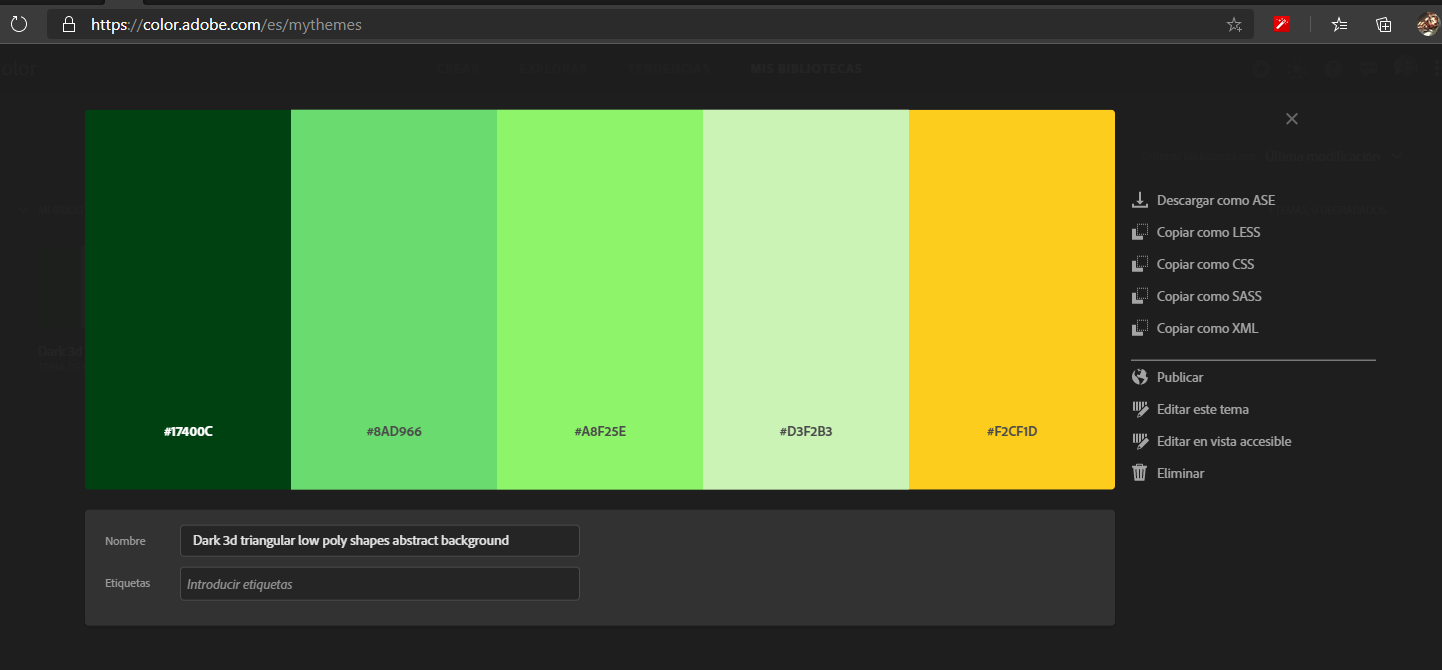
**PALETA DE COLORES DEL UI:**

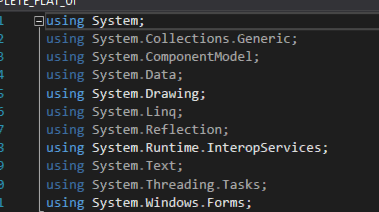
Para la creación de este proyecto se manejo la paleta de colores siguiente:

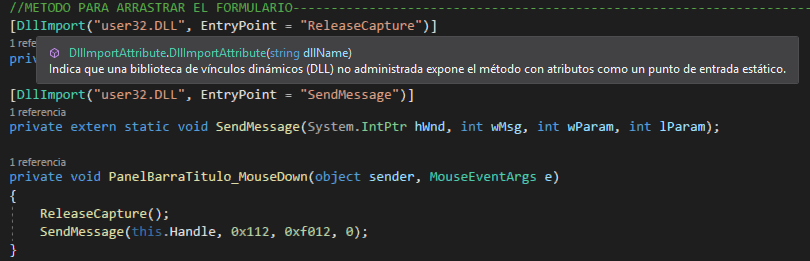


El archivo de esta paleta se incluye en la carpeta de archivos necesarios, se manejo una paleta acorde al logotipo usado, es importante mencionarlo para explicar el porqué de los colores.

**FormMenuPrincipal:**

Este es el formulario principal y es también el formulario que más código contiene:

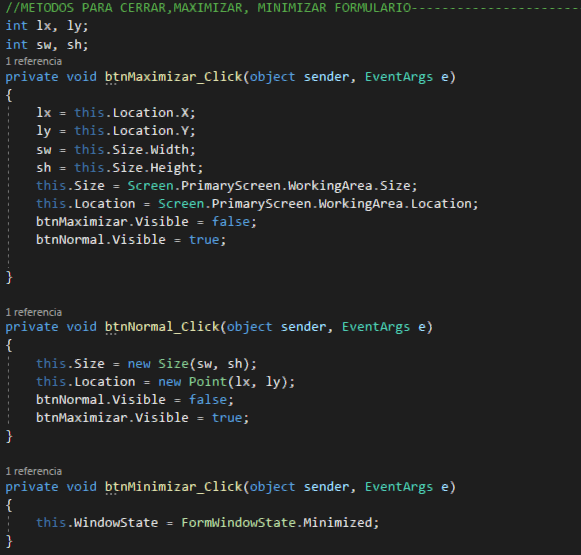


Los Using son las librerías que maneja este formulario, las cuales son necesarias para ejecutar cada uno de los componentes que se usan en el formulario, en estas la que librería que destaca es la de Runtime.IteropServices que permite que podamos ocupar librerías externas para poder ejecutar métodos de arrastre del formulario con la barra de titulo que en este formulario es creada con un panel para poder tener control con la interfaz de usuario así personalizarla aún más. 

Tanto este como los otros formularios tienen adecuados los botones de minimizar, maximizar, restaurar y cerrar como iconos externos dentro del panel antes mencionado:

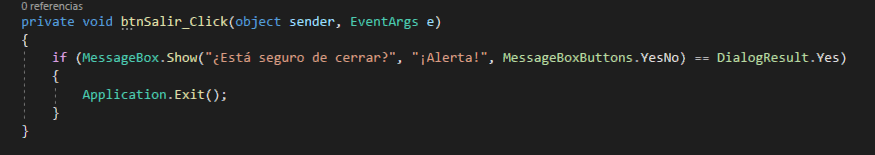


Para que estos iconos puedan funcionar introdujo el siguiente código:



Cómo se puede observar en la acción de clic de cada botón en donde se especifican los tamaños que deberá modificar el Form para la interacción de la ventana, estos tamaños se manejan como en el plano cartesiano con los ejes de la X y de la Y, en el caso de minimizar se invoca el estado de ventana y se cambia a minized para poder minimizar la ventana. Para acceder a la acción de clic de cada botón solo hay que dar doble clic en el modo diseño para poder modificar este código en cada uno, esto aplicar para todos los formularios del proyecto.

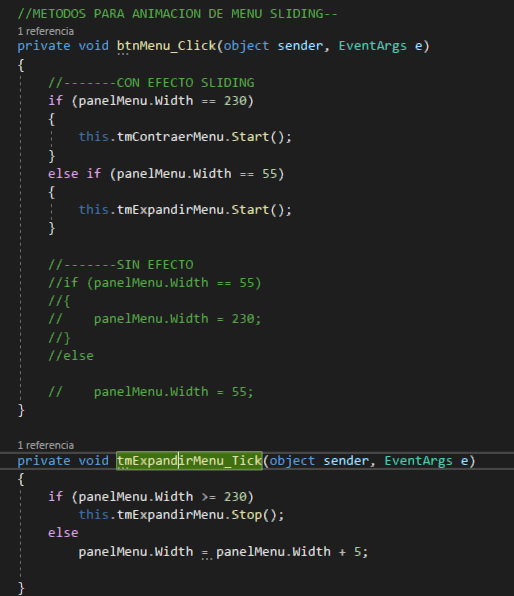
El botón de cerrar contiene el siguiente código:



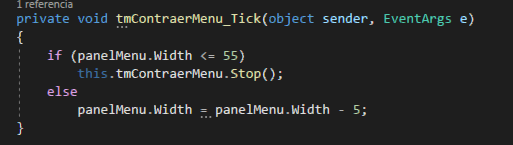
En donde condicionamos la acción mandando un MessageBox con una pequeña advertencia en donde si el usuario selecciona si se invocará al métodos Exit() de la Aplicación.

**Botón Menú: **

Al dar clic en el botón menú se tiene una animación de expandir el panel lateral para que aparezcan las opciones deseadas, para ello se utiliza el siguiente código:



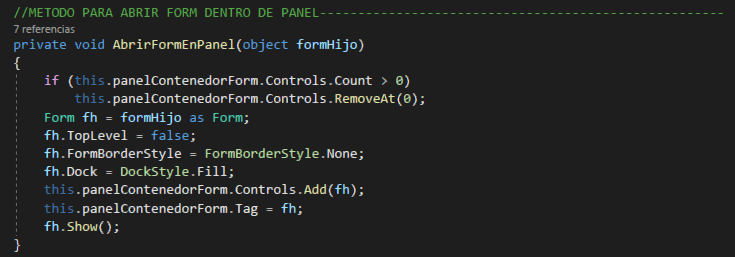
En donde podemos observar que también hacemos referencia a la creación de un método tmExpandirMenu\_Tick y al Método tmContraerMenu.



Los cuales ambos se invocan en la acción de clic del botón menú.

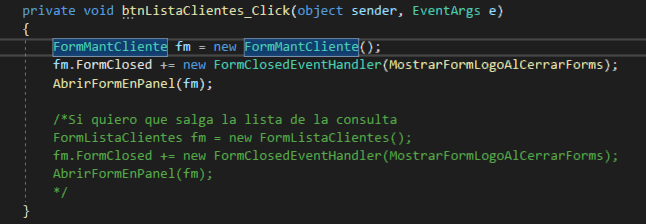
**Manejo de Forms dentro del panel de cada sección.**

Cuando damos clic a una sección en automático nos invoca un formulario el cual se visualiza incrustado en el panel a la derecha, esto lo logramos con el siguiente código:



En donde creamos el método AbrirFormEnPanel() en donde el cual estaremos invocando cada vez que queramos abrir un Form incrustado en alguna sección.

Por ejemplo en la sección “Registro” tiene el siguiente código:

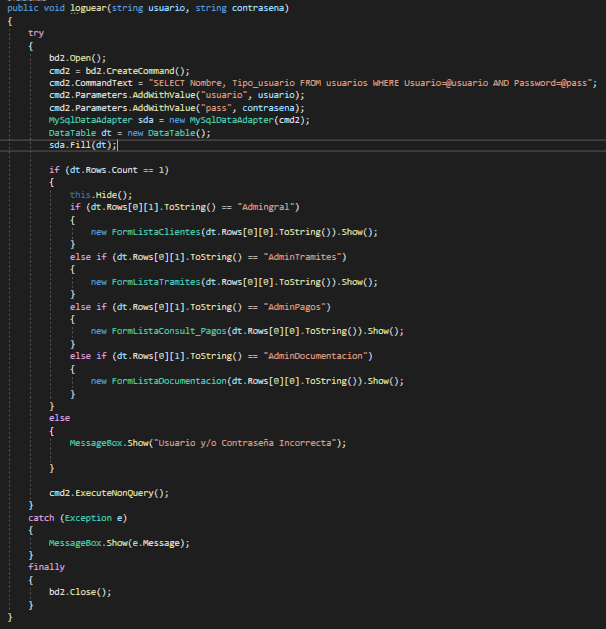


Donde podemos observar que se debe declarara una variable “fm” para almacenar el llamado de un Formulario a elegir en esta caso es FormMantCliente() posteriormente incluimos el constructor MostrarFormAlCerrarForms() que es un método que encapsula el Método AbrirFormEnPanel(fm) el cual aplicará el código anterior esto con el fin de que cuando cerremos el formulario incrustado se mantenga abierto el formulario principal.

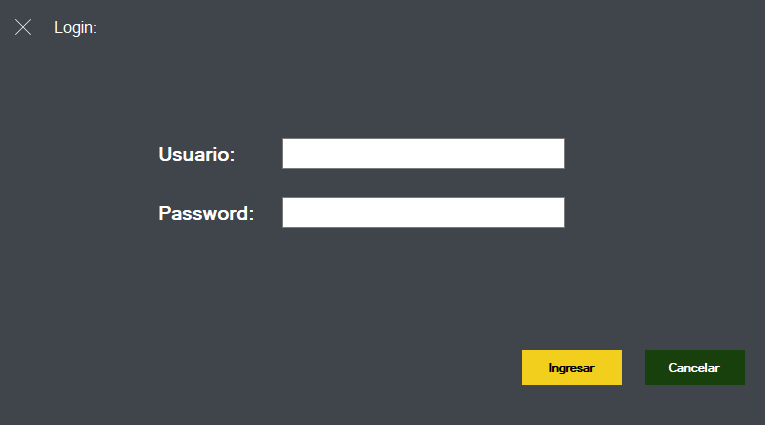
**Form Login:**

El formulario login nos permitirá que los usuarios administradores se puedan loguear e invocar cada uno de sus formularios.

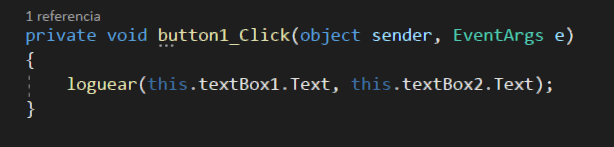
Para esto es necesario crear un método loguear() el cual espere como retorno dos string uno para el usuario y otro para la contraseña, cómo se muestra en el siguiente código:



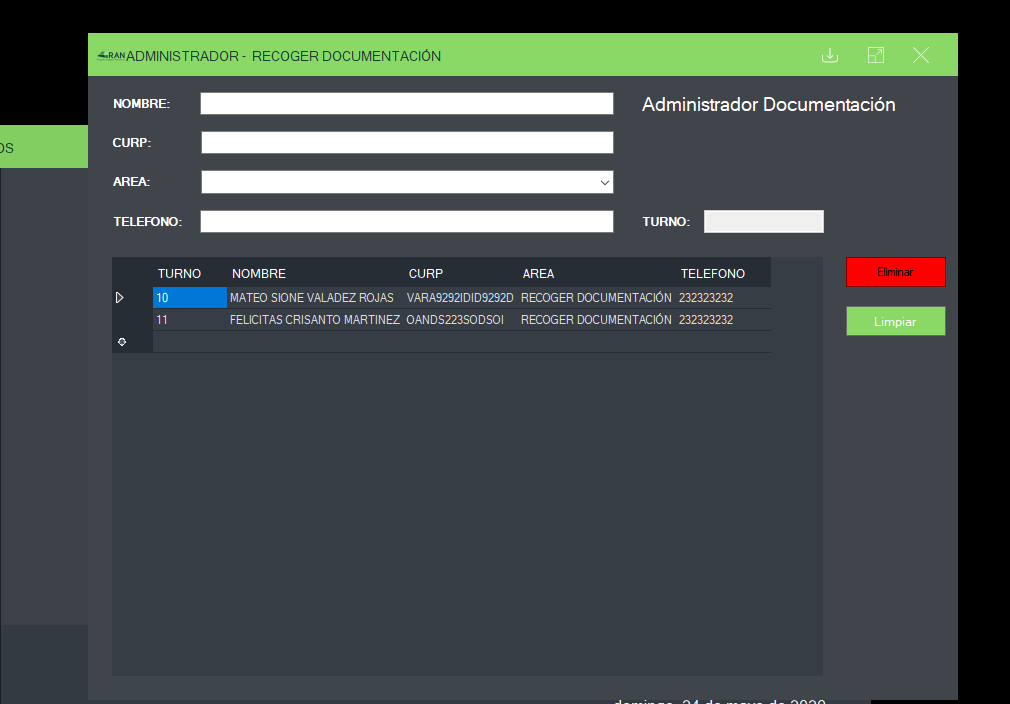
Para la verificación de las contraseñas y apertura de cada formulario se utilizó una condicional IF ELSE IF anidado para poder validar cada situación esto dentro de un Try Catch que llamará la conexión con la base de datos y que nos respaldará por si algo llega a fallar, en este apartado es importante saber que podemos cambiar las contraseñas de acceso para validar contra la base de datos (**NOTA: También en la base datos hay que cambiar la contraseña para que se pueda hacer la validación de la misma, estas se encuentran en la tabla Usuarios dentro de la base de datos.)**



Por último el botón ingresar deberá llamar al constructor loguear(); especificando los valores a enviar para que el método pueda funcionar, estos valores los extraemos de los textbox que se encuentran en el formulario login.cs

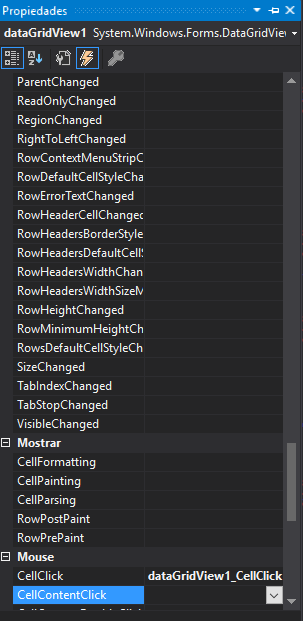


**Formularios (FormListConsul\_Pagos.cs, FormListDocumentacion.cs, FormListTramites.cs, FormListaClientes.cs)**

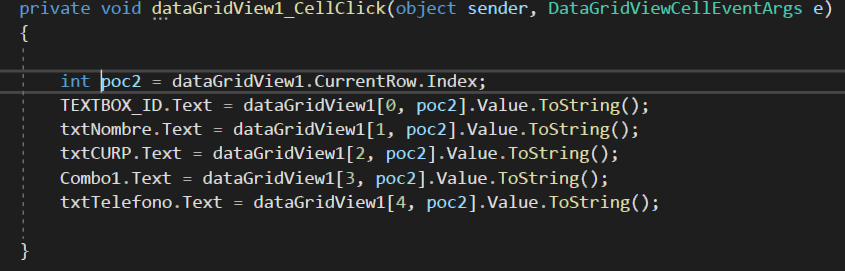


Para estos formularios se realizaron independientemente quiere decir que cada usuario administrador tiene un formulario, pero con las mismas funciones sus diferencias radican en la conexión a sus diferentes tablas ya que esta información esta separada y sólo mandan un respaldo a la tabla “visitantes” quien es la que guardar el historial de los visitantes.

Regresando a la construcción de los formularios cada uno tiene campos con datos de los visitantes estoy textbox se llenarán de forma automática cuando demos clic a algún registro de nuestro DataGridView y así una vez seleccionado podamos eliminarlos el registro mostrador en nuestros textbox. Con el botón de limpiar podemos borrar el contenido de los textbox, el código para todas estas acciones se muestra a continuación.

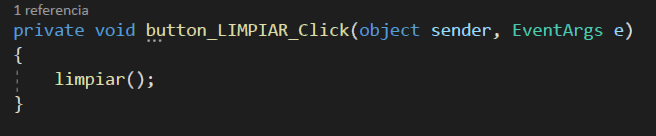
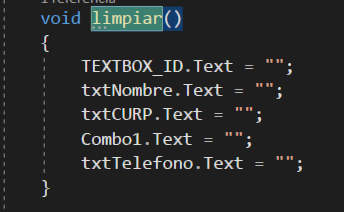


Para el DataGridView nos iremos a sus propiedades en el apartado de “Eventos” daremos clic en el evento de Mouse “CellClick” que es el evento que manipula cuando damos clic en alguna fila de nuestro DataGrid, al hacer eso nos mandará al siguiente código:



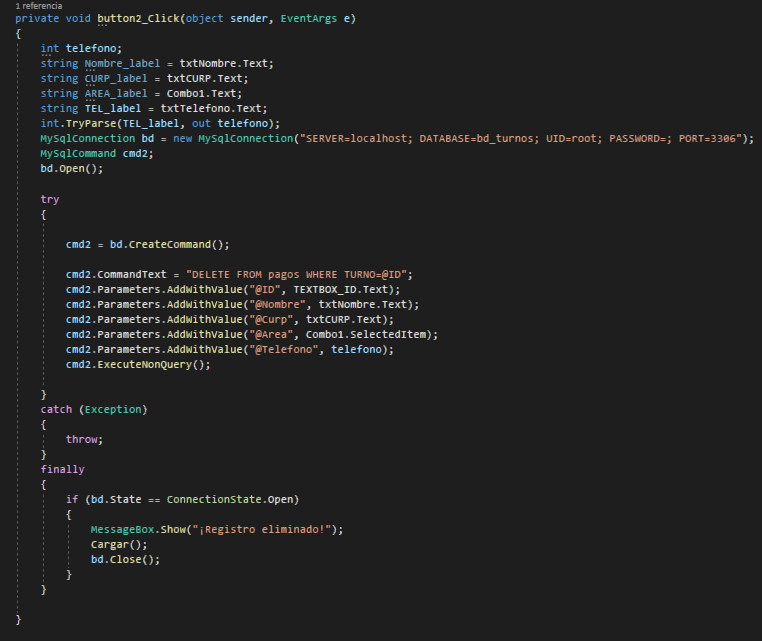
En donde declaramos un variable que estará almacenando nuestra posición sobre la fila del dataGrid, posteriormente estaremos creando matrices para indicar que posición se estará extrayendo la información para posteriormente establecerla en la propiedad Text de cada elemento textbox.

El botón Limpiar invoca el siguiente constructor:

El cual lo único que hace es establecer vacíos en la propiedad Text de cada elemento textbox y combobox.

Todos los botones de eliminar tienen el siguiente código:

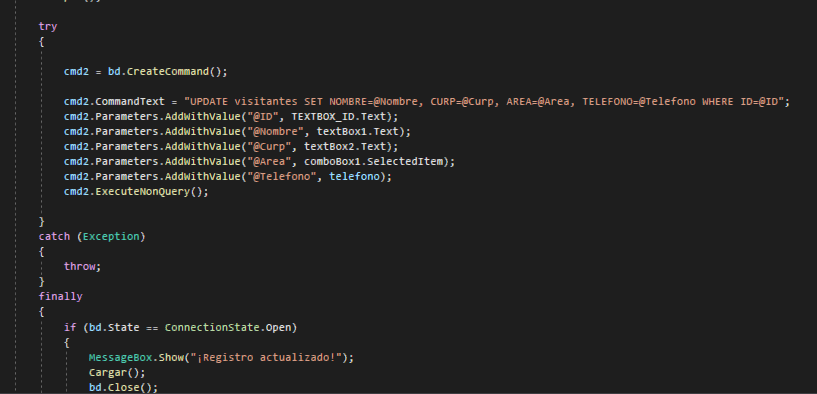


El cual contempla la conexión con la base de datos y los parámetros adecuados para la eliminación del registro, sólo cambiará la instrucción DELETE FROM (Por la tabla a elegir).

**FormListClientes:**

Este formulario es para poder visualizar el historial general de visitas a todos los departamentos, aunque cada departamento elimine el turno de la persona, aquí se mantendrá su información como un historial general. Este es el único formulario que tiene los botones de insertar registro y actualizar registro.

La acción del botón actualizar se resume en este código:



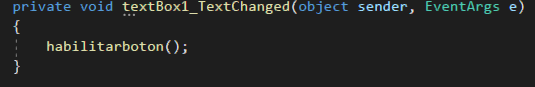
Lo único que cambia es su try catch con la instrucción de “UPDATE” así manteniendo las otras líneas de código como en el caso del botón eliminar de los otros formularios.

El código del botón insertar es el siguiente:

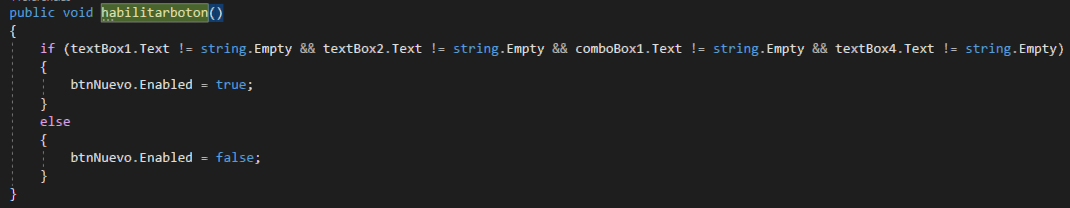


Cómo se puede observar se ocupan dos instrucciones SQL para poder almacenar en la tabla del departamento seleccionado y al mismo tiempo guardar información a la tabla visitante que es la que almacena el historial.

En este formulario hay una validación de datos, el botón insertar solo se activará cuando todos los textbox estén debidamente llenos, esto se realiza invocando el método habilitarboton();



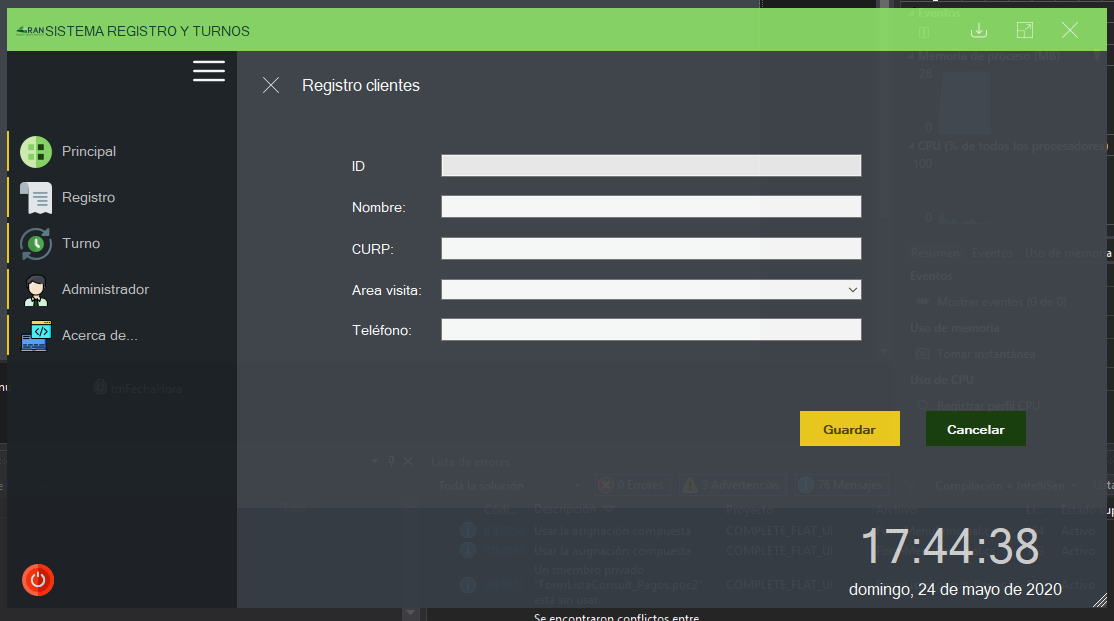
Este deberá estar en todos los textbox, el contenido del método es el siguiente:

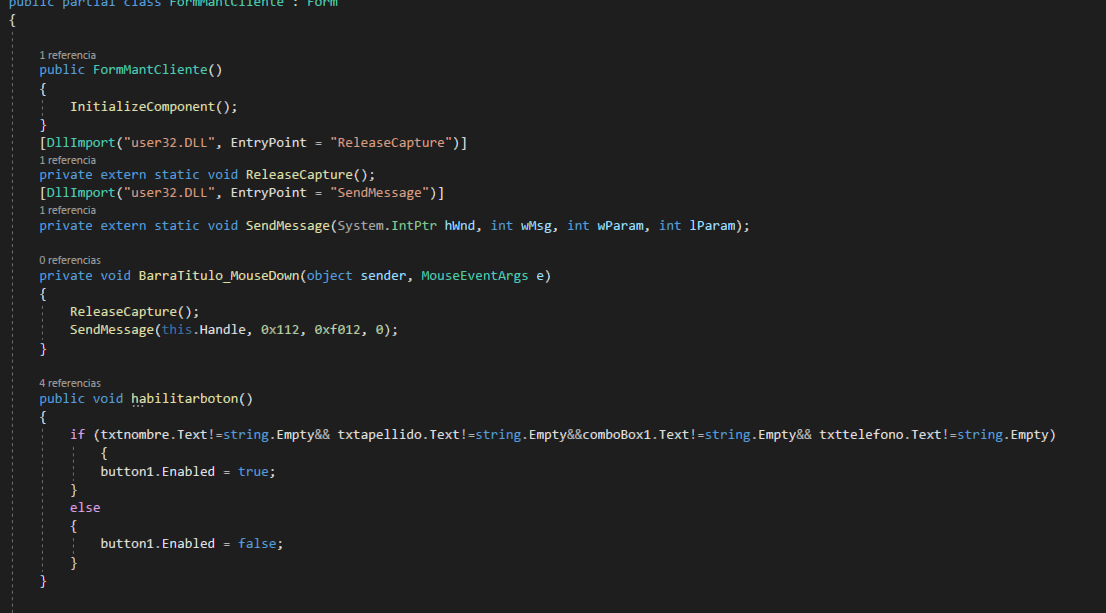


En donde usamos una condicional IF y preguntamos si la propiedad Text es diferente a vacío habilite el botón Insertar cambiando su propiedad Enabled a verdadero.

**FormMantClientes.cs**

El formulario de Registro Clientes es donde podemos encontrar los controles para que los visitantes se puedan registrar al módulo deseado.





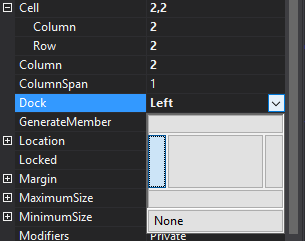
En este formulario también se aplican los métodos de validación de datos con cada uno de los campos textbox, las acciones de guardar registro y conexión a la base de datos los tiene le botón de guardar.

**Form\_Mostrar\_Turnos.cs**

Para la construcción de este formulario se ocupo la herramienta TabletLayoutPanel  .

La cual permite trabajar una cuadricula y acomodar los diferentes elementos utilizando columnas y filas para ello. Dentro de esta tabla se pusieron los diferentes labels a utilizar y se acomodaron en sus propiedades.

La propiedad Dock podemos definir el comportamiento dentro de cada celda:



En la propiedad TextAlign podemos definir la alineación del texto de cada label para poder obtener una mejor presentación visual de los datos.

Cada label de departamento se carga con métodos específicos los cuales son consulta\_tramites(), consulta\_documentacion(), consulta\_pagos().

En el siguiente código se mostrará el código de uno de los métodos realizados, es lo mismo para los demás sólo va a cambiar el nombre de la tabla a seleccionar.



En este código podemos observar que se crean dos variables de tipo comando para MySQL en cada uno de esto almacenera los comando que enviaremos a MySQL para las diferentes consultas, en esta ocasión para que la consulta SELECT nos pueda devolver el primer registro guardado utilizaremos la instrucción MIN, quedando así.

SELECT MIN (TURNOS) FROM pagos

***TURNOS ES EL ID DE CADA TABLA.***

***PAGOS ES EL NOMBRE DE LA TABLA EN DONDE BUSCAREMOS EL CAMPO TURNOS.***

***LA INSTRUCCIÓN MIN ES PARA QUE NOS DEVUELVA EL PRIMER REGISTROS EN ORDEN ASCENDENTE EN LA TABLA ASÍ CUANDO EL ADMINISTRADOR LIBERE ESE TURNO CONTINUE CON EL SIGUIENTE.***

En el caso de no haber datos en la tabla se utiliza la instrucción Catch con una Exception para establecer en las propiedades text de cada label el contenido “N/A” que es una referencia a “No aplica”.

En el caso especial del nombre del cliente se tiene que primero obtener el dato del turno, asignarlo como parámetro de variable al @TURNO para que posteriormente podamos obtener el dato del nombre ayudado de esta variable:

SELECT MIN(NOMBRE) FROM pagos WHERE TURNO=@TURNO

El resultado de esta consulta se convierte a un String para poder mandarlo a la propiedad Text del label que muestra el nombre.

