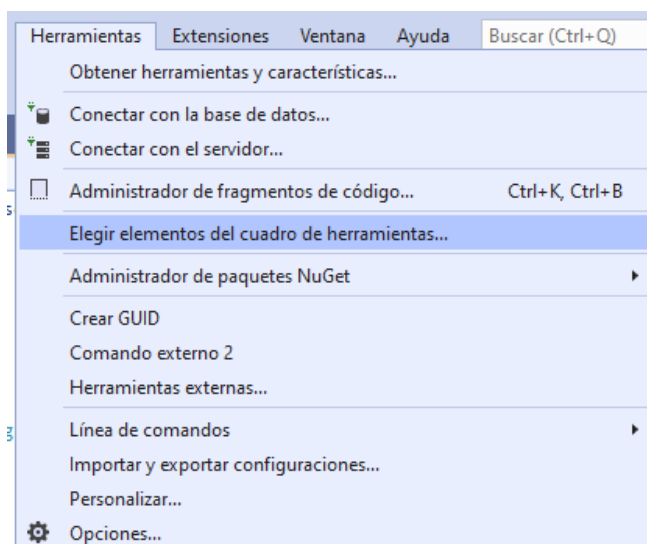
Nota: No olvide que necesitará tener previamente instalados los programas de Crystal Reports y HTML Workshops para C#, puede obtenerlos de los archivos compartidos por auditoría:

https://drive.google.com/drive/folders/1haf-pFNMEsaluYm4_mnEhpj8QK0kTbHy?usp=sharing

Debe usar una cuenta @miumg.edu.gt

Paso 1: Incorporación del navegador en los componentes de usuario y configuraciones de su formulario

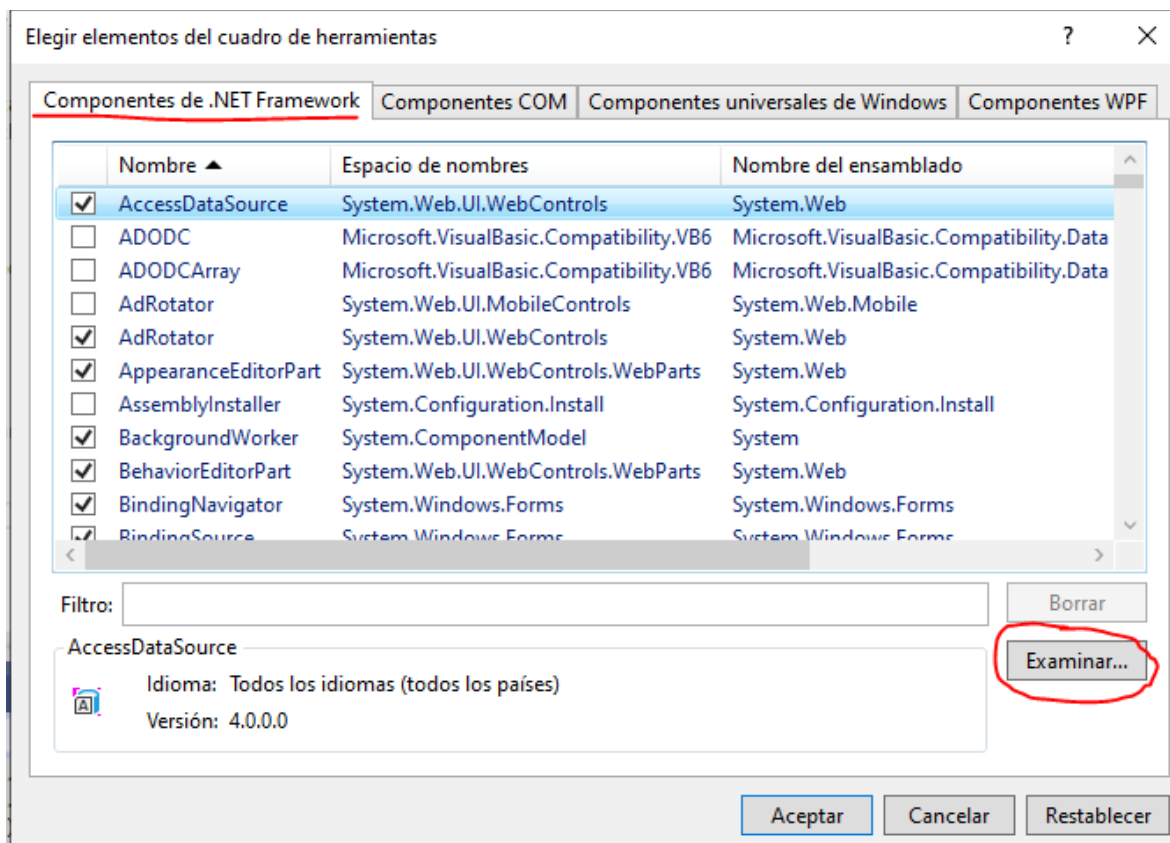
Para poder usar el navegador en un formulario o programa, se debe primero importar la DLL de la capa vista del navegador y agregarlo como control de usuario, para ello, debe seguir estos pasos:



Debe ir a la pestaña Herramientas que se encuentra en la parte superior del menú de Visual Studio, y luego seleccionar el apartado de “Elegir elementos del cuadro de herramientas...”.

Nota: El navegador hace uso de **Net Framework 4.7.2**, si no se tiene esta versión, no nos hacemos responsables por el comportamiento del componente, ya que pueden haber problemas de compatibilidad.

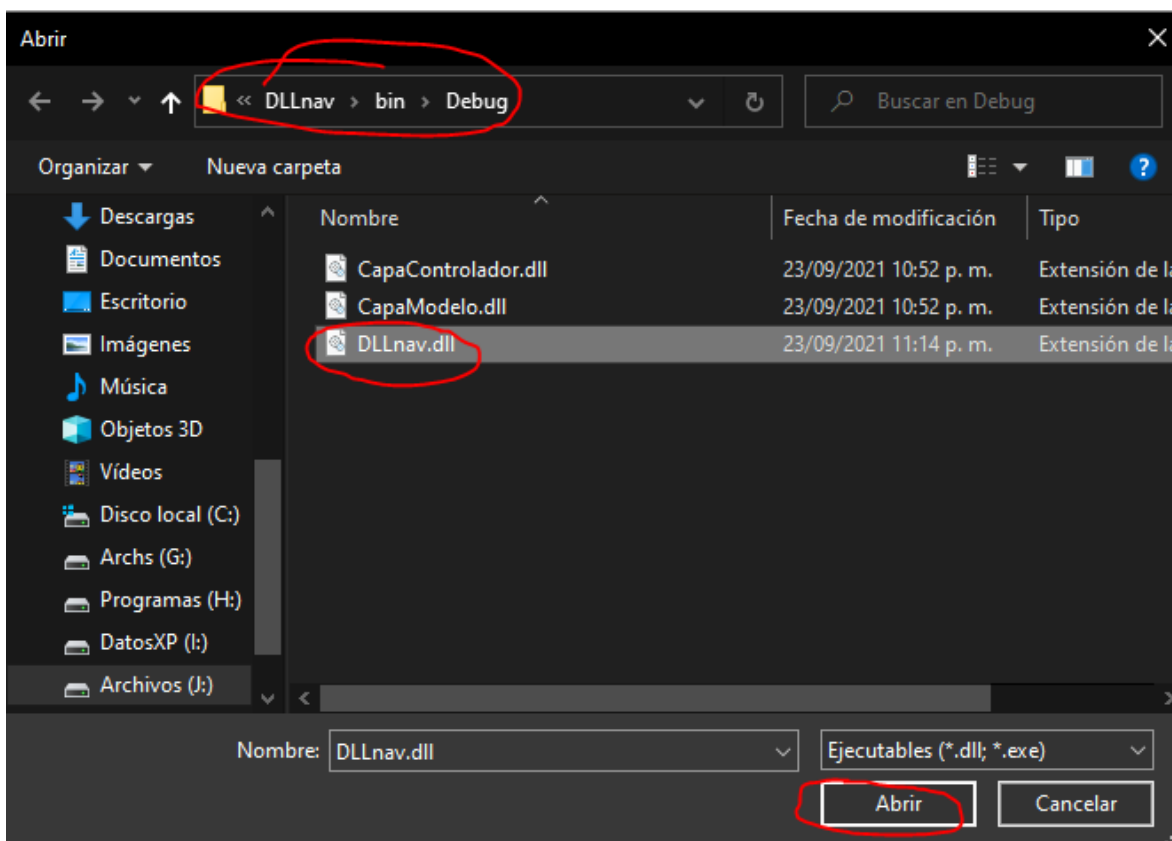
Al darle en elegir elementos del cuadro de herramientas, nos aparecerá esta ventana (Si es primera vez que la abre, puede que tenga que esperar un rato a que cargue):



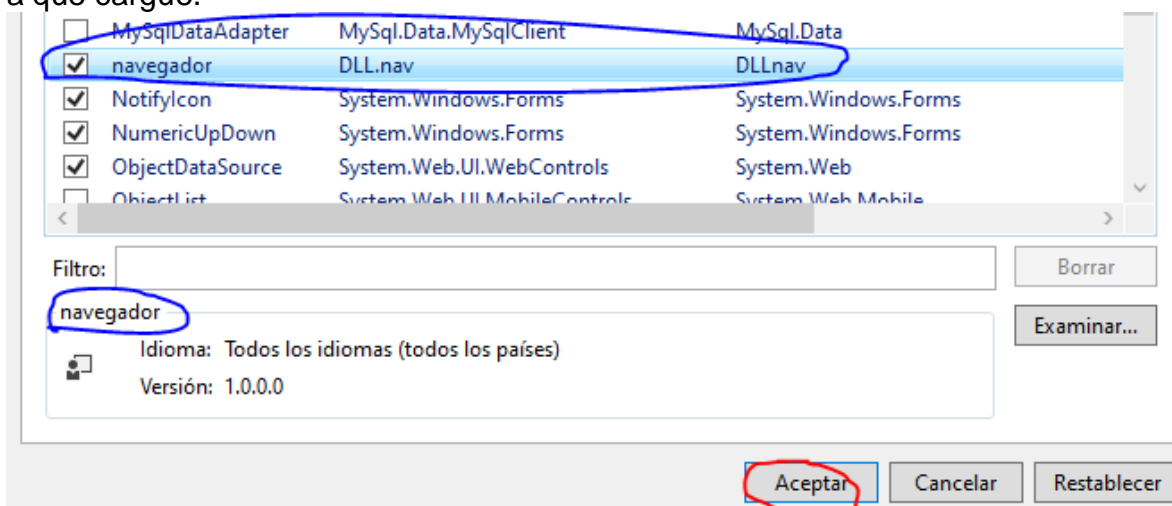
Asegúrese de estar en el apartado “Componentes de .NET Framework”, y luego presione el botón de Examinar...

En esta parte usted debe buscar la ubicación del navegador y seleccionar el DLL de la capa vista, este se encuentra en el directorio:

(Carpeta donde usted tiene descargado el navegador) \ DLLnav \ DLLnav \ bin \ Debug

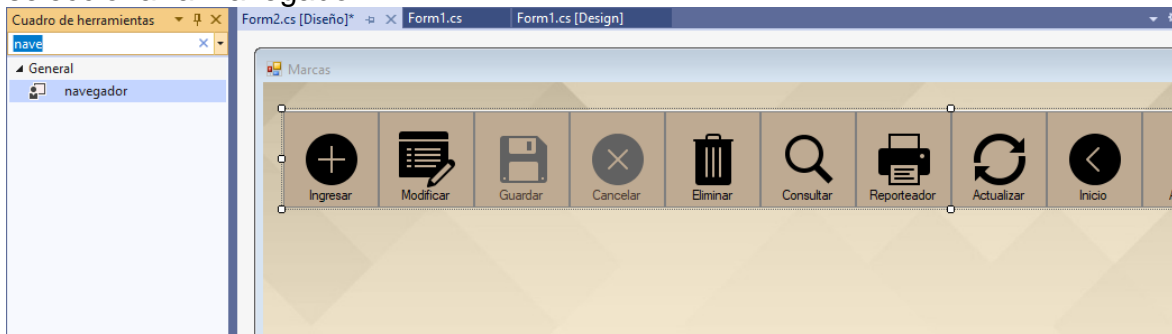


Una vez esté dentro de la carpeta, seleccione el archivo “DLLnav.dll”, este representa la Capa Vista del navegador. Luego de esto seleccione en abrir y espere a que cargue.



Si usted tiene la versión de .NET Framework recomendada y realizó correctamente los pasos anteriores, le deberá aparecer en su listado de componentes una entrada que diga “navegador”, justo como se muestra en la imagen, seguido de esto dele aceptar.

Cuando le de aceptar, ahora usted podrá ir a su cuadro de herramientas y seleccionar al navegador:



Para este ejemplo, usaremos en navegador para hacer un CRUD a una tabla de pruebas llamada Marcas:

```
> create table marca(  
    idMarca varchar(10) not null primary key,  
    nombre varchar(35) not null,  
    estatus char(1) not null  
~ ) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Primera observación importante, es necesario colocar la TextBox en el orden en que vaya a ingresar los datos. **El primer TextBox siempre debe ser el que usará para el Id o código de la tupla, y el último será el que use para estado.**

Diseño preliminar:

The screenshot shows a form titled 'Marcas' with a toolbar at the top containing four buttons: 'Ingresar' (plus icon), 'Modificar' (pencil icon), 'Guardar' (floppy disk icon), and 'Cancelar' (X icon). Below the toolbar are three input fields: 'ID Marca' (a single-line text box), 'Nombre de la Marca' (a single-line text box), and 'Estado' (a group box containing two radio buttons labeled 'Activo' and 'Inactivo'). Red handwritten numbers are placed next to each field: '1' next to 'ID Marca', '2' next to 'Nombre de la Marca', and '3' next to the 'Estado' group box.

Nótese el orden en que se pusieron las textbox, ya que este orden será el que tendrá el arreglo de TextBox que maneja el Navegador para poder realizar sus operaciones.

Nótese también que todos los datos deberán ir en TextBox, sin excepción. Para cuando use radioButtons, checkBoxs, ComboBox, DateTimePicker, etc., Usted deberá programar una TextBox donde se copie el valor que debe insertar en la base de datos, además de que deberá programar que se jale automáticamente (por ejemplo, en el radioButton, que jale automáticamente la selección y la coloque en la caja de texto).

Puede volver las cajas invisibles usando la propiedad Visible con un atributo false.

Otro requisito necesario para que funcione correctamente el navegador es tener un componente DataGridView donde se visualice el contenido de la tabla. Completando el diseño y teniendo ya programado el estado:

The screenshot shows the final design of the 'Marcas' form. The toolbar at the top is expanded to include 13 buttons: 'Ingresar', 'Modificar', 'Guardar', 'Cancelar', 'Eliminar' (trash icon), 'Consultar' (magnifying glass), 'Reporteador' (printer icon), 'Actualizar' (refresh icon), 'Inicio' (left arrow), 'Anterior' (left arrow), 'Siguiente' (right arrow), 'Final' (right arrow), 'Ayuda' (question mark), and 'Salir' (running person icon). The form fields remain the same, but a large, empty rectangular area is added to the right of the 'Estado' group box, intended for a DataGridView. The form is titled 'Marcas' and has a menu bar at the top with options: 'Inicio', 'Catálogos', 'Procesos', 'Reportes', 'Herramientas', 'Seguridad', 'Ayuda', and 'Salir'.

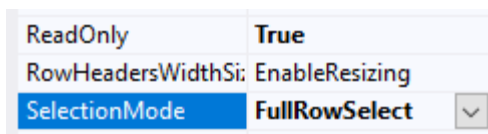
Diseño final del formulario

```
private void rdbActivo_MouseClick(object sender, MouseEventArgs e)
{
    if (rdbActivo.Checked == true)
    {
        txtEstado.Text = "A";
    }
}

1 referencia
private void rdbInactivo_MouseClick(object sender, MouseEventArgs e)
{
    if (rdbInactivo.Checked == true)
    {
        txtEstado.Text = "I";
    }
}
```

Código de los radioButtons para el estado

Nota: Y para la dataGridView, usted debe buscar las propiedades **ReadOnly** y **SelectionMode** y configurarlas de la siguiente manera:



Esto le permitirá hacer que el usuario no pueda editar los datos del DataGridView, además de permitir que seleccione toda la fila.

Códigos para funcionalidad del form (Ejemplo de Productos)

Una vez hecho esto, ya casi estamos listos para parametrizar el navegador, pero antes de ello, debemos tener en cuenta que, **si tenemos comboBox o DateTimePickers, debemos agregar el siguiente código:**

```
/*Wilmer Alexander Torres Lemus - 9959-18-9131*/
1 referencia
private void cbxCodMarca_SelectedIndexChanged(object sender, EventArgs e)
{
    navegador1.funComboTextboxVista(cbxCodMarca, txtCodigoMarca);
}
```

Para una ComboBox, usted debe programar el evento SelectedIndexChanged como se muestra en la imagen. El cbxCodMarca reemplazarlo con el nombre que usted le tenga a su combo Box, igual que el txtCodigoMarca, ese nombre debe reemplazarlo con el nombre que le tenga al textBox donde guardará la selección actual.

Para cuando tenga DateTimePickers, debe programar un evento que pase la fecha a un textBox asociado, y asegurarse de que vaya en el siguiente formato: yyyy-MM-dd, esto es, poner año-mes-día, se le facilita un método en el navegador para realizar esta operación.

```
/*Josue Daniel Zapata Azañon - 9959-18-4829*/  
1 referencia  
private void dtProducto_ValueChanged(object sender, EventArgs e)  
{  
    navegador1.funDPTeXtBoxVista(dtProducto,txtFecha);  
}
```

Reemplace el dtProducto con el nombre de su DateTimePickers, y reemplace txtFecha con el nombre de su TextBox donde guardará los datos.

No olvide colocar este método también en las textBoxs que guarden la fecha, para asegurarse que esté bien el formato. Usar el evento textChanged:

```
/*Josue Daniel Zapata Azañon - 9959-18-4829*/  
1 referencia  
private void txtFecha_TextChanged(object sender, EventArgs e)  
{  
    navegador1.funTeXtBoxDPTVista(dtProducto, txtFecha);  
    navegador1.funDPTeXtBoxVista(dtProducto, txtFecha);  
}
```

Por último, si usted desea que tener una navegación en el DataGridView que se aun poco más completa, debe agregar el siguiente código:

```
/*Josue Daniel Zapata Azañon - 9959-18-4829*/  
1 referencia  
private void dgvConsulta_SelectionChanged(object sender, EventArgs e)  
{  
    navegador1.funSeleccionarDTVista(dvgConsulta);  
}
```

Debe apoyarse con el evento SelectionChanged, y en navegador1, debe poner el nombre que le ha puesto al objeto navegador dentro del contexto de su form. (generalmente si no lo cambia siempre será navegador1). Y el dgvConsulta la data grid view donde usted muestre su parametrización.

Resumen de código agregado a la form según control:

La parametrización la veremos en el paso 3

Estados radioButtons:

```
/*Brayan Mauricio Cifuentes López - 9959-18-11113*/
1referencia
private void radioButton1_MouseClick(object sender, MouseEventArgs e)
{
    navegador1.funCambioEstatusRBVista(txtEstatus, radioButton1, "A");
}

/*Brayan Mauricio Cifuentes López - 9959-18-11113*/
1referencia
private void radioButton2_MouseClick(object sender, MouseEventArgs e)
{
    navegador1.funCambioEstatusRBVista(txtEstatus, radioButton2, "I");
}
```

Actualización del textBox asociado al combo box:

```
/*Wilmer Alexander Torres Lemus - 9959-18-9131*/
1referencia
private void cbxCodMarca_SelectedIndexChanged(object sender, EventArgs e)
{
    navegador1.funComboTextboxVista(cbxCodMarca, txtCodigoMarca);
}
```

Permitir seleccionar datos haciendo clic en el dataGridView:

```
/*Josue Daniel Zapata Azañon - 9959-18-4829*/
1referencia
private void dgvConsulta_SelectionChanged(object sender, EventArgs e)
{
    navegador1.funSeleccionarDTVista(dvgConsulta);
}
```

Para actualizar los radioButtons según el texto de la TextBox:

```
/*Brayan Mauricio Cifuentes López - 9959-18-11113*/
1referencia
private void txtEstatus_TextChanged(object sender, EventArgs e)
{
    navegador1.funSetearRBVista(radioButton1, radioButton2, txtEstatus);
}
```

Para actualizar combo Box según el dato en la textbox asociada:

```
/*Wilmer Alexander Torres Lemus - 9959-18-9131*/
1referencia
private void txtCodigoMarca_TextChanged(object sender, EventArgs e)
{
    navegador1.funTextboxComboVista(cbxCodMarca, txtCodigoMarca);
}
```


Para actualizar el texto de la textBox asociada al date time picker:

```
/*Josue Daniel Zapata Azañon - 9959-18-4829*/  
1 referencia  
private void dtProducto_ValueChanged(object sender, EventArgs e)  
{  
    navegador1.funDPTextBoxVista(dtProducto,txtFecha);  
}
```

Para actualizar el contenido del date time picker cuando se selecciona un valor del datagrid, y para ponerlo en el formato correspondiente.

```
/*Josue Daniel Zapata Azañon - 9959-18-4829*/  
1 referencia  
private void txtFecha_TextChanged(object sender, EventArgs e)  
{  
    navegador1.funTextBoxDPTVista(dtProducto, txtFecha);  
    navegador1.funDPTextBoxVista(dtProducto, txtFecha);  
}
```

Hasta aquí hemos agregado código a nuestra forma, pero aún debemos agregar más detalles si queremos que esté completamente funcional, para ello debemos pasar al siguiente paso: Atribuir las Tags a las TextBox.

Paso 2: Atribución de Tags

Para que el CRUD hacia la base de datos pueda realizarse exitosamente, a cada textBox que usted agregue, debe agregarle un nombre de campo en la propiedad TAG.

Estos nombres que colocará allí, serán los nombres que tienen los campos en la base de datos, y estos serán base para que el navegador los tome y haga el crud.

Siguiendo con el Ejemplo:

```
idMarca varchar(10) not null primary key,  
nombre varchar(35) not null,  
estatus char(1) not null
```

Para la tabla marca tenemos estos campos, entonces, debo colocar en el atributo TAG de cada Text box, el nombre que corresponda:

Datos	
(ApplicationSettings)	
(DataBindings)	
Tag	idMarca
Diseño	
(Name)	txtID
Anchor	Top, Left

Asignación del campo idMarca a la text box que llevará el ID.

Datos	
(ApplicationSettings)	
(DataBindings)	
Tag	nombre
Diseño	
(Name)	txtMarca

Asignación del campo nombre al textbox donde ingresamos el nombre de la marca

Datos	
(ApplicationSettings)	
(DataBindings)	
Tag	estatus
Diseño	
(Name)	txtEstado

Asignación del campo estatus al textbox que lleva el estado.

Recordemos, esto lo debemos hacer para cada TextBox que tengamos.

Paso 3: Parametrización del navegador

Luego de tener las Tags agregadas, en este apartado, vamos a mostrar los parámetros que soporta el navegador, estos se insertaran en la parte del constructor:

```
public Form2()
{
    InitializeComponent();

    //Aquí parametrizaremos el navegador
    //justo en el constructor
}
```

Parámetros obligatorios

Los primero que debemos hacer, es obtener el arreglo de los textbox que tenemos en el form, para ello, usamos “navegador1.funAsignandoTexts(this)”

Este lleva de parámetro this porque obtiene de forma automática todos los elementos TextBox de la forma.

```
//Obteniendo los datos de los TextBox
TextBox[] alias= navegador1.funAsignandoTexts(this);
```

Este alias nos servirá para el siguiente paso:

```
navegador1.funAsignarAliasVista(alias, "marca", "pruebas");
    //Arreglo con los textbox, nombre tabla, nombre base de datos
navegador1.funAsignarSalidaVista(this);
```

En el, le indicamos al navegador cual es el arreglo de textbox a usar, para que pueda obtener el nombre de los campos de la BD y sus valores, le indicamos el nombre de la tabla que estaremos utilizando, y el nombre de la base de datos.

Esto es así para que se pueda validar en la base de datos que sí exista esta tabla y si existan estos datos.

De no poderse realizar la validación correctamente, el programa se lo indicará cuando lo ejecute.

El parámetro funAsignarSalidaVista también se necesita ya que asigna los valores ya validados a las variables globales que utiliza el navegador.

Otro parámetro que debemos de usar es el campoEstado, con este le indicamos al navegador que campo será usado para dar de baja registros.

```
//inicio de elementos para dar de baja
navegador1.campoEstado = "estatus";
//aquí indicamos que campo usaremos como estado
```

Algo que también debemos tomar en cuenta, es que debemos indicar que ayuda tendrá asignada la forma, para ello utilizamos:

```

/* Inicio ID Aplicacion usada para reportes y ayudas */
navegador1.idAplicacion = "1";
/* Inicio ID Aplicacion usada para reportes y ayudas */

//inicio de elementos para ejecutar la ayuda
navegador1.tablaAyuda = "Aplicacion";
navegador1.campoAyuda = "pkId";
//fin de elementos para ejecutar la ayuda

```

tablaAyuda nos indica que tabla usaremos para buscar la ayuda, el campoAyuda indicará como se llama el campo del id que estamos enviando, y luego el idAplicacionn nos indicará que id de aplicación se está asociado a la forma. Esto permite que se pueda abrir la ayuda respectiva del módulo en que estemos haciendo una consulta a las tablas de seguridad.

Para asociar un reporte, debe usar:

```

// Inicio datos para ejecutar reportes
navegador1.funReportesVista("ruta", "idAplicacion", "Reporte");
// Final datos para ejecutar reportes

```

Donde ruta es el campo que tiene la tabla para indicar la ruta del reporte, el idAplicacion es el nombre del campo id, y el campo Reporte es el Reporte que buscará. Este método no cambiará nunca, por lo que lo puede copiar y pegar, solo asegúrese que este después de los parámetros antes mencionados.

Posteriormente, tenemos uno de los parámetros más importantes la data Grid View

```

//pasamos la referencia a la datagridview
navegador1.pideGrid(dgvMarca);
//indicamos que queremos que el datagridview empiece con datos
navegador1.llenaTabla();

```

Con pideGrid pasamos la referencia al data grid view que tenemos en nuestra forma, permitiendo que los botones de actualizar y navegación funcionen correctamente.

Y con llenaTabla, le indicamos que queremos que cada vez que inicie el formulario, ya tenga cargados los datos actuales en la form.

El último parámetro de uso obligatorio es:

```

//enviamos una referencia a la forma para que el navegador pueda cerrarla
navegador1.pedirRef(this);

```

Este nos permite pasar una referencia a la forma actual para que el navegador pueda cerrarla con el botón salir.

Parámetros opcionales

Estos parámetros sólo los deberemos usar en ciertos casos, como cuando queramos jalar datos de la base de datos y colocarlos en una combo Box.

El parámetro para indicar que una comboBox se llena desde la base de datos es:

```
//llenado de combo Box  
navegador1.funLlenarComboControl(cbxCodMarca, "marca", "idMarca", "nombre","estatus");
```

Con este parámetro, indicamos que combo box tendrá datos desde la base de datos, de que tabla jalara los datos, que campo será el que guardaremos en la BD, que campo se mostrará en la combo, y que campo se usará de estatus para decidir si mostrar o no la información.

Dicho de otro modo, la sintaxis es:

```
navegador1.funLlenarComboControl( comboQueLlenare , "TablaQueUsare" ,  
"IdentificadorDeCadaCampo" , "NombreDelCampoaMostrar" , "CampoDeEstado")
```

El identificador de cada campo es el texto que se colocará en la textbox y se ingresará en la BD, el Tabla Que usaré es la tabla de la que obtendremos la información, Nombre del Campo a Mostrar es el campo que se mostrará en la combo, y Campo de Estado es el campo de estado que se validará para saber si mostrar o no la opción en la comboBox.

Ejemplo de parametrización e inicialización de form en CRUD productos:

```
public Form1()
{
    InitializeComponent();

    TextBox[] alias = navegador1.funAsignandoTexts(this);
    navegador1.funAsignarAliasVista(alias, "producto", "hotelSanCarlos");
    navegador1.funAsignarSalidadVista(this);
    navegador1.funLlenarComboControl(cbxCodMarca, "marcaP", "idMarca", "nombre","estatus");

    //inicio de elementos para dar de baja
    navegador1.campoEstado = "estado";
    //fin de elementos para dar de baja

    /* Inicio ID Aplicacion usada para reportes y ayudas */
    navegador1.idAplicacion = "1";
    /* Inicio ID Aplicacion usada para reportes y ayudas */

    //inicio de elementos para ejecutar la ayuda
    navegador1.tablaAyuda = "Aplicacion";
    navegador1.campoAyuda = "pkId";
    //fin de elementos para ejecutar la ayuda

    // Inicio datos para ejecutar reportes
    navegador1.funReportesVista("ruta", "idAplicacion", "Reporte");
    // Final datos para ejecutar reportes

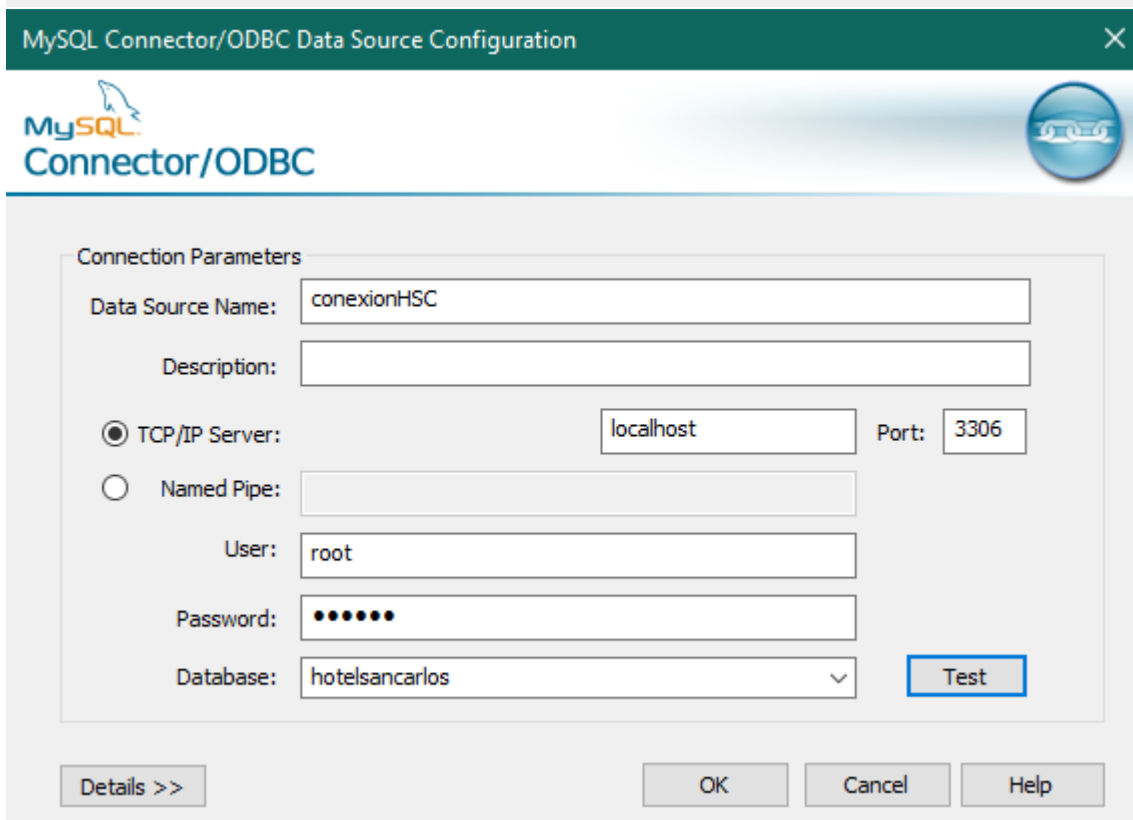
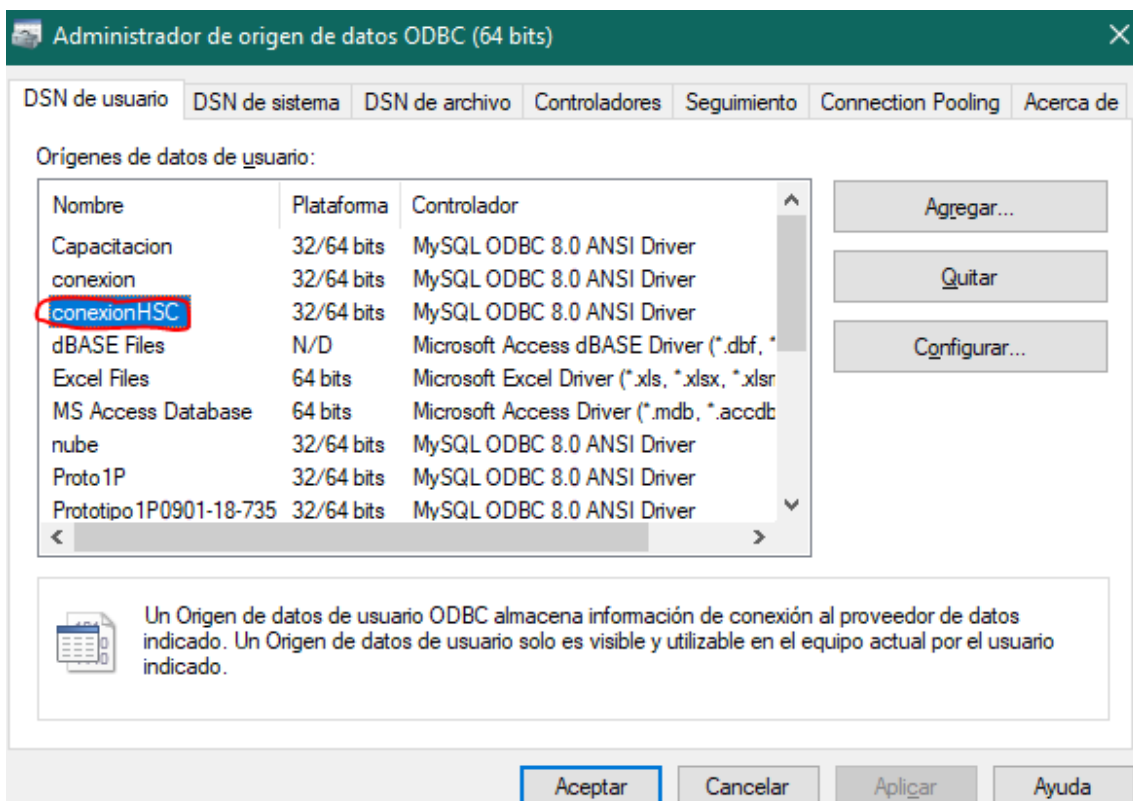
    navegador1.pideGrid(this.dvgConsulta);
    navegador1.llenaTabla();
    navegador1.pedirRef(this);
}
```

ODBC

Nota:

El ODBC que usted cree para la conexión con la base de datos, debe llevar el nombre conexionHSC, que es el nombre estándar dado.

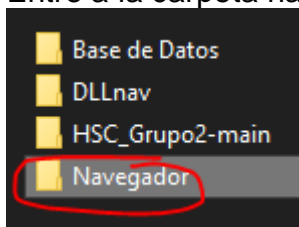
Ejemplo:



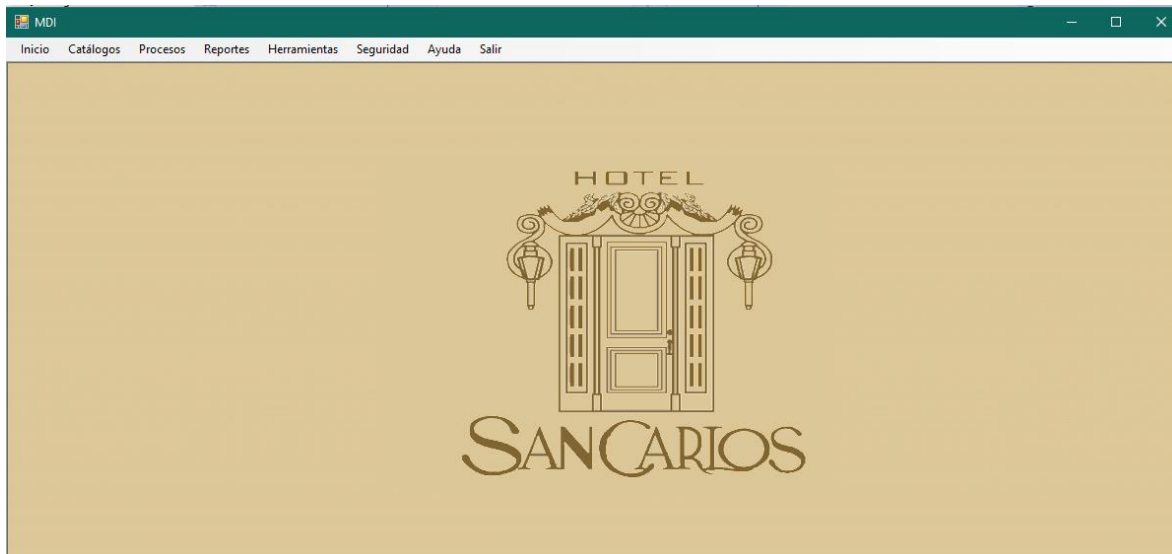
Paso Extra: Ejemplo de funcionamiento del navegador

Dentro de los archivos del objeto común, se ha incluido un proyecto de prueba, puede utilizar este para verificar el funcionamiento del navegador, para ejecutarlo, realice los siguientes pasos:

Entre a la carpeta navegador:



Y abra la solución, luego por favor compile y ejecute, esta acción abrirá un MDI que los grupos del navegador crearon para pruebas:



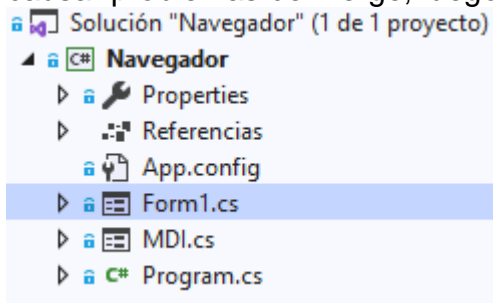
Ahora diríjase a la parte de Catálogos, y luego a Producto

Esta acción apertura el from de pruebas de productos:

The MDI window displays a product management interface. On the left, there are input fields for product details: Codigo Producto (1), Stock Producto (10), Codigo De Marca (Nike), Precio Producto (15), Nombre Producto (Producto1), Fecha Producto (22/09/2021), and Descripción Producto (Descripción). On the right, a table shows product data:

idProducto	idMarca	nombre	fecha	descripcion	stock	prec
1	1	Producto 1	22/09/2021	Descripcion	10	15
2	2	Producto 2	7/10/2021	Descripcion 2	200	5

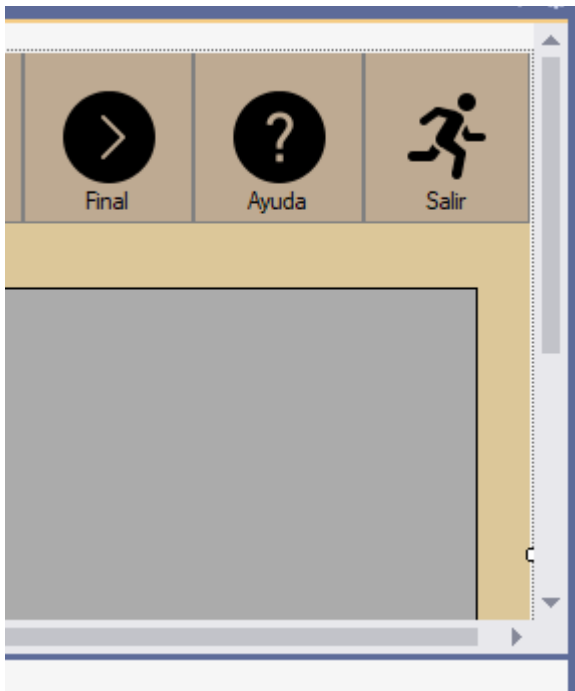
Si a usted también le aparece la ventana cortada, o desea agregarle frame, primero realice una copia de este ejecutable en otro lado para evitar sobre escribir el git y causar problemas de merge, luego, vaya a esta parte:



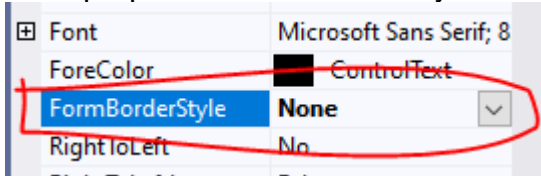
Abra la Form1.cs

The Form1.cs design view shows the same product management form as the MDI window. A red arrow points to the bottom-right corner of the form, indicating where to click to expand the form to a convenient size.

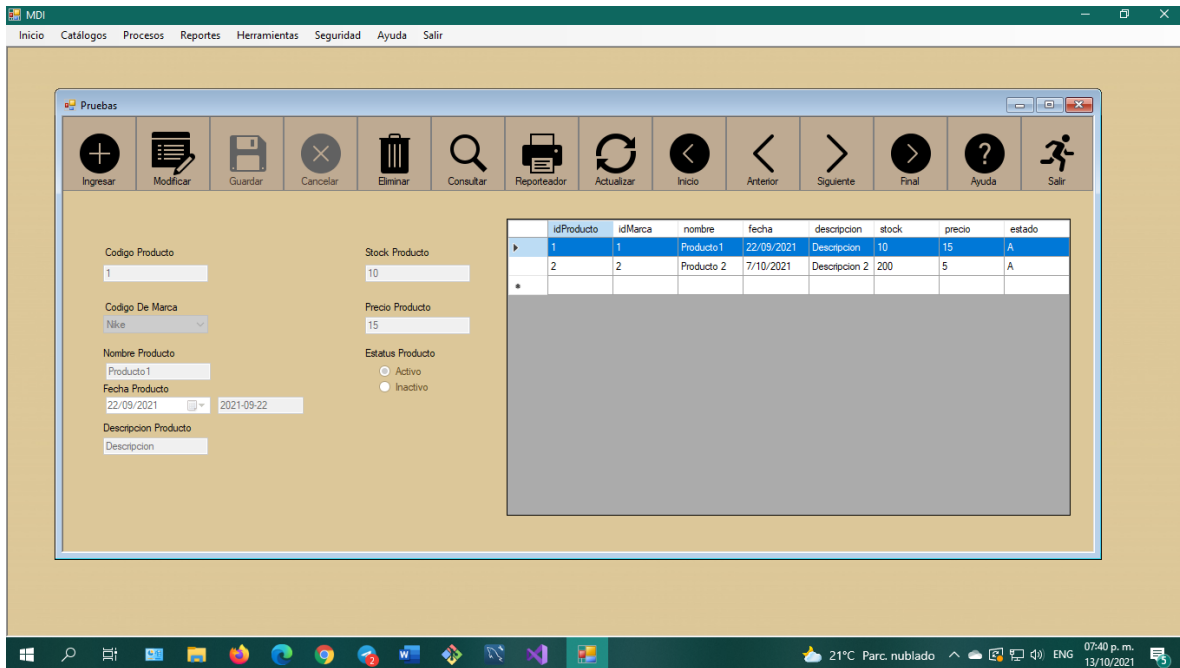
Seleccione el cuadrado blanco y expanda la form a un tamaño conveniente



Una vez este de un tamaño convincente, puede ver las propiedades de la form, y en la propiedad `FormBorderStyle` seleccione uno a su agrado o déjelo como esta.



Una vez termine sus cambios, ejecútelo y ahora podrá experimentar con el o usarlo como guía, recuerde que el código que se mostro en pasos anteriores, así como la parametrización, se encuentran en este ejemplo.



Gracias por su tiempo

Demostración en video de lo platicado en este manual:

Clase 12/10/21

<https://drive.google.com/file/d/1Oa8cqR055wwLW9HhwCJ9YKUQHZidAlwV/view?usp=sharing>

Clase 15/10/21

-Link Pendiente, revisar carpeta de grabaciones-

(Se realizo durante clase, por lo que se adjunta el video de la clase, más o menos la hora es a las 8:00am)

Manual Técnico del Navegador

MT-NAV-HSC-01

Control de Versiones

Fecha de Referencia	Descripción de la modificación
24/09/2021 a 08/10/2021	Creación del manual Técnico
12/10/2021 a 13/10/2021	Modificación del manual Técnico, ahora se agregaron los detalles de compilación y se agregaron instrucciones para ejecutar el ejemplo
14/10/2021 a 15/10/2021	Se agrego un índice, se actualizaron algunas partes del manual, y se agrego un enlace al video de clase donde se hizo la capacitación.

Versión	Razón de cambio	Responsable del Cambio	Responsable de Aprobación	Fecha de modificación
1.0	Versión Inicial	Grupo 1 y Grupo 3	Jaime López	08/10/2021
1.1	Se agrego un apartado de compilación	Jaime López	<i>Se regreso para realizar cambios</i>	12/10/2021 a 13/10/2021
1.2	Índice, enlace a capacitación, arreglos misc.	Jaime López	<i>Pendiente</i>	14/10/2021 a 15/10/2021