Clase 3 - **No Presencial. Por Guillermo Guastavino**

Ambiente de Desarrollo 2, repaso de C para C#, bases de VB .NET a partir de C 2, y cont. del proyecto con Navegador Web Integrado.

Buen día a todos. La cuarentena se extendió de nuevo (¡), así que llegamos a la 3ra clase no presencial.

Vamos a continuar con los temas de VB partiendo de C#, que para ustedes parten de C, así siguen repasando C, ven las diferencias para C# y aprenden también VB .NET

Continuamos desde dónde dejamos la explicación en la Clase 2, conviene que mires la clase 2 para refrescar todo.

Habíamos repasado de C los pre y post incrementos, veamos ahora la posibilidad de incluir una asignación dentro de otra:

```
a = ++b;
c = a + 4;
c = ((a = ++b) + 4); (Esta es igual a las dos de arriba juntas)
```

ves que donde iba a en la última expresión, agregamos directamente la asignación del valor de a. Se esta forma a vale ++b primero (ejecuta de adentro hacia fuera); luego le suma 4 y el resultado se le asigna a la variable c. Quedan con valor nuevo entonces a y c.

Negar un booleano (si es V pasa a F o si es F pasa a V)

C

```
n = !n (El signo de admiracion es para cambiar el estado siendo n boolean)
```

VB

```
n= not n
```

Obtener el módulo o resto de la operación

C

```
int x = 20 % 7
```

VB

```
Dim x as integer=20 mod 7
```

El modulo es útil para poder saber si un número es divisible por otro. Si da 0, es porque el resto es 0, entonces es un múltiplo.

Por ejemplo si quiero saber si AÑO que es un entero que tiene un año, es un año bisiesto (cada 4 años y por ende es múltiplo de 4 con resto 0):

Si año % (o mod) 4 es 0, es año bisiesto

Lo mismo para saber si es par, hacer x % 2 o x mod 2, si es 0 es par.

Estructuras

Nota: El console.writeline, genera una línea tipo PRINT de DOS, para no tener que cargar el valor en una label. Ustedes pueden poner el código a probar en el evento click de un botón, agregar una label label1, y en vez del console poner por ejemplo: label1.text =”Es verdadero”

IF ELSE

C#

```
bool condicion = true;

if (condicion)
{
    Console.WriteLine("Es verdadero.");
}
else
{
    Console.WriteLine("Es falso.");
}
```

Podría haber puesto if (condition == true), pero si una variable o expresión es boolean, por defecto se considera que pregunto por si es verdadero.

VB

Dim condicion as boolean

```
If condicion then
      Console.WriteLine("Es verdadero.")
Else
      Console.WriteLine("Es                                    falso.")
end if
```

```
La estructura se cierra con “end if”, también se puede escribir “endif”
```

Recordar que en VB, la comparación de igualdad se hace con un solo signo=, por ejemplo if a= 2 then. Si quisiera hacer if a=2 en C, estaría asignando 2 a la variable a, y luego evaluando la expresión como booleana…, muy diferente…

Cuando en VB la parte verdadera tiene una sola sentencia, y de existir, el else también tiene una sóla sentencia, se puede poner todo en una sola línea, sin end if:

If condicion then `Console.WriteLine("Es V.") Else Console.WriteLine("Es F.")`

Un IF THEN ELSE anidado en VB:
Dim a as integer=2, b as integer=0

```
If a=1 then
  b=3
else
    if a=2 then
      b=8
    else
        if a=3 then
           b=16
        else
           B=27
        endif
   endif
endif
```

Si a no es ni 1,2 ni 3, b vale 27

Como tienen solo una sentencia por V y una por F (la parte falsa tiene una sola sentencia, que es el próximo if), se puede poner todo junto:

If a=1 then b=3 else if a=2 then b=8 else if a=3 then b=16 else b=27

Si en V o en F hay más de una sentencia, hay que armarlo completo. La parte verdadera va desde el IF al end if, o desde el if al else.

Condicional función

Son expresiones que funcionan como condicionales, pero pueden usarse como parte de otras sentencias. En Excel el equivalente es la función =SI(condición, valor_V, valor_F)

Es igual a lo que ya vieron en C:

```
c = (a == 2) ? ( b * 3) : (b * 4);
```

si a vale 2, evalúa la parte V que es b * 3, sino es igual a 2, evalúa la parte falsa que es b + 4. El resultado se lo asigna a c

En C es bastante difícil de recordar la estructura, en cambio en VB es más simple. El mismo caso:

```
c=iif(a=2, b*3, b*4)
```

Selección multiple con Switch (C) y Select Case (VB)

En C#

```
INT A = 2;

SWITCH (A)
{
CASE 0:
-
-
-
BREAK;  //Se coloca break para que no continue buscando otra CASE
        si no se lo coloco y se cumple la condicion en mas de un
        CASE ejecuta esos otros CASE.
        En cambio el break, frena la busqueda.
CASE 1:
-
-
-
BREAK;
CASE 2:
-
-
-
BREAK;
DEFAULT: // cualquier otro valor
-
-
-
BREAK;
}
```

En VB tiene mayor funcionalidad:

```
Select case a             ‘(evaluamos a)
        case 0            ‘(sería: en el caso que a sea 0, entonces b sería 3)
                b=3
                c=8
        case 1
                b=5
                c=1
        case 2
                b=8
                c=10
        case else         ‘(sería: si no es ninguno de los anteriores, entonces…)
                b=5
                c=17
end select
```

En VB, poner : entre dos sentencias, es lo mismo que poner una sentencia en una línea, y la otra en la de abajo. Siguen siendo 2 sentencias, pero se lee mejor A=2: b=5: c=8 : f=7
Que:

A=1
B=5
C=8
F=7

Lo mismo se puede aplicar a cada case (a mi gusto se ve mejor…). Del ejemplo anterior:

```
Select case a
        case 0  : b=3 : c=8
        case 1 :  b=5 : c=1
        case 2 :  b=8 : c=10
        case else : b=5 : c=17
end select
```

Los case, también soportan estructuras más amplias.

Case 3,6,7,9,0 : b=9 (si a es 3, 6, 7, 9 o 0, b va a valer 9)

Case 5 to 145 : c=6 (si a está entre 5 y 145 inc., c valdrá 6)

Una pregunta: Tengo un bucle que lee 10.000.000 de registros contables desde una base de datos. A cada registro que viene con un valor, tengo que categorizarlo, es decir, que si va de tal a cual valor, le corresponde tal índice, y si va de este otro valor a este otro valor le corresponde este otro índice. Son 50 los tipos de índices que podría evaluar en cada registro. Qué es mejor?, usar switch o case , o bien usar IF anidado?

….. piensen
….. piensen
….. piensen
….. piensen

Hay un problema con la pregunta, dice “Qué es mejor?”, en forma absoluta. En realidad la misma cosa puede ser buenísima en un escenario (un entorno determinado de trabajo, una situación particular o un programa específico), y horrible en otro.
Si juzgamos 50 IF anidados contra un Switch o Select case, de acuerdo a su “claridad” para entenderlo y seguirlo, por ejemplo; diría que es mejor el Switch / Select Case, pero que pasa de acuerdo a la perfomance?. En un blucle de lectura y procesamiento, que se ejecuta 10 millones de veces, una ventaja o desventaja de hasta 1 milisegundo entre uno y otro sería significativa. Ese milisegundo, provocaría un retraso total de 10000 segundos al terminar el proceso, lo que equivale a 2.8 horas de más… es mucho, podría significar que no alcanzara a ejecutarse, que haya que pagar horas extras, volver de madrugada etc.
Sabemos que las clases heredan unas de otras, y que cada nivel del namespace agrega o modifica la funcionalidad. Un IF, un condicional, existe hasta en el set de instrucciones

del microprocesador. Entonces las clases que se construyen para llegar a un IF, son muy primitivas. El case / switch, debe estar usando clases derivadas de IF, porque es bastante parecido a un IF anidado. Podemos pensar entonces que ese agregado de funcionalidad y de niveles de herencia, deben estar consumiendo recursos. Hace mucho tiempo planteé esta duda con un curso, pusimos un if anidado en un bucle, y un case en otro. La diferencia fue de aproximadamente (se ve afectado por los otros procesos de Windows) de 27 milisegundos entre uno y otro….

Cuando se decide que código usar o como implementar una rutina crítica, hay que pensar que vamos a priorizar. Desde mi punto de vista, siempre trato de priorizar la perfomance. Pero para procesos fijos, como un fondo de un formulario o un menú, me pongo a hacer gráficos muy eleaborados, me puedo dar ese lujo porque no atenta contra la perfomance, siempre que logre optimizarlo para que no me tire en contra el peso del gráfico.

Todo depende del escenario.

Matrices, array, arreglos, variables o vectores, todos sinónimos

(si vector y matriz, para mi son sinónimos, desde el punto de vista que una matriz son varios vectores, y entonces son vectores también, el número no cambia la esencia de lo que son).
El primer índice es 0, así que una matriz de 5 elementos llega hasta el indice.... cual?... 4

Repaso C:

```
*** Array ***
// Matriz de una sola dimension
int[] A = new int [5]; //De 6 elementos ya que es en base 0
string[] B = new string[6];

A[0] = 5; // carga con 5 la matriz A en la posicion 0

// Matriz Multidimensional
int[,] A = new int[4,2];
int[,,] B = new int [4,2,3];

// Matriz dentro de otra matriz (Matriz en Matriz)
int[][] A = new int[3][];

declaracion y asignacion en un paso
int[] NumPares = new int[] { 0, 2, 4, 6, 8, 10, 12 };
```

En VB

```
Dim a(5) as integer  ‘es una matriz (o vector si te gusta más), de 6 elementos, sin valorar aún
a(0)=1
a(1)=4
a(2)= a(1)*2+a(0) ‘acá les estoy dando valores
```

También puedo declarar una matriz al mismo tiempo que la lleno de valores:

```
Dim a() as integer= {3, 4, 1, 2, 5, 0}
```

No se indica la cantidad al declararlo, la cantidad es la cantidad de ítems que agregue entre las llaves.

Por ejemplo, para los días de la semana:

```
Dim dias() as string={”lunes”,”martes”,”miércoles”, “jueves”, “viernes”, “sábado”, “domingo”}
```

Entonces dias(1) es… cual?...
Martes, porque empieza en 0…

Entonces le agrego un “” al comienzo, para que dias(1) sea el “lunes”

```
Dim días as string={“”, ”lunes”, ”martes”, ”miércoles”, “jueves”, “viernes”, “sábado”, “domingo”}
```

Matrices

Una “planilla” de 3 x 3:

5 6 2
7 8 4
1 7 8

```
Dim a( , ) as integer={{5, 6, 2}, {7, 8, 4}, {1, 7, 8} }
Cambiar un valor:
a(2,1)=56
```

Bucles: (el msgbox y messagebox.show abren un cuadro de diálogo - ventanita con botones para interactuar – en el medio)

C#

```
FOR (INT I = 0; I < 100;--> I ++)  { msgbox I }
```

VB

Dim I as integer
for i=0 to 99 step 1: msgbox I : next

el step es el ingremento o decremento. Por ejemplo step 2 en el caso anterior me muestra los números pares (ignoro al cero…). Step -1, siempre que empiece con el número más alto, por ejemplo for x=10 to 0 step -1 va a contar para "abajo".

(WHILE)

C#

```
int A = 8;
while (A < 20)
   {
     if (A == 10) { continue; } //para y vuelve a evaluar
     if (A == 19) { break; }  // sale del while
   }
```

(DO)

```
int C = 1;
  do
   {
         //codigo

   } while (C > 10);
```

VB

Dim a as integer=8

```
Do while (a<20)
     Msgbox a
     A+=1
Loop
```

Si a hubiera valido 20, no entraba al bucle

Dim a as integer=20

```
Do
     Msgbox a
     A+=1
```

Loop while (a<20)

Como la condición está al final, siempre entra al menos una vez

Dim a as integer=8

Do until (a=20)
    Msgbox a
    A+=1
Loop

Es “hasta” en vez de “mientras”

Do
    Msgbox a
    A+=1
Loop until (a=20)

Manejo de errores.

C#

```
    try
    {
        // acá va el código que quiero controlar por si falla algo
    }
    catch (IOException e)
    {
        // acá va el código cuando falló… por ejemplo avisar que hubo.
    }
```

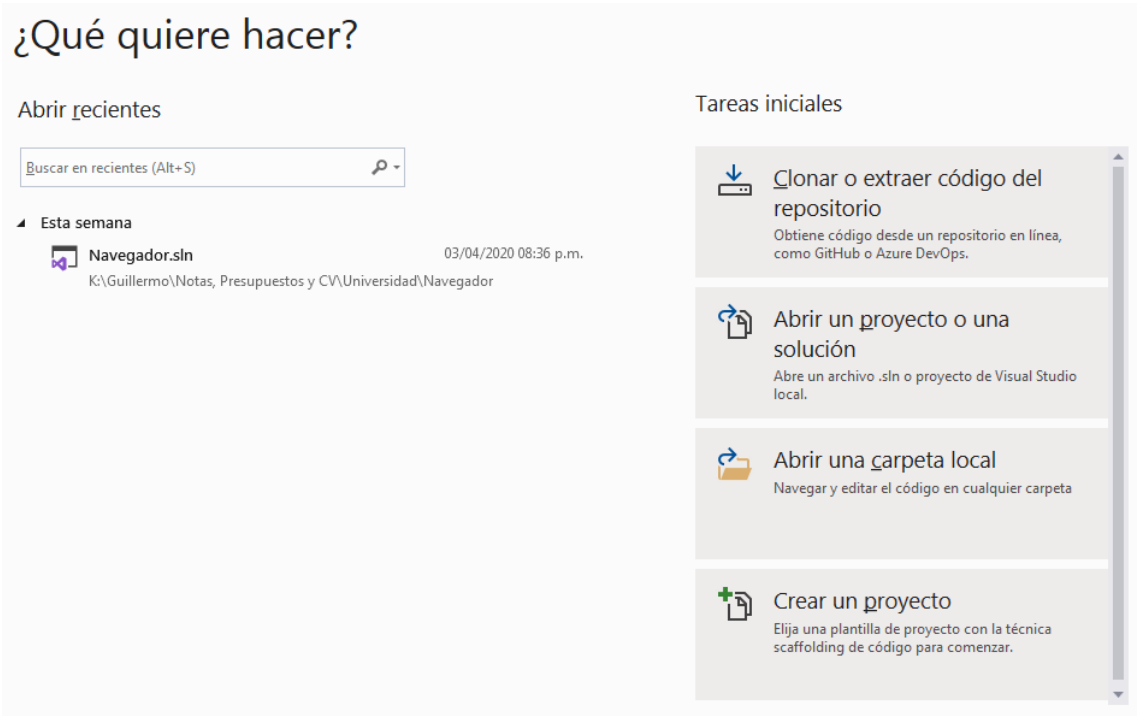
```
VB
      Try
            ‘Código que quiero controlar
      Catch
            ‘lo que hago si falla
      End try
```

En el catch de ambos casos podría poner por ejemplo

```
Messagebox.show(“se produjo un error.”)
```

Vamos a mejorar todo esto más adelante. Por ahora sólo adelanto lo que vamos a necesitar hoy.

Volvamos al proyecto Navegador de la semana pasada:

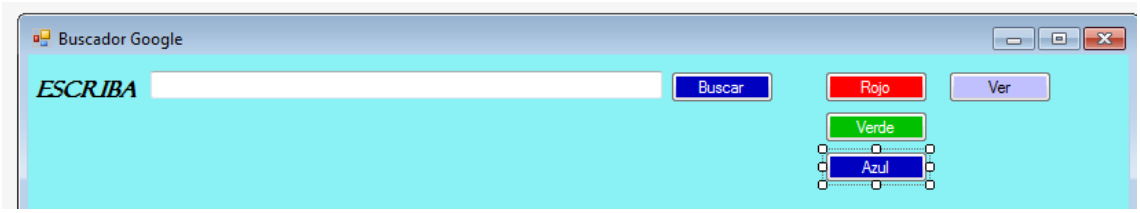


Puedo tocar a la izquierda en la lista de recientes, o tocar a la derecha en abrir un proyecto e ir a buscarlo

En mi caso, el menor esfuerzo…

La clase pasada vimos cómo cambiar propiedades en tiempo de diseño.

Agreguen 3 botones , uno rojo llamado bRojo, otro azul bAzul, y otro verde bVerde. Uno más llamado bVer. En tiempo de diseño, como en la clase pasada, cambien las propiedades de los botones, menos bVer. A todos pónganle la propiedad text a Rojo, Verde, Azul y Ver respectivamente.

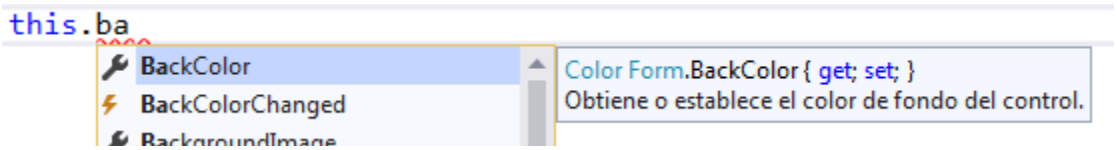


Click en bAzul para seleccionarlo y doble click para entrar en el evento Click

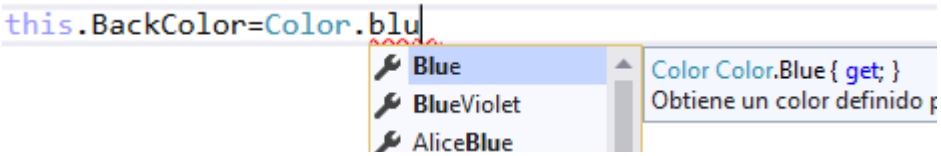
```
private void bAzul_Click(object sender, EventArgs e)
{

}
```

Si quisiéramos cambiar el backcolor de textbox tBuscar, pondríamos tbuscar.backcolor, pero queremos cambiar el del formulario. Yo no necesito decir cámbiame a Guillermo, basta que diga cámbiame a mi. Lo mismo pasa con el formulario (que puede haber cientos), basta con decir ME, a mí, al formulario actual. Lamentablemente eso funciona sólo para VB (que van a armar ustedes), para C# se usa **this**.

Vamos a utilizar el intellisense como la clase pasada, así que escriban this.b hasta que aparezca backcolor (si es otra propiedad que empieza con b, van a tener que seguir escribiendo hasta que la vean):



Como lo que seguía era el signo =, escríbanlo, y backcolor aparecerá automáticamente, luego empiecen a escribir color.blue, y ; enter para cuando tengan todo completo o tengan lo necesario para autorellenar.




```
1 referencia
private void bAzul_Click(object sender, EventArgs e)
{
    this.BackColor = Color.Blue;
}
```

Todo esto es más simple en VB, es más inteligente el intellisense.

Hagan lo mismo con los otros botones para que cada uno cambie el color como corresponda. Y ejecuten.

Para el botón bVer vamos a hacer que el botón de Buscar (en mi caso es button1), aparezca y desaparezca (propiedad visible), cada vez que aprieto bVer.

```
private void bVer_Click(object sender, EventArgs e)
{
    button1.Visible = !button1.Visible;
}
```

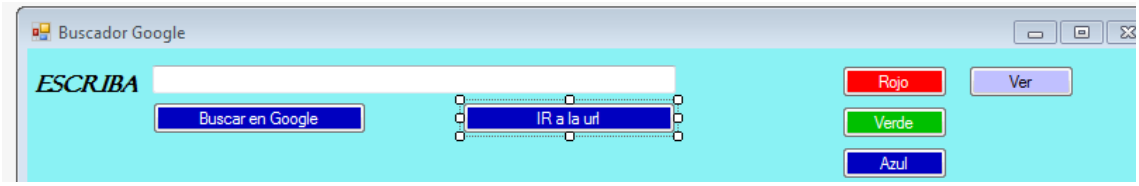
Para la versión VB (lo vimos antes), la negación es not

Lo que hace esa sentencia es que si button1.visible está en true (o sea se ve), lo pone en false y viceversa.

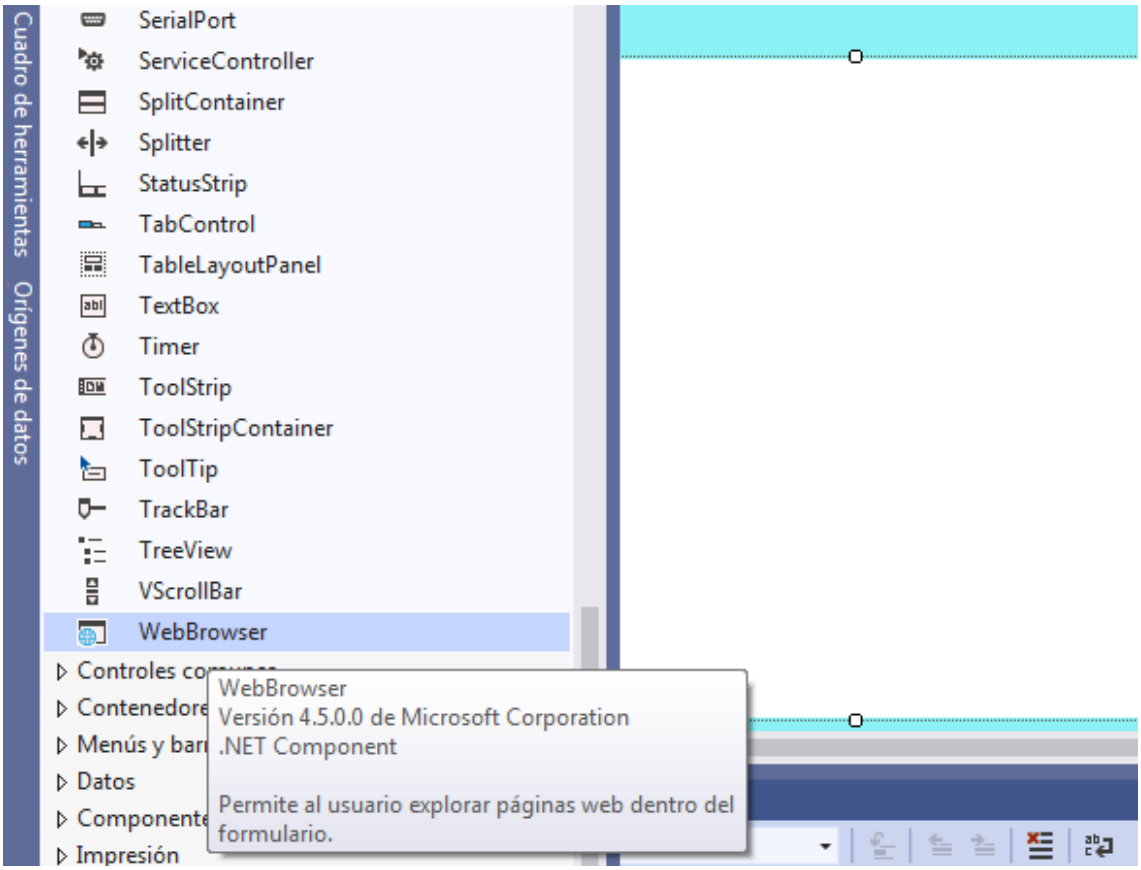
Estos botones son sólo de práctica, agreguen más objetos y traten de relacionarlos y cambiarles las propiedades en ejecución. No nos importa por ahora que estén en este proyecto, ustedes pueden ir armando uno mejor por afuera (que es lo que deben hacer…).

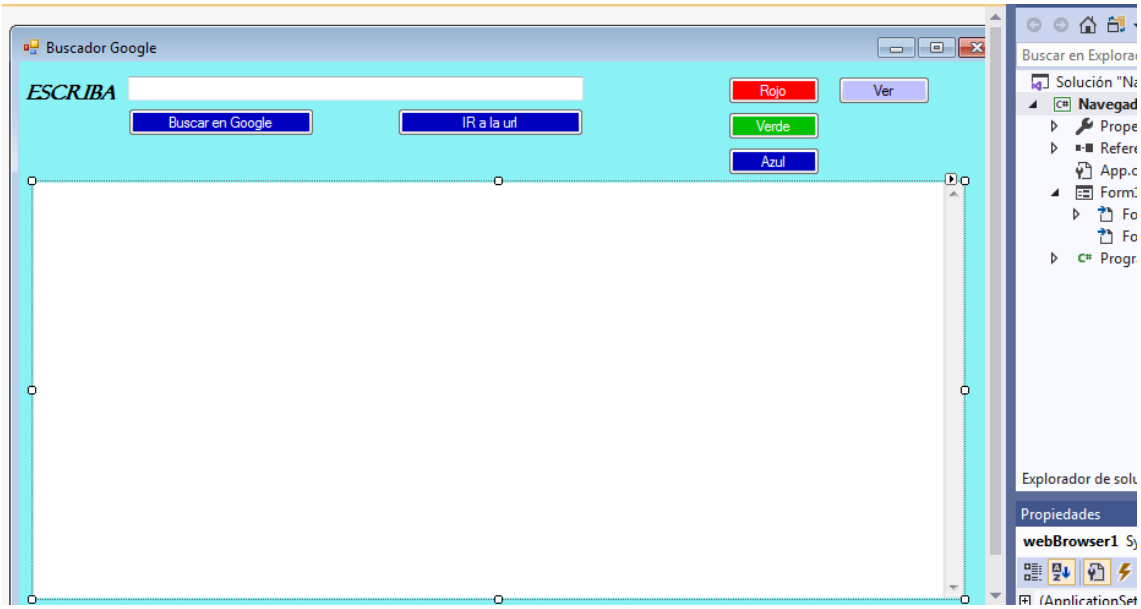
Un Navegador integrado

Cambiamos de lugar el botón de Buscar, y le cambiamos también el texto. Agregamos el botón bIr, con text “IR a la url”



Ahora buscamos en el Cuadro de Herrmanientas, al objeto WebBrowser, y creamos la instancia WebBrowser1. Háganlo grande, porque debe caber adentro una página web…





En el botón bIr:

```
private void bIr_Click(object sender, EventArgs e)
{
    if (String.IsNullOrEmpty(tBuscar.Text)) return;
    if (tBuscar.Equals("about:blank")) return;
    if (!tBuscar.Text.StartsWith("http://") &&
        !tBuscar.Text.StartsWith("https://"))
    {
        tBuscar.Text = "http://" + tBuscar.Text;
    }
    try
    {
        webBrowser1.Navigate(new Uri(tBuscar.Text));
    }
    catch
    {
        MessageBox.Show("Se produjo un error.");
    }
}
```

Línea por línea las cosas nuevas:

```
if (String.IsNullOrEmpty(tBuscar.Text)) return;
```

La clase isnullorempty dependientes de string, devuelve verdadero si tBuscar.text (la url que vamos a buscar), está vacío es o null (se aplica más para cuando veamos base de datos). Si está vacío hace return, con lo cual sale de la rutina.
En VB, será la Sub bIr_Click, entonces para salir de la sub, habrá que usar **exit sub**

```
if (tBuscar.Equals("about:blank"))
```

Si está vacío o no inicializado el objeto, vuelve.

```
if (!tBuscar.Text.StartsWith("http://") &&
                !tBuscar.Text.StartsWith("https://"))
```

tBuscar es un texto, tendrá entonces la propiedad Text que es puro texto. Si a cualquier variable de texto, o una propiedad de texto, o una sentencia que produzca texto, le aprietan punto . al final, se mostrarán todos los métodos disponibles (funciones que puedo aplicar entre otras cosas). En éste caso StartsWith, simplemente mira al comienzo del string (la supuesta url que escribimos), si existe el texto http://. Como niega la sentencia, está preguntando si NO le pusimos http adelante, para que luego lo agregue el programa.

Si no tenía http o https, le agrega http:// al comienzo.

Yo escribí sólo www.infobae.com, y le agregó http

La clase URI formatea una URL.

Al oprimir el botón, carga la página. Como webbroser es una clase de Internet Explorer, puede ser que les vaya tirando errores de incompatililidad de página, acéptenlos. Extrañamente, llamar a google la carga perfecto.

Deben completar la versión VB del proyecto. A partir de la próxima clase vamos a hacer al revés, vamos a trabajar en VB, porque ya vimos lo más importante de ambos lenguajes, y VB es más simple para focalizarnos en framework, y ustedes tendrán que ir armando las versiones C#.

Que tengan un excelente finde (hasta dónde la cuarentena lo permita) y muy Felices Pascuas.

Guillermo Guastavino