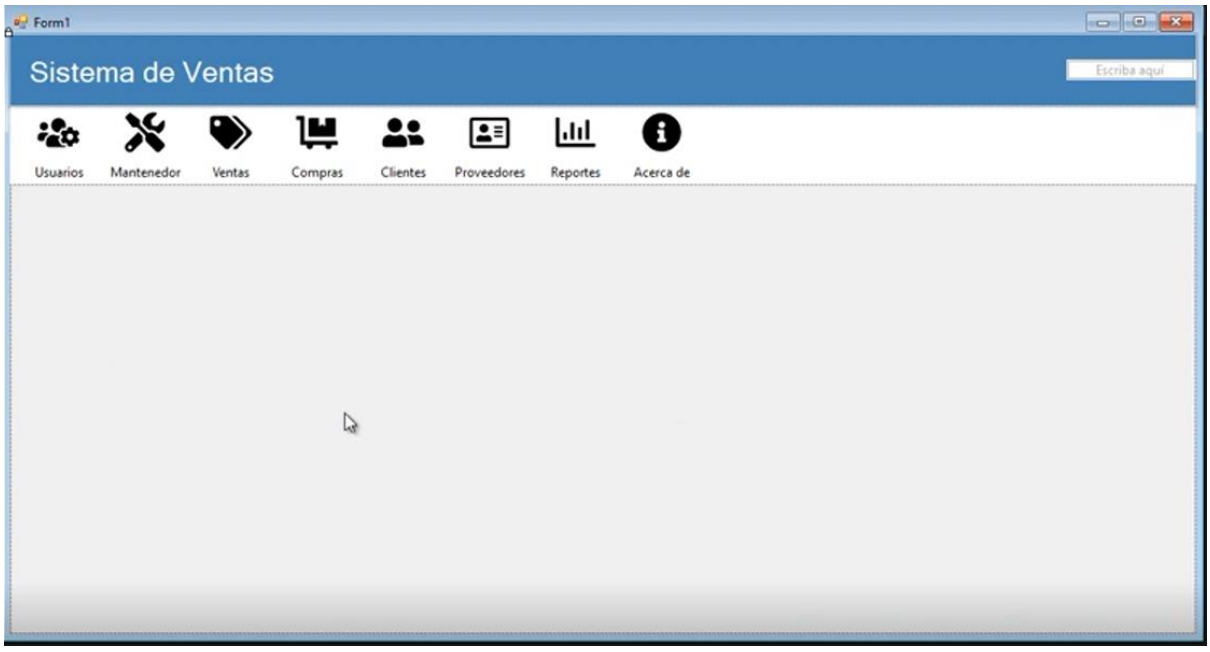
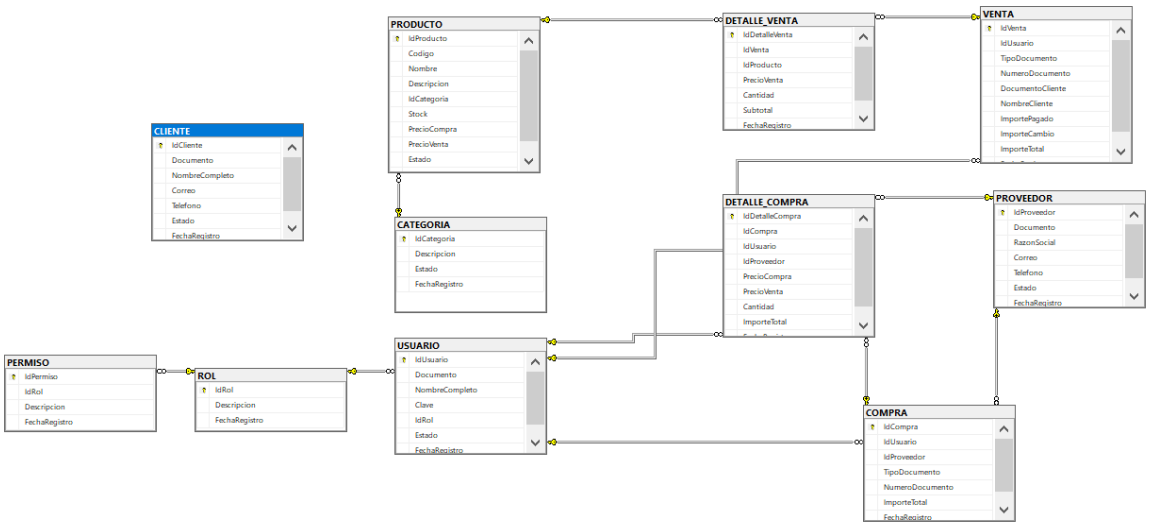
Diseñaremos un sistema de ventas de un comercio que registrará vendedores, clientes, proveedores, compras y ventas tras la introducción de un login inicial de usuario (vendedor). Los usuarios tendrán diferentes roles, en el ejemplo pueden ser administradores o no. Los administradores tendrán opciones más amplias.



Como gestor de la base de datos utilizaremos Microsoft SQL Server que hemos instalado en la práctica anterior.

Este diseño se corresponde con el video de apoyo número 1.

Lo primero que tenemos que hacer es crear todas las tablas SQL donde vamos a almacenar la información de nuestro sistema de ventas y la relación entre ellas con el script proporcionado.



Vamos a trabajar con un proyecto por capas con cuatro capas (proyectos).

Cuando aumenta la complejidad de las aplicaciones, una manera de administrarla consiste en dividir la aplicación según sus responsabilidades o intereses. Este enfoque sigue el principio de separación de intereses y puede ayudar a mantener organizado un código base que crece para que los desarrolladores puedan encontrar fácilmente dónde se implementa una función determinada. Pero la arquitectura en capas ofrece una serie de ventajas que van más allá de la simple organización del código.

Al organizar el código en capas, la funcionalidad común de bajo nivel se puede reutilizar en toda la aplicación. Esta reutilización es beneficiosa ya que significa escribir menos código y puede permitir que la aplicación se estandarice en una sola implementación, siguiendo el principio Una vez y solo una (DRY).

Con una arquitectura en capas, las aplicaciones pueden aplicar restricciones sobre qué capas se pueden comunicar con otras capas. **Esta arquitectura permite lograr la encapsulación**. Cuando se cambia o reemplaza una capa, solo deberían verse afectadas aquellas capas que funcionan con ella. Mediante la limitación de qué capas dependen de otras, se puede mitigar el impacto de los cambios para que un único cambio no afecte a toda la aplicación.

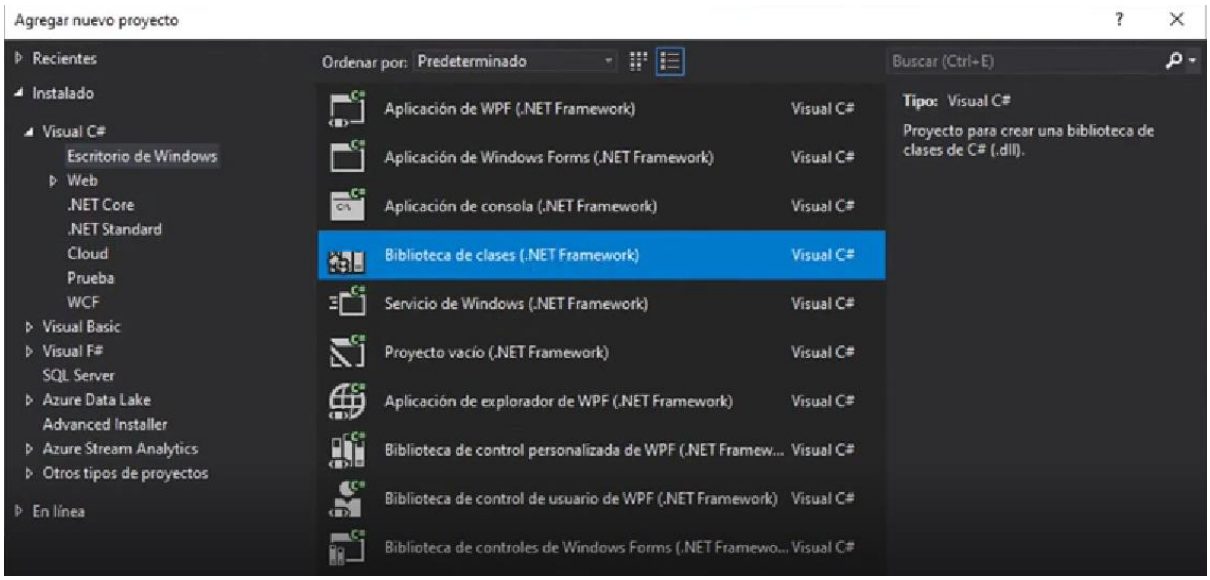
Las capas (y la encapsulación) facilitan considerablemente el reemplazo de funcionalidad dentro de la aplicación. Por ejemplo, es posible que una aplicación use inicialmente su propia base de datos de SQL Server, pero más adelante podría optar por usar una basada en la nube. Si la aplicación ha encapsulado correctamente su implementación dentro de una capa lógica, esa capa

específica de SQL Server se podría reemplazar por una nueva que implementara la misma interfaz pública.

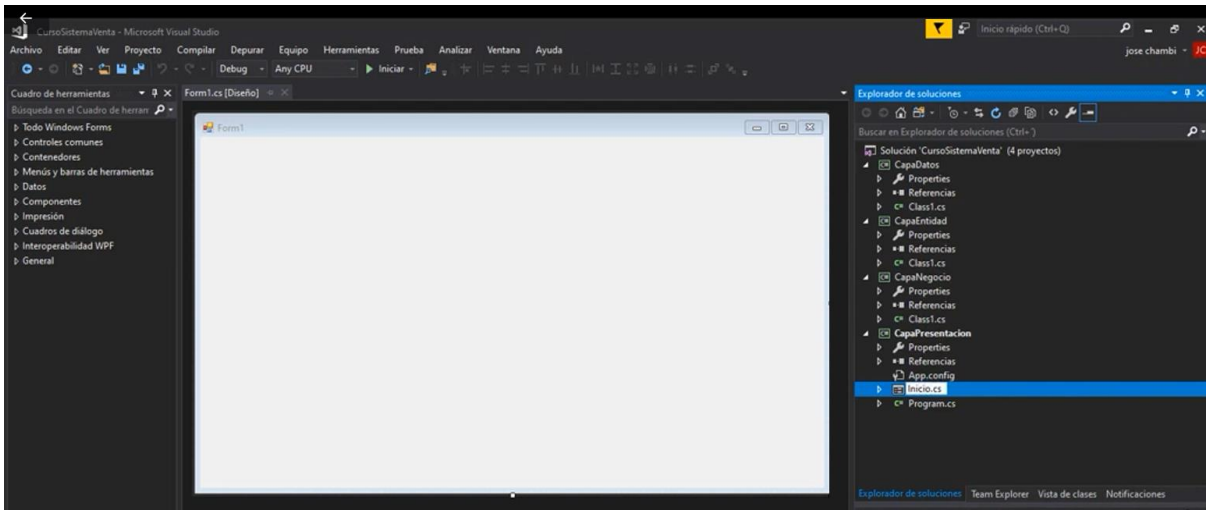
Además de la posibilidad de intercambiar las implementaciones en respuesta a cambios futuros en los requisitos, las capas de aplicación también facilitan el intercambio de implementaciones con fines de prueba. En lugar de tener que escribir pruebas que funcionan en la capa de datos reales o de la interfaz de usuario de la aplicación, estas capas se pueden reemplazar en tiempo de prueba con implementaciones falsas que proporcionen respuestas conocidas a las solicitudes. Este enfoque normalmente hace que las pruebas sean mucho más fáciles de escribir y mucho más rápidas de ejecutar en comparación con la ejecución de pruebas sobre la infraestructura real de la aplicación.

Las capas lógicas son una técnica común para mejorar la organización del código en las aplicaciones de software empresarial, y hay varias formas de organizar el código en capas.

Creamos una nueva solución C# forms con un proyecto llamado CapaPresentacion. Agregaremos nuevos proyectos de Biblioteca de Clases con los nombres CapaEntidad, CapaNegocio y CapaDatos.



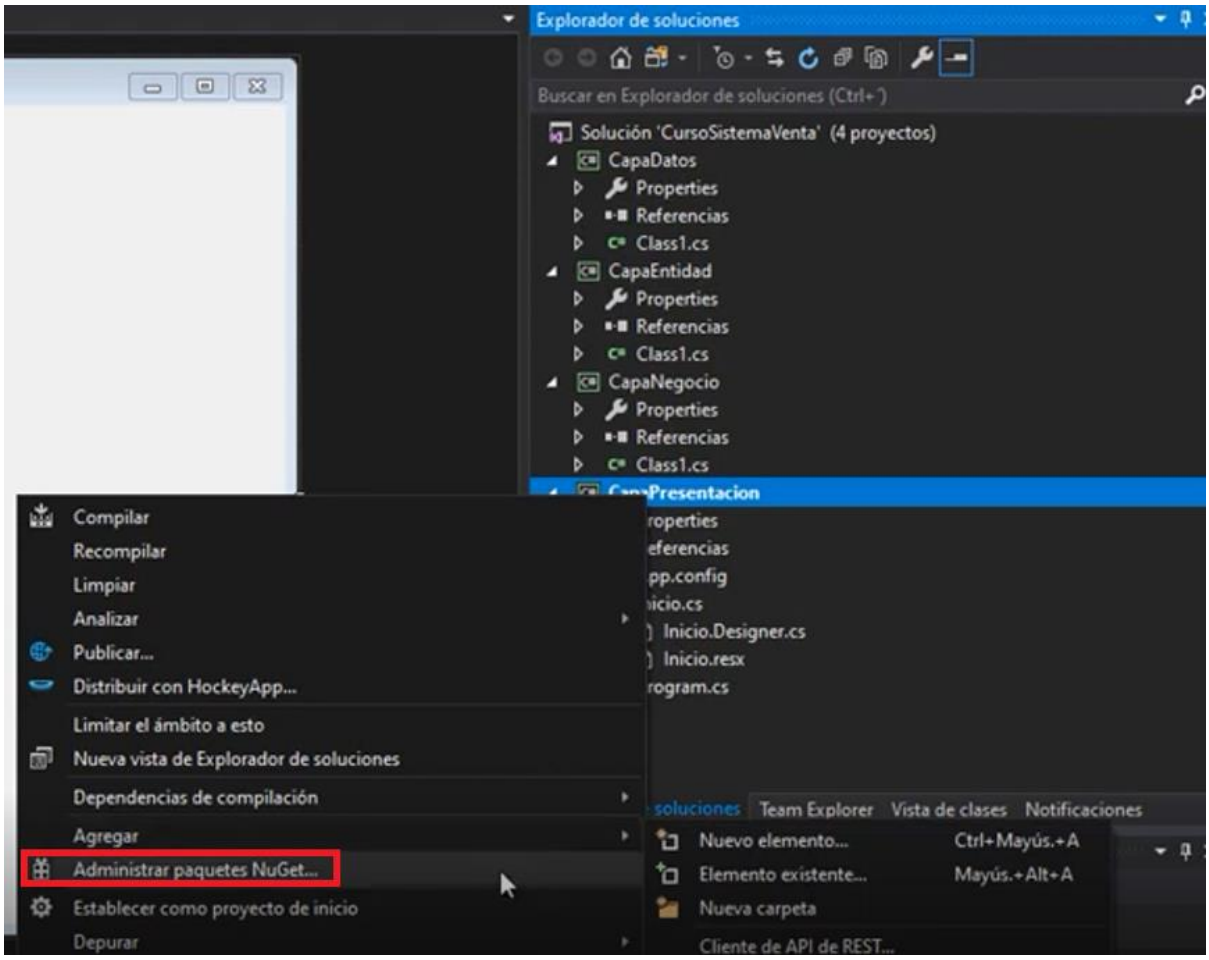
Comenzaremos a trabajar sobre nuestro proyecto CapaPresentacion accediendo a su formulario. Como siempre lo primero es cambiar el nombre del formulario, para seguir el ejemplo indicaremos como nombre “Inicio” y para que se ejecute en mitad de la pantalla StartPosition=CenterScreen



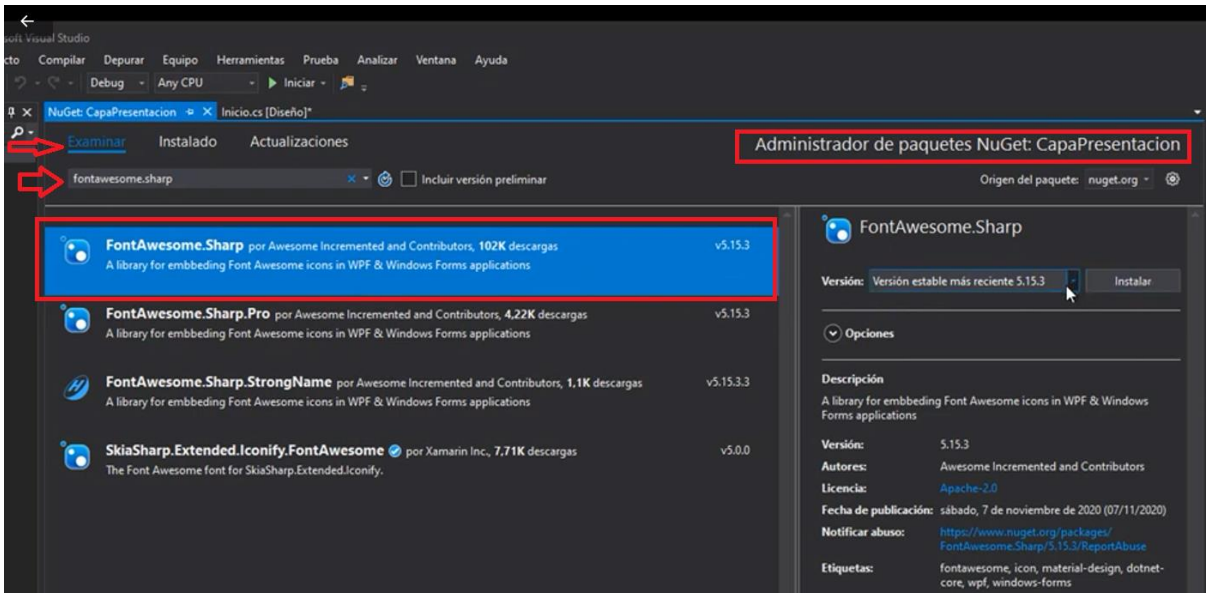
Para los iconos y botones vamos a utilizar un plugin (pequeños programas complementarios que amplían las funciones de aplicaciones. Cuando instalamos un plugin, el software para el que se instala adquiere una nueva funcionalidad)

Instalaremos Font Awesome para aplicaciones de escritorio (disponible en Bootstrap para aplicaciones web. Boostrap es biblioteca multiplataforma o conjunto de herramientas de código abierto para diseño de sitios y aplicaciones web)

Para instalarlo pulsamos con el botón derecho del ratón sobre nuestro proyecto CapaPresentación y seleccionamos **Administrar paguetes de NuGEt**:

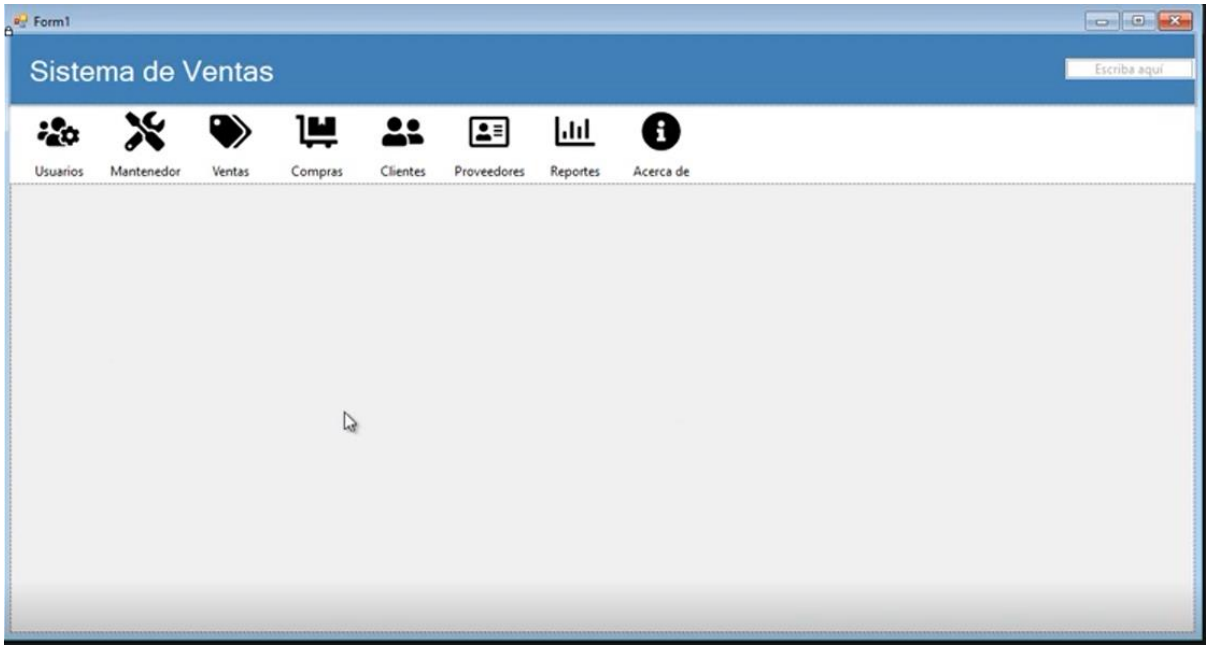


A través de **examinar** buscamos FontAwesome.Sharp de Awesome e instalamos (por defecto aparece situado en la pestaña Instalado donde podríamos comprobar previamente si ya lo tenemos disponible):

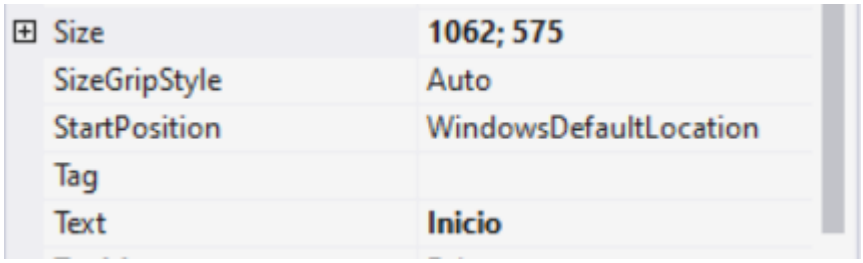


En el cuadro de herramientas se nos incorporará una pestaña más de FontAwesome.Sharp con los nuevos objetos disponibles.

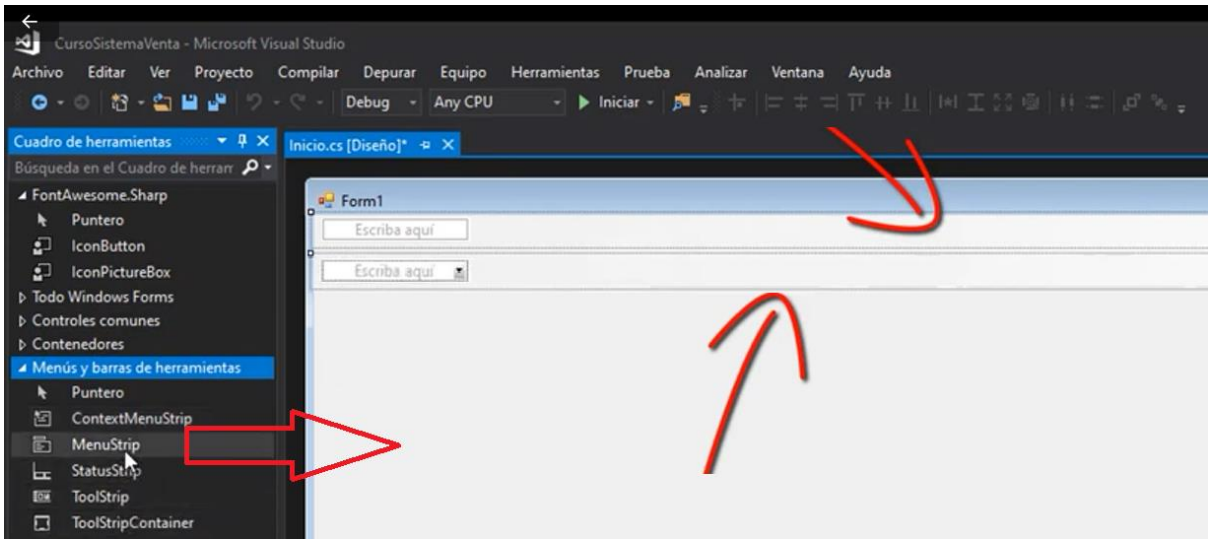
Vamos a diseñar el primer formulario:



Nos situamos en Form1del proyecto CapaPresentacion, cambiamos su nombre y propiedad Text a Inicio (si no lo hemos hecho previamente) y comprobamos su tamaño:



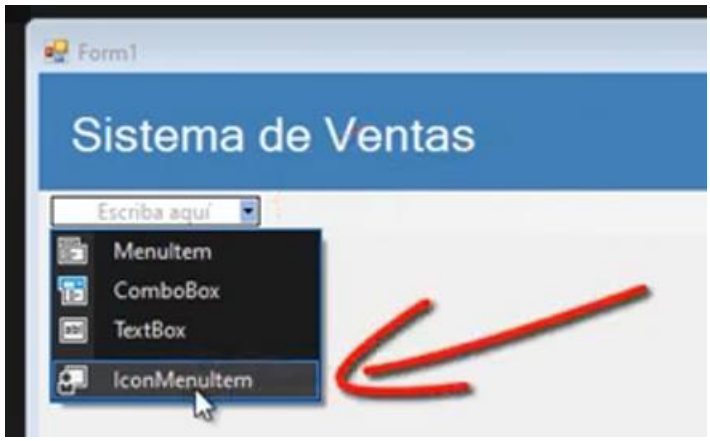
Para la parte del título y de los iconos incorporamos dos menús de tipo strip (https://docs.microsoft.com/es-es/dotnet/desktop/winforms/controls/menustrip-control-overview-windows-forms) que renombraremos como MenuTitulo y Menu:



Cambiamos varias propiedades del primer menú strip:

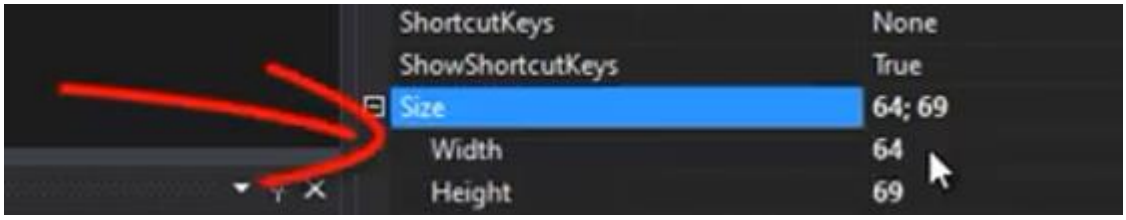
Ampliamos verticalmente el menú y con el botón derecho del ratón seleccionamos bloquear controles para que ya no se pueda modificar. Sobre el ponemos un Label que tiene que indicar "Sistema de Ventas" con tamaño fuente 20, color letra blanco y color de fondo el mismo que el menú (SteelBlue)

En el segundo menú strip pinchaos sobre el desplegable de "Escriba aquí" y seleccionamos IconMenuItem:



Buscamos la propiedad IconChar incorporaremos los distintos iconos.

Para el icono de usuarios elegimos icono Compass. Cambiamos la propiedad text a Usuario, ImageScaling a None, Icon Size a 50, TextImagenRelacion a ImagenAboveText,la propiedad AutoSize a False para que nos deje cambiar la anchura y cambiamos el tamaño de anchura (Width 80)



Copiamos este IconMenuItem siete veces para que nuestro Menú tenga ocho iconos en total.

Cambiamos el text de todos los iconos para identificarlos (menuusuarios, menumantenedor…)

Para cambiar la imagen del icono podemos hacerlo desde la propiedad IconChar o bien editando el fichero Designer.cs del formulario, desplegando el Código generado por el Diseñador de Windows Forms y cambiando la propiedad IconChar en el código de cada item:

Por cada item tendremos una línea de este tipo:

```
this.menuusarios.IconChar = FontAwesome.Sharp.IconChar.Compass;
```

Cambiamos el IconChar Compass por otro distinto para cada item:

```
// menuusarios
            this.menuusarios.IconChar = FontAwesome.Sharp.IconChar.UserCog;
```

```
/ menumantenedor
            this.menumantenedor.IconChar = FontAwesome.Sharp.IconChar.Tools;

// menuventas
            this.menuventas.IconChar = FontAwesome.Sharp.IconChar.Tags;

// menucompras
            this.menucompras.IconChar = FontAwesome.Sharp.IconChar.DollyFlatbed;

// menuclientes
            this.menuclientes.IconChar = FontAwesome.Sharp.IconChar.UserFriends;

// menuproveedores
            this.menuproveedores.IconChar = FontAwesome.Sharp.IconChar.AddressCard;

// menuinformes
            this.menuinformes.IconChar = FontAwesome.Sharp.IconChar.ChartBar;

// menuacercade
            this.menuacercade.IconChar = FontAwesome.Sharp.IconChar.InfoCircle;
```