Curso de Programación .NET nivel I

Unidad 1

“Introducción al Framework .NET y C#”

Contacto: consultas@elearning-total.com
Web: www.elearning-total.com

Índice

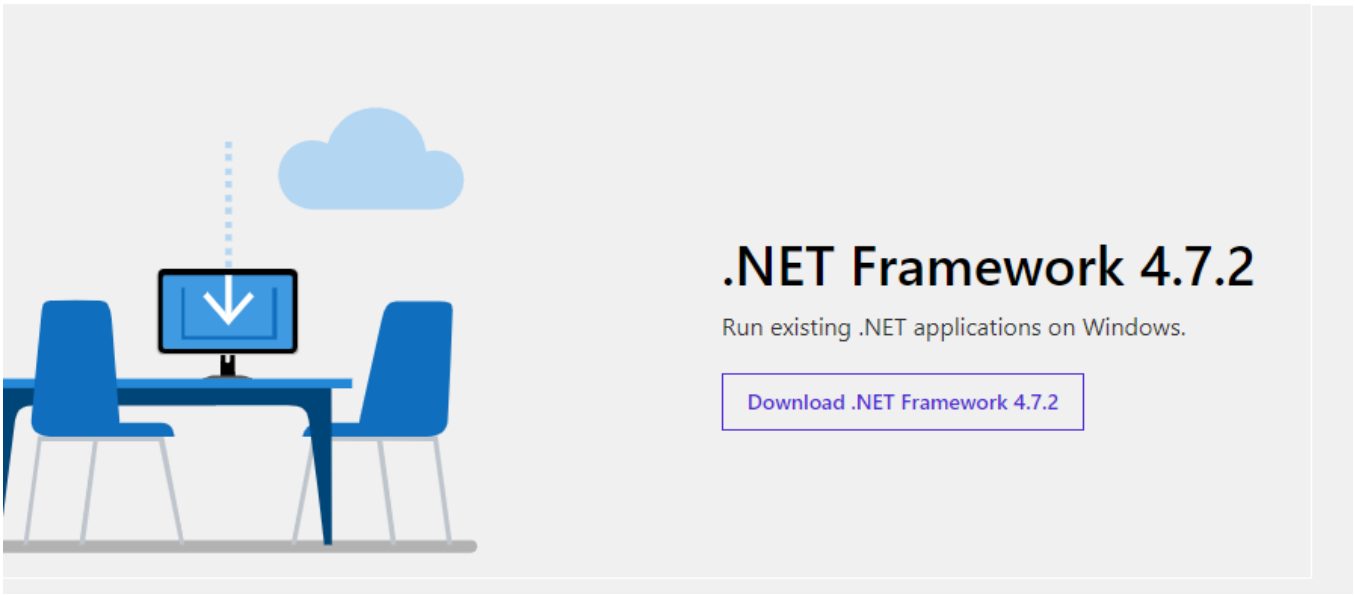
La plataforma .NET

introducción

La plataforma .NET fue lanzada en febrero del 2002 por los Ingenieros de Microsoft® como respuesta a la creciente demanda de software y proliferación de los sistemas a nivel mundial. .NET es una plataforma que ayuda a los programadores y diseñadores de software a acelerar el proceso de construcción de aplicaciones por medio de herramientas visuales y componentes pre-ensamblados.

Algo importante a notar que siempre trae confusión es que .**NET no es un lenguaje, sino que es una plataforma que permite la utilización de diversos lenguajes de programación.**

Seguramente tengas instalado .NET en este momento en tu computadora y hayas corrido cientos de aplicaciones sobre él. El problema de explicar .NET es que todos estos componentes no son visibles. Lo más



cercano que estuviste a saber de su existencia es haber bajado el Framework runtime:

este Runtime es solo una porción del framework y **es necesario para correr aplicaciones hechas en .NET**. Para crear aplicaciones hay que instalar Visual Studio, el cual veremos más adelante.

Por fortuna a partir de las nuevas versiones de Windows, el runtime ya viene instalado!

Componentes

.NET se compone de:

- **Biblioteca de clases base (FCL)**
  Uno de los componentes fundamentales de .NET es su biblioteca de clases **(FCL)** que ayudan al programador a agilizar su productividad por medio de soluciones de código ya implementadas, sin tener que preocuparse por las mismas y enfocar su esfuerzo en la lógica del negocio.
  Los FCL son como los bloques estandarizados de construcción de viviendas prefabricadas, en ese proceso solo resta prestar atención al ensamblado del mismo para las necesidades del cliente.
  FCL provee componentes de interfaz de usuario o UI, acceso a datos y base de datos, criptografía, desarrollo Web, algoritmos numéricos y comunicaciones de redes.

- **Lenguajes de programación**
  Los lenguajes de programación son otro componente fundamental y es la manera en que un programador traduce la lógica a código que los sistemas pueden interpretar. Los principales

lenguajes que maneja .NET son C#, Visual Basic y F#. Siguiendo con el ejemplo de las viviendas, el código es el detalle del plano de construcción.

- **Entorno común de ejecución de lenguaje (CLR)**

  Estos lenguajes pueden a grandes rasgos combinarse entre sí, y por lo tanto reutilizar componentes escritos, por ejemplo en Visual Basic para una aplicación en C#. Esta funcionalidad esta dada por el Entorno común de ejecución de lenguaje (CLR).

  En .NET los programas corren en este entorno de Software contrario a lo que sucede con lenguajes como C o Pascal que corren de manera nativa en el Hardware. Esta capa, el CLR, permite abstraer la aplicación brindando servicios de seguridad, gestión de memoria y de manejo de excepciones mejoradas.

Como vemos en el siguiente dibujo el CLR junto con el código intermedio generado, que puede provenir de distintos lenguajes, forman lo que Microsoft® denomina la infraestructura de Lenguaje común o CLI por su acrónimo en ingles.

El compilador compila el código fuente a un CIL(Common Intermediate Language) anteriormente conocido como MSIL(Microsoft Intermediate Language).

El CIL es un lenguaje de bajo nivel legible por humanos.

EL CLR es el encargado de tomar el CIL y transformarlo en algo que la computadora, más bien el sistema operativo, comprenda. El resultado final es el lenguaje de máquina.

Por ejemplo: en .NET nuestro código para mostrar un cartel es:

```
MessageBox.Show(“HOLA”).
```

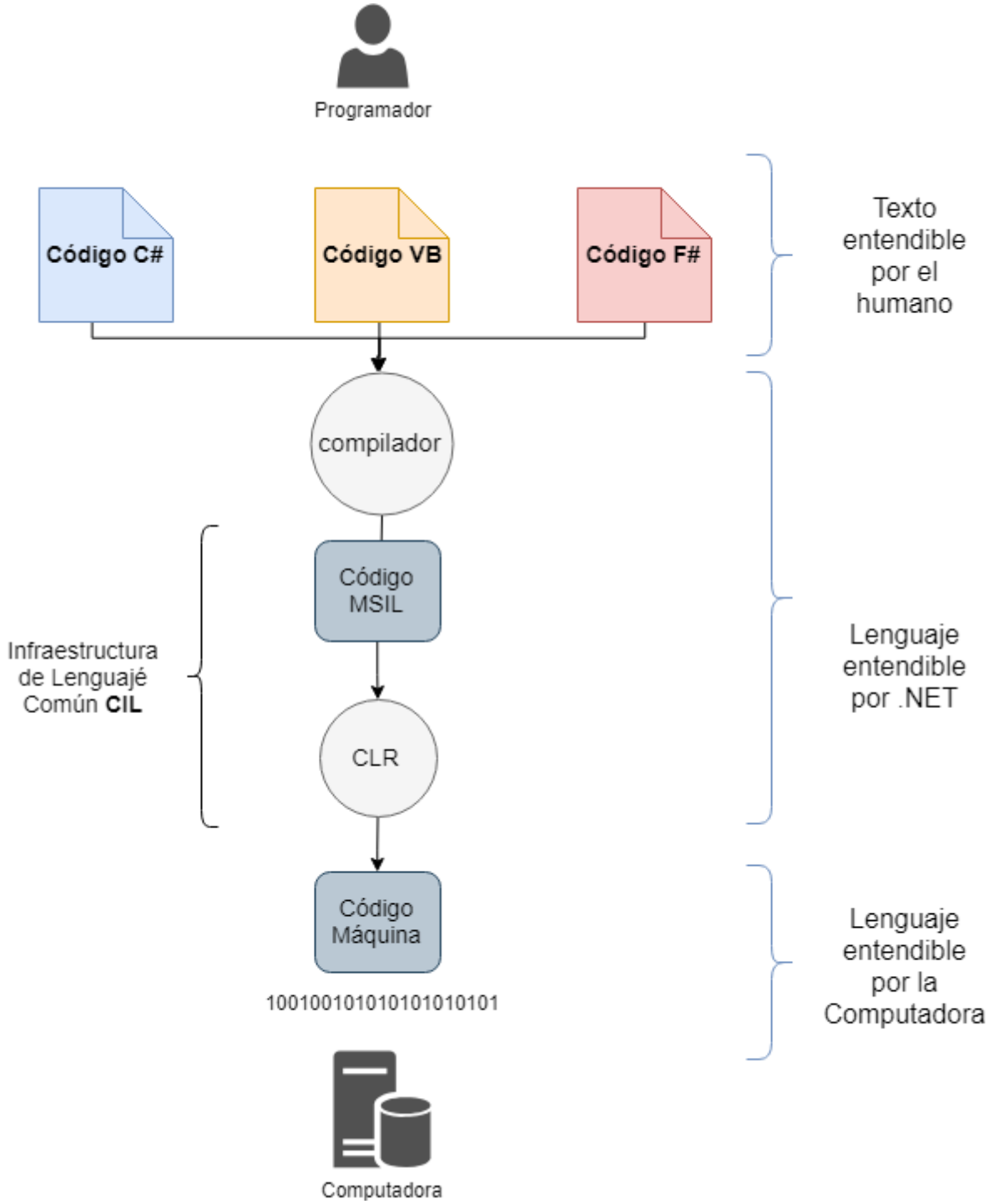
El CLR la tiene que traducir en algo que Windows entienda:

```
int MessageBox( HWND hWnd, LPCTSTR lpText, LPCTSTR lpCaption, UINT uType );
```

No le presten atención a todo el texto, simplemente comparen lo simple que es en .NET mostrar un cartel comparado con C en este caso.

Importante

Durante todo el curso vamos a usar mucho la palabra “compilar”, que significa transformar nuestro texto de programación entendible para humanos (o para algunos) a un lenguaje que nuestra computadora entienda. El gráfico a continuación contiene todos los pasos del proceso.


Programador
Código C#
Código VB
Código F#
Texto entendible por el humano
compilador
Código MSIL
Infraestructura de Lenguajé Común CIL
CLR
Lenguaje entendible por .NET
Código Máquina
10010010101010101010
Lenguaje entendible por la Computadora
Computadora

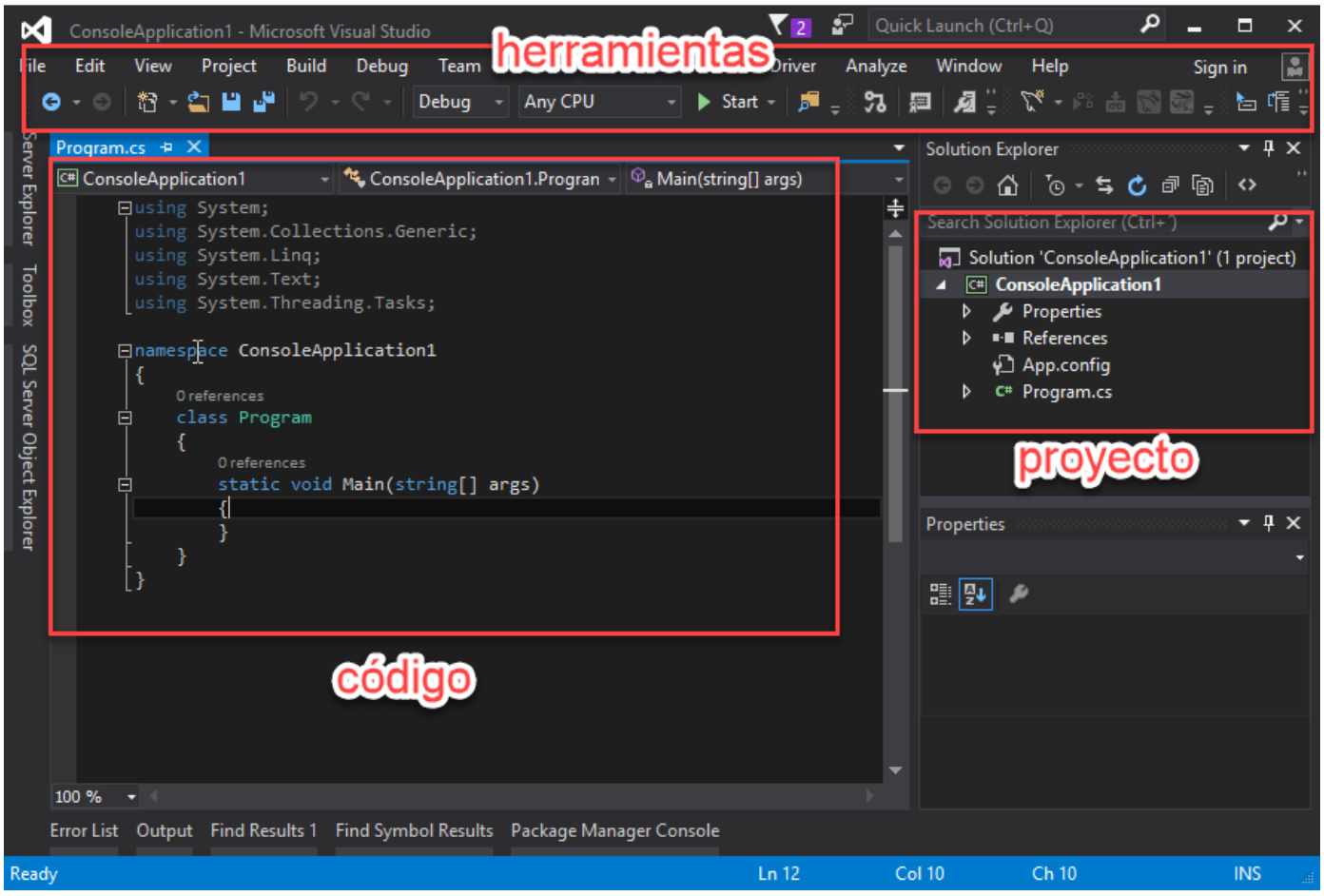
Visual Studio

Introducción

Bueno, ya nos sacamos de encima los conceptos mas “pesados” de la arquitectura .NET que son fundamentales para un desarrollador Full Stack.
Ya sea por temas de corrección de errores o investigación de performance, siempre es bueno saber que sucede por debajo de los programas que estamos ejecutando o haciendo ejecutar a otras personas.

Visual Studio: La aplicación para crear aplicaciones.

La plataforma .NET centra casi todo su administración en Visual Studio.
Visual Studio es lo que se conoce como un Entorno de desarrollo integrado o IDE por sus siglas en ingles. Un IDE es una aplicación informática que permite facilitar las tareas a los desarrolladores o programadores proporcionándole servicios integrales y automatizaciones. Normalmente en un IDE podemos encontrar un editor de código fuente, un compilador y un depurador entre otras herramientas.



En la imagen vemos que Visual Studio a su izquierda viene con un editor de código fuente, a la derecha la estructura de un proyecto en Visual Studio lo cual veremos más adelante en esta unidad y arriba en el un

conjunto de herramientas entre las que se distinguen un icono verde de Play, ese ícono sirve para probar nuestra aplicación y visualmente seguir el flujo de nuestra aplicación para verificar su correcto funcionamiento, esto es el **Debugger**.

Versiones

Existen principalmente tres versiones de Visual Studio:

Características compatibles	Visual Studio Community	Visual Studio Professional	Visual Studio Enterprise
Escenarios de uso admitidos	●●●○	●●●●	●●●●
Compatibilidad de la plataforma de desarrollo [2]	●●●●	●●●●	●●●●
Entorno de desarrollo integrado	●●●○	●●●○	●●●●
Depuración y diagnóstico avanzados	●●○○	●●○○	●●●●
Herramientas de pruebas	●○○○	●○○○	●●●●
Desarrollo multiplataforma	●●○○	●●○○	●●●●
Herramientas y características de colaboración	●●●●	●●●●	●●●●

La versión **Community** como su nombre lo indica es soportada por la comunidad de Microsoft y es gratuita bajo ciertos requisitos:

- Desarrollador individual
- Aprendizaje en clase o investigación académica
- Contribución a proyectos de código abierto
- Organizaciones no empresariales para un máximo de 5 usuarios
- NO esta permitido para uso empresarial

La principal diferencia con la **Professional** es que en esta última si está permitido el uso empresarial y no es gratuita. Luego las demás diferencias son mínimas.

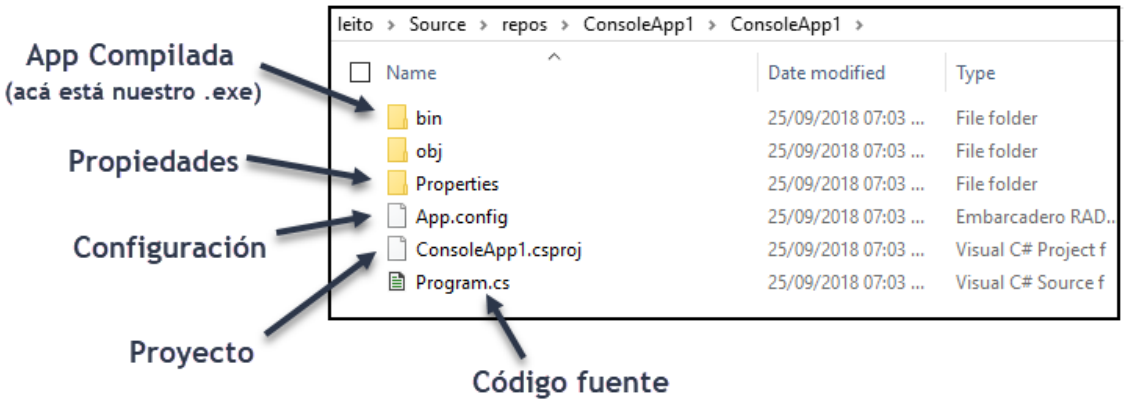
Por último la versión **Enterprise** es la más completa de todas ellas y esta pensada para escenarios complejos ya que cuenta con todo el set de herramientas de depuración y de Testing como por ejemplo las pruebas de

UI automatizadas, en la que se simula al usuario final presionando botones y clicks en la ventana para corroborar que la aplicación no tenga errores del lado de los componentes visuales.

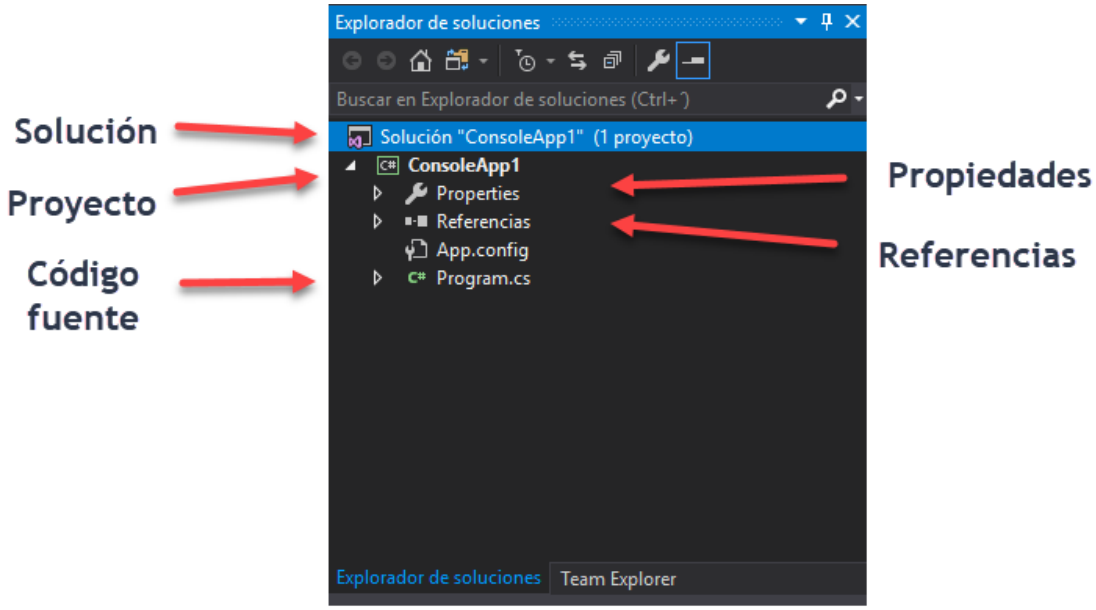
Este curso esta pensado para realizarse completamente con la versión gratuita **Community** de Visual Studio la cual pueden bajar directamente de Microsoft, en el campus pueden encontrar un video como bajar, instalar y configurar Visual Studio.

Los Proyectos y Soluciones en Visual Studio

En Visual Studio los proyectos tienen una estructura en la cual se guardan sus configuraciones y código. Esto me permite al igual que otras aplicaciones guardar el progreso, volver a cargarlo o compartir mis proyectos con otras personas. El concepto es el mismo que un archivo de Word o Excel pero en vez de ser un solo



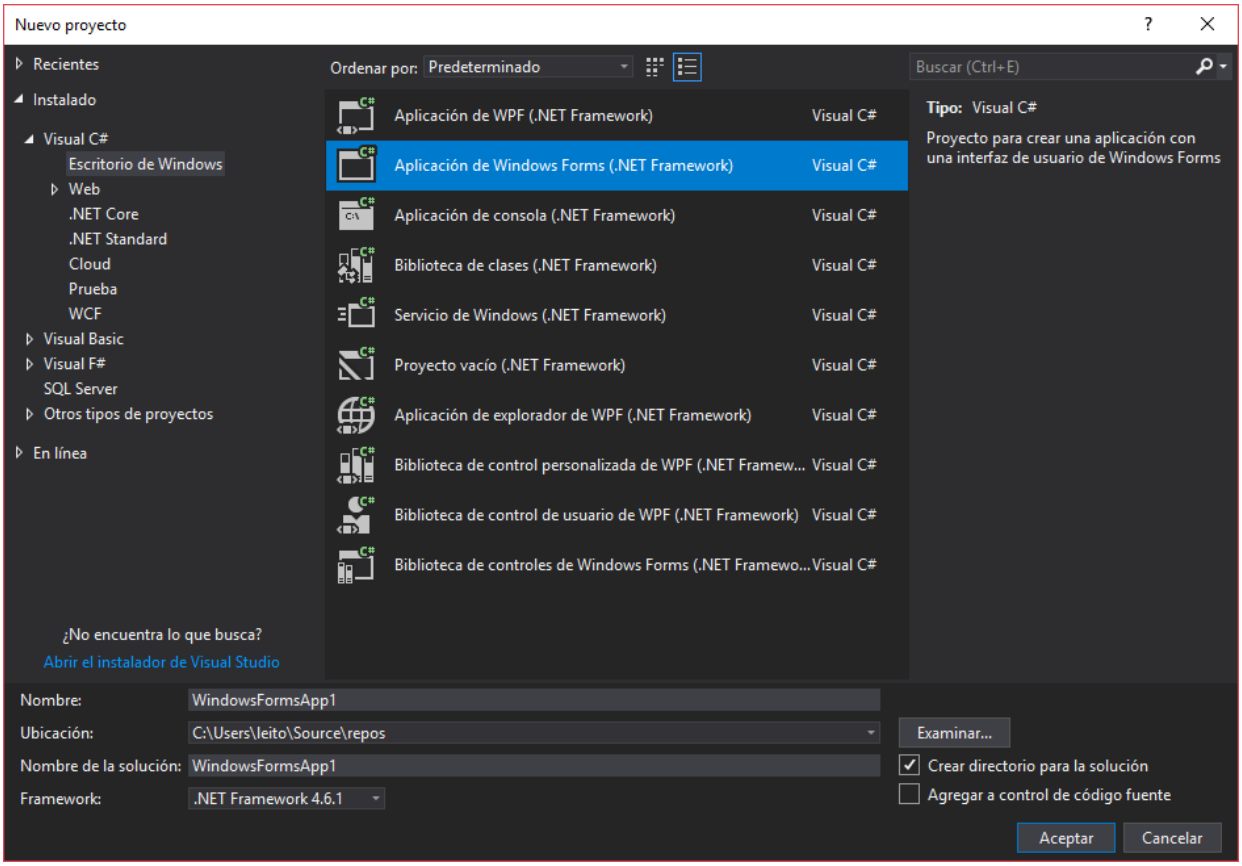
archivo son varios dentro de un directorio.



La estructura básica de una solución en Visual Studio:

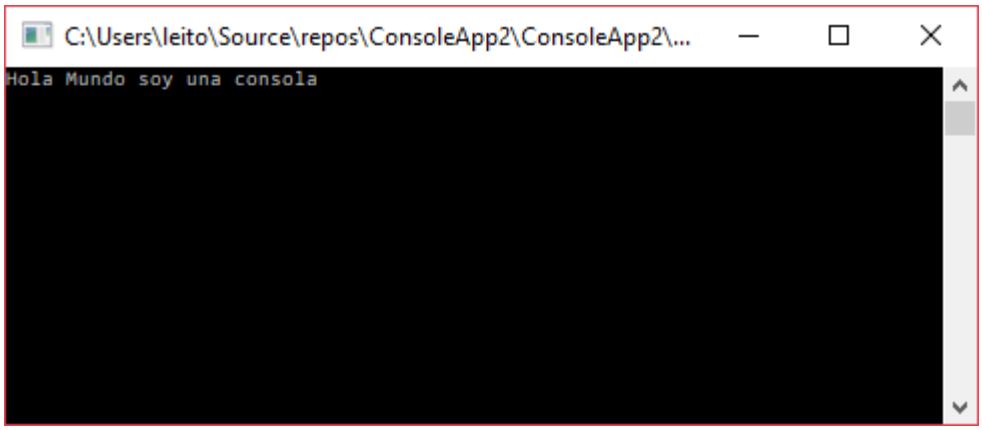
- **Solución**: Es el contenedor de proyectos. Puede contener 1 o más proyectos.
    - **Proyecto**: Es la unidad de agrupación lógica de código y configuración en Visual Studio para una aplicación. Un proyecto si o si tiene que tener una solución que lo contenga. Un proyecto a su vez consta de:
        - **Archivos de código fuente:** contiene la lógica de nuestra aplicación descripta a través de un lenguaje de programación.
        - **Referencias:** contiene referencias a las funciones y objectos de otras aplicaciones lo que permite su reutilización.
        - **Propiedades:** todo lo referido a la configuración de nuestro proyecto por ejemplo la carpeta a donde se compila, la versión etc. esta descripto acá.
        - **Configuración:** Al contrario de propiedades, la Configuración es todo lo que una vez que la aplicación se compila, puedo modificar y persista en el tiempo. Por ejemplo si cambio el tamaño de una ventana puedo habilitar una configuración para que la “recuerde”.

Tipos de Proyectos



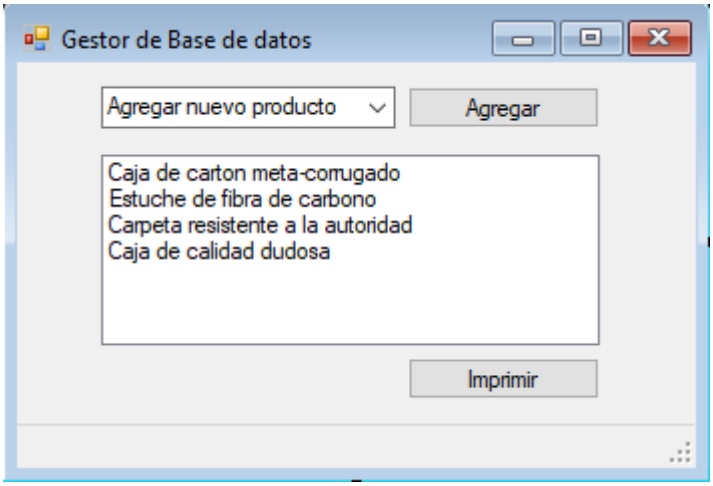
En visual Studio existen muchos tipos de proyectos entre los más importantes:

- **Proyectos de consola**
  Se ejecutan localmente en la computadora del cliente y opcionalmente tienen una interfaz tipo



consola.

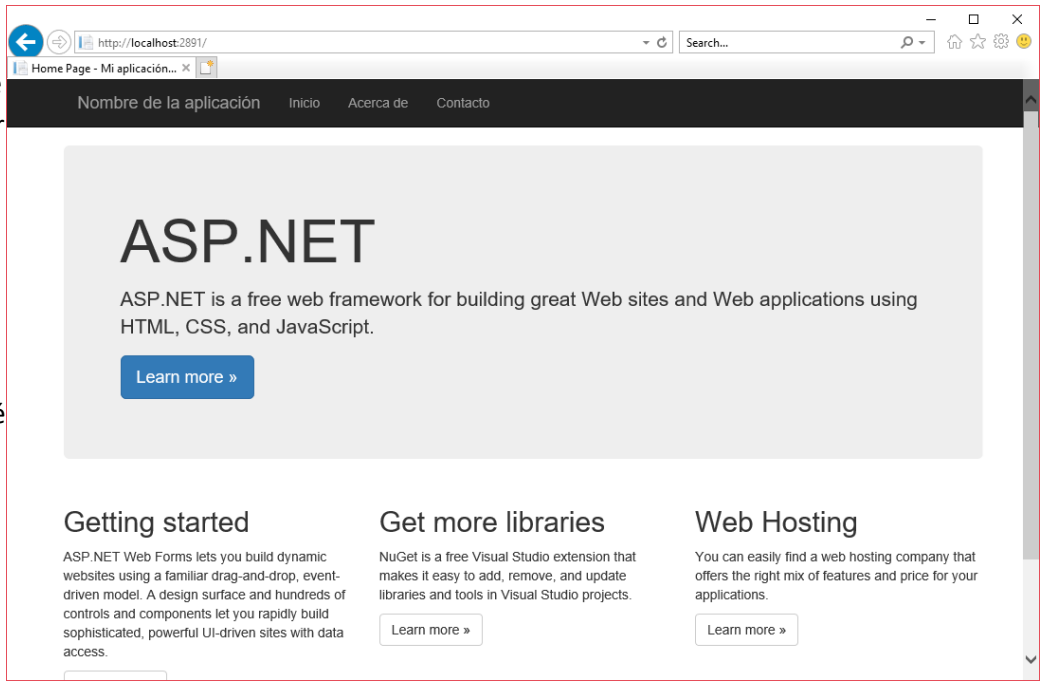
- **Proyectos de escritorio Visuales**
  Son los que se ejecutan localmente en la computadora del cliente final, por ejemplo: una calculadora o un editor de texto. Hay disponibles 2 tecnologías clásicas para realizar aplicaciones visuales en



.NET.

  - **WinForms**: Es la más utilizada por su rapidez y simpleza para realizar aplicaciones, con arrastrar un par de componentes visuales ya podemos crear una interfaz. Su contra es que muchas veces al modificar el diseño visual hay que modificar la lógica, esto genera en algunos casos problemas para su mantenimiento.
  - **WPF**: Es la nueva apuesta de Microsoft para desacoplar la vista de la lógica. La contra es que la curva de aprendizaje es elevada y puede agregar complejidad a un proyecto simple.

- **Proyectos Web**
  Se ejecutan en un IIS, el cual es la aplicación de Microsoft para ejecutar aplicaciones web y el usuario accede a través de un navegador.

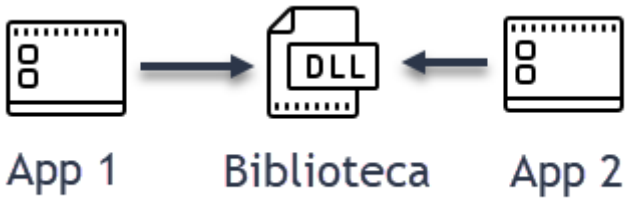


De ntro de Web tambié n

hay distintas tecnologías para crear páginas web:

- **WebForms**: Es análogo a WinForms pero aplicado a la web.
- **MVC**: Una implementación del modelo de tres capas para aplicaciones Web con un modelo análogo al WPF. Al igual que sucede con Winforms y WPF, MVC viene a subsanar temas de mantenimiento de código complejo de WebForms.
- **WCF:** Proyectos para realizar Web Services los cuales son funciones expuestas para ser consumidas remotamente por otras aplicaciones. Dicho de otro modo son "librerías" en la web.

- **Biblioteca de Clases o "DLL"**
  Son proyectos que sirven para almacenar funciones que luego van a ser reutilizadas por al menos más de un proyecto dentro de .NET.
  Aquí vemos el caso modelo de la característica de .NET en el cual hablabamos del entorno común de ejecución del lenguaje.
  Por ejemplo: Imaginen que tienen dos proyectos distintos que utilizan el mismo formato de fecha y hora y ustedes pensaron el código para obtener ese formato especifico, en vez de tener que repetir el mismo código y hacer copy-paste, pueden crearse una biblioteca de clases con ese código ahi y llamarlo "LaFechaYHoraQueQuiero", luego con ese nombre los otros dos proyectos pueden simplemente ir a buscar ese código y ejecutarlo. Para referenciar esa biblioteca tienen las "Referencias" del proyecto. como vimos anteriormente.



Este curso cubre los tipos de proyectos **Consola** y **WinForms**, los demás proyectos se enumeran aquí a modo informativo.