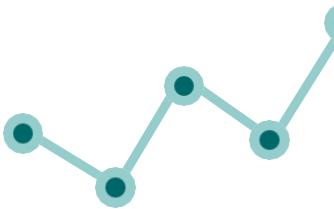




LENGUAJES DE PROGRAMACIÓN III





BIENVENIDA

Bienvenido(a) a la asignatura *Lenguajes de Programación III*, con la cual comprenderás los principios del lenguaje C#. Esto es importante porque complementa las bases de lenguajes anteriores que has practicado, como C++, pero además, aprenderás cómo trabajar con .NET, y estarás listo para afrontar los retos de la industria a nivel corporativo.

1
UNIDAD

INTRODUCCIÓN A MICROSOFT .NET



1.1

FRAMEWORK



TEMARIO



1.2

ENTORNO DE DESARROLLO





INTRODUCCIÓN

La asignatura de *Lenguajes de Programación III* tiene como objetivo exponer y enseñar el proceso de desarrollo en C#, comenzando desde el entorno de trabajo, las aplicaciones en consola y de escritorio, hasta realizar transacciones con BD.

En esta primera unidad el estudiante podrá identificar y comprender los distintos elementos y componentes de la plataforma .NET, así como los diferentes tipos de proyectos a través de un archivo de solución.

COMPETENCIAS A DESARROLLAR



El alumno será capaz de identificar los distintos elementos y componentes de la plataforma de .NET.



El alumno será capaz de comprender proyectos a través de un archivo de solución y de proyecto IDE, para el desarrollo de aplicaciones visuales.



VIDEO



TE INVITAMOS A VER EL SIGUIENTE VIDEO:



FRAMEWORK

Microsoft .NET es una plataforma de desarrollo de software con énfasis en transparencia de redes, con independencia de plataforma de hardware, que permite un rápido desarrollo de aplicaciones. Con esta propuesta se facilita la interconexión de distintas plataformas de hardware, software, información y usuarios.

Con base en esta plataforma se ha desarrollado una estrategia horizontal que integra productos que van desde sistemas operativos y herramientas de desarrollo, hasta las aplicaciones para usuario final.





```
1. /* not ready */
2. /* - form validation */
3. .edit tr:hover td {background:#eee}.button2-left, float:left;.button2-right span {
4. display: block;
5. float: left;
6. color: #065;
7. cursor: pointer;
8. .edit .button button {background:#D9EDF8; color:#09519;}.btn-toolbar .btn {
9. background-color: #D9EDF7;
10. text-decoration: none;
11. div.tooltip-editor {
12. img.caption_left {
13. float: left;
14. margin-left:auto;
15. margin-right:auto;
16. /* Calendar */
17. #form_publish_down img {
18. width: 16px;
19. height: 16px;
20. margin-left: 3px;
21. background: url(/images/system/calendar) no-repeat;
22. vertical-align: middle;
23. }
24. /* System Messages */
25. .error {
26. padding:0px;
27. }
```



.NET
Framework

Framework .NET es la plataforma de desarrollo de código administrado de Microsoft. Está formado por una serie de herramientas y librerías con las que se puede crear todo tipo de aplicaciones, desde las tradicionales aplicaciones de escritorio, hasta aplicaciones para Xbox pasando por desarrollo web, desarrollo para Windows Store y Windows Phone, y aplicaciones de servidor con WCF.

.NET Framework soporta completamente las características de la POO, también, el uso de Herencia, Polimorfismo, Clases, propiedades, métodos, eventos, constructores, etc.

FRAMEWORK

Características de Framework .NET

Las aplicaciones desarrolladas para .NET se ejecutan dentro de .NET framework. Admite el paradigma de programación orientado a objetos en el que el usuario puede trabajar con diferentes lenguajes como C#, VB.NET para aplicaciones. Algunas características de .NET Framework son:

- ▶ Interoperabilidad de idiomas
- ▶ Portátil
- ▶ Seguridad de tipos
- ▶ Compatibilidad con subprocessos múltiples administrados
- ▶ Rendimiento
- ▶ Ejecución en paralelo de idiomas
- ▶ Sistema de tipos comunes
- ▶ Cómputo paralelo
- ▶ Dynamic Language Runtime
- ▶ Gestión automática de la memoria
- ▶ Independencia del idioma
- ▶ Seguridad

FRAMEWORK

Componentes principales de Framework .NET

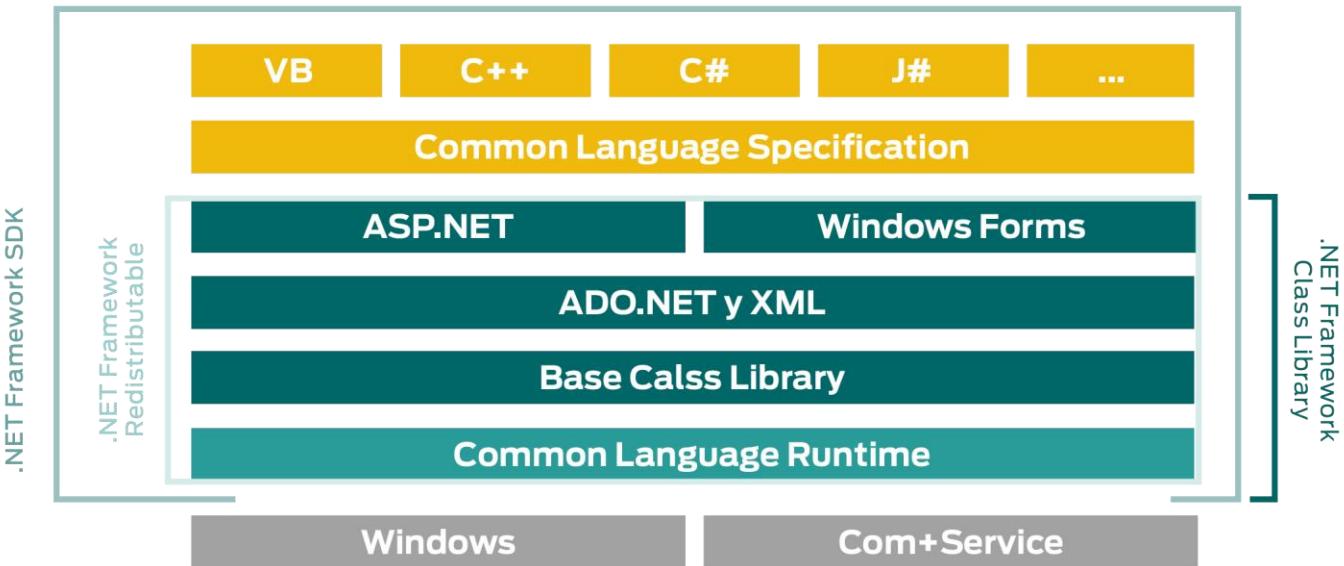
Este se compone de cuatro partes:

- Conjunto de bibliotecas de clases
- Grupo de lenguajes de programación
- Entorno ASP.NET
- .NET Framework

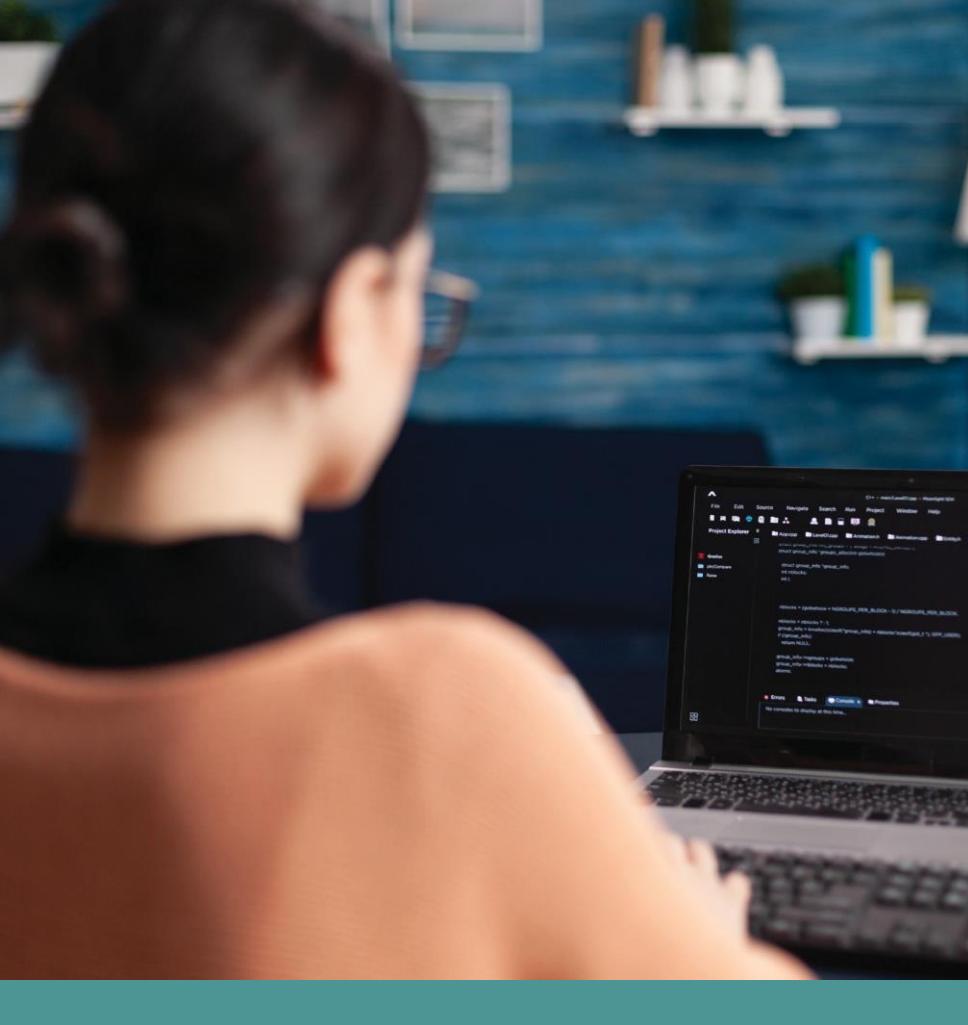
EL.NET Framework proporciona 3 elementos principales:

- ▶ Common Language Runtime o motor en tiempo de ejecución común para todos los lenguajes .NET
- ▶ .Net Framework Class Library o biblioteca de clases base del .NET Framework
- ▶ Colección de Frameworks de desarrollo

FRAMEWORK



Arquitectura de .NET Framework



CLR gestiona la ejecución de las aplicaciones diseñadas para la plataforma .NET. Por eso el código de estas aplicaciones es llamado **código gestionado**. Al código no escrito para ser gestionado directamente en la plataforma .NET se le llama **código no gestionado**.

CLR garantiza la seguridad de los tipos de datos. Avala que no se produzcan errores en la conversión de tipos en la ejecución de una aplicación.

FRAMEWORK

Lenguajes de Programación

Microsoft .NET desarrolla activamente tres lenguajes de programación: **C#, F# y Visual Basic (VB)**; es un error muy común, decir que C++ está dentro de .NET. El lenguaje C++ cuenta con su propio compilador, en el que C++ no puede entrar directamente a la plataforma .NET, ya que hay cosas que .NET no soporta.

Por lo tanto, Microsoft creó C++/CLI, que es básicamente C++ con extensiones de Microsoft que le permiten escribir código dirigido a .NET Framework, pero C++ no se incluye en .NET. En conclusión, los únicos lenguajes de .NET son C#, F# y VB.





C# es un lenguaje de programación simple, eficaz, moderno, orientado a objetos y seguridad de tipos, además incluye soporte para programación orientado a componentes, y mantiene sus raíces en la familia de lenguajes estilo C.

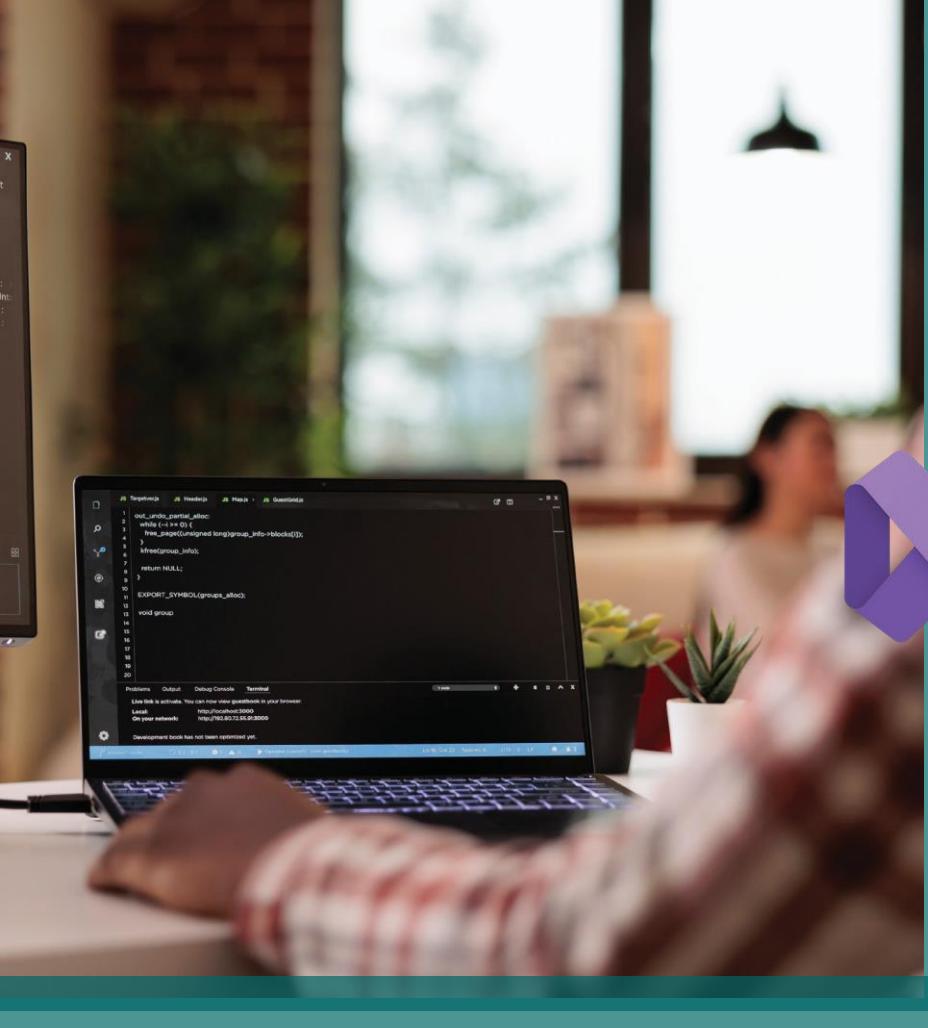
F# es un lenguaje de programación funcional, consiste principalmente en definir tipos y funciones que son de tipo inferido y generalizadas automáticamente. Esto permite que su enfoque permanezca en el dominio del problema y manipule sus datos; ofrece interoperabilidad limpia con C# y bases de código existentes.

FRAMEWORK

Visual Basic es un lenguaje accesible con una sintaxis simple para crear aplicaciones de tipo seguro y orientadas a objetos. Entre los lenguajes .NET, la sintaxis de VB es la más cercana al idioma normal, lo que a menudo facilita el trabajo a las personas sin experiencia en desarrollo de software.



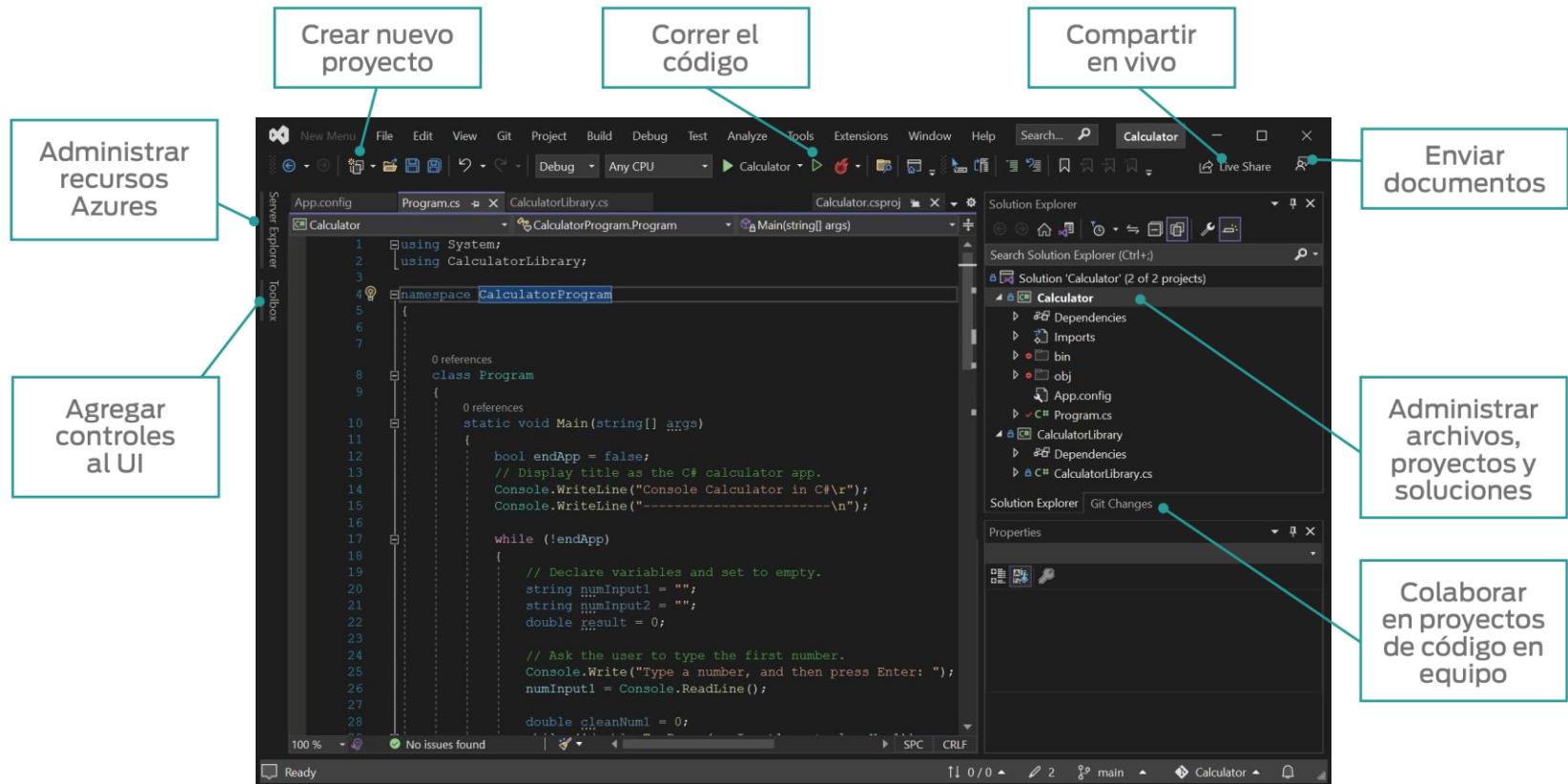
ENTORNO DE DESARROLLO



Interfaz de desarrollo

.NET se instala directamente en el entorno de desarrollo de IDE Visual Studio, ya que este provee un modelo de programación simple, seguro y soporta herramientas potentes con ayuda de la distribución, empaquetado y soporte.

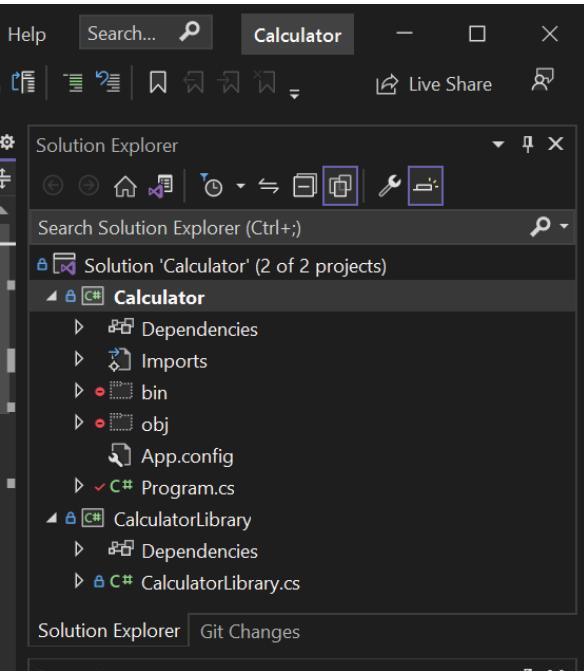
FRAMEWORK



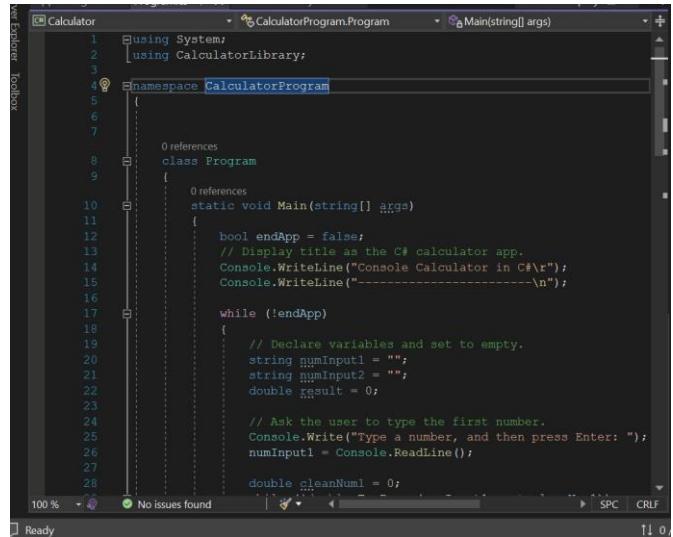
ENTORNO DE DESARROLLO

En la imagen anterior se muestra Visual Studio con un plan abierto con las ventanas principales y su funcionalidad.

En la parte superior derecha del Explorador de soluciones, se puede ver y regir los archivos de código y navegar por ellos. El Explorador de soluciones puede contribuir a ordenar el código, al agrupar los archivos en resoluciones y proyectos.



FRAMEWORK



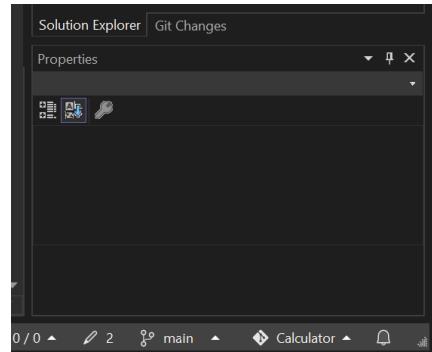
The screenshot shows the Microsoft Visual Studio IDE. The main window displays a C# code editor with the following code:

```
1  using System;
2  using CalculatorLibrary;
3
4  namespace CalculatorProgram
5  {
6      class Program
7      {
8          static void Main(string[] args)
9          {
10             bool endApp = false;
11             // Display title as the C# calculator app.
12             Console.WriteLine("Console Calculator in C#\r");
13             Console.WriteLine("-\r\n");
14
15             while (!endApp)
16             {
17                 // Declare variables and set to empty.
18                 string numInput1 = "";
19                 string numInput2 = "";
20                 double result = 0;
21
22                 // Ask the user to type the first number.
23                 Console.Write("Type a number, and then press Enter: ");
24                 numInput1 = Console.ReadLine();
25
26                 double cleanNum = 0;
```

The status bar at the bottom indicates "Ready".

La ventana central es donde se muestra el contenido del documento o codificación. Ahí, se puede editar código o diseñar una interfaz gráfica de cliente, como una ventana con botones y cuadros de escrito.

En la parte inferior derecha de **Git Changes** (Cambios de Git), se puede hacer el seguimiento de recursos de trabajo y compartir código con otros usuarios por medio de tecnologías de control de variantes, como Git y GitHub.



ENTORNO DE DESARROLLO

Elementos de la Plataforma de Ejecución Intermedia .NET

El entorno de ejecución de Framework .NET es CLR (Common Language Runtime). Este gestiona la memoria, la seguridad, el control de errores, etc. Esto permite que varios lenguajes puedan interactuar entre sí.

La plataforma .NET es denominada de ejecución intermedia. Justamente porque se ubica entre el sistema operativo y las aplicaciones finales con las que interactúan los usuarios, fungiendo como intermediaria entre ambos.



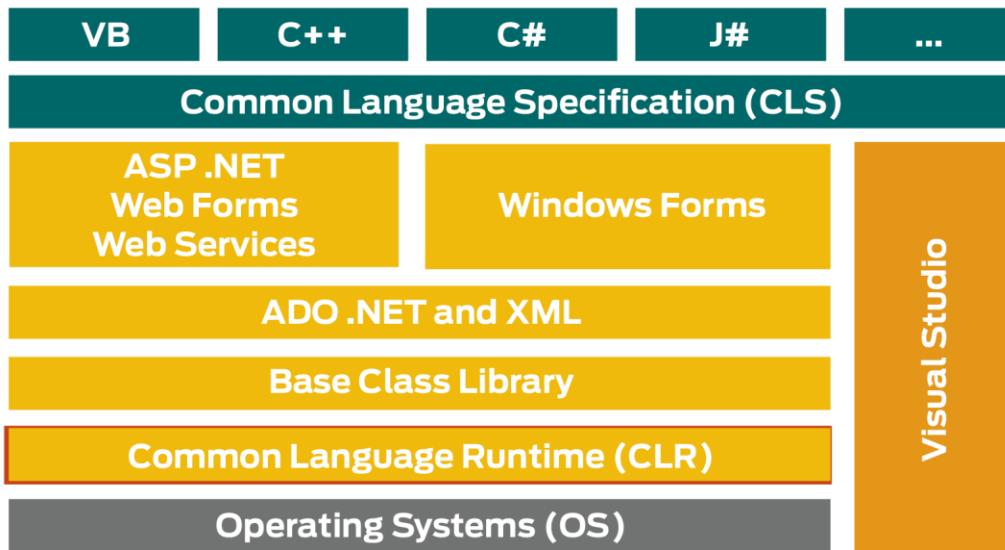
Elementos de la Plataforma de Ejecución Intermedia

ENTORNO DE DESARROLLO

Common Language Runtime (CLR)

Se trata de un programa de ejecución común a todos los lenguajes. Se encarga de leer el código generado por el compilador, y de iniciar su ejecución.

El funcionamiento del CLR se da sobre el sistema operativo, para aislar su plataforma. Esto permite ejecutar aplicaciones .NET multiplataforma.



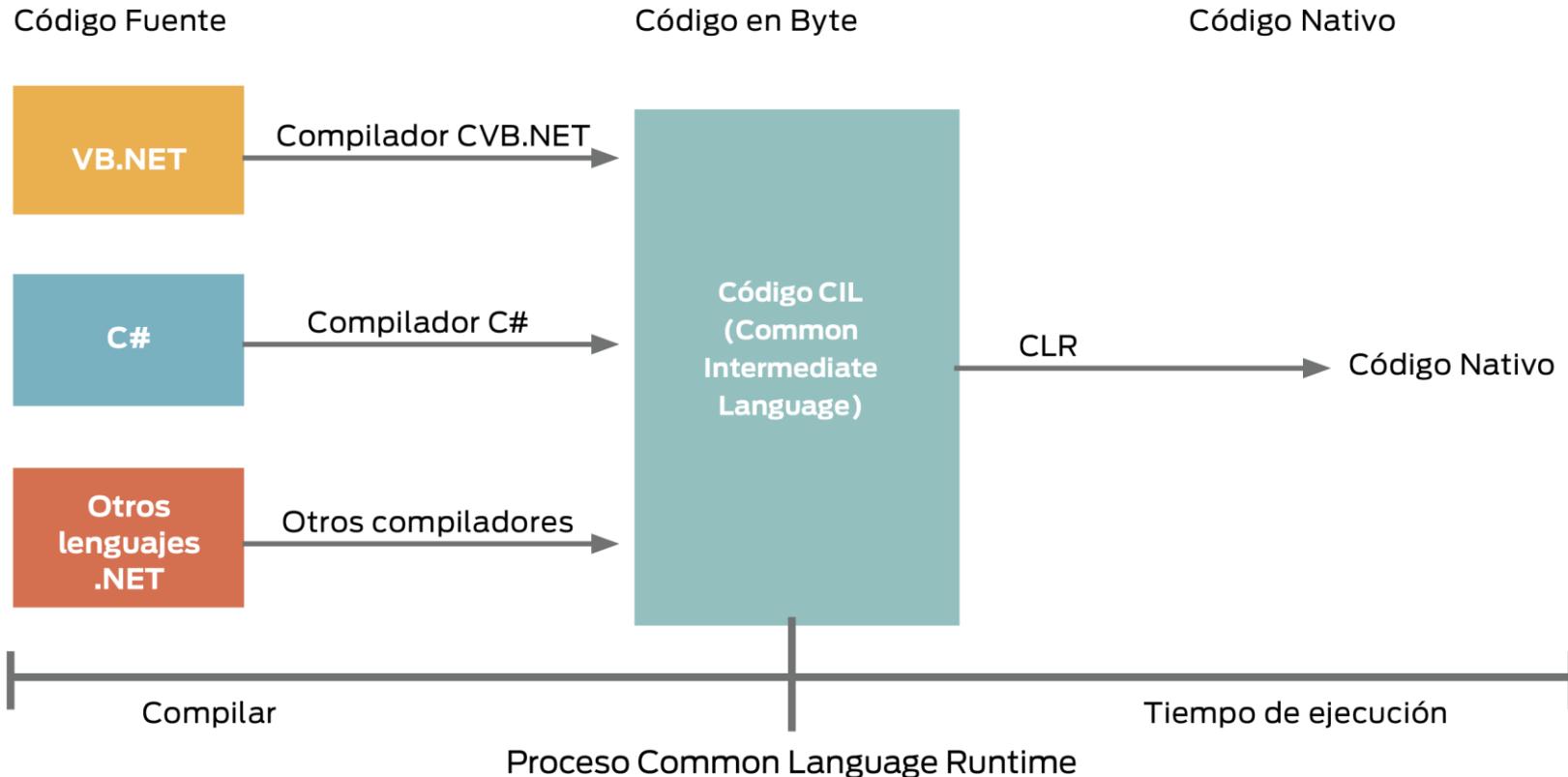
ENTORNO DE DESARROLLO

La función principal de CLR es gestionar la ejecución de las aplicaciones diseñadas para la plataforma .NET. Por tal razón, al código de estas aplicaciones es llamado **código gestionado**, y al código no escrito para ser ejecutado directamente en la plataforma .NET se le suele llamar **código no gestionado**.

CLR garantiza la seguridad de los tipos de datos, avalando que no se producen errores en la conversión de tipos en la ejecución de una aplicación .NET. Este aspecto y algunos otros son regulados por lo que se conoce el Common Type System (CTS).

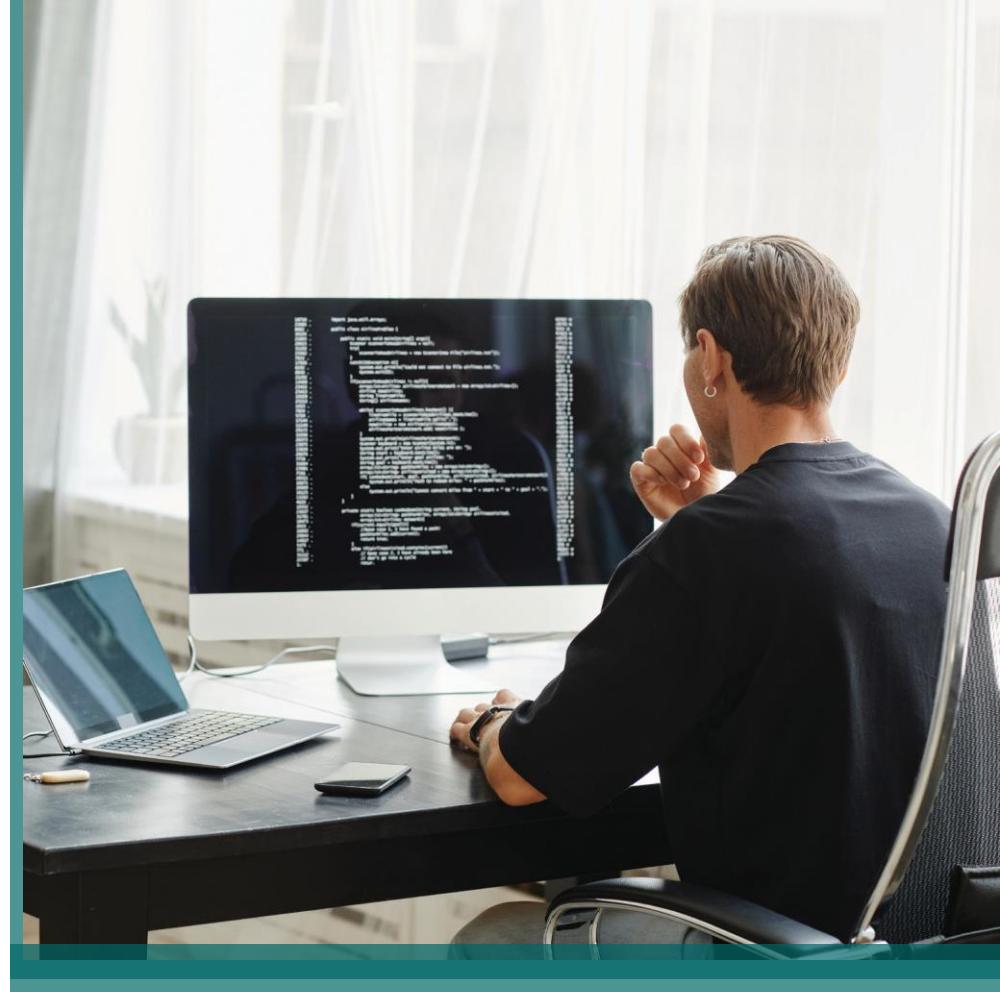


ENTORNO DE DESARROLLO



Microsoft Intermediate Language (MSIL)

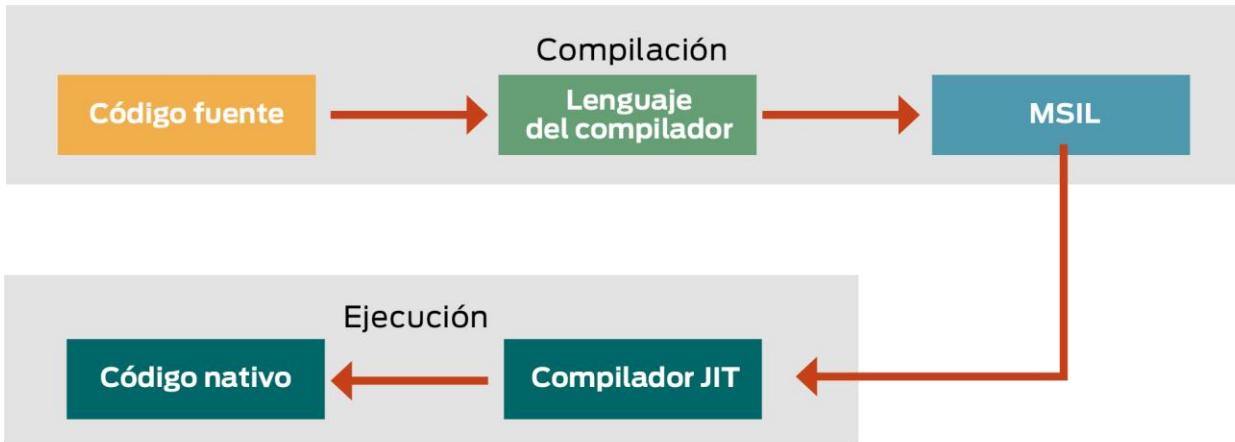
Representa el código objeto al que es transformado el código fuente, escrito en cualquiera de los lenguajes aceptados por .NET. Estos que luego van a ser compilados a nativo por el compilador JIT (Just In Time), que provee la CLR en tiempo de ejecución.

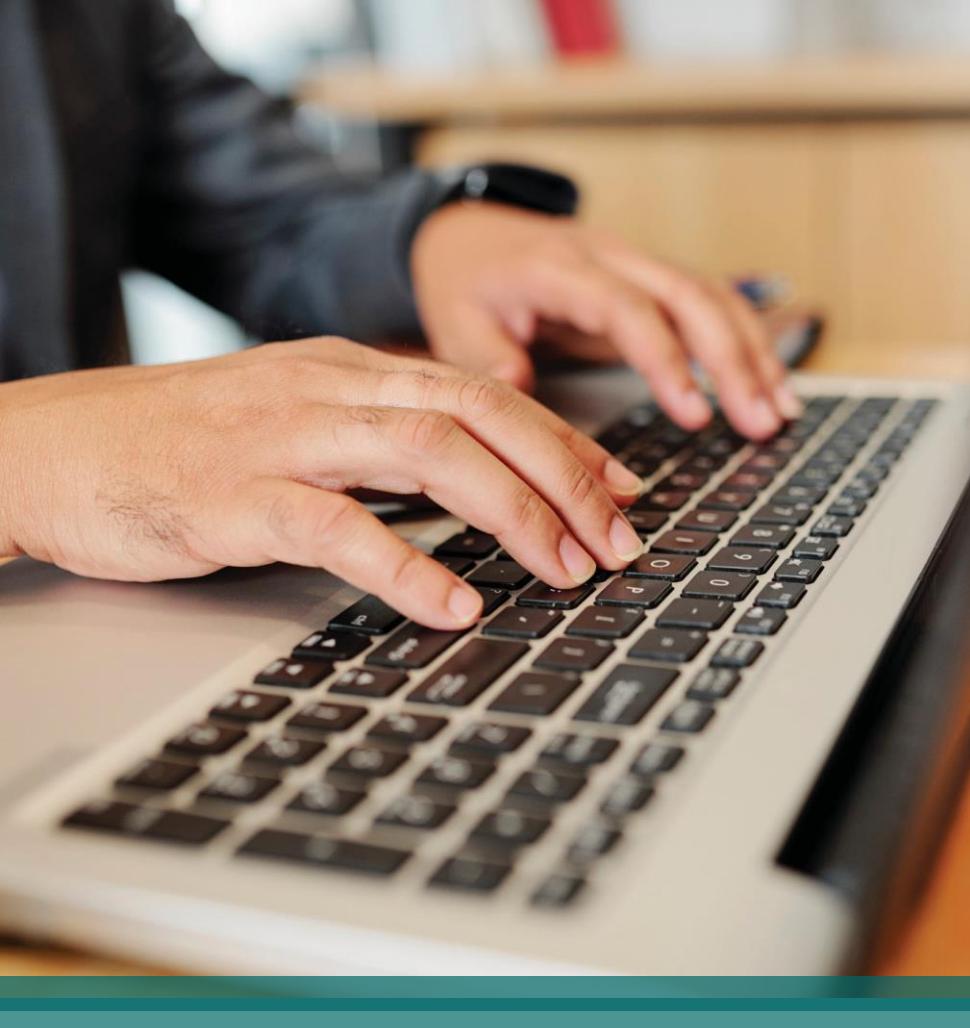


ENTORNO DE DESARROLLO

La principal ventaja del MSIL es que facilita la ejecución multiplataforma y la integración entre lenguajes, al ser independiente de la CPU y proporcionar un formato común para el código máquina, generado por todos los compiladores que generan código para .NET.

Sin embargo, dado que las CPUs no pueden ejecutar directamente MSIL, antes de ejecutarlo habrá que convertirlo al código nativo de la CPU sobre la que se va a ejecutar.





ENTORNO DE DESARROLLO

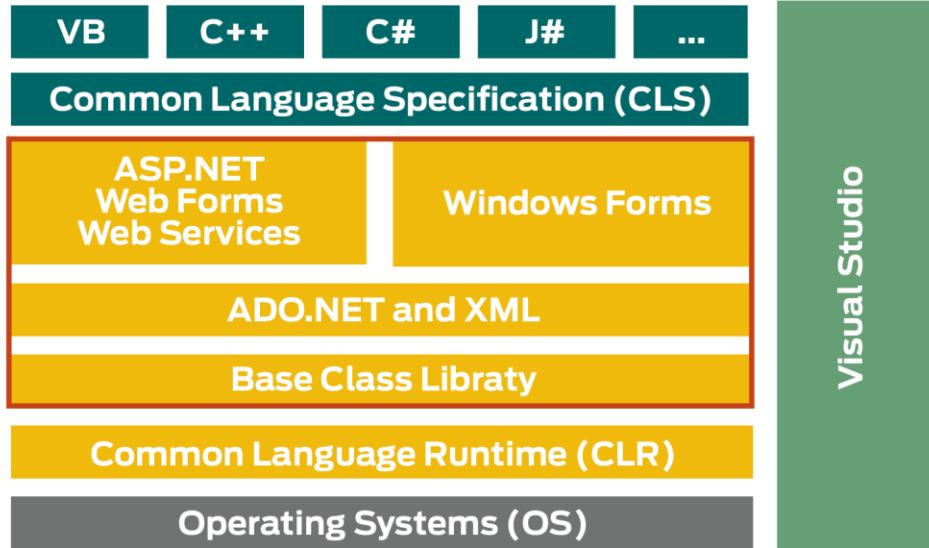
Librería de Clase Base (BCL)

.NET tiene un extenso grupo de estándares de bibliotecas de clases, llamadas **bibliotecas de clases base** (conjunto principal) o **bibliotecas de clases de marco** (conjunto completo). Estas bibliotecas se implementan para bastantes tipos, algoritmos y funcionalidades de utilidad en general y específicos de la aplicación. Tanto las bibliotecas comerciales, como las comunitarias, se fundamentan en las bibliotecas de clases de marco, lo cual otorga bibliotecas listas y simples de usar y para un extenso grupo de labores informáticas.

ENTORNO DE DESARROLLO

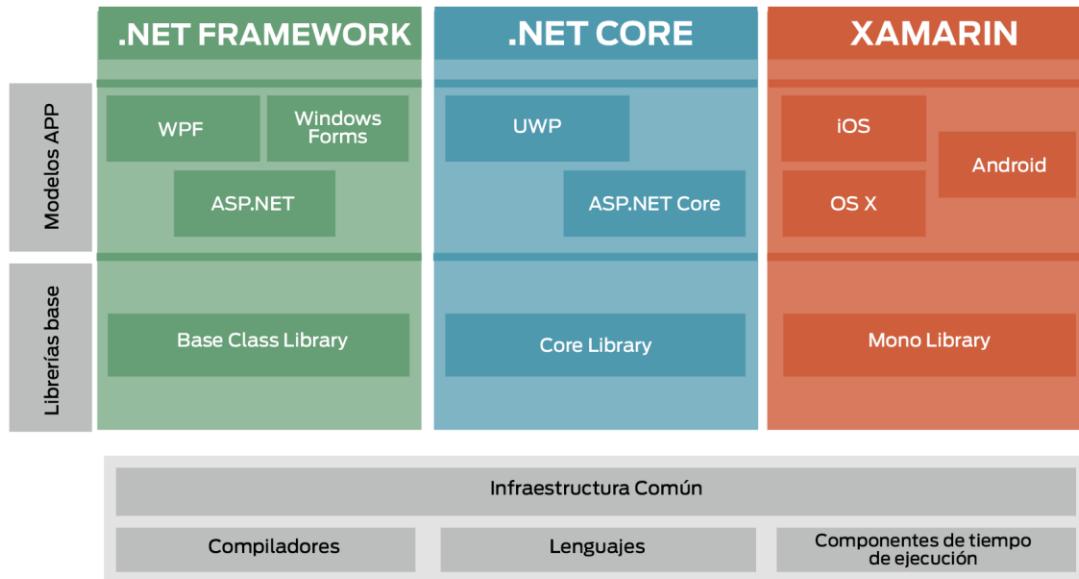
La **Librería de Clase Base (BCL)** es una librería incluida en el .NET Framework, formada por cientos de tipos de datos que permiten acceder a los servicios ofrecidos por el CLR y a las funcionalidades más frecuentemente usadas para escribir programas.

BCL otorga los tipos más primordiales y la funcionalidad de más utilidad, y es la base de cada una de las otras bibliotecas de clases de .NET. El BCL tiene como fin proveer implementaciones en general, sin sesgos para ninguna carga de trabajo.



ENTORNO DE DESARROLLO

Además, a partir de estas clases prefabricadas, el programador puede crear nuevas clases que, mediante herencia, extiendan su funcionalidad y se integren a la perfección con el resto de clases de la BCL. Por ejemplo, al implementar ciertas interfaces podemos crear nuevos tipos de colecciones que serán tratadas exactamente igual que cualquiera de las colecciones incluidas en la BCL.



Common Language Specification (CLS)

A veces es necesario un código escrito en un lenguaje para acceder a otro lenguaje, pero los lenguajes de programación que tienen como objetivo CLR son diferentes entre sí, por ejemplo, C# distingue entre mayúsculas y minúsculas, pero VB no.

Esto puede causar un problema cuando accedemos a código escrito en un idioma desde otro idioma. Así que .NET ha creado CLS; es decir, **especificación de lenguaje común**.

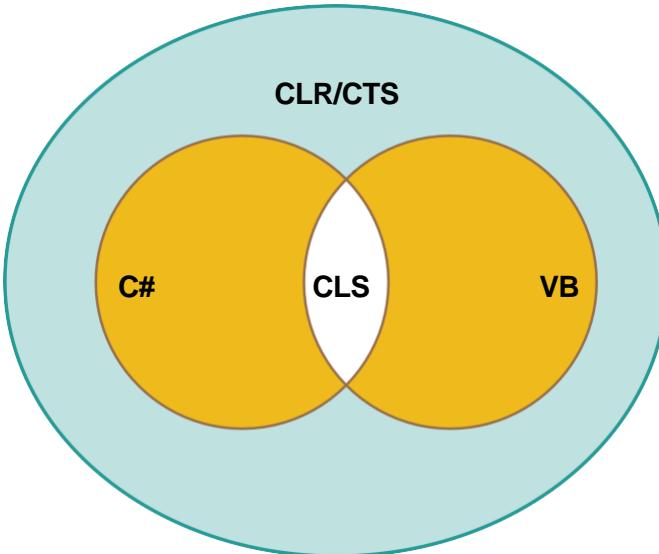
CLS define un conjunto mínimo de características que deben ser compatibles con todos los lenguajes destinados a **CLR**.

ENTORNO DE DESARROLLO

CLR/CTS admite muchas funciones, pero eso no significa que todos los idiomas a los que se dirige **CLR** admitan todas estas funciones.

Según el diagrama de Venn:

- CLR es un superconjunto de todos los idiomas que se dirigen a CLR.
- C# y VB son subconjuntos de CLR.
- CLS es un subconjunto mínimo imprescindible para todos los idiomas destinados a CLR.
- Además, algunas de las características son comunes en C# y VB, además de las características de CLS.





ENTORNO DE DESARROLLO

Atributo conforme

Antes de comenzar con cualquier programa en CLS, existe un atributo útil que debe incluirse al escribir el código conforme a CLS.

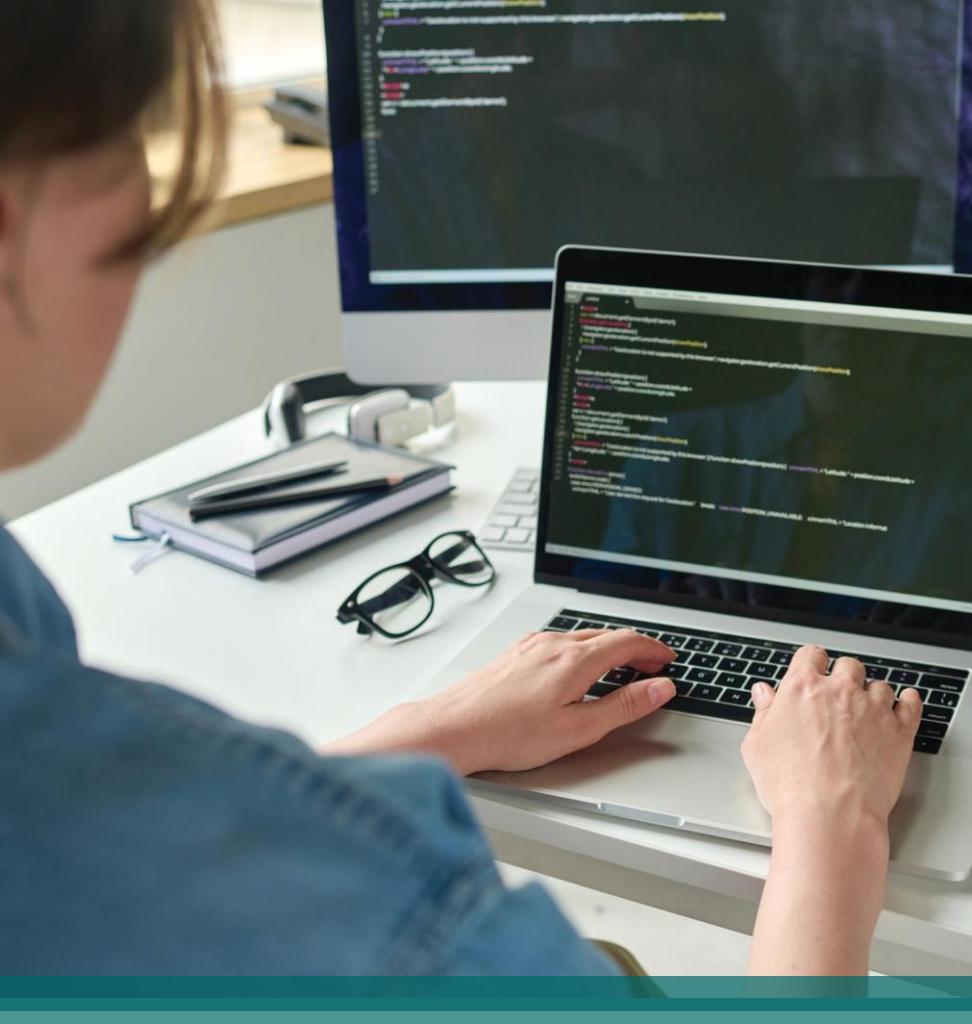
Si se establece el atributo `CLS Compliant` en verdadero, da una advertencia de tiempo de compilación cuando escribe cualquier miembro público o protegido que no sea `CLSClaimant`. Dará una advertencia solo para los miembros públicos y protegidos, no para los privados, ya que solo se puede acceder a los miembros privados dentro de esa clase.

ENTORNO DE DESARROLLO

Ejemplo: C# distingue entre mayúsculas y minúsculas, pero VB no; es por eso que los miembros públicos que difieren según el caso le darán una advertencia.

```
01.  using System;
02.
03. [assembly: CLSCompliant(true)]
04. namespace CLSEExamples
05.
06.
07.     public class CaseSensitiveExample
08.     {
09.         static void Main(string[] args)
10.        {
11.
12.        }
13.
14.        public void XYZ()
15.        {
16.            Console.WriteLine("Inside XYZ");
17.        }
18.        public void Xyz()
19.        {
20.            Console.WriteLine("Inside Xyz");
21.        }
22.    }
23. }
```

Al ejecutar el código anterior, recibirá una advertencia como: Identificador 'CaseSensitiveExample.Xyz()' que difiere solo en caso de que no cumpla con CLS.



ENTORNO DE DESARROLLO

Common Type System (CTS)

Estandariza los tipos de datos de todos los lenguajes de programación que utilizan .NET, en un tipo de datos común para una comunicación fácil y fluida entre estos lenguajes .NET. El sistema de tipo común realiza las siguientes funciones:

- Establece un marco que permite la integración entre lenguajes, la seguridad de tipos y la ejecución de código de alto rendimiento.
- Proporciona un modelo orientado a objetos que admite la implementación completa de muchos lenguajes de programación.

ENTORNO DE DESARROLLO

El Sistema de Tipo Común (CTS) admite dos categorías generales de tipos:

1. Tipos de valor

Los tipos de valor contienen directamente sus datos, y las instancias de los tipos de valor se asignan en la pila o se asignan en línea en una estructura. Los tipos de valor pueden ser tipos integrados, definidos por el usuario o de enumeración.

2. Tipos de referencia

Los tipos de referencia almacenan una referencia a la dirección de memoria del valor y se asignan en el montón. Los tipos de referencia pueden ser tipos autodescriptivos, tipos de punteros o tipos de interfaz.



ENTORNO DE DESARROLLO

CTS además define cada una de las otras características de los tipos, como modificadores de ingreso, cuáles son los miembros de tipo válidos, cómo funciona la herencia y la sobrecarga, etcétera.

Modificador	Código desde el que es accesible el miembro
public	Cualquier código
private	Código del mismo tipo de dato
Family	Código del mismo tipo de dato o de hijos de este
Assembly	Código del mismo ensamblado
family and assembly	Código del mismo tipo o de hijos de este ubicado en el mismo ensamblado
family and assembly	Código del mismo tipo o de hijos de este, o código ubicado en el mismo ensamblado

FORO UNIDAD 1

Con los conocimientos que aprendiste en esta unidad, responde la siguiente pregunta:

¿De qué manera podrías aprovechar el potencial que ofrece C# para crear sistemas completos e innovadores?

Presiona el botón para participar en el foro.





LECTURAS PARA REFORZAR LA UNIDAD

Capítulo 1:

- Thai, T. L., & Lam, H. (2002, febrero). *.NET Framework Essentials*. O'Reilly.
- Chen, X. (2004, 29 abril). *Developing Application Frameworks in .NET (Softcover reprint of the original 1st ed.)*. Apress.

Capítulo 2:

- Meijer, E., & Gough, J. (s. f.). *Technical Overview of the Common Language Runtime*.
- Hugon, J. (2018, marzo). *C# 7 Desarrolle aplicaciones Windows con Visual Studio 2017*. Ediciones Eni.

El lenguaje C# toma las mejores características de lenguajes preexistentes como Visual Basic, Java o C++ y las combina en uno solo. El hecho de ser relativamente reciente no implica que sea inmaduro, pues Microsoft ha escrito la mayor parte de la BCL usándolo, por lo que su compilador es el más depurado y optimizado de los incluidos en el .NET Framework SDK.

EL.NET Framework soluciona los problemas mediante el CLR, que es independiente del lenguaje, de la plataforma y del estándar, y con el uso del estándar del mercado (XML). La independencia del lenguaje de .NET permite a los desarrolladores generar una aplicación en cualquier lenguaje basado en .NET, y saber que la aplicación web servirá en cualquier cliente que soporte .NET.

CONCLUSIÓN



¡FELICIDADES!

Acabas de concluir la primera unidad de tu curso *Lenguajes de Programación III*. Te invitamos a finalizar este esfuerzo realizando el examen parcial correspondiente. Para ello, debes regresar a la pantalla principal y dar clic en *Presentar examen*.

2 UNIDAD

APLICACIONES



2.1

APLICACIONES
DE CONSOLA



TEMARIO



2.2

APLICACIONES
DE ESCRITORIO





INTRODUCCIÓN

En esta segunda unidad, el estudiante aprenderá a diferenciar entre una aplicación de consola y una de escritorio, además podrá realizar una aplicación sencilla para ejecutar en consola con un 'Hello World' y podrá diseñar un formulario básico para una aplicación de escritorio, además conocerá los elementos que lo conforman y cómo configurarlos de acuerdo con lo que se necesita.

COMPETENCIAS A DESARROLLAR



El alumno será capaz de programar una fácil aplicación para consola.



El alumno será capaz de realizar un formulario sencillo y sus partes para una aplicación en escritorio.



VIDEO



TE INVITAMOS A VER EL SIGUIENTE VIDEO:



APLICACIONES DE CONSOLA

¿Qué es una aplicación en consola?

En el contexto de C#, es una aplicación que toma entradas y muestra la salida en una consola de línea de comandos con acceso a tres flujos básicos: entrada estándar, salida estándar y error estándar.

La aplicación de consola está diseñada principalmente por las siguientes razones:

- Las aplicaciones de consola no tienen ninguna interfaz gráfica de usuario, tendrán interfaces basadas en caracteres.

- ▶ Proporciona una interfaz de usuario simple para aplicaciones que requieren poca o ninguna interacción del usuario, como ejemplos para aprender funciones del lenguaje C# y programas de utilidad de línea de comandos.
- ▶ Pruebas automatizadas, que pueden reutilizar recursos de implementación de automatización.



APLICACIONES DE CONSOLA

No tiene ninguna interfaz gráfica de usuario. Las aplicaciones de consola tendrán una interfaz basada en caracteres. Para trabajar con aplicaciones de consola en .NET, se debe usar una clase llamada **Console** que está disponible dentro del espacio de nombres **System**, que es el espacio de nombres raíz.

Console Application in C#:

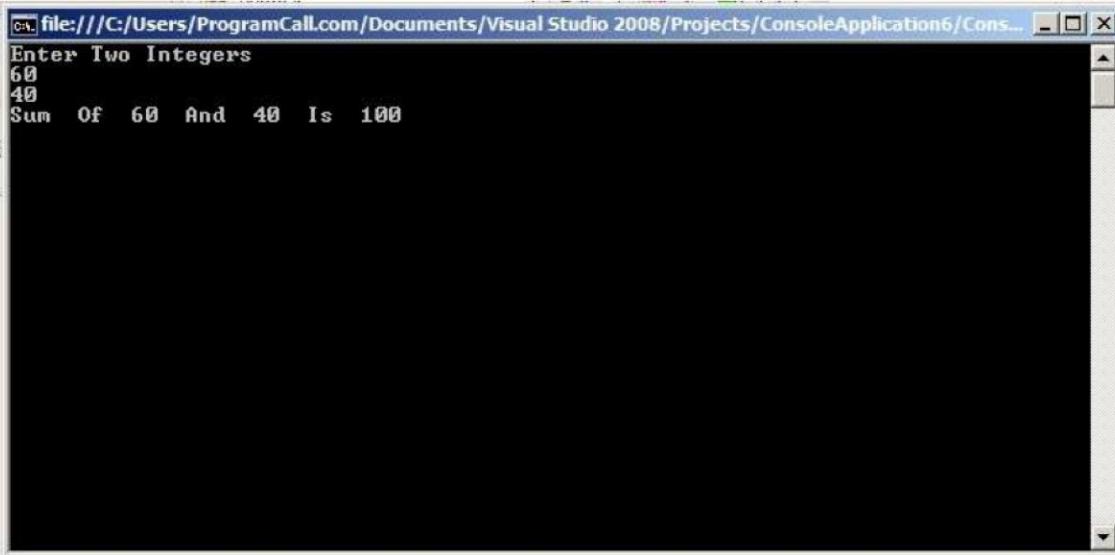
Para leer una línea de texto desde la ventana de la consola, utiliza el método **Console.ReadLine()**. Esto leerá un flujo de entrada desde la ventana de la consola y devolverá la cadena de entrada cuando el usuario presione la tecla Intro.

APLICACIONES DE CONSOLA

También hay dos métodos para escribir en la consola, que se utilizan ampliamente:

Console.WriteLine(): escribe el valor especificado en la ventana de la consola.

Console.ReadLine(): hace lo mismo, pero agrega un carácter de nueva línea al final de la salida.



The screenshot shows a Windows Command Prompt window with a black background and white text. The title bar reads "file:///C:/Users/ProgramCall.com/Documents/Visual Studio 2008/Projects/ConsoleApplication6/Cons...". The window contains the following text:
Enter Two Integers
60
40
Sum Of 60 And 40 Is 100

APLICACIONES DE CONSOLA

Ejemplo: El siguiente código C#.NET le permite al usuario ingresar una línea de texto y muestra esa cadena de texto s = **Console.ReadLine();**
Console.WriteLine(s);

```
using System;

namespace LlamarPrograma
{
    class Program
    {

        static void Main(string[] args)
        {

            int A, B, SUM;

            Console.WriteLine("Ingresa dos enteros");

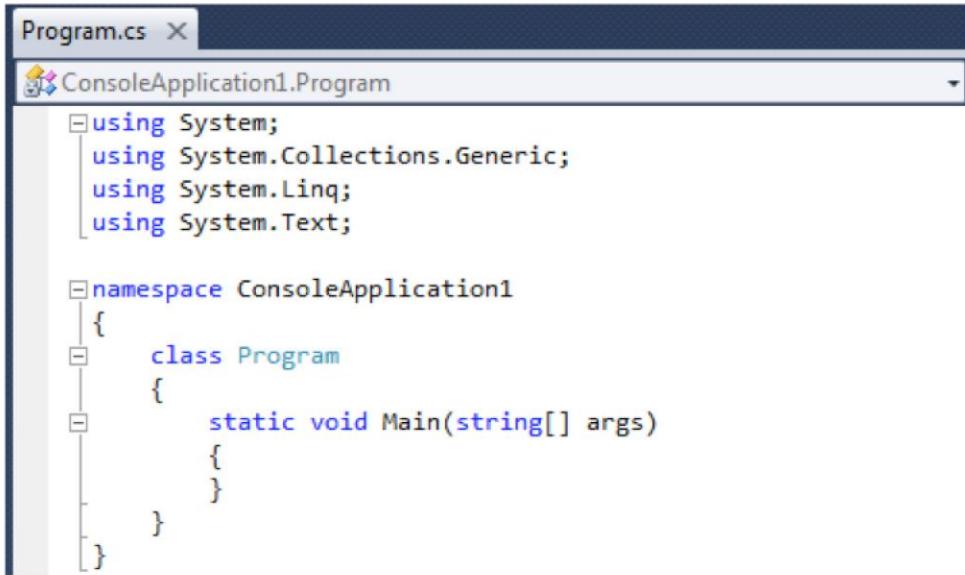
            A = int.Parse(Console.ReadLine());

            B = int.Parse(Console.ReadLine());
            SUM = A + B;
            Console.WriteLine("La suma de {0} Y {1} es {2}", A, B, SUM);
            Console.Read();// Para evitar que la consola desaparezca
        }
    }
}
```

¡Hello World! – Primer programa

Vamos a crear una aplicación basada en consola C#.

Cuando iniciamos un nuevo proyecto en Visual Studio para C# veremos una pantalla como la siguiente:



The screenshot shows a code editor window for a C# console application named "ConsoleApplication1". The file is "Program.cs". The code is as follows:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
        }
    }
}
```



APLICACIONES DE CONSOLA

Para el ejemplo del programa que vamos a ver, simplemente vamos a imprimir un “¡Hello World!” de mensaje. Vamos a utilizar el comando en el bloque de código principal (Static Void Main):

```
Console.WriteLine("Hello World!");
```

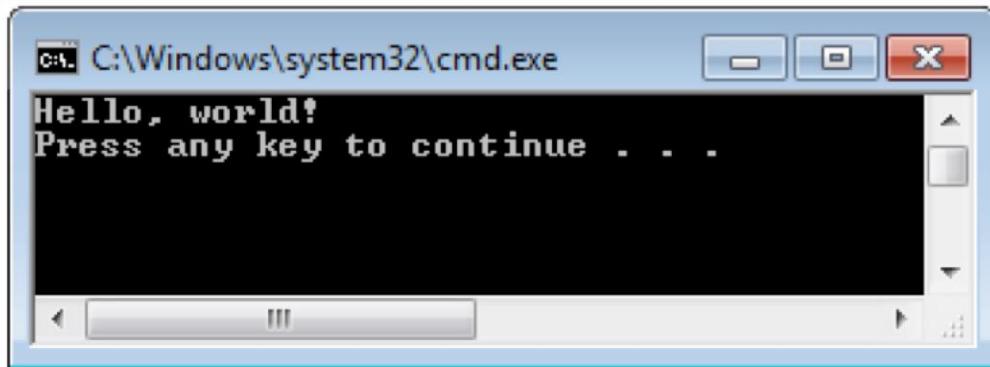
APLICACIONES DE CONSOLA

Y el código fuente quedaría así:

```
using System;
using System.Collections.Generic;
using System.Text;
namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello, world!");
        }
    }
}
```

APLICACIONES DE CONSOLA

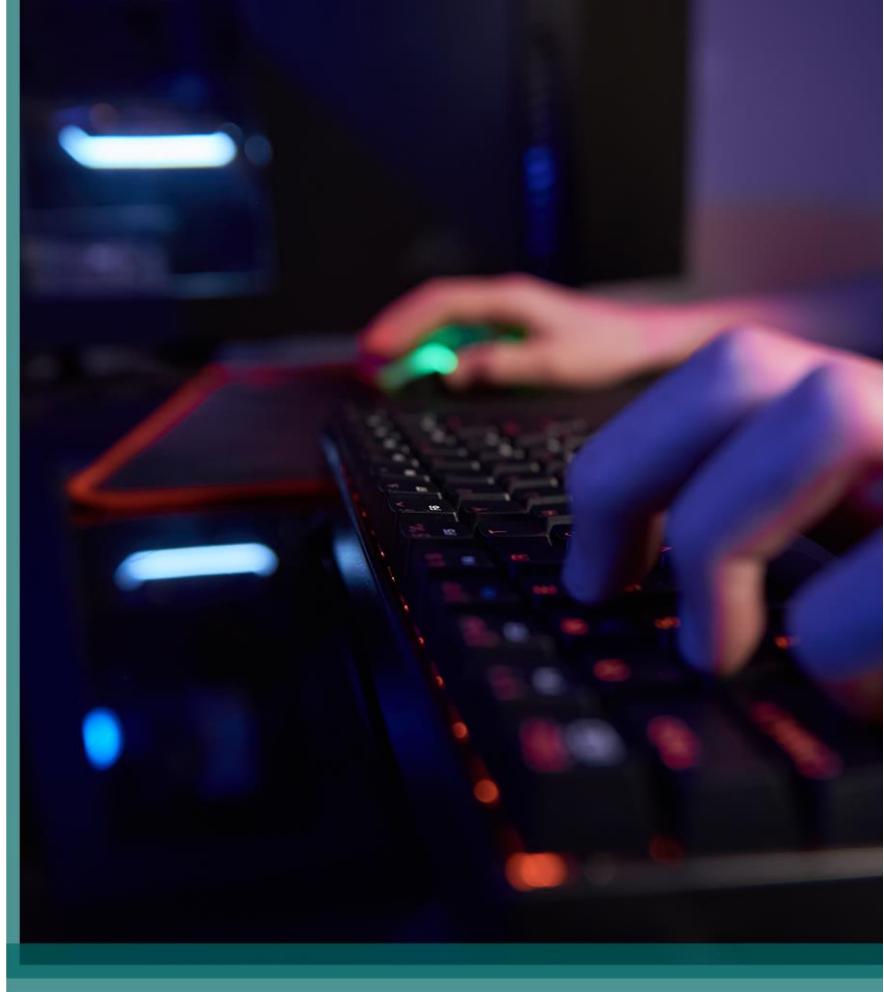
Después de ingresar el comando, el siguiente paso es ejecutar el programa. Puede ejecutar el programa usando Ctrl+F5. Luego, Visual Studio mantendrá abierta la ventana de la consola, hasta que presione una tecla. Obtendrá el aspecto de la pantalla como la siguiente imagen.



Aplicaciones de escritorio

En un escenario de la vida real, el equipo normalmente usa Visual Studio y C# para crear **Windows Forms** o aplicaciones basadas en la Web.

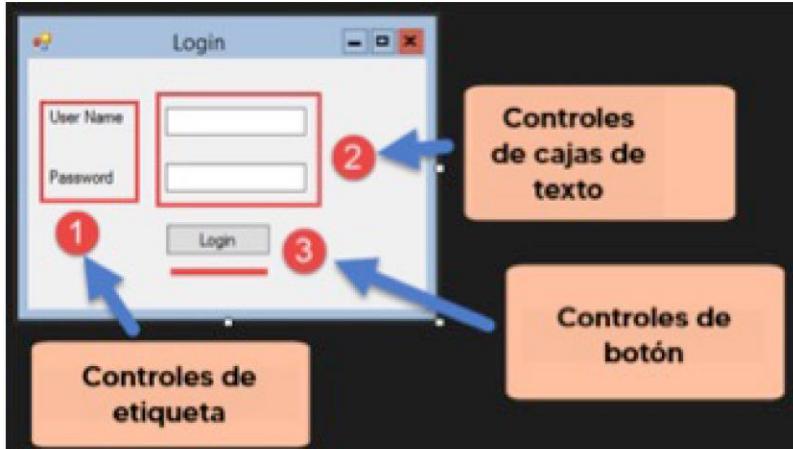
Una aplicación de formulario de Windows es una aplicación diseñada para ejecutarse en una computadora. No se ejecutará en el navegador web porque entonces se convierte en una aplicación web.

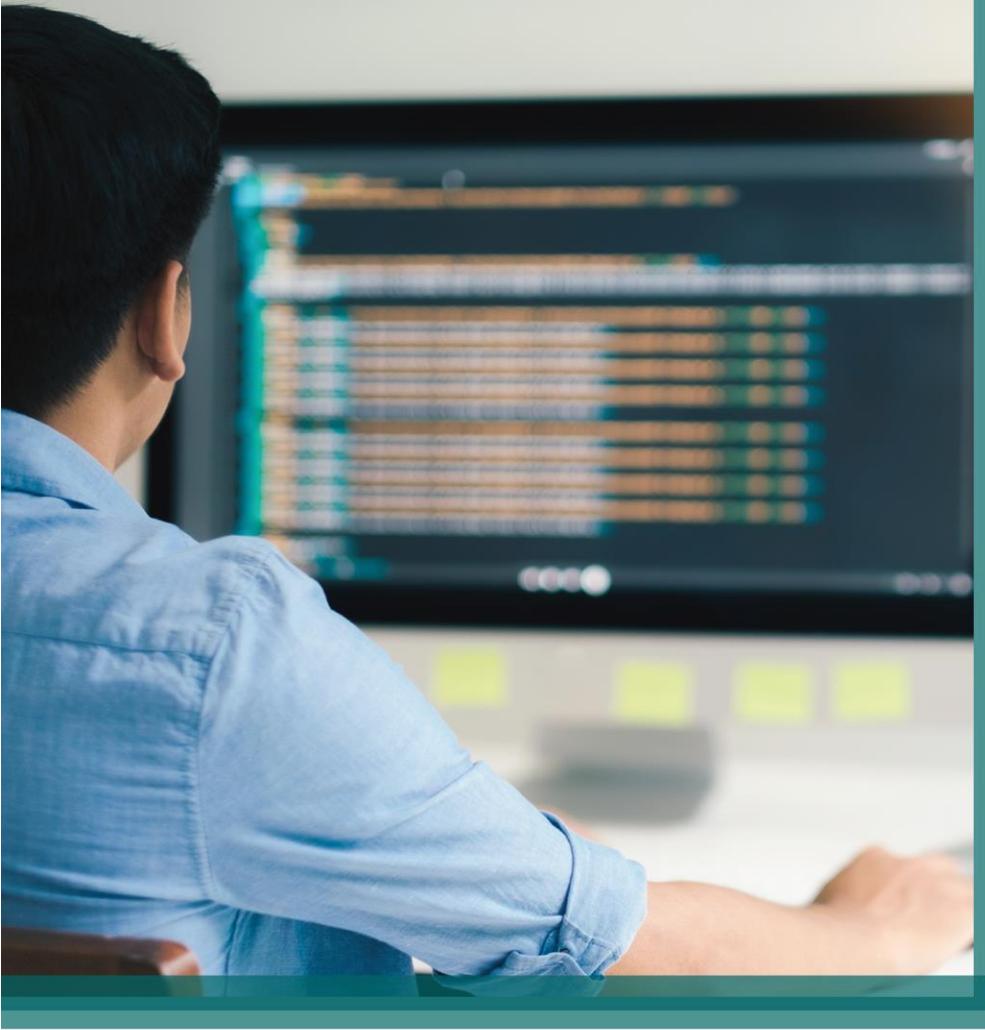


Conceptos básicos de Windows Forms

Una aplicación de Windows Forms es aquella que se ejecuta en la computadora de escritorio. Una aplicación de formularios de Windows normalmente tiene una colección de controles como etiquetas, cuadros de texto, cuadros de lista, etc.

Veamos un ejemplo de una aplicación de formulario de Windows simple C#. Muestra una pantalla de inicio de sesión simple, a la que puede acceder el usuario. El usuario ingresa las credenciales requeridas y luego da clic en el botón Iniciar sesión para continuar.





APLICACIONES DE ESCRITORIO

- 1.** Esta es una colección de controles de etiquetas que normalmente se usan para describir controles adyacentes. Entonces, en nuestro caso, tenemos 2 cuadros de texto y las etiquetas se usan para decirle al usuario que un cuadro de texto es para ingresar el nombre de usuario y el otro para la contraseña.
- 2.** Los 2 cuadros de texto se utilizan para contener el nombre de usuario y la contraseña que ingresará el usuario.

APLICACIONES DE ESCRITORIO

- 3.** Por último, tenemos el botón de control. El control de botón normalmente tendrá algún código adjunto para realizar un determinado conjunto de acciones. Entonces, por ejemplo, en el caso anterior, podríamos hacer que el botón realice una acción de validación del nombre de usuario y la contraseña que ingresa el usuario.

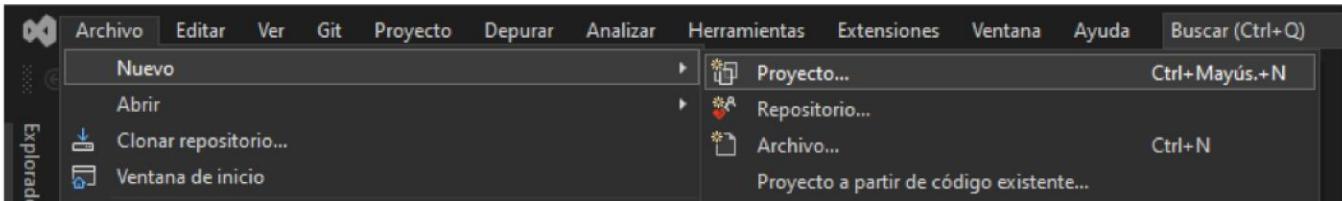


APLICACIONES DE ESCRITORIO

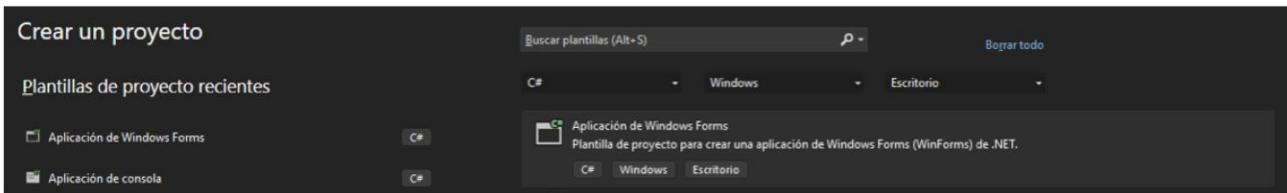
iHola Mundo!

Ahora veamos un ejemplo de cómo podemos implementar una aplicación simple de 'Hola Mundo' en Visual Studio.

Paso 1) Crear un nuevo proyecto en Visual Studio. Después de iniciar Visual Studio, debes elegir la opción de menú Nuevo->Proyecto.

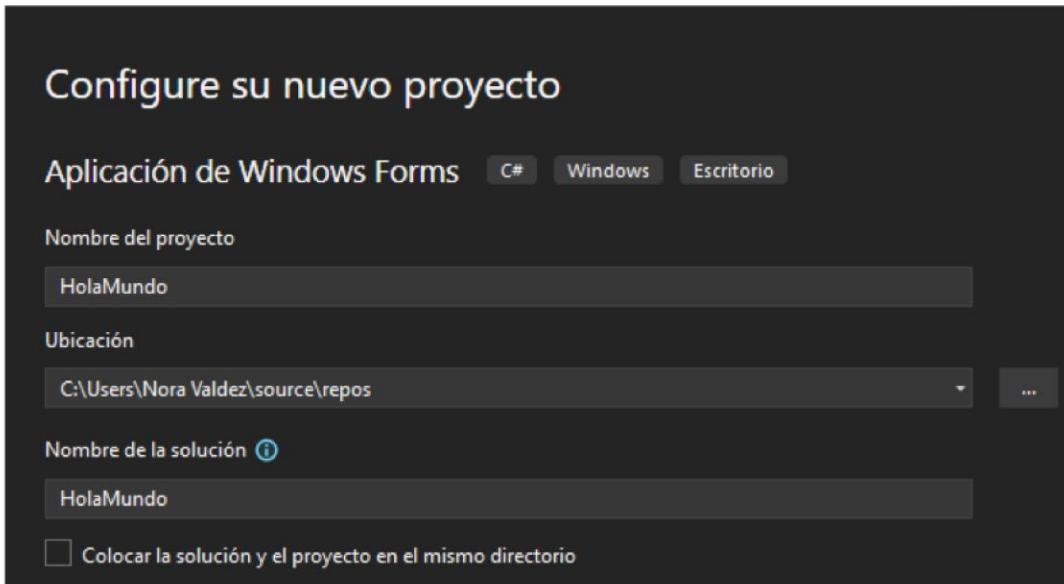


Paso 2) El siguiente paso es elegir el tipo de proyecto, que en este caso es una aplicación de Windows Forms.



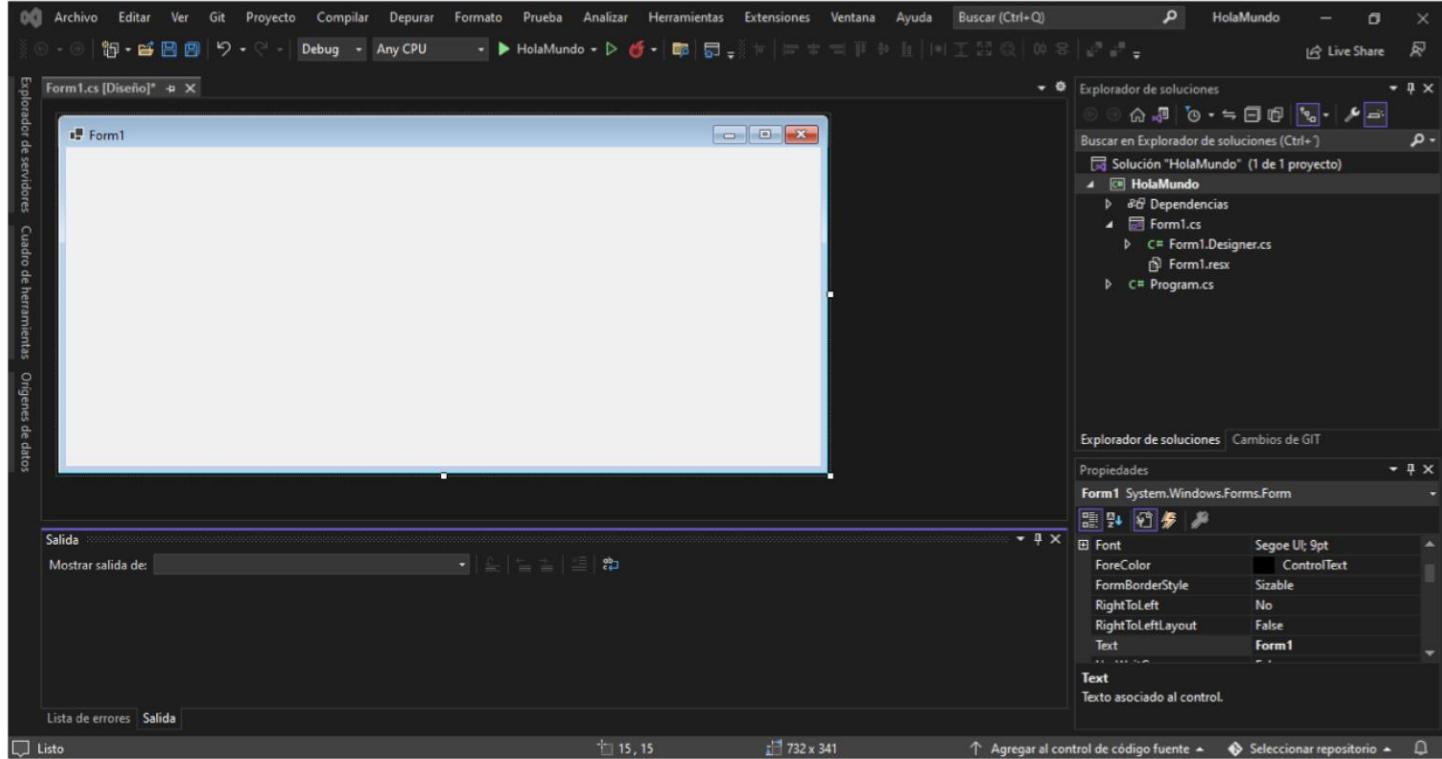
APLICACIONES DE ESCRITORIO

Paso 3) Aquí vamos a configurar nuestro proyecto poniendo el nombre, en este caso es HolaMundo, indicamos la ubicación donde vamos a guardar nuestro proyecto, y damos clic en siguiente.



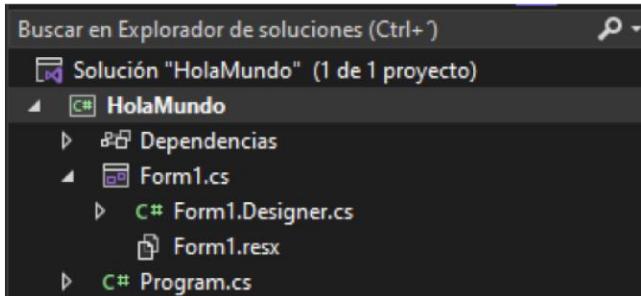
APLICACIONES DE ESCRITORIO

Y habremos creado el proyecto. Podrás ver la siguiente interfaz:



APLICACIONES DE ESCRITORIO

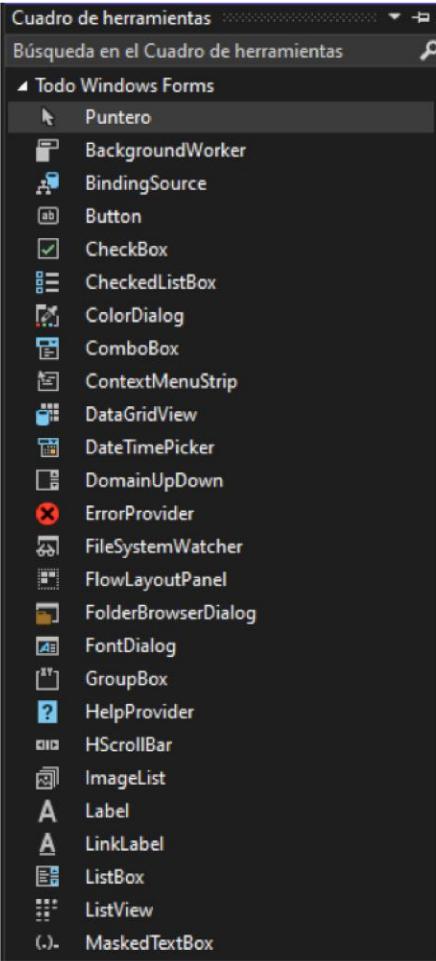
1. Diseñador de formularios en Visual Studio. En este Diseñador de formularios es donde comenzarás a crear tu aplicación de Windows Forms.
2. Explorador de soluciones tiene 2 archivos de proyecto: uno es el formulario llamado HolaMundo.cs., que contiene todo el código para la aplicación de Windows Forms. El programa principal llamado Program.cs es un archivo de código predeterminado que se crea cuando se establece una nueva aplicación en Visual Studio. Este código contendrá el código de inicio de la aplicación en su conjunto.



APLICACIONES DE ESCRITORIO

3. En el lado izquierdo de Visual Studio, se encuentra la caja o cuadro de herramientas. Este contiene todos los controles que se pueden agregar a Windows Forms. Los controles, como un cuadro de texto o una etiqueta, son algunos de los controles que se pueden agregar a Windows Forms solamente arrastrando el elemento al diseñador.

A continuación se muestra una imagen de cómo se ve el cuadro de herramientas:



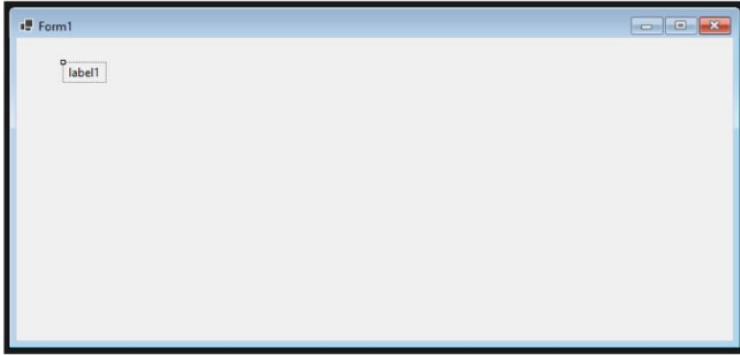
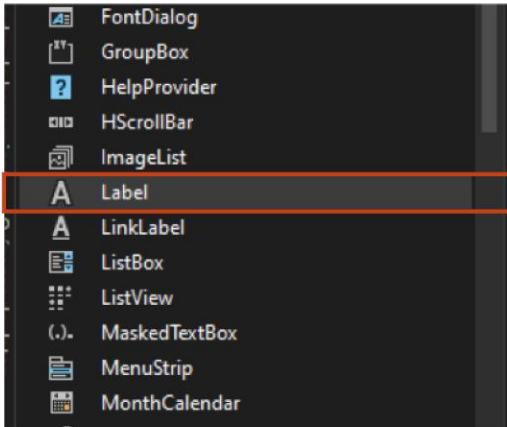
APLICACIONES DE ESCRITORIO

Interfaz

Ahora que ya vimos las partes del IDE de Visual Studio, realizaremos la pantalla del “Hola Mundo”.

Entonces agregaremos una etiqueta al formulario que mostrará “Hola Mundo”. En el cuadro de herramientas debes elegir el control Etiqueta o Label y simplemente arrastrarlo al Formulario.

Una vez que arrastremos la etiqueta al formulario, podrás ver la etiqueta añadida en el formulario, como se muestra a continuación.



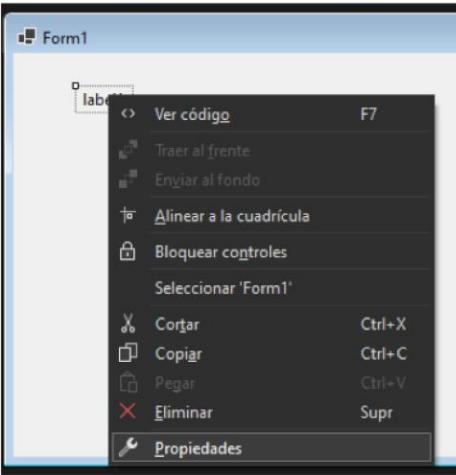
APLICACIONES DE ESCRITORIO

El siguiente paso es ir a las propiedades del control y cambiar el texto a 'Hola Mundo'.

Para ir a las propiedades de un control, debes hacer clic con el botón derecho en el control y elegir la opción de menú **Propiedades**.

El panel de propiedades también aparece en Visual Studio. Entonces, para el control de label, en el control de propiedades ve a la sección Texto e ingresa "Hola Mundo".

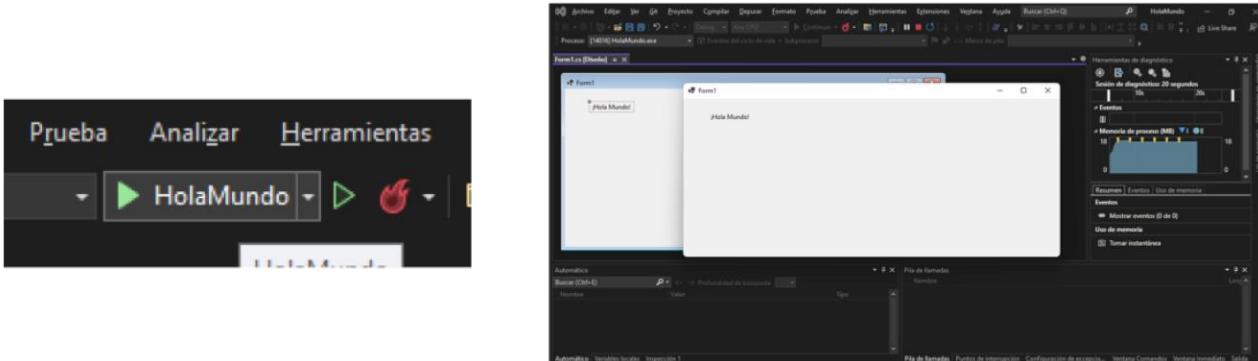
Cada Control tiene un conjunto de propiedades que describen el control.



APLICACIONES DE ESCRITORIO

Ahora que ya hemos modificado el label, vamos a correr el programa, y damos clic en el botón de “play” que tiene el nombre del proyecto en la parte superior:

En la salida, puedes ver que se muestra Windows Form. También puedes ver que se muestra ‘Hola Mundo’ en el formulario.



Adición de controles a un formulario

Ya vimos cómo agregar un control a un formulario, cuando agregamos el control de etiqueta en la sección anterior para mostrar “Hola Mundo”.

Ahora veamos los otros controles disponibles para los formularios de Windows y veamos algunas de sus propiedades comunes.

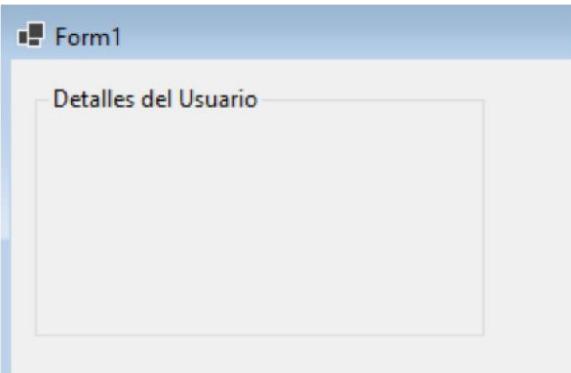
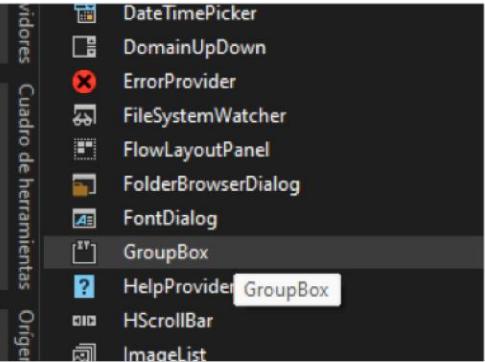
En nuestra aplicación de formulario de Windows en ejemplos de C#, crearemos un formulario que tendrá la siguiente funcionalidad.

- ▶ La alternativa para que el usuario ingrese el nombre y la dirección.
- ▶ Una opción para elegir la ciudad en la que reside el usuario.
- ▶ La posibilidad para que el usuario ingrese una opción para el género.

Cuadro de grupo

Un cuadro de grupo se utiliza para agrupar controles lógicos en una sección. Dentro del cuadro de texto, estarán los detalles de una persona, por lo que es importante separar estos detalles en una sección separada en el formulario. Para este propósito, puede tener un cuadro de grupo.

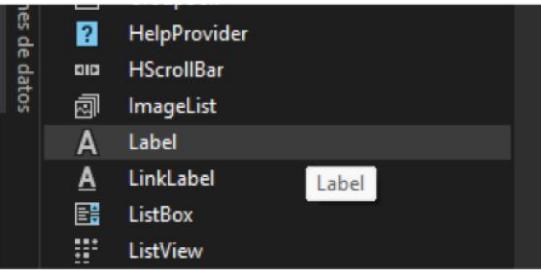
En la ventana de propiedades vamos a cambiar el texto por “Detalles del Usuario” y se verá de la siguiente manera:



Control de etiquetas

Se utiliza para mostrar un texto o un mensaje al usuario en el formulario. Normalmente se usa junto con otros controles. La etiqueta indica al usuario lo que debe llenar en el cuadro de texto.

Vamos a agregar dos etiquetas, una para indicar el nombre y otra para la dirección.



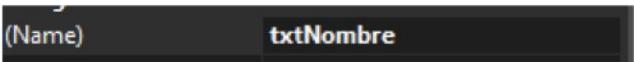
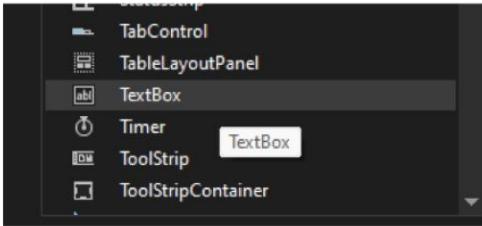
APLICACIONES DE ESCRITORIO

Caja de texto

Se utiliza un cuadro de texto para permitir que un usuario ingrese algún texto en la aplicación de Windows en C#.

Añadiremos 2 cuadros de texto al formulario, uno para el nombre y otro para la dirección a introducir por el usuario.

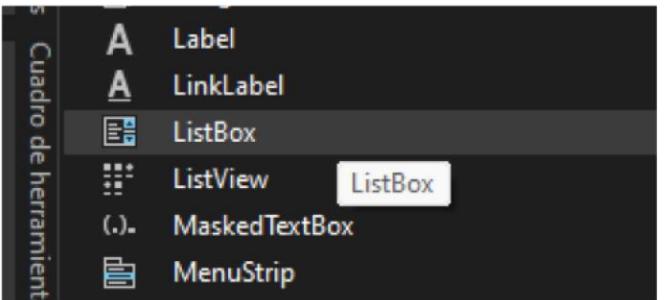
Cuando los hayamos agregado, vamos a la ventana de propiedades de cada uno y le cambiamos el nombre para que sean más identificables, por ejemplo txtNombre y txtDireccion.



Cuadro de lista

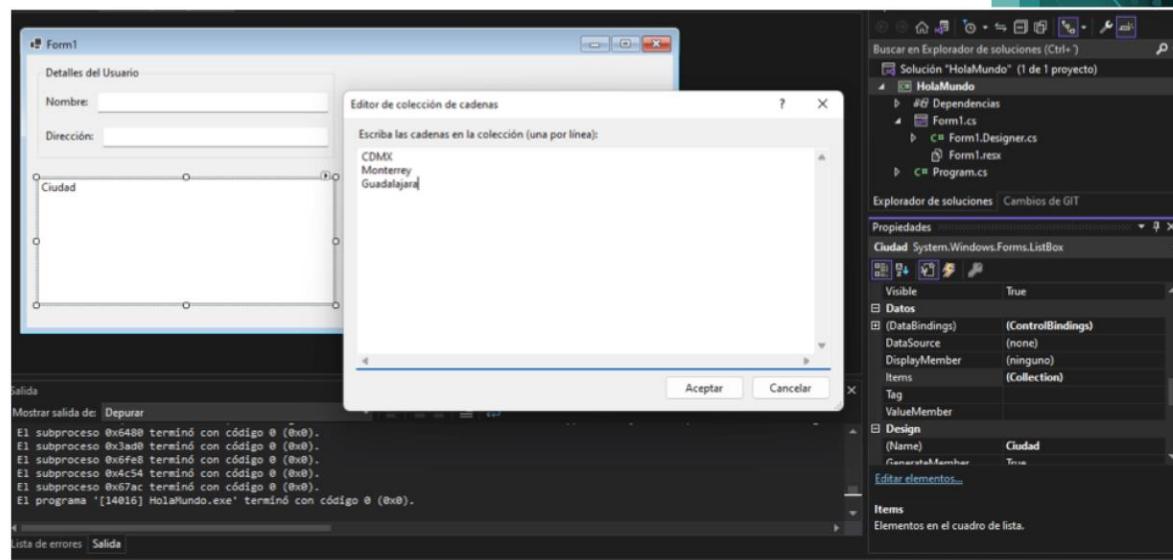
Un cuadro de lista se utiliza para mostrar una lista de elementos en el formulario de Windows. Agregaremos un cuadro de lista al formulario para almacenar algunas ubicaciones de ciudad.

Una vez que se haya agregado el cuadro de lista, ve a la ventana de propiedades haciendo clic en el control del cuadro de lista.



APLICACIONES DE ESCRITORIO

- 1.** Cambia la propiedad del control de cuadro Listbox. En nuestro caso, hemos cambiado esto a “Ciudad”.
- 2.** Haz clic en la propiedad Items. Podemos agregar diferentes elementos que aparezcan en el cuadro de lista. En nuestro caso, hemos seleccionado artículos de “colección”.
- 3.** En el Editor de colecciones de cadenas que aparece, ingresa los nombres de las ciudades. En nuestro caso, ingresamos CDMX, Monterrey y Guadalajara. Y clic en aceptar.



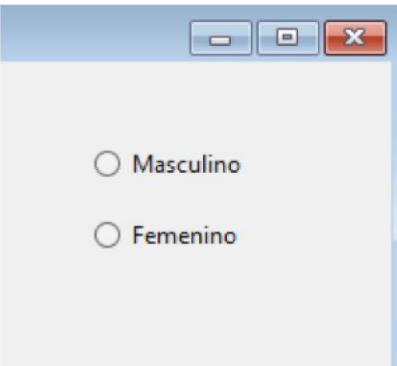
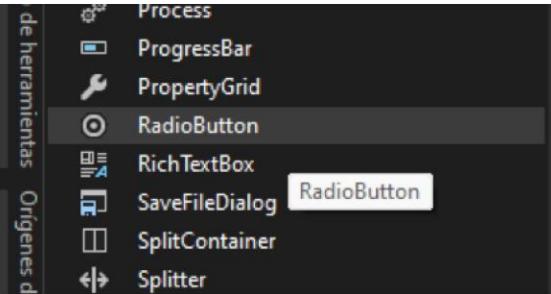
APLICACIONES DE ESCRITORIO

Botón de radio

Se utiliza para mostrar una lista de elementos que el usuario puede elegir. Agregaremos dos radios button para las opciones masculino/femenino.

Luego hacemos clic en propiedades:

1. Cambia la propiedad de texto de ambos controles de radio. En propiedades cambia el texto a masculino de un botón de opción, y a femenino el del otro.
2. Cambia la propiedad de nombre de ambos controles de radio. El nombre a 'hombre' de un botón de opción y a 'mujer' para el otro.

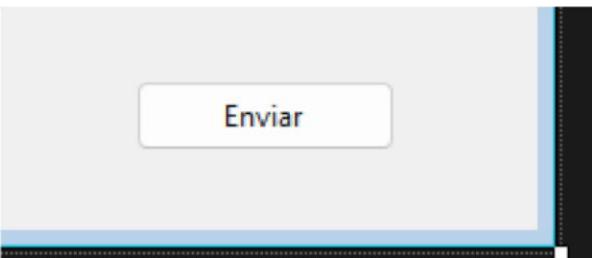
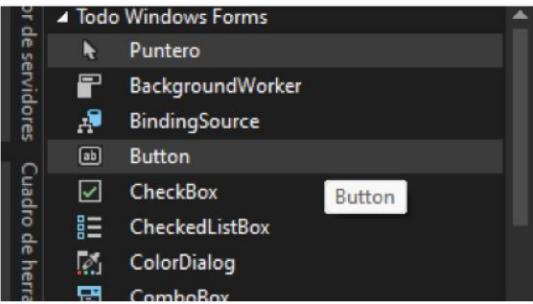


APLICACIONES DE ESCRITORIO

Botón

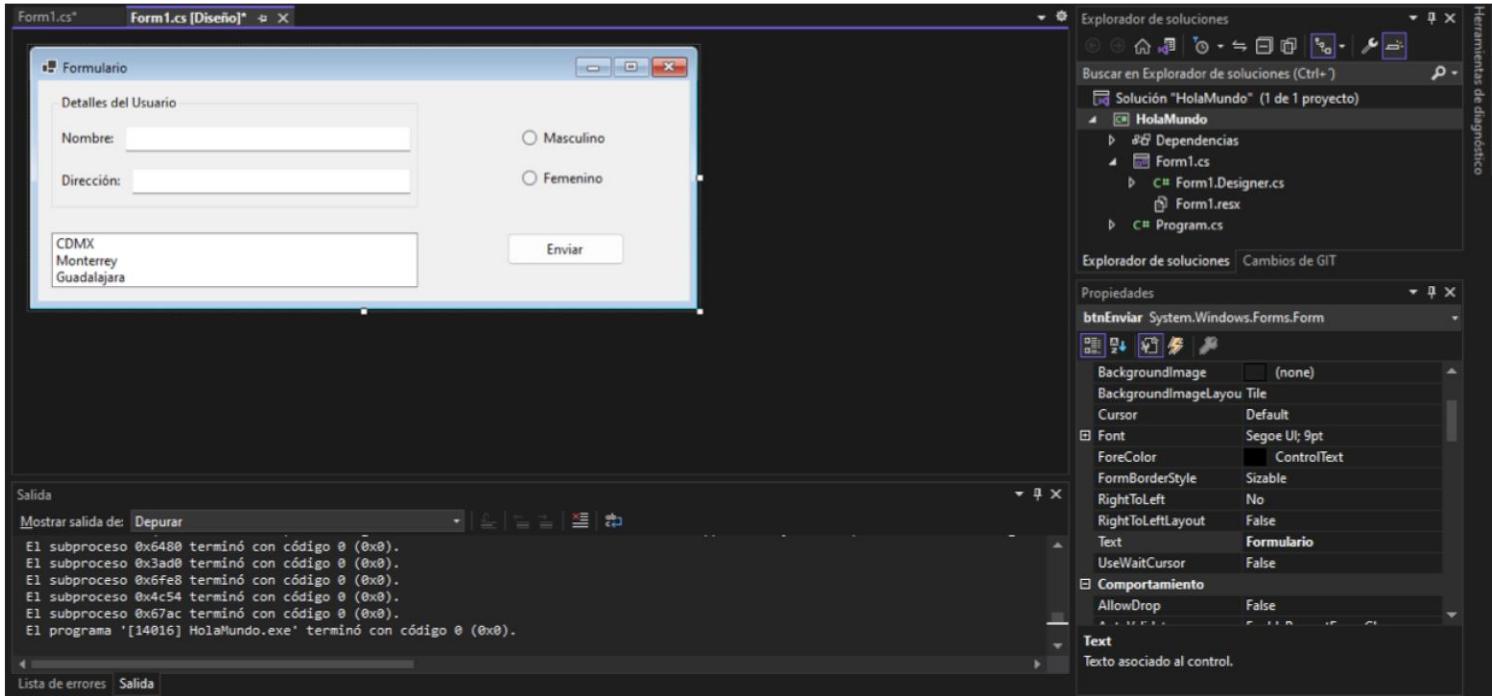
Se utiliza un botón para permitir que el usuario haga clic, y luego inicia el procesamiento del formulario. Agregaremos un botón simple llamado 'Enviar', que se utiliza para enviar toda la información en el formulario:

1. Cambia la propiedad de texto del control de botón. En propiedades cambia el texto a 'enviar'.
2. Cambia la propiedad de nombre del control a 'btnEnviar' en name.



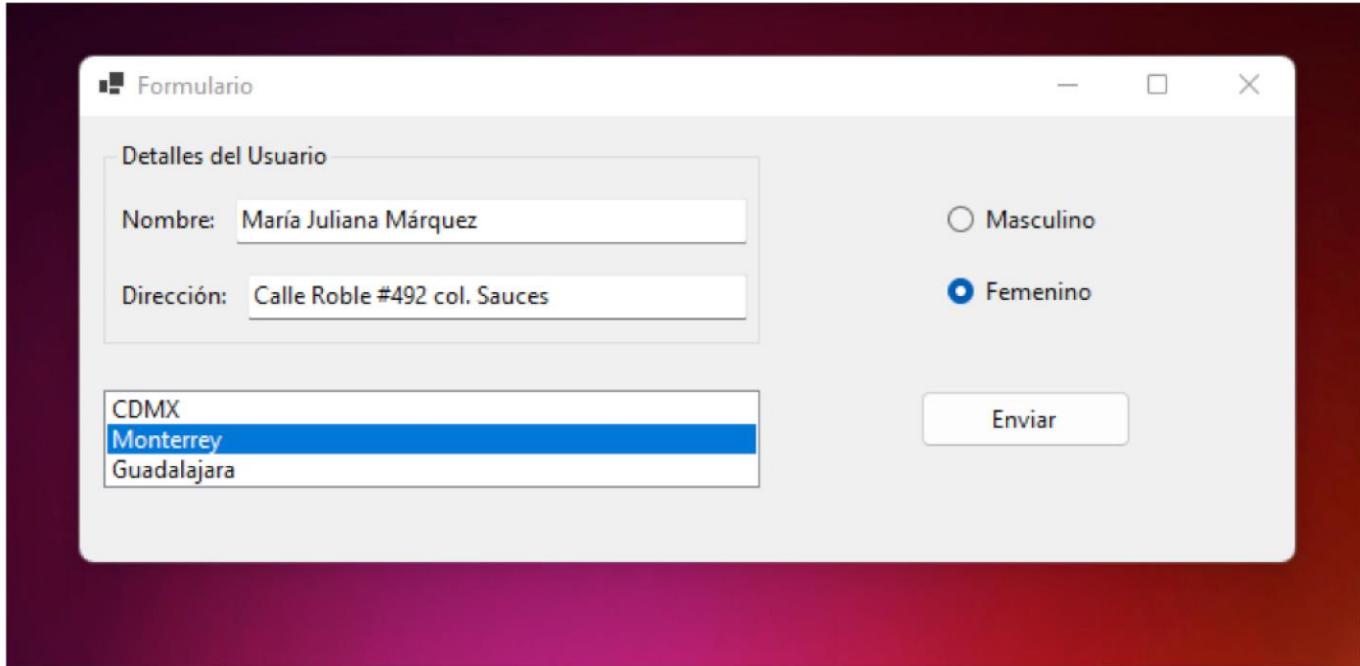
APLICACIONES DE ESCRITORIO

Algo así debería de verse el cascarón del formulario:



APLICACIONES DE ESCRITORIO

Luego lo ejecutamos, y con esto tenemos una aplicación de escritorio, aunque aun no puede realizar ninguna acción, ya que solo creamos lo que es la interfaz.



ACTIVIDAD INTEGRADORA 1

Te invitamos a realizar la siguiente actividad.

Presiona el botón para descargar la actividad.



Presiona el botón para entregar la actividad.





LECTURAS PARA REFORZAR LA UNIDAD

Capítulo 1:

- Almeida, F. (2018, julio). *Visual C#.NET: Console Applications and Windows Forms*. Academia.

Capítulo 2 y 3:

- Wolf, T. R. (2022, 22 enero). War and Peace - C# Programming 5 Vol.: Programming in C# Visual Studio - All about SQL, Record reads, writes, *DataGridViews* (War and Peace - C# Programming Visual Studio 2022).

Visual C# ofrece acceso a un extenso grupo de tipos incluidos basados en el sistema de tipos habituales, así como en la librería de clases de .NET Framework. Al igual que con todos los idiomas orientados a objetos, en C# aprovecha los tipos incluidos y las bibliotecas de clases para producir nuevos tipos para sus aplicaciones. Al exponer sus nuevos tipos para el consumo de otros, sus tipos tienen que consumar con la explicación de lenguaje común (CLS) para maximizar su reutilización por los desarrolladores que utilizan otros idiomas de programación.

CONCLUSIÓN





¡FELICIDADES!

Acabas de concluir la segunda unidad de tu curso *Lenguajes de Programación III*. Te invitamos a finalizar este esfuerzo realizando el examen parcial correspondiente. Para ello, debes regresar a la pantalla principal y dar clic en *Presentar examen*.

3

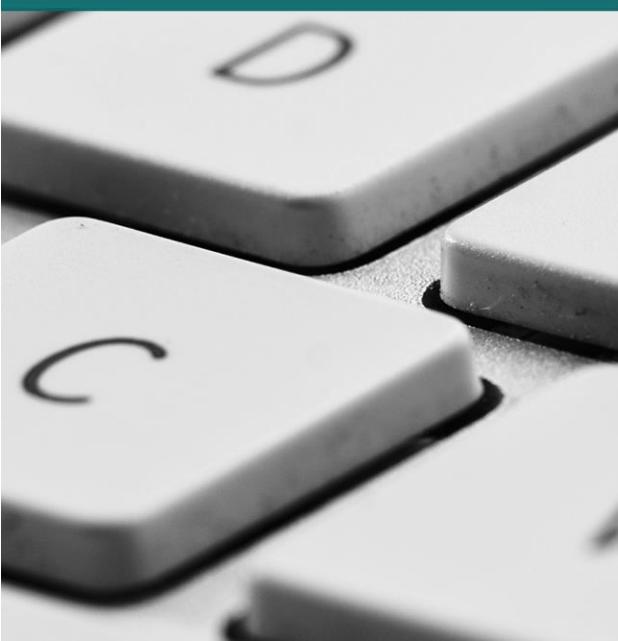
UNIDAD

DISEÑO DE INTERFAZ DE USUARIO



3.1

FORMULARIOS



TEMARIO



3.2

CONTROLES Y MENÚS





INTRODUCCIÓN

En esta tercera unidad, el estudiante aprenderá acerca de Windows Forms, el cual se utiliza para mostrar componentes de interfaz fáciles de usar para que puedan interactuar con la computadora en código C#. De hecho, la mayoría de las ventanas de interfaz en el sistema operativo Microsoft Windows se crean con Windows Forms. Esta es la razón por la que Windows Forms es parte importante de la programación .NET.

COMPETENCIAS A DESARROLLAR



El alumno será capaz de realizar una interfaz de un formulario utilizando los controles.



El alumno será capaz de reconocer los controles que maneja Windows Form para la creación de interfaces.



VIDEO



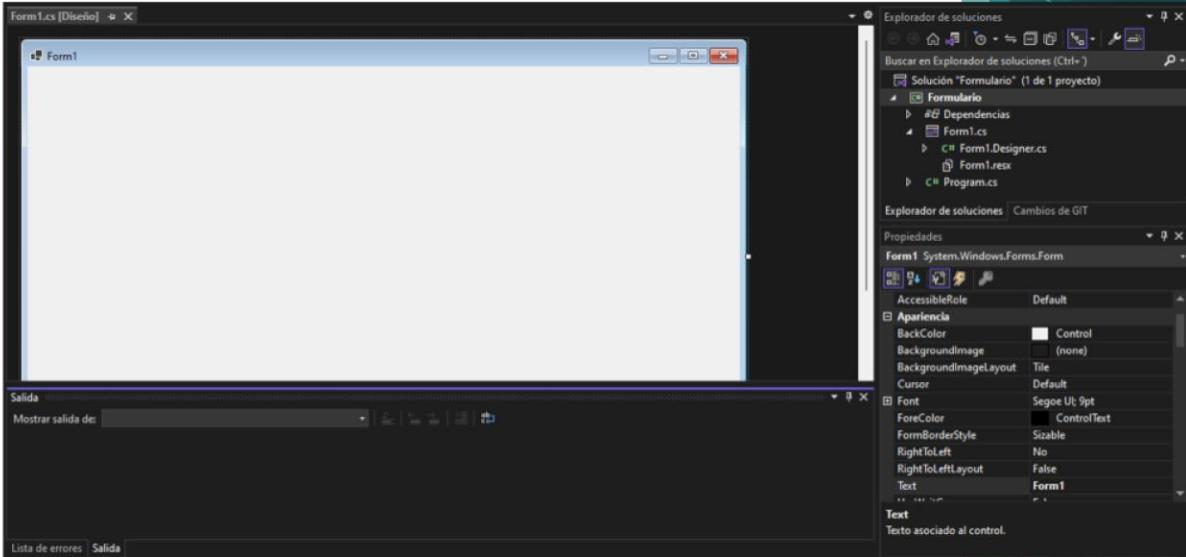
TE INVITAMOS A VER EL SIGUIENTE VIDEO:



FORMULARIOS

Windows Forms

La biblioteca de clases de interfaz gráfica de usuario (GUI) se incluye en .NET Framework, y su objetivo principal es proporcionar una interfaz más sencilla para desarrollar las aplicaciones para escritorio, tableta y PC. También se denomina WinForms, y las aplicaciones que se desarrollan utilizando WinForms se conocen como aplicaciones que se ejecutan en la computadora de escritorio.



FORMULARIOS



La configuración de la aplicación es otra característica de Windows Forms para crear, almacenar y mantener información del estado de tiempo de ejecución en un formulario XML, que se puede usar para recuperar la configuración preferida del usuario, como las posiciones de la barra de herramientas y las listas usadas más recientemente. Estos ajustes se pueden reutilizar en una aplicación futura.

FORMULARIOS

Algunas de las mejores prácticas para crear aplicaciones de Windows Forms incluyen:

- ▶ Se pueden ampliar, mediante herencia, para diseñar un marco de aplicación que pueda proporcionar un alto nivel de abstracción y reutilización de código.
- ▶ Los formularios deben ser compactos, con controles limitados a un tamaño que pueda ofrecer una funcionalidad mínima.
- ▶ Los formularios se pueden dividir en fragmentos empaquetados en ensamblajes, que se pueden actualizar automáticamente, y se pueden administrar.
- ▶ El diseño de la aplicación, para que no tenga estado, proporciona escalabilidad y flexibilidad con facilidad para la depuración y el mantenimiento.
- ▶ Las aplicaciones de Windows Forms deben diseñarse según el nivel de confianza requerido.

FORMULARIOS

En la unidad pasada aprendiste a utilizar las herramientas más básicas para la creación de un formulario sencillo. En esta unidad vamos a crear un formulario un poco más complejo utilizando más tipos de herramientas que, a su vez, le darán funcionalidad.

Crearemos un formulario médico de la enfermería escolar de una universidad donde se pedirán los siguientes datos:

Estos a su vez se dividirán en tres cuadros: datos del paciente, datos académicos y datos médicos.

Nombre del paciente

Edad

Número de teléfono

Género

Domicilio

Carrera

Semestre

Grupo

Padecimiento

Nombre del médico

Medicamento

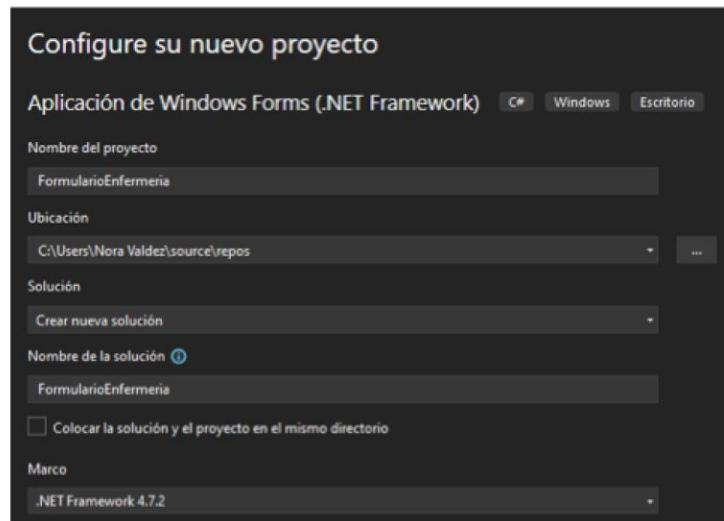
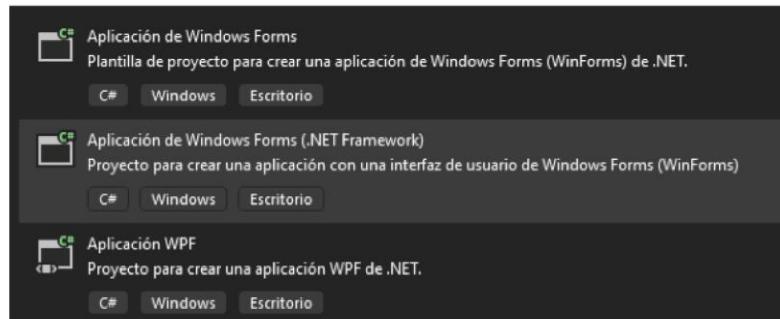
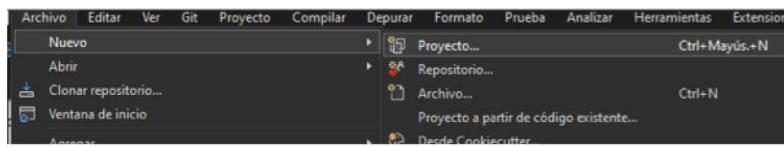
Fecha

FORMULARIOS

Vamos a comenzar creando un nuevo proyecto en Visual Studio llamado FormularioEnfermeria, en Archivo >> Nuevo >> Proyecto.

Después seleccionamos Aplicación de Windows Forms con la opción de .NET Framework.

Y lo nombramos como FormularioEnfermeria.

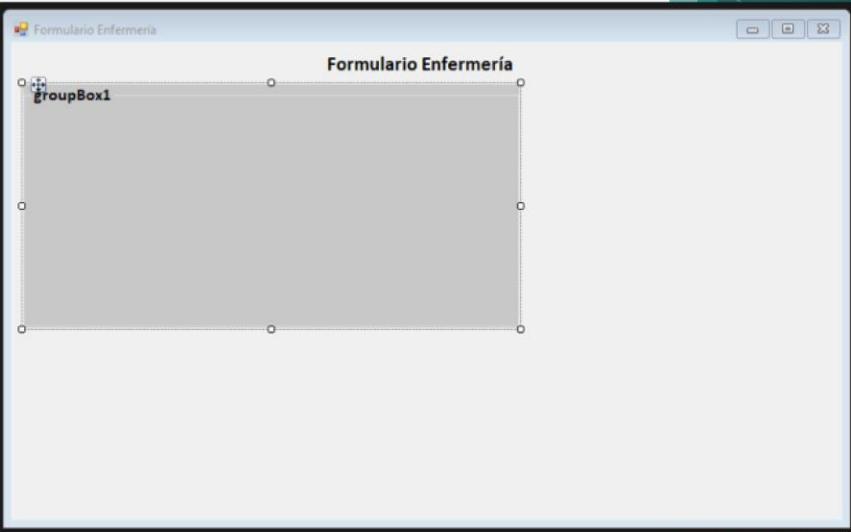


FORMULARIOS

Una vez creado el proyecto, vamos a darle forma a nuestro formulario. Agregamos un label a modo de título, que diga “Formulario Enfermería” y le damos estilo. Desde propiedades, en *Font* se puede cambiar la fuente y el grosor.

Después agregamos la primera sección del formulario, que pertenece a datos personales: nombre del paciente, edad, número de teléfono, género y domicilio.

Para delimitar la sección, agregaremos un groupBox y lo nombraremos datos personales, con un fondo color gris y la fuente en Calibri 12 en negritas; debería quedar algo así:



FORMULARIOS

Una vez añadido el primer groupbox, vamos a agregar el resto de datos con el label (etiqueta) para indicar qué ingresar en cada punto.

Veamos qué tipo de herramienta se utiliza para ingresar los datos en cada punto:

- Nombre de paciente: text box
- Edad: text box
- Número de teléfono: text box
- Género: 2 radio button, uno indicará masculino y el otro femenino
- Domicilio: text box

Debería quedar como la siguiente imagen:

Datos del Paciente

Nombre del paciente

Edad

Número de teléfono

Género

Masculino

Femenino

Domicilio

FORMULARIOS

El siguiente groupbox que vamos a realizar es el de datos académicos. En este agregamos otro groupbox y cambiamos el título por datos académicos.

Se puede cambiar el color de este en la sección de apariencia en Backcolor por un gris claro. Ahora, dentro de este cuadro, agregamos los labels para indicar los puntos que habrá en el cuadro. Las herramientas a utilizar para ingresar datos en cada punto (carrera, semestre y grupo) son Combo box.

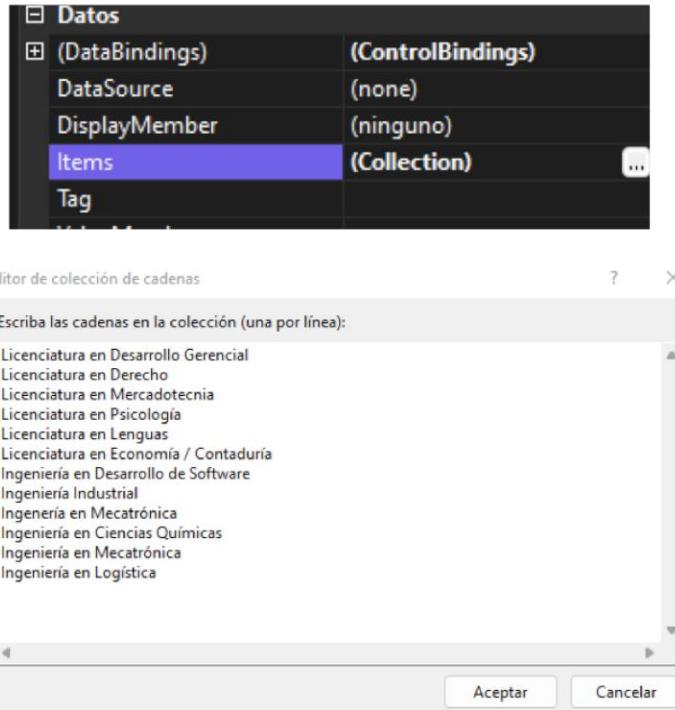
Datos Académicos

Carrera	Selecciona la carrera
Semestre	Selecciona el semestre
Grupo	Selecciona el grupo

FORMULARIOS

Para ingresar los datos a seleccionar, se agregan en propiedades >> datos >> ítems y debe estar en collection. En cada Combo box, se añaden datos de acuerdo con lo que pide la etiqueta; por ejemplo, en carrera agregamos diferentes licenciaturas e ingenierías para seleccionar.

Al hacer clic en los tres puntos que están al lado de *collection*, se abrirá el cuadro del editor de colección donde agregamos las opciones que mostrará el Combo box. Al dar clic en aceptar, estos se mostrarán al momento de correr el programa.



FORMULARIOS

Ahora agregamos el último groupbox del programa, donde están los datos médicos del paciente. Primero añadimos los labels para padecimiento y síntomas, nombre del médico, medicamento y fecha.

En los tres primeros puntos se ingresan los datos en un text box, pero ahora la diferencia es que síntomas y medicamentos aparecerá en modo párrafo. Para configurar un text box de este modo, seleccionamos la herramienta en cuestión y vemos una “flechita” en el extremo derecho; damos clic y seleccionamos la opción multilínea. Esto permite agregar párrafos.



FORMULARIOS

Ahora el tipo de herramienta a utilizar en fecha para ingresar los datos es DateTimePicker, ya que este proporciona un calendario en el formulario; normalmente tendrá seleccionado el presente día, pero se puede seleccionar cualquier día.

La sección de datos médicos debería quedar así:

Datos Médicos

Padecimiento y Síntomas

Nombre del Médico / Enfermero

Medicamento

Fecha

septiembre de 2022						
lun.	mar.	mié.	jue.	vie.	sáb.	dom.
29	30	31	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	1	2
3	4	5	6	7	8	9

Hoy: 14/9/2022

FORMULARIOS

Finalmente el formulario debe visualizarse como la siguiente imagen.

Formulario Enfermería

Datos del Paciente

Nombre del paciente

Edad

Número de teléfono

Género Masculino Femenino

Domicilio

Datos Académicos

Carrera

Semestre

Grupo

Datos Médicos

Padecimiento y Síntomas

Nombre del Médico / Enfermero

Medicamento

Fecha

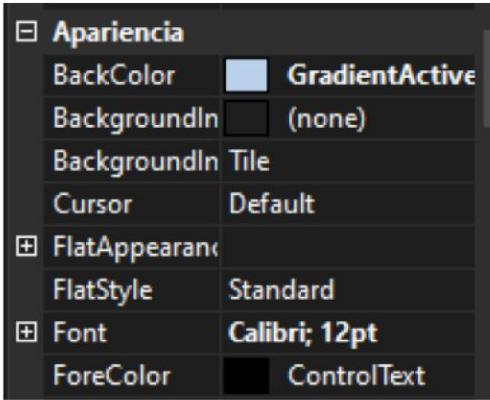
septiembre de 2022						
lun.	mar.	mié.	jue.	vie.	sáb.	dom.
29	30	31	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	1	2
3	4	5	6	7	8	9

Hoy: 14/9/2022

FORMULARIOS

Después de completar los espacios para ingresar datos en el formulario, es momento de agregar los botones de Enviar datos y Resetear datos. El primer botón enviará los datos ingresados a una nueva interfaz, y el segundo borrará todo lo ingresado en el formulario.

En herramientas selecciona Button para cada botón. Para darles formato, el fondo de ambos será de color azul y la fuente será Calibri, tamaño 12. Recuerda que para realizar estos cambios debes entrar al panel de propiedades >> apariencia; y deberían verse así:



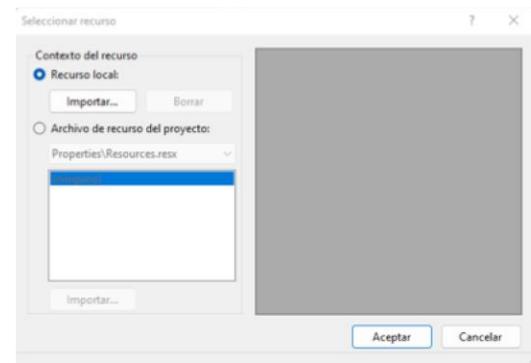
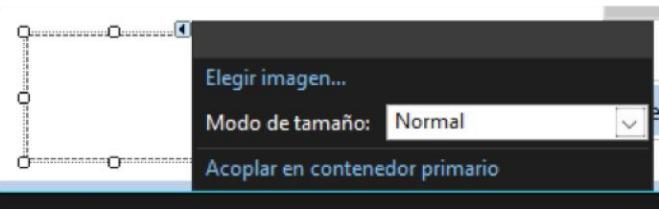
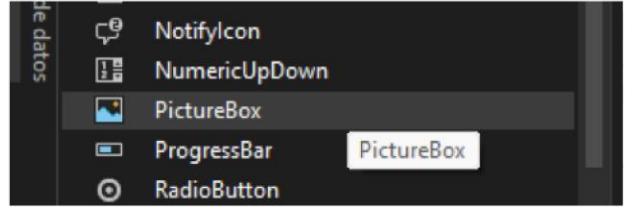
Enviar datos

Resetear datos

FORMULARIOS

Como plus, agrega una imagen alusiva al formulario. En este caso, integraremos un ícono de una enfermera. Para agregarlo, sigue los siguientes pasos:

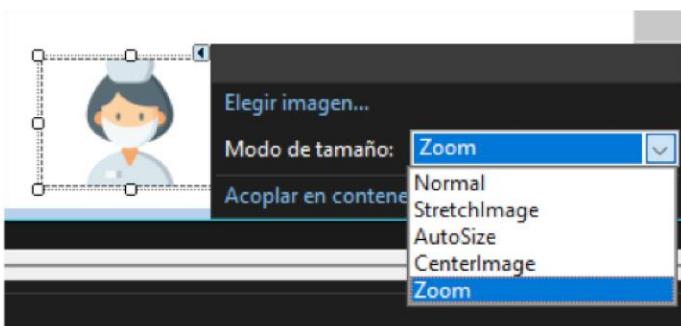
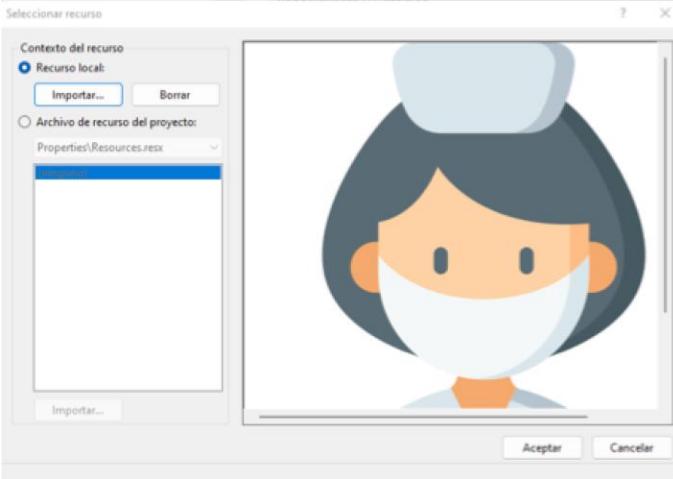
1. En las herramientas, selecciona la opción de PictureBox.
2. Aparecerá una flechita en el extremo derecho, dale clic y aparecerá la opción de seleccionar imagen.
3. Se abrirá un cuadro llamado Explorador de recursos, ahí selecciona la primera opción de contexto de recurso: Recurso local, y luego da clic en importar.



FORMULARIOS

4. Se abrirá el explorador de archivos, selecciona la imagen deseada.
5. Aparecerá la imagen en el cuadro de recursos, y dale aceptar.
6. Al principio la imagen no se visualizará bien debido a su tamaño; para adaptarla, da clic en modo de tamaño y selecciona la opción de zoom; con esto la imagen se autoajustará de acuerdo con el tamaño dado al Picture box.

Listo, agregamos una imagen al formulario, y la acomodaremos a un lado de los botones.



FORMULARIOS

Ahora la interfaz del formulario completo debe verse de la siguiente manera.

Enfermería

Formulario Enfermería

Datos del Paciente

Nombre del paciente

Edad

Número de teléfono

Género Masculino Femenino

Domicilio

Datos Académicos

Carrera

Semestre

Grupo

Datos Médicos

Padecimiento y Síntomas

Nombre del Médico / Enfermero

Medicamento

Fecha
lun. mar. mié. jue. vie. sáb. dom.
29 30 31 1 2 3 4
5 6 7 8 9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29 30 1 2
3 4 5 6 7 8 9
Hoy: 14/9/2022



Enviar datos

Resetear datos

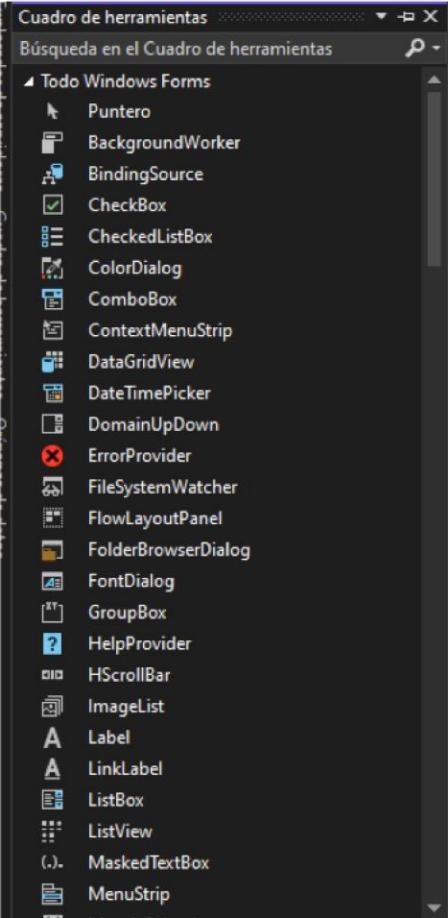
CONTROLES Y MENÚS

Controles

Los controles conforman la mayor parte de la interfaz de cliente en Visual Studio.

Los controles se utilizan en la interfaz de Visual Studio; estos tienen que continuar las directrices de relación del escritorio de Windows.

Este asunto es específico de Visual Studio y trata situaciones especiales o detalles que incrementan aquellas directrices de Windows.

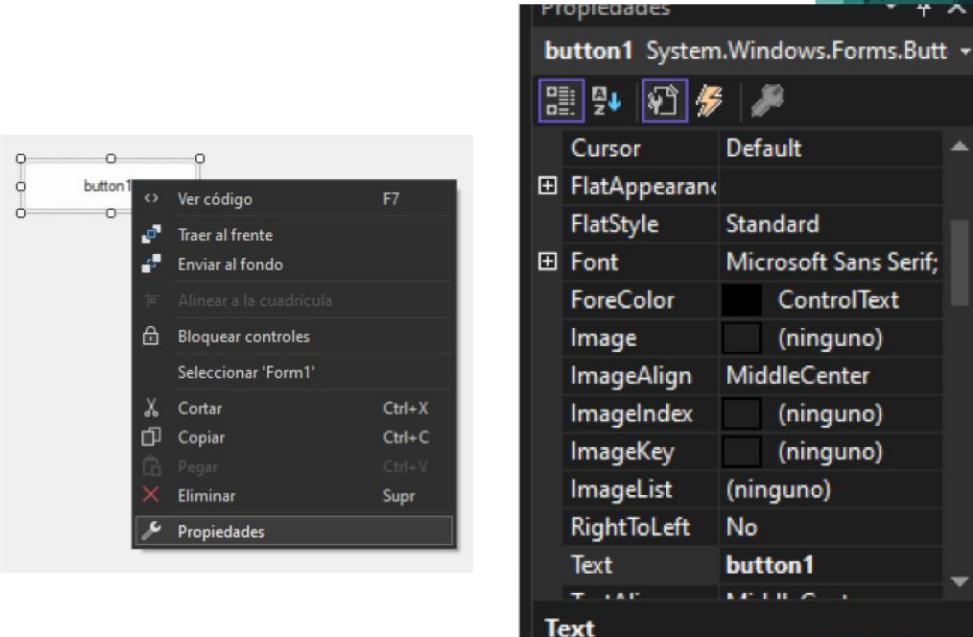


CONTROLES Y MENÚS

Controles de botón

Button: Componente que permite implementar un evento pulsándolo. Después de colocar un control de botón en un formulario, el siguiente paso es establecer las propiedades del botón.

Para esto, solo haz clic en un botón y en la parte derecha inferior del programa aparecerán las propiedades de este, donde se cambia el color, tipo de letra, el contenido del texto, etc., en general las propiedades visuales se cambian en la sección de apariencia.



CONTROLES Y MENÚS

Controles de botón

Check Box y Radio Button:

Los controles CheckBox y RadioButton tienen una función similar: permiten al usuario elegir de una lista de opciones.

Los controles CheckBox permiten al usuario elegir una combinación de opciones. Por el contrario, los controles de RadioButton permiten al usuario elegir entre opciones mutuamente excluyentes.

Hobbies (Check Box)

- Cantar
- Dibujar
- Ver series
- Dormir
- Cocinar
- Programar

¿Te gustaría entrar a un curso? (Radio Button)

- Sí
- No

CONTROLES Y MENÚS

Controles PictureBox y TextBox

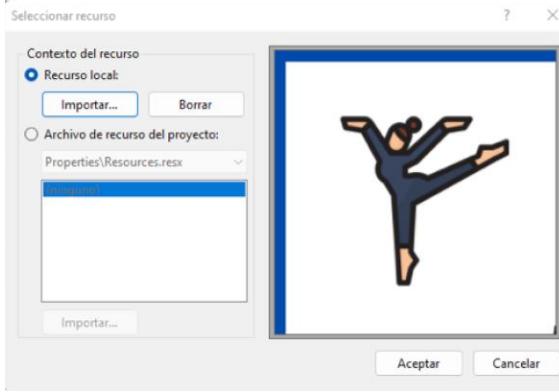
PictureBox se usa para mostrar gráficos en formato de mapa de bits, GIF, JPEG o ícono. La propiedad de PictureBox permite establecer una imagen tanto en tiempo de diseño, como en tiempo de ejecución.

Propiedades:

AllowDrop: cuando el PictureBox acepta datos que un usuario arrastra sobre él.

ImageLocation: establece la dirección URL de la imagen que se muestra en el control.

InitialImage: establece la imagen que se muestra en el control, cuando se carga la imagen principal.



CONTROLES Y MENÚS

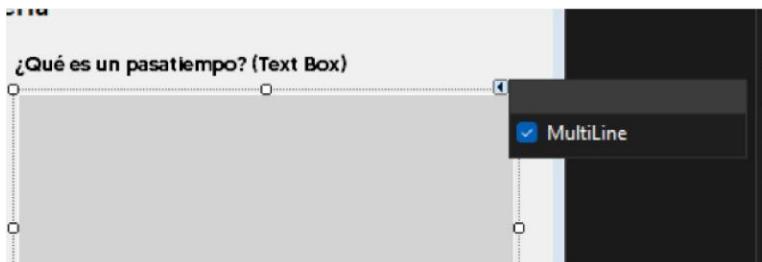
TextBoxt

Este control se usa para texto editable, y también se puede hacer de solo lectura. Puede mostrar varias líneas, ajustar el texto al tamaño del control y agregar formato básico. El texto que muestra el control está contenido en la propiedad Text. De manera predeterminada, puede ingresar hasta 2048 caracteres en un cuadro de texto. Si establece la propiedad Multilínea, se puede ingresar hasta 32 KB de texto. La propiedad Texto se puede establecer en tiempo de diseño con la Ventana de propiedades.

¿Qué es un pasatiempo? (Text Box)

Se conoce como pasatiempo a la actividad que una persona lleva a cabo para mantenerse entretenida durante un tiempo. De este modo, el pasatiempo sirve para combatir el aburrimiento y para tener la mente concentrada en algo placentero.

Por ejemplo: "Jugar al tenis es mi pasatiempo preferido", "Cuando era pequeño, comencé a colecciónar estampillas como pasatiempo y aún hoy mantengo ese hobby", "No existe pasatiempo más enriquecedor que la lectura de un buen libro".



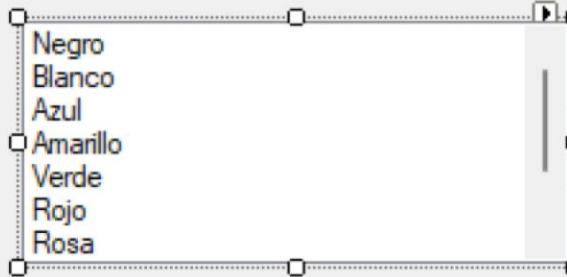
CONTROLES Y MENÚS

Controles de lista: ListBox, ComboBox y CheckListBox

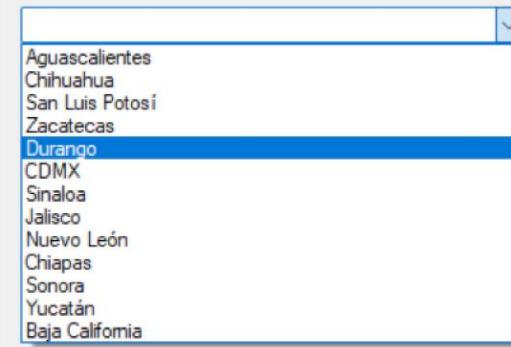
ListBox: muestra una lista en la que el usuario puede seleccionar uno o más elementos. Si el número total de elementos supera el número que se puede mostrar, se agrega una barra de desplazamiento al control ListBox.

ComboBox: se usa para mostrar datos en un cuadro combinado desplegable; este control aparece en dos partes: la parte superior es un cuadro de texto que permite al usuario escribir un elemento de la lista. La segunda parte es un cuadro de lista que muestra una lista de elementos de los que el usuario puede seleccionar uno.

¿Cuáles es tu color favorito? (ListBox)



Selecciona el estado donde vives (ComboBox)

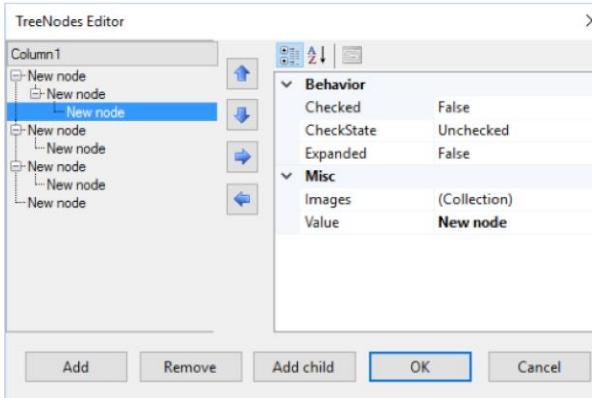
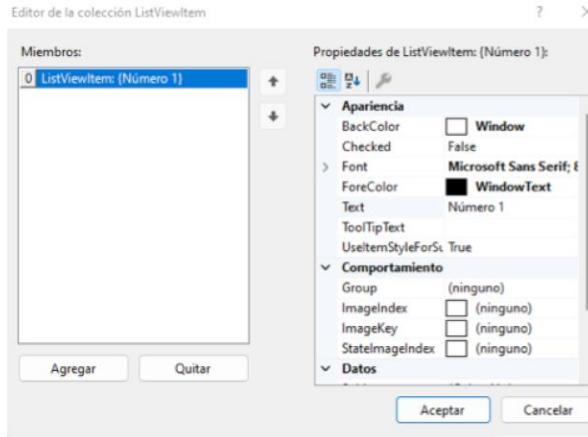


CONTROLES Y MENÚS

Controles ListView y TreeView:

ListView: muestra una lista de elementos con íconos. Puede usar una vista de lista para crear una interfaz de usuario, como el panel derecho del Explorador de Windows. El control tiene cuatro modos de vista: LargeIcon, SmallIcon, List y Details.

TreeView: muestra una jerarquía de nodos a los usuarios, como la forma en que se muestran los archivos y las carpetas en el panel izquierdo de la función Explorador de Windows del sistema operativo Windows. Cada nodo de la vista de árbol puede contener otros nodos, denominados nodos secundarios.



CONTROLES Y MENÚS

Control de fechas

MonthCalendar: presenta una interfaz gráfica intuitiva para que los usuarios vean y configuren la información de la fecha. El control muestra un calendario: una cuadrícula que contiene los días numerados del mes, dispuestos en columnas debajo de los días de la semana, con el rango de fechas seleccionado resaltado. Se puede seleccionar un mes diferente haciendo clic en los botones de flecha a cada lado del título del mes.

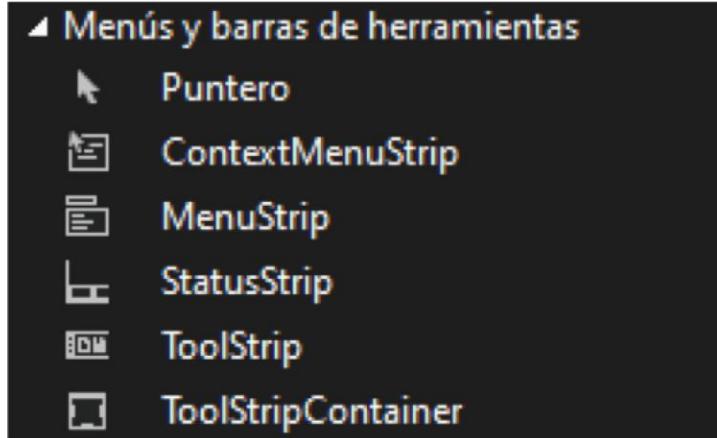


CONTROLES Y MENÚS

Menús

Un menú en un Windows Form se crea con un objeto **MainMenu**, que es una colección de objetos Menú y barras de herramientas. MainMenu es el contenedor de la estructura Menu del formulario y los menús están hechos de objetos MenuItem, que representan partes individuales de un menú.

Puedes agregar menús a Windows Forms en el momento del diseño, agregando el componente MainMenu y luego agregando elementos de menú mediante el Diseñador de menús.

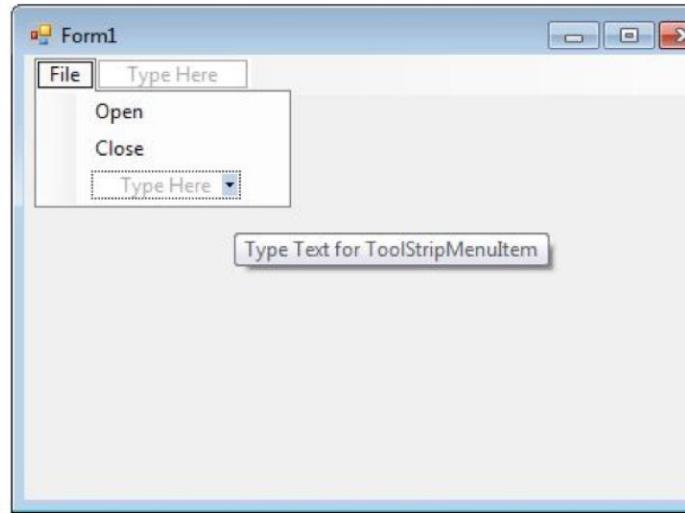


CONTROLES Y MENÚS

MenuStrip

El control **MenuStrip** admite la interfaz de documentos múltiples (MDI) y la combinación de menús, la información sobre herramientas y el desbordamiento. Puede mejorar la facilidad de uso y la legibilidad de sus menús agregando teclas de acceso, teclas de método abreviado, marcas de verificación, imágenes y barras separadoras.

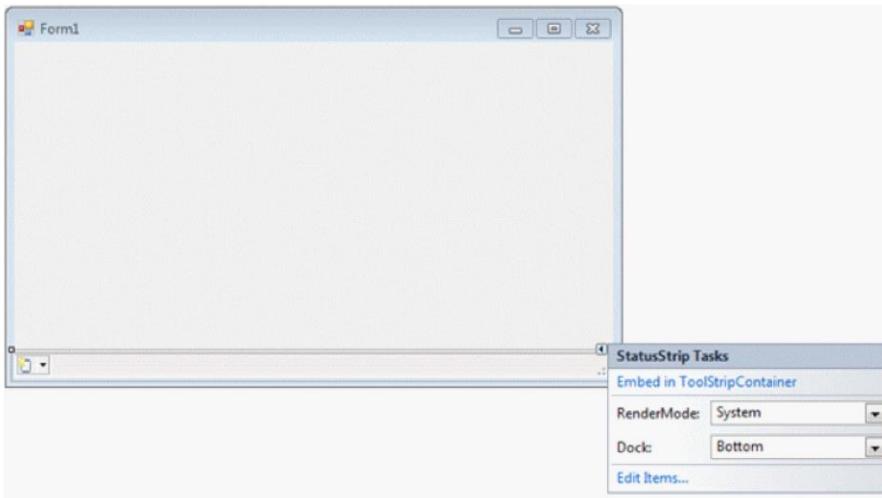
El control MenuStrip reemplaza y agrega funcionalidad al control MainMenu; sin embargo, el control MainMenu se conserva por motivos de compatibilidad con versiones anteriores y uso futuro si lo desea.



CONTROLES Y MENÚS

StatusStrip

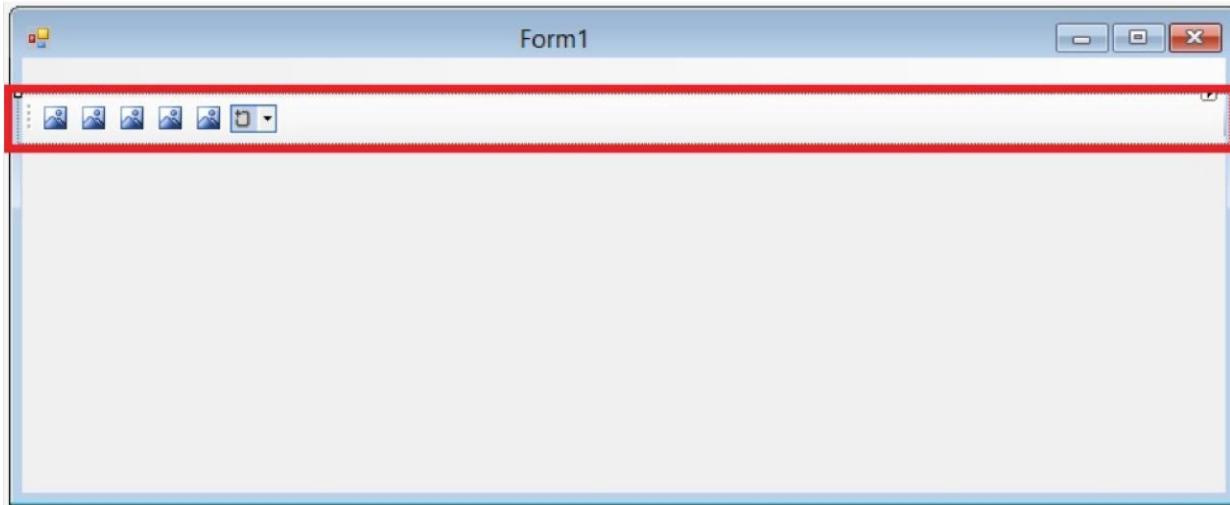
Un control **StatusStrip** muestra información sobre un objeto que se ve en un formulario, los componentes del objeto o información contextual relacionada con la operación de ese objeto dentro de su aplicación. Normalmente, un control StatusStrip consta de objetos ToolStripStatusLabel, cada uno de los cuales muestra texto, un ícono o ambos.



CONTROLES Y MENÚS

ToolStrip

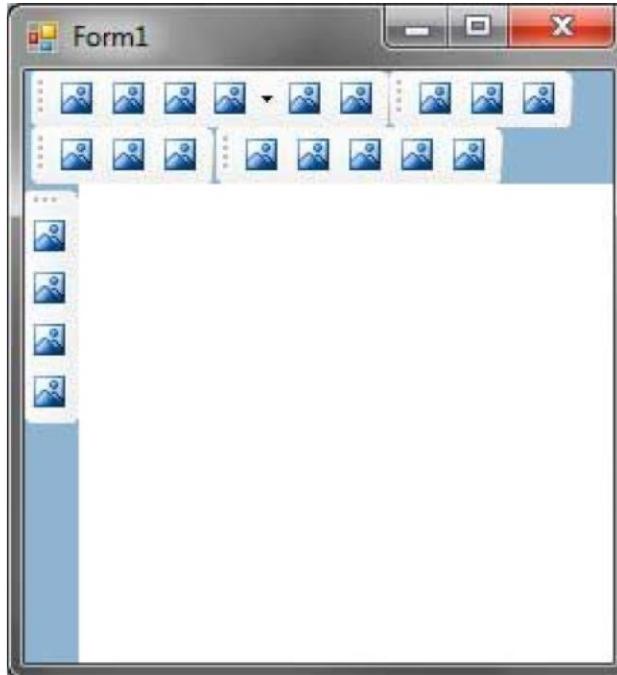
El control **ToolStrip** de Windows Forms y sus clases asociadas proporcionan un marco común para combinar elementos de la interfaz de usuario en barras de herramientas, barras de estado y menús. Los controles ToolStrip ofrecen una rica experiencia en tiempo de diseño que incluye activación y edición en el lugar, diseño personalizado y rafting, que es la capacidad de las barras de herramientas para compartir espacio horizontal o vertical.



CONTROLES Y MENÚS

ToolStripContainer

Un ToolStripContainer tiene paneles en los lados izquierdo, derecho, superior e inferior para colocar y distribuir los controles ToolStrip, MenuStrip y StatusStrip. Múltiples controles ToolStrip se apilan verticalmente si los coloca en el ToolStripContainer izquierdo o derecho. Se apilan horizontalmente si los coloca en el ToolStripContainer superior o inferior. Puede utilizar el ToolStripContentPanel central de ToolStripContainer para colocar los controles tradicionales en el formulario.



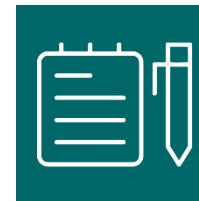
ACTIVIDAD INTEGRADORA 2

Te invitamos a realizar la siguiente actividad.

Presiona el botón para descargar la actividad.



Presiona el botón para entregar la actividad.





LECTURAS PARA REFORZAR LA UNIDAD

Capítulo 1:

- Hugon, J. (2021). *C# 9: desarrolle aplicaciones Windows con Visual Studio 2019*. Ediciones Eni.
- Brown, E. (2002, 1 abril). *Windows Forms Programming with C# (1st ed.)*. Manning Publications.

Capítulo 2:

- MacDonald, M. (2002, 4 octubre). *User Interfaces in C#: Windows Forms and Custom Controls (Softcover reprint of the original 1st ed.)*. Apress.



Windows Forms proporciona un modelo de programación unificado para el desarrollo de aplicaciones estándar de Windows. Ayuda de manera eficiente en la programación en C#, siempre puede ser productivo porque el sustrato común se ha desarrollado para beneficiar a todos.

Windows Forms trae un verdadero modelo de programación orientado a objetos para el desarrollo de GUI de Windows, lo que permite un marco extensible que es mucho más limpio y fácil de usar, en comparación con los intentos anteriores.

CONCLUSIÓN



¡FELICIDADES!

Acabas de concluir la tercera unidad de tu curso *Lenguajes de Programación III*. Te invitamos a finalizar este esfuerzo realizando el examen parcial correspondiente. Para ello, debes regresar a la pantalla principal y dar clic en *Presentar examen*.

4 UNIDAD

TRANSACCIONES CON BASES DE DATOS





4.1

ALTAS Y BAJAS



TEMARIO



4.2

MODIFICACIONES
Y CONSULTAS





INTRODUCCIÓN

En esta cuarta y última unidad, el alumno aprenderá a realizar un CRUD (altas, consultas, modificaciones y eliminación) de datos utilizando C# en el entorno de trabajo de Visual Studio, además de realizar la conexión de una base de datos SQL de Microsoft SQL Server Management al programa que estaremos trabajando.

COMPETENCIAS A DESARROLLAR



El alumno será capaz de realizar una alta y una eliminación programando en C#.



El alumno será capaz de efectuar una modificación y una consulta programando en C#.



VIDEO



TE INVITAMOS A VER EL SIGUIENTE VIDEO:



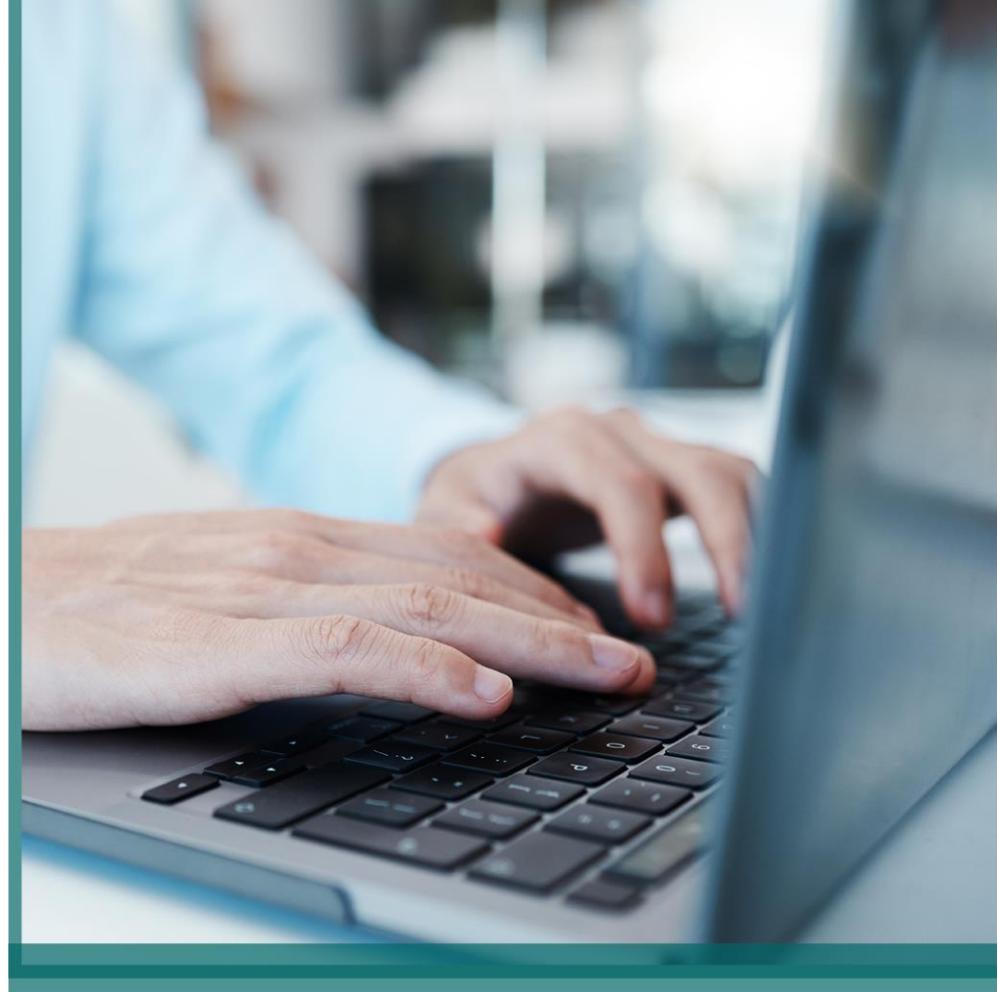
ALTAS Y BAJAS

CRUD: Create, Read, Update y Delete

Crear, Leer, Actualizar, Eliminar

En dichas aplicaciones, los usuarios deben poder crear datos , tener acceso a ellos en la interfaz de usuario leyendo los datos, actualizar o editarlos y eliminarlos.

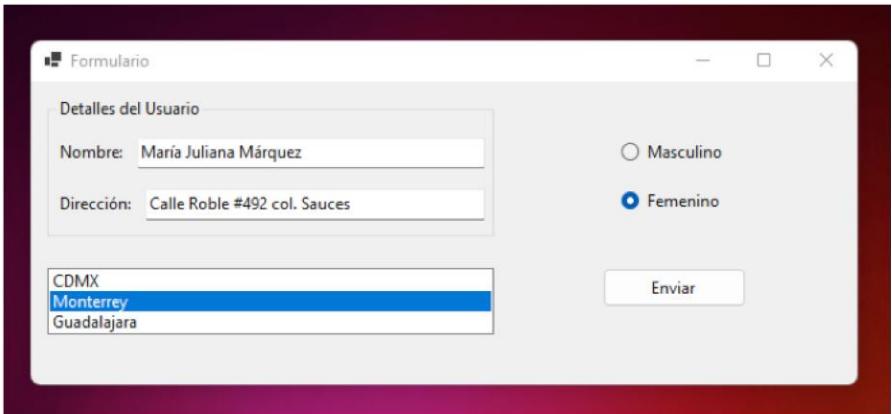
En las aplicaciones completas, las aplicaciones CRUD constan de 3 partes: una API (o servidor), una base de datos y una interfaz de usuario (UI).



Conexión a la base de datos

Utilicemos el formulario creado en la Unidad 2 donde se piden datos personales. Ahora vamos a guardar los datos ingresados en una base de datos; utilizaremos SQL Server para este ejemplo.

Entonces lo primero que haremos es la base de datos y luego un archivo de conexión a esta dentro del proyecto del formulario.



Conexión a la base de datos

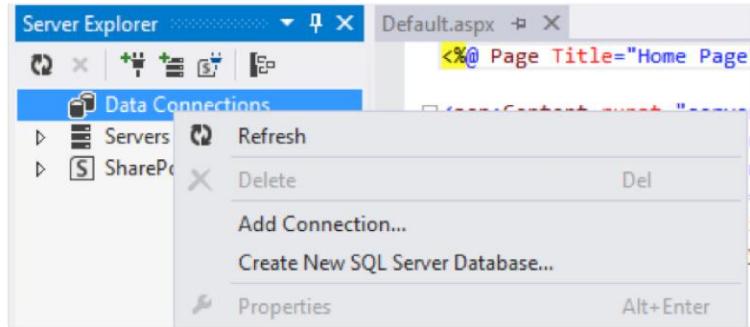
Con la base de datos creada en Microsoft SQL Server Management y llamada datos, vemos que incluye una tabla llamada datospersonales con cinco filas: ID, nombre, dirección, ciudad y género, cada una corresponde a los datos solicitados anteriormente en el formulario.

	Column Name	Data Type	Allow Nulls
▶	ID	int	<input type="checkbox"/>
	nombre	text	<input type="checkbox"/>
	direccion	text	<input type="checkbox"/>
	ciudad	text	<input type="checkbox"/>
	genero	text	<input type="checkbox"/>
			<input type="checkbox"/>

Conexión a la base de datos

Una vez que la interfaz y la base de datos están listas, es momento de realizar la conexión con la base de datos.

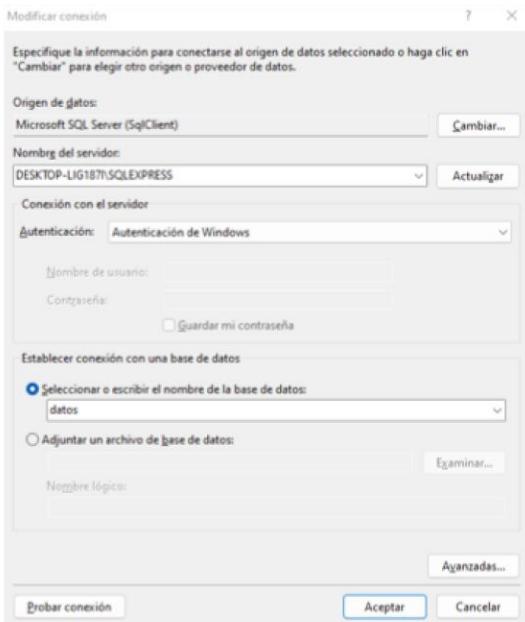
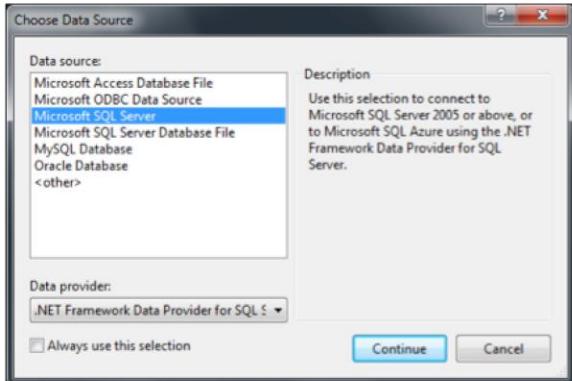
Para conectarse a la base de datos de SQL Server en Visual Studio, inicia un nuevo proyecto y abre el Explorador de servidores haciendo clic en la pestaña junto a la Caja de herramientas o desde Ver > Explorador de servidores. Haz clic derecho en ‘Conexiones de datos’ y luego clic en ‘Agregar conexión’.



ALTAS Y BAJAS

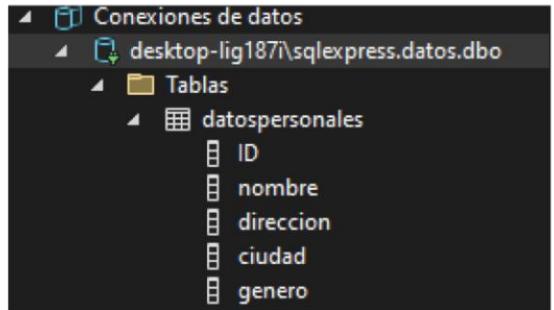
En la siguiente pantalla (Origen de datos), cambia la fuente de datos, asegúrate de seleccionar Microsoft SQL Server y luego haz clic en el botón ‘Continuar’.

- En el campo ‘Nombre del servidor’ ingresa el nombre del servidor de SQL Server Management.
- Cambia el método de ‘Autenticación’ a ‘Autenticación de Windows’.
- Finalmente, ingresa el nombre de la base de datos.



ALTAS Y BAJAS

Haz clic en el botón '**Aceptar**' para salir del asistente 'Aregar conexión', y se habrá agregado una conexión a la base de datos que aparecerá en 'Conexiones de datos' en 'Explorador de servidores'.





Create (Alta)

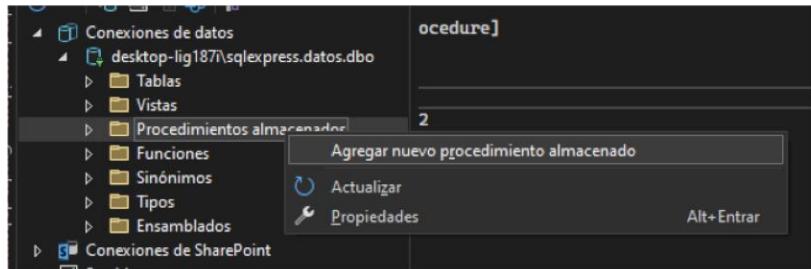
Significa crear una entrada. Esta entrada podría ser una cuenta, información de usuario, una publicación o una tarea.

En una base de datos SQL, crear es **Insert**.

Alta

Para crear un alta, el primer paso es crear la consulta **Insert** utilizando un archivo nuevo de la base de datos.

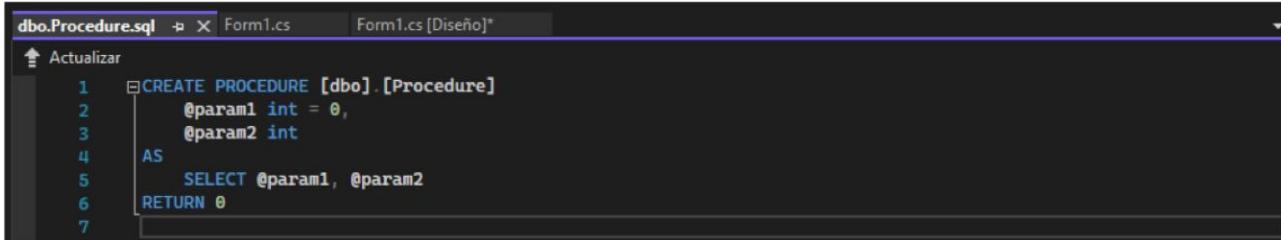
En la barra de menú de la base de datos, existe una carpeta llamada “Procedimientos almacenados”. Damos clic derecho sobre esa carpeta y seleccionamos la opción de “Aregar nuevo procedimiento almacenado”.



ALTAS Y BAJAS

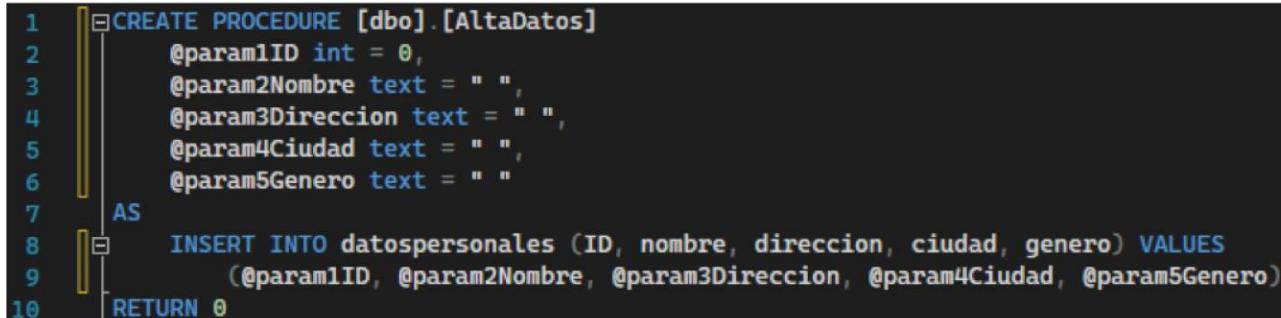
Alta

Ahora tendremos un archivo como el de la siguiente imagen:



```
dbo.Procedure.sql  X  Form1.cs      Form1.cs [Diseño]*  
↑ Actualizar  
1  CREATE PROCEDURE [dbo].[Procedure]  
2    @param1 int = 0,  
3    @param2 int  
4    AS  
5      SELECT @param1, @param2  
6    RETURN 0  
7
```

El método de insertar un registro queda así. Recuerda que el nombre del Store Procedure será como se almacena y se llamará según tu programa en VS. Para este caso quedaría del siguiente modo:



```
1  CREATE PROCEDURE [dbo].[AltaDatos]  
2    @param1ID int = 0,  
3    @param2Nombre text = " ",  
4    @param3Direccion text = " ",  
5    @param4Ciudad text = " ",  
6    @param5Genero text = " "  
7    AS  
8      INSERT INTO datospersonales (ID, nombre, direccion, ciudad, genero) VALUES  
9        (@param1ID, @param2Nombre, @param3Direccion, @param4Ciudad, @param5Genero)  
10     RETURN 0
```

Alta

Es importante especificar que los parámetros creados son como las variables en el código de alta del programa, y los datos ingresados se guardarán en la base de datos que importamos anteriormente. Por eso en la consulta del INSERT se ponen los parámetros creados, en lugar de los datos de manera directa.

```
CREATE PROCEDURE [dbo].[ConsultasDatos]
    @ID int,
    @Nombre text,
    @Direccion text,
    @Ciudad text,
    @Genero text,
    @Resultado text
AS
BEGIN
    INSERT INTO datospersonales VALUES (@ID, @Nombre, @Direccion, @Ciudad, @Genero, @Resultado)
END
```

Alta

El siguiente paso es codificar el botón de “enviar” de la interfaz del programa. Para esto, haz doble clic en cualquier parte del formulario para generar un evento Form_Load. Escribe el código del evento y también importa el espacio de nombres System.Data.SqlClient.

```
0 referencias
private void ConexionBD(object sender, EventArgs e)
{
    cn = new SqlConnection(@"Data Source=(local);Initial Catalog=datos;Integrated Security=True");
    cn.Open();
    //Entrelazar datos en la vista
    ObtenerDatos();
}
```

Ahora crearemos un método para obtener todos los datos de la tabla y configurarlos en la vista de cuadrícula de datos. Usaremos este código muchas veces, por lo que creamos un método simple.

El siguiente es el código para obtener todos los registros de la tabla y configurarlos en la vista de cuadrícula de datos.

```
private void GuardarFormulario()
{
    cmd = new SqlCommand("GuardarFormulario", cn);
    cmd.CommandType = CommandType.StoredProcedure;
    cmd.Parameters.AddWithValue("@ID", 0);
    cmd.Parameters.AddWithValue("@Nombre", "");
    cmd.Parameters.AddWithValue("@Direccion", "");
    cmd.Parameters.AddWithValue("@Ciudad", "");
    cmd.Parameters.AddWithValue("@Genero", "");
    cmd.Parameters.AddWithValue("@Resultados", "");
    da = new SqlDataAdapter(cmd);
    DataTable dt = new DataTable();
    da.Fill(dt);
    dataGridView1.DataSource = dt;
}
```

Explicación del código

- En primer lugar, coloca el nombre del procedimiento de almacenamiento y el objeto de conexión en el objeto SqlCommand, que definimos en la parte superior de la página.
- Define el tipo de comando como procedimiento de almacenamiento.
- Indica en todos los parámetros un valor nulo y cero, que es el tipo de operación para obtener todos los registros del procedimiento de almacenamiento.
- Inicializa el objeto de comando a objeto DataAdapter.
- Crea un nuevo objeto DataTable y coloca el valor del objeto del adaptador de datos al objeto de la tabla de datos mediante el método de relleno.
- Establece el objeto de la tabla de datos en la vista de cuadrícula de datos.

ALTAS Y BAJAS

Ahora generamos un método para guardar lo haciendo doble clic en dicho botón; agregamos el siguiente código en el evento y damos clic en el botón Guardar.

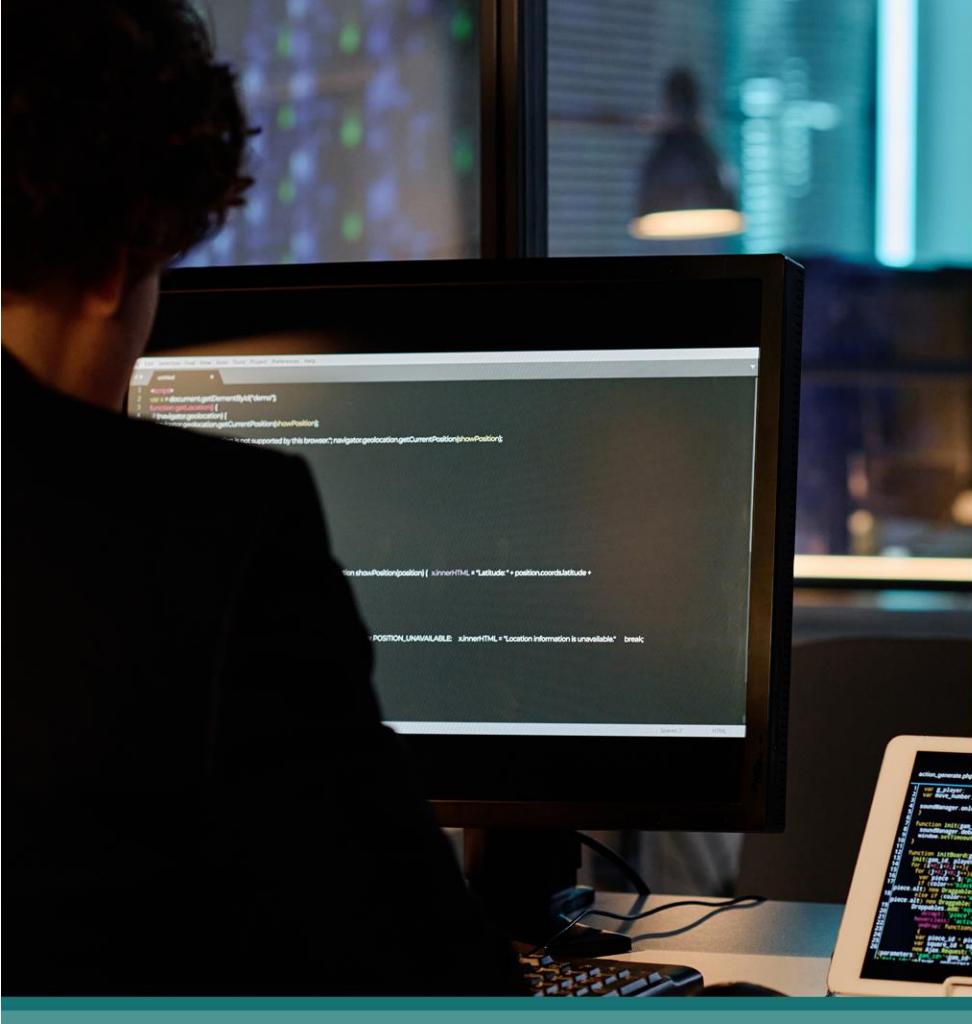
```
public void btnEnviar()
{
    if (txtNombre.Text != string.Empty && txtDireccion.Text != string.Empty &&
        Ciudad.Text != string.Empty && Hombre.Text != string.Empty && Mujer.Text != string.Empty &&
        txtresultados.Text != string.Empty)
    {
        cmd = new SqlCommand("GuardarFormulario", cn);
        cmd.CommandType = CommandType.StoredProcedure;
        cmd.Parameters.AddWithValue("@ID", 0);
        cmd.Parameters.AddWithValue("@Nombre", "");
        cmd.Parameters.AddWithValue("@Direccion", "");
        cmd.Parameters.AddWithValue("@Ciudad", "");
        cmd.Parameters.AddWithValue("@Genero", "");
        cmd.Parameters.AddWithValue("@Resultados", "");
        cmd.ExecuteNonQuery();
        ObtenerDatos();
        txtNombre.Text = "";
        txtDireccion.Text = "";
        Ciudad.Text = "";
        Hombre.Text = "";
        Mujer.Text = "";
        txtresultados.Text = "";
    }
}
```

ALTAS Y BAJAS

Primero, es necesario verificar que el usuario haya ingresado un valor en todos los campos; de lo contrario mostramos un mensaje o continuamos.

Después, así como se obtienen todos los parámetros de paso del método de registro en el procedimiento de almacenamiento, aquí utilizamos el método ExecuteNonQuery para insertarlo en la tabla.

```
public void btnEnviar()
{
    if (txtNombre.Text != string.Empty && txtDireccion.Text != string.Empty &&
        Ciudad.Text != string.Empty && Hombre.Text != string.Empty && Mujer.Text != string.Empty
        && txtResultados.Text != string.Empty)
    {
        cmd = new SqlCommand("GuardarFormulario", cn);
        cmd.CommandType = CommandType.StoredProcedure;
        cmd.Parameters.AddWithValue("@ID", 0);
        cmd.Parameters.AddWithValue("@Nombre", "");
        cmd.Parameters.AddWithValue("@Direccion", "");
        cmd.Parameters.AddWithValue("@Ciudad", "");
        cmd.Parameters.AddWithValue("@Genero", "");
        cmd.Parameters.AddWithValue("@Resultados", "");
        cmd.ExecuteNonQuery();
        ObtenerDatos();
        txtNombre.Text = "";
        txtDireccion.Text = "";
        Ciudad.Text = "";
        Hombre.Text = "";
        Mujer.Text = "";
        txtResultados.Text = "";
    }
}
```



Delete (Eliminar)

Eliminar es deshacerse de una entrada de la interfaz de usuario y de la base de datos.

En una base de datos SQL, **Delete** se utiliza para eliminar una entrada.

Eliminar

El proceso de eliminar es muy parecido al de alta, como fue el caso de un paciente nuevo; utilizamos el mismo archivo creado en el procedimiento de alta para crear una nueva consulta para Delete, como se aprecia en la siguiente imagen con la codificación:

```
||| BEGIN  
|||   DELETE  datospersonales WHERE id = ID;  
||| END
```

Eliminar

Ahora que tenemos la consulta, el siguiente paso es crear un espacio donde se muestren los datos dados de alta. Para ello, agregamos dos botones: uno para modificar y otro para eliminar; este espacio está en la misma interfaz que el formulario de alta.

The screenshot shows a user interface titled "Pacientes registrados" (Registered Patients). At the top, there is a search bar with a placeholder and a blue "Buscar" (Search) button. Below the search bar is a large, empty rectangular area intended for displaying patient data. At the bottom of the interface, there are two buttons: a light blue "Modificar" (Modify) button and an orange "Eliminar" (Delete) button.

Eliminar

Ahora damos clic en el botón de eliminar para darle funcionalidad.

La estructura del código del evento eliminar es muy parecida al del botón de alta, pero aquí cambia la consulta que se manda llamar del procedimiento, en este caso es eliminar.

A partir de ahí los parámetros siguen iguales, y al final declaramos False los botones de eliminar y modificar.

```
private void btnEliminar_Click(object sender, EventArgs e)
{
    if (txtTabla.Text != string.Empty)
    {
        cmd = new SqlCommand("Eliminar", cn);
        cmd.CommandType = CommandType.StoredProcedure;
        cmd.Parameters.AddWithValue("@ID", 0);
        cmd.Parameters.AddWithValue("@Nombre", "");
        cmd.Parameters.AddWithValue("@Direccion", "");
        cmd.Parameters.AddWithValue("@Ciudad", "");
        cmd.Parameters.AddWithValue("@Genero", "");
        cmd.Parameters.AddWithValue("@Resultados", "")
        cmd.ExecuteNonQuery();
        ObtenerDatos();
        txtNombre.Text = "";
        txtDireccion.Text = "";
        Ciudad.Text = "";
        Hombre.Text = "";
        Mujer.Text = "";
        txtResultados.Text = "";

        btnEliminar.Enabled = false;
        btnModificar.Enabled = false;
    }
}
```

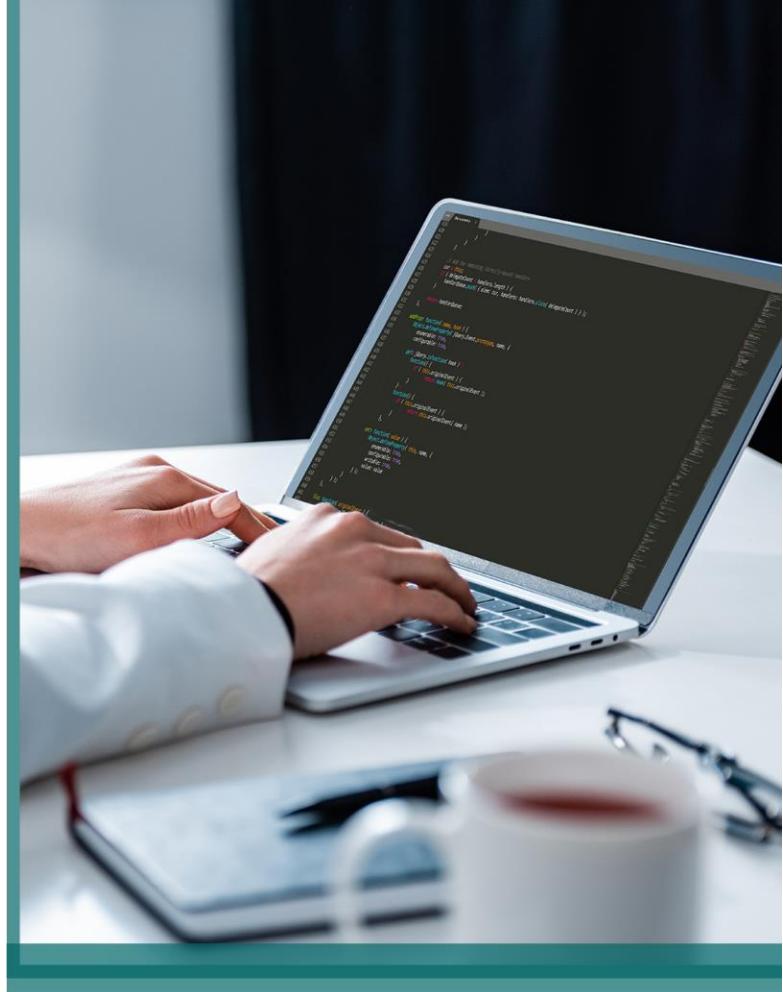
MODIFICACIONES Y CONSULTAS

Update (Modificaciones)

Update es la operación que permite modificar los datos existentes, es decir, editar los datos.

A diferencia del Read, la operación Update altera los datos existentes al realizar cambios en ellos.

En una base de datos SQL, se utiliza Update para actualizar una entrada.



MODIFICACIONES Y CONSULTAS

Modificaciones

Ahora que tenemos alta y baja de datos, el siguiente evento del CRUD es modificar. Para esto, seguimos los mismos pasos del botón dar de alta y del botón eliminar. En el archivo donde se encuentra este procedimiento agregamos una nueva consulta Update:

```
BEGIN  
  UPDATE datospersonales SET Nombre=@Nombre, Direccion=@Direccion, Ciudad=@Ciudad,  
  Genero=@Genero, Resutado=@Resultado WHERE id=@ID  
END
```

MODIFICACIONES Y CONSULTAS

Modificaciones

Ahora generamos un evento de clic en el botón de modificar: btnModificar, al dar doble clic en él y reemplazar con el siguiente código.

El código sigue la misma estructura que la codificación del código de alta y de eliminar.

```
private void btnModificar_Click(object sender, EventArgs e)
{
    if (txtTabla.Text != string.Empty && txtNombre.Text != string.Empty &&
        txtDireccion.Text != string.Empty && Ciudad.Text != string.Empty &&
        Hombre.Text != string.Empty && Mujer.Text != string.Empty
        && txtResultados.Text != string.Empty)

        cmd = new SqlCommand("Eliminar", cn);
        cmd.CommandType = CommandType.StoredProcedure;
        cmd.Parameters.AddWithValue("@ID", 0);
        cmd.Parameters.AddWithValue("@Nombre", "");
        cmd.Parameters.AddWithValue("@Direccion", "");
        cmd.Parameters.AddWithValue("@Ciudad", "");
        cmd.Parameters.AddWithValue("@Genero", "");
        cmd.Parameters.AddWithValue("@Resultados", "");
        cmd.ExecuteNonQuery();
        ObtenerDatos();
        txtNombre.Text = "";
        txtDireccion.Text = "";
        Ciudad.Text = "";
        Hombre.Text = "";
        Mujer.Text = "";
        txtResultados.Text = "";

        btnEliminar.Enabled = false;
        btnModificar.Enabled = false;
```

MODIFICACIONES Y CONSULTAS

Read (Consultas)

La operación Read significa obtener acceso a las entradas, o entradas en la interfaz de usuario; es decir, verlas. Nuevamente, la entrada puede ser cualquier cosa, desde información del usuario hasta publicaciones en redes sociales. Este acceso podría significar que el usuario obtenga acceso a las entradas creadas, justo después de crearlas o buscarlas. La búsqueda se implementa para permitir que el usuario filtre las entradas que no necesita.

En una base de datos SQL, la consulta Select pertenece a este tipo de operación.



MODIFICACIONES Y CONSULTAS

Consultas

Nuevamente en el archivo de procedimientos, realizamos las dos últimas consultas para leer los datos guardados en la base de datos. Para este caso hacemos dos tipos de consulta: una donde se busque específicamente el registro mediante el ID, y la otra consulta de manera general con todos los registros existentes.

```
□ BEGIN
  SELECT * FROM datospersonales WHERE id=@ID
END

□ BEGIN
  SELECT * FROM datospersonales
END
```

MODIFICACIONES Y CONSULTAS

Consultas

Al igual que la codificación de todos los botones anteriores, el código del evento de botón buscar, btnBuscar sigue la misma estructura, a diferencia del IF, donde se especifica que los datos buscados se mostrarán en los espacios donde los ingresamos.

```
private void btnBuscar_Click(object sender, EventArgs e)
{
    if (txtBuscar.Text != string.Empty)
    {

        cmd = new SqlCommand("Consultas datos", cn);
        cmd.CommandType = CommandType.StoredProcedure;
        cmd.Parameters.AddWithValue("@ID", txtBuscar.Text);
        cmd.Parameters.AddWithValue("@Nombre", "");
        cmd.Parameters.AddWithValue("@Direccion", "");
        cmd.Parameters.AddWithValue("@Ciudad", "");
        cmd.Parameters.AddWithValue("@Genero", "");
        cmd.Parameters.AddWithValue("@Resultados", "");
        dr = cmd.ExecuteReader();
        if (dr.Read())
        {
            txtNombre.Text = dr["Nombre"].ToString();
            txtDireccion.Text = dr["Direccion"].ToString();
            Ciudad.Text = dr["Ciudad"].ToString();
            Hombre.Text = dr["Hombre"].ToString();
            Mujer.Text = dr["Mujer"].ToString();
            txtResultados.Text = dr["Resultados"].ToString();
            btnEliminar.Enabled = false;
            btnModificar.Enabled = false;
        }
    }
}
```



LECTURAS PARA REFORZAR LA UNIDAD

Capítulo 1:

- Sabet, S. (2012). *Comparison between C# and Java : in implementation of a university desktop app.* (Dissertation).

Capítulo 2:

- Colectiva, N. (2019, 26 marzo). CRUD con C# y SQL Server. Blog *Nube Colectiva*.

La capacidad de crear, leer, actualizar y eliminar elementos en una aplicación web es crucial para la mayoría de los proyectos de pila completa.

CRUD es demasiado importante para ignorarlo, por lo que aprenderlo realmente puede mejorar la confianza dentro de pilas desconocidas. Registrarse en un sitio web, guardar un marcador, ajustar una configuración o eliminar una publicación de Facebook: sin CRUD, nada de esto sería posible.

CONCLUSIÓN



¡FELICIDADES!

Acabas de concluir la cuarta unidad de tu curso *Lenguajes de Programación III*. Te invitamos a finalizar este esfuerzo realizando el examen parcial correspondiente. Para ello, debes regresar a la pantalla principal y dar clic en *Presentar examen*.

BIBLIOGRAFÍA

- A. (2022, 1 septiembre). Tutorial: *Un paseo por el IDE de Visual Studio*. Microsoft Docs.
- Chavhan, G. (2018, 9 abril). Common Language Specification In .NET. *C# Corner*.
- C# console based application. (s. f.) – *Net-Informations*.
- *Documentos de C#: inicio, tutoriales y referencias*. (s. f.). Microsoft Docs.
- JHadiya, Y. (s. f.). *CRUD Operation In C# Windows Application Using Store Procedure*.
- Staff, C. (2021, 9 agosto). Windows Forms Controls. *CodeGuru*.
- Thakur, D. (2013, 8 marzo). Common Type System (CTS). *Computer Notes*.
- Thompson, B. (2022, 25 agosto). C# Windows Forms Application Tutorial with Example. *Guru99*.



PROYECTO FINAL

PROYECTO FINAL

Te invitamos a realizar la siguiente actividad.

Presiona el botón para descargar la actividad.



Presiona el botón para entregar la actividad.

