¿Qué diferencia existe entre plataforma y un framework?

Plataforma se refiere al hardware o software sobre el cual una pieza del software o programa está construida o a la cual está destinada, por ejemplo, Windows sería una plataforma.

(Resumen: plataforma es el hardware en el que el software se ejecuta)

Por otro lado, un Framework se refiere a una colección de bibliotecas/clases con el objetivo en común de proporcionar una base sobre la cual crear el software. Los framework sirven como una especie de plantilla o guía que utiliza el programador para aquellas partes del desarrollo que pueden ser automatizadas Ejemplo: una aplicación usando Xamarin la cual es una implementación del framework .NET En particular.

(Resumen: conjunto de bibliotecas y clases para crear software)

¿Qué es el .NET Framework y qué tipo de aplicaciones se pueden desarrollar?

.Net framework es un entorno de ejecución, provee servicios a las aplicaciones en ejecución. Soporta varios tipos de lenguajes de programación, por ejemplo: C#, visual basic, c++, entre otras.

.NET Framework consta de dos componentes principales: Common Language Runtime (CLR) es el encargado de administrar el código en tiempo de ejecución y la biblioteca de clases de .NET Framework, las cuales pueden ser utilizadas para el desarrollo de aplicaciones, algunas de ellas son:

- Aplicaciones de consola.
- Aplicaciones para Windows
- Aplicaciones orientadas a servicios que utilizan Windows Communication Foundation(WCF)
- Aplicaciones habilitadas para flujo de trabajo que utilizan Windows Workflow Foundation(WF)

¿Cuáles son sus dos partes principales?

.NET Framework tiene dos componentes principales:

Common Language Runtime (CLR): es un entorno de ejecución para los códigos de los programas que corren sobre la plataforma Microsoft .NET, está encargada de compilar de CIL al código de máquina nativo.

Biblioteca de clases de .NET Framework: maneja la mayoría de las operaciones básicas que se encuentran involucradas en el desarrollo de aplicaciones.

¿Qué es CTS?

“Common Type System” o conocido como CTS dentro del .NET FRAMEWORK, CTS es uno de los núcleos principales de la arquitectura parte del .Net. CTS estandariza los tipos de datos para todos los lenguajes de programación dentro del .Net.

(Resumen: hace que puedas convertir tu lenguaje a otro)

¿Qué es BCL?

Significa biblioteca de clases Base, también conocida como biblioteca de clases (CL). BCL es un subconjunto de la biblioteca de clases del Framework (FCL). La biblioteca de clases es una colección de tipos reutilizables que están estrechamente integrados con CLR. La biblioteca Base Class proporciona clases y tipos que son útiles para realizar operaciones diarias, por ejemplo, para tratar con tipos primitivos y de cadena, conexión de base de datos, operaciones de E / S.

(Resumen: Subconjunto de bibliotecas para trabajar en el framework)

¿Qué es CLS?

CLS es un subconjunto de CTS. Define un conjunto de reglas y restricciones que todo idioma debe seguir y que se ejecuta bajo el marco .NET.

(Resumen: Define las reglas/restricciones de los idiomas de .NET)

¿Cuál es su importancia?

-Define reglas para tipos de datos que todos los lenguajes .net deben seguir

-Hace posible el intercambio de lenguajes.

-Provee una librería que contiene tipos básicos usados en el desarrollo de elementos tales como strings, bytes, char, date, etc.

-CTS define ciertos tipos de categoría como clases, numerales, estructuras, Interfaces.

-También define tipos de propiedades, tipos de modificadores de acceso y cómo trabaja la herencia y sobrecarga.

¿Qué versiones del framework fueron liberadas en el tiempo?

.NET FrameWork 1.0:

La primera versión del .net, lanzado el 13 de febrero 2002:

-Soporte integrado para controles ASP.NET móviles, que anteriormente estaba disponible como complemento

-Permite que los ensamblados de Windows Forms se ejecuten de manera semi-confiable desde Internet

-Habilita la seguridad de acceso al código en aplicaciones ASP.NET

.Net FrameWork 2.0:

Liberado el 22 de Enero de 2006

-Soporte informático completo de 64 bits para las plataformas de hardware x64 e IA-64

-Integración con Microsoft SQL Server: en lugar de utilizar T-SQL, se pueden crear procedimientos almacenados y activadores en cualquiera de los lenguajes compatibles con .NET

-Nuevas funciones de personalización para ASP.NET, como compatibilidad con temas, máscaras, páginas maestras y elementos web

.Net FrameWork 3.0

.NET Framework 3.0, anteriormente llamado WinFX, fue lanzado el 21 de Noviembre de 2006

-Windows Presentation Foundation (WPF), anteriormente con nombre en código Avalon: un nuevo subsistema de interfaz de usuario y API basados en el lenguaje de marcado XAML, que utiliza hardware de gráficos por computadora en 3D y tecnologías Direct3D.

-Windows Workflow Foundation (WF): permite la automatización de tareas de creación y transacciones integradas mediante flujos de trabajo

Objetivos del Net Framework:

Proveer un ambiente: Consistente, Minimice problemas, Segura, performance e Integrable.

¿Qué es el estudio visual studio?

Visual studio es una IDE (entorno de desarrollo integrado) creado por Microsoft, es un editor de código permite a los desarrolladores crear sitio y aplicaciones web, así como servicios web en cualquier entorno compatible con la plataforma .NET. Microsoft provee visual studio para Windows y macOS también nos ofrece una version Community la cual es con una licencia gratuita, también existe la versión paga la cual nos brinda algunas otras herramientas que la versión gratuita no nos da. Esta ide es compatible con múltiples lenguajes de programación, tales como C++, C#, Visual Basic, .NET, F#, Python, Ruby y PHP, al igual que entornos de desarrollo web, como ASP.NET MVC, Django, etc.

Características encontradas:

• Permite crear proyectos en distintos lenguajes de programación.

• Permite editar código y ejecutarlo.

• Permite conectarse con los repositorios de git.

• Permite abrir una terminal para utilizar línea de comandos.

• Depurar código.

• Permite agregar algunos plugins.

¿Cuál es la estructura básica de soluciones y proyectos?

Para la creación de una de una aplicación se debe comenzar con un proyecto, el cual contiene todos los archivos que se compilan en un archivo ejecutable, biblioteca o sitio web. Además, contiene la configuración del compilador y otros archivos que podrían ser necesarios en diversos servicios. Visual Studio utiliza MSBuild para compilar cada proyecto en una solución y todos los proyectos contienen un archivo de proyecto de MSBuild, el cual tendrá diferentes extensiones según el tipo de proyecto. Una solución es un contenedor con uno o mas proyectos relaciones, junto con información de compilación, la configuración de ventanas de Visual Studio y archivos varios que no estén asociados a un proyecto determinado. El archivo de soluciones puede tener dos tipos de extensiones .sln y .suo, el primero organizado los proyectos, sus elementos y elementos de solución en la solución. El archivo con extensión .suo almacena la configuración de nivel de usuario y las personalizaciones, como los puntos de interrupción.

Ensamblado:

Funciones:

- **Contenedor de código:**
- MSIL -> PE -> Punto de Entrada
- **Mínima unidad de versionado que existe dentro de .NET**
- **Límite de:**
- seguridad
- tipos
- ámbito de referencia
- versión
- **Unidad de implementación**
- **Ejecución simultánea Capaz de cargar ensamblados durante ejecución**

Estructura:

- Manifiesto (metadatos)
- Metadatos de tipos
- Código MSIL
- Recursos

Metadato es un dato que describe otro dato

Manifiesto:

- Es una descripción de lo que contiene el ensamblado
- **Compuesto por:**
- Nombre
- Versión
- Lista de Archivos incluidos
- etc

Soluciones a problemas:

- Problema: infierno DLL (librerías dinámicas que ya tenían códigos nativos)
- Solución: Ensamblados

Metadatos de Tipo:

- **Tipos:**
- clase
- estructura
- enumeración
- **Miembros:**
- propiedades
- métodos
- eventos
- **Usado por:**
- compilador
- intellisense

```
18
TypeDef #2 (02000003)
-------------------------------------------------------
TypDefName: CalculatorExample.Calc (02000003)
Flags : [NotPublic] [AutoLayout] [Class]
[AnsiClass] [BeforeFieldInit] (00100001)
Extends : 01000001 [TypeRef] System.Object
Method #1 (06000003)
-------------------------------------------------------
MethodName: Add (06000003)
Flags : [Public] [HideBySig] [ReuseSlot] (00000086)
RVA : 0x00002090
ImplFlags : [IL] [Managed]
…
```

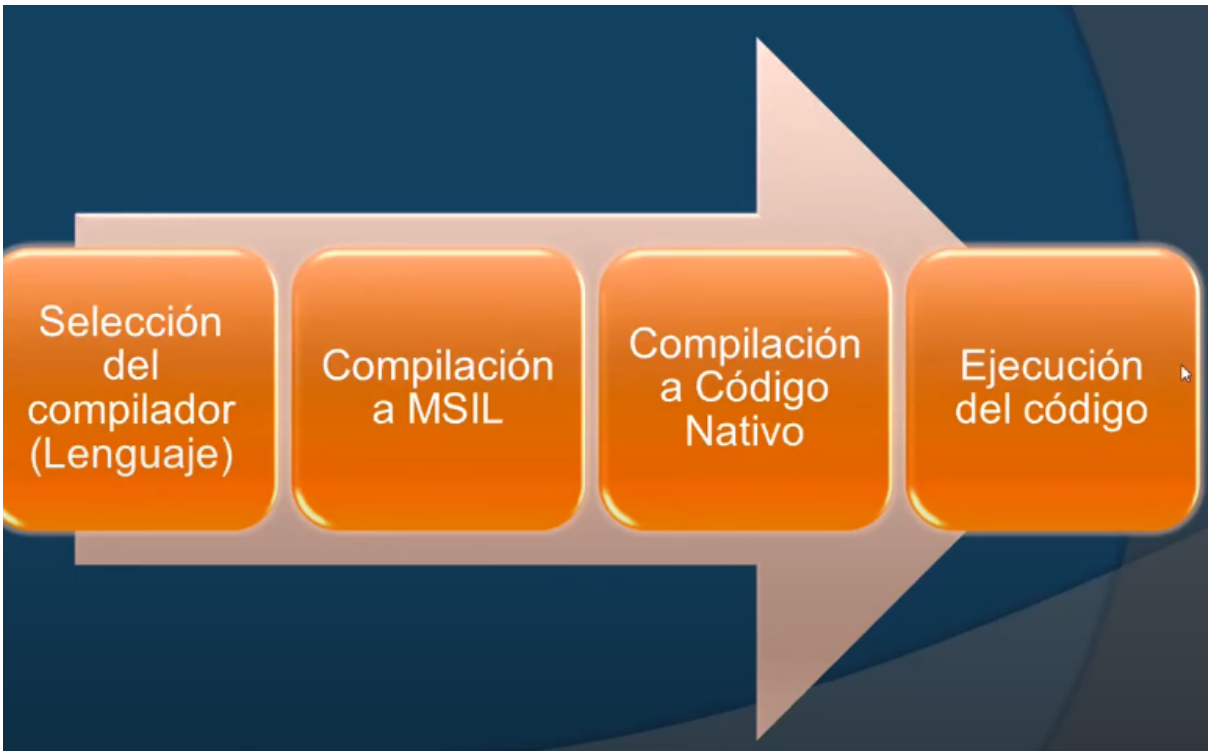
Tipos de ensamblados:

- Privados
- CFompartidos

Caché Global de Ensamblados (GAC)

- Reservorio de ensamblados
- No colocar todos los ensamblados allí
- herramienta de instalación

Ejecución de código administrado:



Selección del compilador:

- Soporte multilenguaje
- Compiladores provistos (C#, VB.NET, C++)
- Compiladores de terceros

Compilación a MSIL:

-Al compilar convertimos el código fuente en:

- MSIL (independiente al lenguaje de la cpu)
- Metadatos

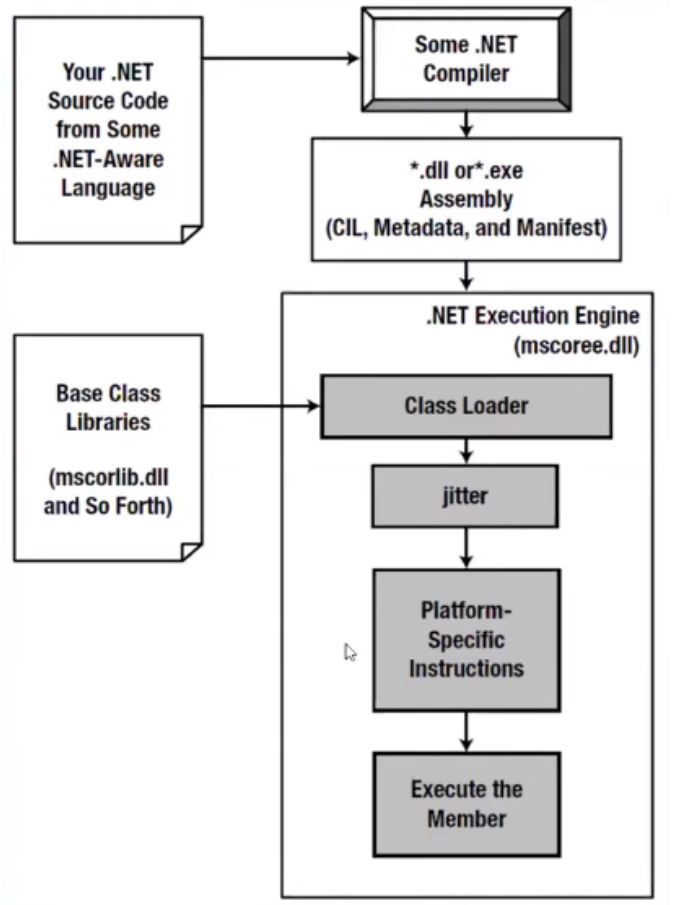
Compilación a Código Nativo:

- Antes de poder usar MSIL debe convertirse a codigo de Maquina
- Aquí actúa el compilador JIT adecuado

Ejecucion del Codigo:

- La CLR maneja la ejecución de la aplicación (administrada)
- Proceso de compilación JIT y ejecución posterior continuo

- Además de:
- recolección de elementos no utilizados
- seguridad
- interoperabilidad con código no administrado
- etc
- El corazón del CLR esta fisicamente representado por mscoree.dll (common object runtime execution engine)



-
- Tipos de variables en el assembler:
- int = int32
- float = float32
- byte = uint8
- char = char
- double = float64
- short = int16
- decimal = System.Decimal
- string = string
- long = int64

Espacios de nombre (namespaces):
- Organización de grandes cantidades de clases
- Construcción de bibliotecas de clases
- Para invocar a utilizar una clase dentro de una jerarquía namespace, utiliza puntos.

Net Standard:

.NET Standard es una especificación formal de las API de .NET que están disponibles en varias implementaciones de .NET. La motivación detrás de .NET Standard fue establecer una mayor uniformidad en el ecosistema .NET. Sin embargo, .NET 5 adopta un enfoque diferente para establecer la uniformidad, y este nuevo enfoque elimina la necesidad de .NET Standard en muchos escenarios.

.NET estándar	1.0	1.1	1.2	1.3	1.4	1,5	1,6	2.0	2.1
.NETO	5,0	5,0	5,0	5,0	5,0	5,0	5,0	5,0	5,0
.NET Core	1.0	1.0	1.0	1.0	1.0	1.0	1.0	2.0	3,0
.NET Framework [1]	4.5	4.5	4.5.1	4.6	4.6.1	4.6.1 [2]	4.6.1 [2]	4.6.1 [2]	N / A [3]
Mononucleosis infecciosa	4.6	4.6	4.6	4.6	4.6	4.6	4.6	5.4	6.4
Xamarin.iOS	10.0	10.0	10.0	10.0	10.0	10.0	10.0	10.14	12.16
Xamarin.Mac	3,0	3,0	3,0	3,0	3,0	3,0	3,0	3.8	5.16
Xamarin.Android	7.0	7.0	7.0	7.0	7.0	7.0	7.0	8.0	10.0
Plataforma universal de Windows	10.0	10.0	10.0	10.0	10.0	10.0.16299	10.0.16299	10.0.16299	TBD
Unidad	2018.1	2018.1	2018.1	2018.1	2018.1	2018.1	2018.1	2018.1	TBD

Modificadores de Acceso:

Pueden especificarse los siguientes seis niveles de accesibilidad con los modificadores de acceso:

- public: El acceso no está restringido.
- protected: El acceso está limitado a la clase contenedora o a los tipos derivados de la clase contenedora.
- internal: El acceso está limitado al ensamblado actual.
- protected internal: El acceso está limitado al ensamblado actual o a los tipos derivados de la clase contenedora.
- private: El acceso está limitado al tipo contenedor.
- private protected: El acceso está limitado a la clase contenedora o a los tipos derivados de la clase contenedora que hay en el ensamblado actual.

Access Modifiers	Inside Assembly		Outside Assembly	
	With Inheritance	With Type	With Inheritance	With Type
Public	✓	✓	✓	✓
Private	X	X	X	X
Protected	✓	X	✓	X
Internal	✓	✓	X	X
Protected Internal	✓	✓	✓	X

-

Arreglos:

- Todo arreglo hereda de la clase Array
- Permite la utilización de funcionalidad común:
- Clear
- CopyTo
- Length
- Reverse
- Sort

Formas de declarar un arreglos:

```
int[] arregloEnteros2 = { 100, 200, 300 };

int[] arregloEnteros3 = new int[] { 100, 200, 300 };

int[] arregloEnteros4 = new int[3] { 100, 200, 300 };
```

Arreglos multidimensionales

```
int[,] matriz = new int[2, 3];
// int[,] matriz = new int[2, 3] { { 1, 2, 3 }, { 2, 4, 6 } };

// Se llena.
for (int i = 0; i < 2; i++)
    for (int j = 0; j < 3; j++)
        matriz[i, j] = (i+1) * (j+1);;

//Se imprime.
for (int i = 0; i < 2; i++)
{
    for (int j = 0; j < 3; j++)
        Console.Write(matriz[i, j] + "\t");
    Console.WriteLine();
}
Console.WriteLine();
```

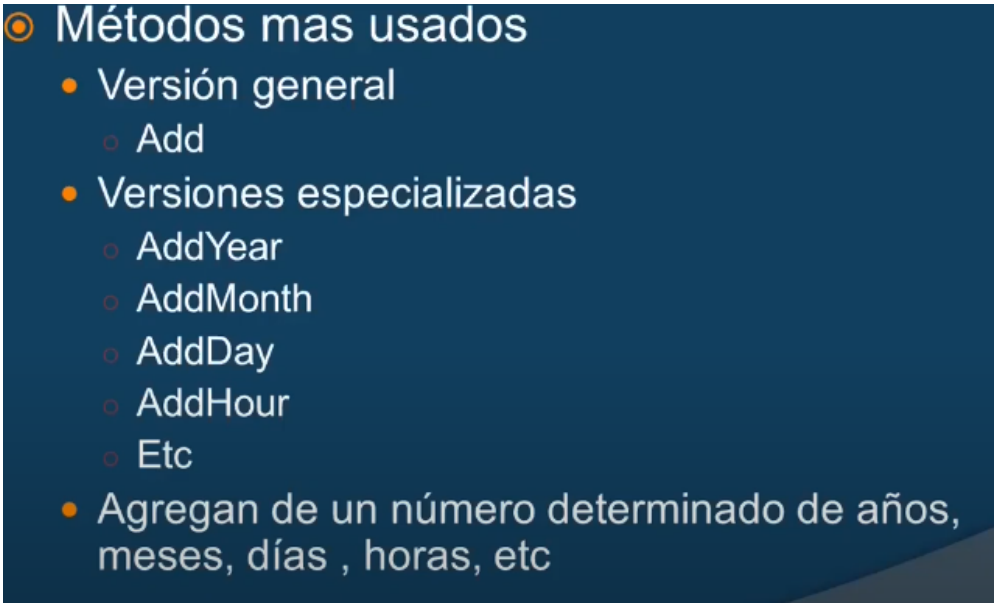
```
1       2       3
2       4       6
```

Arreglos multidimensionales (Jagged)

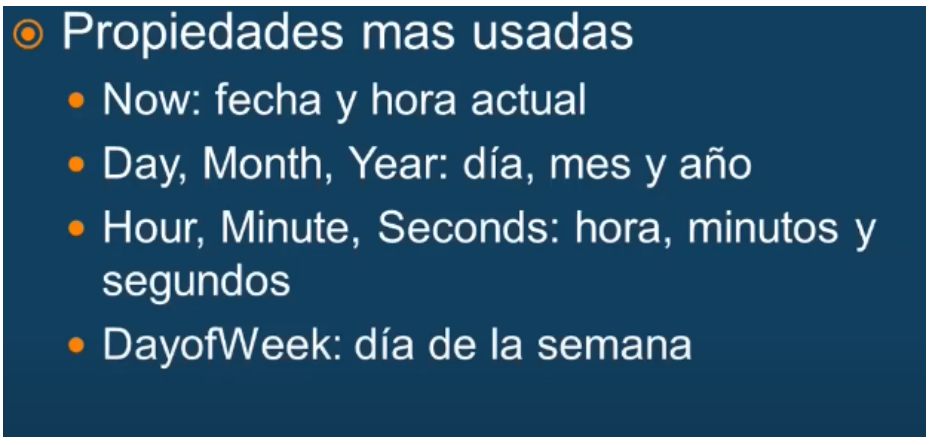
```
int[][] arrDeArr = new int[5][];

// Crear
for (int i = 0; i < arrDeArr.Length; i++)
    arrDeArr[i] = new int[i + 7];

// Imprimir
for (int i = 0; i < 5; i++)
{
    for (int j = 0; j < arrDeArr[i].Length; j++)
        Console.Write(arrDeArr[i][j] + " ");
    Console.WriteLine();
}
```

```
0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
```

DateTime
System.DateTime




```
DateTime dt = DateTime.Now;

DateTime dt2 = new DateTime(2020, 03, 14,22,5,5);

DateTime dt3 = dt.AddDays(3);

Console.WriteLine(dt);

Console.WriteLine(dt2.ToShortDateString());

Console.WriteLine(dt3.ToShortDateString());

/*      24/4/2020 17:10:18
        14/3/2020
        27/4/2020
*/
```



TimeSpan
Para hacer operaciones con fechas y hora, se recurre a:
System.TimeSpan

```
TimeSpan dif;
dif = dt3 - dt;
Console.WriteLine(dif.Days);

//3
```

Se puede mostrar en segundos, meses, etc.

Principios de POO

Abstracción: Nos permite decidir qué es lo importante y descartar aquello que es innecesario.

Encapsulamiento: El mecanismo permite ocultar los detalles de implementación de un objeto.

Herencia: Es el pilar más fuerte que asegura la reutilización de código, ya que a partir de esta característica es posible reutilizar (heredar) las características y comportamientos de una clase superior llamada clase padre, a sus clases hijas, denominadas clases derivadas.

Polimorfismo: A través de esta característica es posible definir varios métodos o comportamientos de un objeto bajo un mismo nombre, de forma tal que es posible modificar los parámetros del método, o reescribir su funcionamiento, o incrementar más funcionalidades a un método.

Herencia:
Es un mecanismo mediante el cual una clase base brinda sus miembros (propiedades, métodos, etc) a su clase derivada pudiendo esta especializarse (Menos constructores y destructores).

Constructores:
Una clase derivada no hereda los constructores de la clase padre.
Aunque no existe ningún constructor declarado invoca al de la clase sin parámetros.
Para hacer el constructor hay que hacer algo llamado Cadena de constructores:

public Constructor(parámetros) : base (parámetros)

La palabra reservada “base” se encarga de ser el constructor de la clase padre.

Invalidación o Overriding:
Es el proceso al cual se le cambia, expande o modifica la funcionalidad de un método heredado, se lo hace mediante métodos virtuales, también funciona para propiedades.

Declaración:

public virtual void MetodoPadre(); ---------> El virtual es importante

public override void MetodoHija(); --------> El override es importante

Si no se usa las palabras reservadas, se hace algo llamado “shadowing” que es ocultar la funcionalidad del método padre.

No se puede modificar los parámetros que recibe:
public override void MetodoHija(int a) ---------> int a esta mal.

Sobrecarga:
En este, no se usa las palabras reservadas y se pueden cambiar el paso de parámetros, pudiendo llamar tanto al método norma, como al sobrecargado.

public void Metodo_padre();

public void Metodo_hija(int a)

Y podra llamarse a ambos métodos que por el compilador, serán 2 métodos diferentes.