Contenidos

Tecnologías de Desarrollo de Software IDE

Unidad: 1

“Plataforma de Desarrollo”

(Última Actualización: 23 Julio 2013)

**Índice**

[Unidad 1: Plataforma y Entorno de Desarrollo](#h.8qrkdx2nclre)

[Introducción](#h.nwkf9yjbixn8)

[Plataforma de Desarrollo .Net](#h.uqzxpva3okce)

[Entorno de trabajo .Net Framework](#h.yfb2pyho8517)

[Evolución de versiones de .Net Framework](#h.plxcaoh339x5)

[Componentes Principales](#h.p3sblmbyrupg)

[Common Language Runtime (CLR)](#h.7vpeftivbs82)

[Proceso de compilación y ejecución de código administrado](#h.yjzdeu2yn6hs)

[Código en Lenguaje Intermedio (CIL o MSIL)](#h.ynafkut3e8y4)

[Ensamblados, Manifiestos y Metadatos](#h.4dpy53aoth3w)

[Desensamblador de MSIL](#h.dpgxcl9yijwh)

[Componentes del CLR](#h.ka5gya469shp)

[Infraestructura Común de Lenguajes (CLI)](#h.l85didsaawjn)

[Sistema de Tipos Comunes (CTS: Common Type System)](#h.tsgodtcn0xs4)

[Clasificación de Tipos](#h.ru8p9c1id0a8)

[Especificación Común de Lenguajes (CLS)](#h.b2d9uow7mzuk)

[Base Class Library (BCL)](#h.9rl415hdvihb)

[Organización de la BCL](#h.rs9nd2n7pq1i)

[Espacio de Nombres (Namespaces)](#h.r4nc5xlrigh1)

[Soporte a múltiples Lenguajes de Programación](#h.3l8ppunkv06w)

[Resumen de la Unidad / Capítulo](#h.5un75axlpa8l)

[Auto-evaluación](#h.ooijoj6vd8nq)

[Bibliografía#](#h.lznadjthi7kk)

*Nota*: los contenidos fueron generados a partir de diversas fuente de información, entre las que se destacan la oficial presentada en el sitio MSDN, bibliografía de la editorial oficial de la empresa y autores varios de reconocido mérito académico y/o profesional.

## 

## 

## Unidad 1: Plataforma y Entorno de Desarrollo

**IMPORTANTE:** los **contenidos** aquí desarrollados **no son necesariamente completos** en su total profundidad, por lo que **se sugiere consultar otras fuentes**, tales como la **Bibliografía** sugerida por la cátedra en el Programa Analítico y docentes.

**Objetivo(s)**

* Conocer los aspectos básicos de la plataforma de desarrollo y su funcionamiento.

**Temas**

1. *La plataforma de desarrollo .Net y su funcionamiento.*
   1. *Problemas que resuelve.*
   2. *Beneficios que ofrece.*
2. El entorno de trabajo .Net Framework y su funcionamiento.
   1. Componentes.
   2. Evolución de versiones liberadas.
3. El motor de ejecución CLR (Common Language Runtime) y su funcionamiento.
   1. Componentes y servicios ofrecidos en tiempo de ejecución.
      1. Recolección de basura (garbage collector)
      2. Resto de servicios ...
4. Arquitectura de compilación y Ejecución. El proceso de compilación JIT (Just-In-Time)
5. El Sistema de Tipos Comunes CTS (Common Type System).
   1. Categorías.
      1. Tipos por Valor y por Referencia
6. La Librería (o Biblioteca) de Clases Base BCL (Base Class Library).
7. Espacio de Nombres (namespaces)
8. Código administrado o Interpretado. Código en Lenguaje Intermedio CIL o MSIL (Microsoft Intermediate Language)
9. Ensamblados (assemblies), Manifiestos y Metadatos.
10. Especificación Común de Lenguajes CLS (Common Language Specification)
11. La Infraestructura Común del Lenguaje CLI (Common Language Infrastructure). Estandarización y licenciamiento (Norma ECMA-335 ISO/IEC 23271).
12. Soporte a múltiples lenguajes.

### 

### Introducción

La **plataforma .Net** con los componentes principales de su **Framework**, **Librería de Clases** (BCL) y **Motor de Ejecución** (CLR), permiten entender cómo funciona esta plataforma y el entorno de desarrollo sobre el cual generar y ejecutar código escrito en alguno lenguaje de programación soportado por .Net.

La **plataforma .NET** es un conjunto de tecnologías diseñado por Microsoft que permiten el desarrollo y ejecución de software, de forma independiente del lenguaje de programación, modelo de objetos, sistema operativo, infraestructura de hardware y dispositivos donde se ejecuta.

Esta combinación tecnológica la componen un entorno de trabajo, denominado **.NET Framework** y un conjunto de **lenguajes .Net** y **herramientas de programación** que posibilitan el desarrollo de aplicaciones robustas, seguras y de alto rendimiento para diferentes ambientes.

Esta plataforma sumada a las utilidades y herramientas que dispone el **Entorno de Desarrollo Integrado** (IDE) “**Visual Studio .Net**” permite que tanto programadores como otros actores que intervienen en el ciclo de vida de desarrollo del software, puedan realizar de modo integrado un rápido desarrollo de aplicaciones en cualquiera lenguaje soportado por .Net.

### Plataforma de Desarrollo .Net

La plataforma .NET está construido sobre una arquitectura abierta (estándar ECMA-335, ISO/IEC 23271), que permite generar y ejecutar tanto aplicaciones de escritorio, aplicaciones web, servicios web como aplicaciones móviles.

Uno de los objetivos de esta plataforma es simplificar el desarrollo al independizarse de la plataforma de hardware, del lenguaje de programación utilizado y de los dispositivos donde serán ejecutados.

La plataforma .NET proporciona entre otros aspectos:

* Un modelo de programación coherente e independiente del lenguaje para todas las capas o niveles de una aplicación.
* Una interoperabilidad transparente entre tecnologías.
* Una fácil migración desde tecnologías existentes.
* Un completo soporte de tecnologías de Internet independientes de la plataforma y basada en estándares, incluyendo HTTP, XML y SOAP.

Esta conformada por las siguientes áreas tecnológicas:

* Un entorno de trabajo denominado **.Net Framework**.
* Un conjunto de **Lenguajes .Net y Herramientas de desarrollo**.

### Entorno de trabajo .Net Framework

En el ámbito del [*desarrollo de software*](http://es.wikipedia.org/wiki/Desarrollo_de_software), un ***framework***[[1]](#footnote-2) o ***infraestructura digital***, es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de software concretos, con base a la cual otro proyecto de [*software*](http://es.wikipedia.org/wiki/Software) puede ser más fácilmente organizado y desarrollado. Típicamente, puede incluir soporte de [programas](http://es.wikipedia.org/wiki/Programa_(computaci%C3%B3n)), [bibliotecas](http://es.wikipedia.org/wiki/Biblioteca_(programaci%C3%B3n)), y un [lenguaje interpretado](http://es.wikipedia.org/wiki/Lenguaje_interpretado), entre otras herramientas, para así ayudar a diferentes actores que intervienen en el ciclo de vida del software y unir los diferentes componentes de un proyecto.

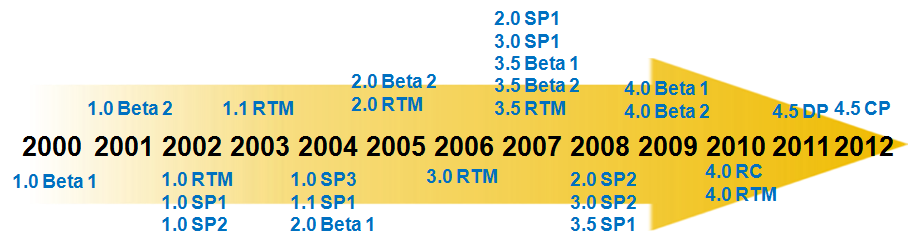
Representa una [*arquitectura de software*](http://es.wikipedia.org/wiki/Arquitectura_de_software) que modela las relaciones generales de las entidades del dominio, y provee una estructura y una especial metodología de trabajo, la cual extiende o utiliza las aplicaciones del dominio.

El **.NET Framework**[[2]](#footnote-3) es la infraestructura ofrecida por Microsoft donde aplicaciones y servicios son construidos y ejecutados, que por su naturaleza unificada permite que cualquier tipo de aplicación sea desarrollada mediante herramientas comunes haciendo la integración mucho más simple.

#### Evolución de versiones de .Net Framework

El .Net framework se encuentra en constante evolución y crecimiento por lo que la empresa va liberando con el tiempo nuevas versiones que incorporan tanto mejoras de la funcionalidad existente como nuevos desarrollos y tecnologías.

Se muestra un diagrama con las diferentes versiones que fueron surgiendo en un correlato cronológico en el tiempo.



*Figura: Evolución de versiones del .Net Framework (hasta 2012)*

La tabla siguiente correlaciona .NET Framework, CLR y las versiones de Visual Studio y proporciona un breve revisión de cada versión. Tenga en cuenta que Visual Studio proporciona compatibilidad con múltiples versiones (multi-targeting), por lo que no está limitado a la versión de .NET Framework se muestra.

|  |  |  |
| --- | --- | --- |
| .Net Framework | Incorporado con el IDE | Descripción |
| 4.5.1 | Visual Studio 2013 | Incluye mejoras de rendimiento y depuración, compatibilidad para el redireccionamiento de enlaces automático y compatibilidad expandida para las aplicaciones de la Tienda Windows. |
| 4.5 | Visual Studio 2012 | Incluía una versión actualizada de CLR 4, compatibilidad para compilar aplicaciones Tienda Windows y las actualizaciones para WPF, WCF, WF y ASP.NET. |
| 4 | Visual Studio 2010 | Presenta una nueva versión de CLR, bibliotecas de clases base expandidas y nuevas características, como Managed Extensibility Framework (MEF), Dynamic Language Runtime (DLR) y contratos de código. |
| 3.5 | Visual Studio 2008 | Se agregaron nuevas características, como los sitios web habilitados para AJAX y LINQ. La actualización SP1 incorporaba datos dinámicos y un pequeño conjunto de mejoras adicionales. |
| 3.0 | Visual Studio 2005 | Esta versión es esencialmente .NET Framework 2.0 con la incorporación de Windows Presentation Foundation (WPF), Windows Communications Foundation (WCF), Windows Workflow Foundation (WF) y CardSpace. Se actualizó con SP1 y SP2. |
| 2.0 | Visual Studio 2005 | Se incorporó una nueva versión de CLR con ampliaciones de las bibliotecas de clases base, que incluían genéricos, colecciones de genéricos y ampliaciones significativas de ASP.NET. Esta versión se actualizó con el SP1 y el SP2. |
| 1.1 | Visual Studio .NET 2003 | Incluía actualizaciones de ASP.NET y ADO.NET. Esta versión se actualizó dos veces posteriormente, con Service Pack 1 y (SP1) SP2. Esta versión incorporaba también la ejecución en paralelo, lo que permite que las aplicaciones de un solo equipo ejecuten varias versiones de CLR. |
| 1.0 | Visual Studio .NET | Contenía la primera versión de CLR y la primera versión de las bibliotecas de clases base. |

*Nota*: ver mas sobre la evolución de las versiones de Framework en el sitio MSDN “[.NET Framework Versions and Dependencies](http://www.google.com/url?q=http%3A%2F%2Fmsdn.microsoft.com%2Fen-us%2Flibrary%2Fbb822049(v%3Dvs.110).aspx&sa=D&sntz=1&usg=AFQjCNFODM7WneZbRhDz3pTGKmAGBngDmw)” (en MSDN) o en “[.NET Framework version history](http://www.google.com/url?q=http%3A%2F%2Fen.wikipedia.org%2Fwiki%2F.NET_Framework_version_history&sa=D&sntz=1&usg=AFQjCNFxeFCYDL9vbZi2pCPmwbjwLRO9oA)” (en Wikipedia)

#### Componentes Principales

Este entorno de trabajo esta formado por dos componentes principales:

* Un *motor de ejecución* llamado **Common Language Runtime** (CLR)
* Una *librería de clases* llamada **Base Class Library** (BCL)

*Nota*: más adelante en la sección referida a la BCL, se muestra gráficamente la evolución de la librería de clases base respecto al framework.

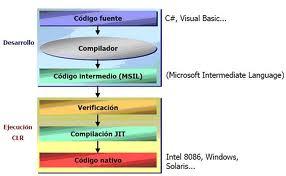
### Common Language Runtime (CLR)

El **Common Language Runtime[[3]](#footnote-4)** (**CLR**) maneja el código en tiempo de ejecución y provee una serie de servicios esenciales tales como la integración de lenguajes, seguridad, manejo de tipos seguros, manejo de hilos (thread) y manejo de memoria, entre otras ventajas que proporciona este *entorno de ejecución administrado*.

Además brinda un conjunto de herramientas y utilidades que permiten simplificar y facilitar la tarea del desarrollador, tales como herramientas de depuración de errores (debugging), manejo de excepciones y uso de funcionalidad básica y avanzada expuesta en la librería de clases común a todos los lenguajes .Net (BCL).

El CLR es el verdadero núcleo del framework de .NET, entorno de ejecución en el que se cargan las aplicaciones desarrolladas en los distintos lenguajes, ampliando el conjunto de servicios del sistema operativo. Permite integrar proyectos en distintos lenguajes soportados por la plataforma .Net, como C#, Visual Basic, C++, F#, entre otros.

El CLR es el encargado de [compilar](http://es.wikipedia.org/wiki/Compilador) una forma de *código intermedio* llamada [**Common Intermediate Language**](http://es.wikipedia.org/wiki/Common_Intermediate_Language) ([**CIL**](http://es.wikipedia.org/wiki/Common_Intermediate_Language)) o también conocido como **Microsoft Intermediate Language** ([**MSIL**](http://es.wikipedia.org/wiki/MSIL)), a [*código máquina*](http://es.wikipedia.org/wiki/C%C3%B3digo_m%C3%A1quina) *nativo* (optimizado para el sistema operativo y el hardware donde correrá la aplicación), mediante una compilación [JIT](http://es.wikipedia.org/wiki/Compilaci%C3%B3n_en_tiempo_de_ejecuci%C3%B3n) *(Just-In-Time)* [en tiempo de ejecución](http://es.wikipedia.org/wiki/Compilaci%C3%B3n_en_tiempo_de_ejecuci%C3%B3n)[[4]](#footnote-5) a medida que el programa invoca métodos. El código ejecutable obtenido se almacena en la memoria caché del ordenador, siendo recompilado de nuevo sólo en el caso de producirse algún cambio en el código fuente.



*Figura: Proceso de desarrollo y compilación en .Net*

El Entorno de Desarrollo[[5]](#footnote-6) cuenta con la asistencia de conjunto de compiladores[[6]](#footnote-7), existiendo uno para cada lenguaje .Net soportado, cada uno de los cuales es el encargado de transformar el código fuente escrito con la sintaxis propia del lenguaje utilizado, a un código en un Lenguaje Intermedio (IL) común a todos, conocido como [CIL](http://es.wikipedia.org/wiki/Common_Intermediate_Language) *(Common Intermediate Language)*, que es entendido por el CLR. Para generarlo, el compilador se basa en la especificación [**CLS**](http://es.wikipedia.org/w/index.php?title=CLS&action=edit&redlink=1) *(****Common Language Specification****)* que determina las reglas necesarias para crear el código CIL compatible con el CLR.

##### Proceso de compilación y ejecución de código administrado

El proceso de ejecución administrada incluye los pasos siguientes:

1. [**Elegir un compilador**](http://msdn.microsoft.com/es-ar/library/cds9hbbb(v=vs.80).aspx)[[7]](#footnote-8).

Para obtener los beneficios que proporciona el Common Language Runtime, se deben utilizar uno o más compiladores por cada lenguaje .Net utilizado para generar código.

1. **Compilar** el código **a** [**Lenguaje intermedio de Microsoft** (**MSIL**)](http://msdn.microsoft.com/es-ar/library/c5tkafs1(v=vs.80).aspx).

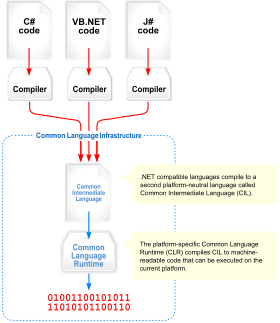
La compilación convierte el código fuente en **CIL** (o MSIL)y genera los **metadatos** requeridos.

1. [**Compilar MSIL a código nativo**](http://msdn.microsoft.com/es-ar/library/ht8ecch6(v=vs.80).aspx).

En tiempo de ejecución, un **compilador** Just-In-Time (**JIT**) **convierte CIL** **en código nativo**. Durante esta compilación, el código debe pasar un proceso de comprobación que examina el **CIL** y los metadatos para validar si el código garantiza la seguridad de tipos, que es uno de los servicios que realiza el CLR.

1. [**Ejecutar código**](http://msdn.microsoft.com/es-ar/library/8t3521k6(v=vs.80).aspx).

El **CLR** proporciona la infraestructura que permite que la ejecución tenga lugar, así como una amplia gama de servicios que se pueden utilizar durante la ejecución.



*Figura: Proceso de compilación y ejecución de código administrado*

#### Código en Lenguaje Intermedio (CIL o MSIL)

El código en Lenguaje Intermedio de Microsoft (MSIL o CIL) es un conjunto de instrucciones independiente de la CPU que se puede convertir de forma eficaz en código nativo.

MSIL incluye instrucciones para cargar, almacenar, inicializar y llamar a métodos en los objetos, así como instrucciones para operaciones lógicas y aritméticas, flujo de control, acceso directo a la memoria, control de excepciones y otras operaciones.

Antes de poder ejecutar código, se debe convertir MSIL al código específico de la CPU, mediante un proceso de compilación JIT.

EL CLR proporciona uno o varios compiladores JIT para cada arquitectura de equipo compatible, por lo que se puede compilar y ejecutar el mismo conjunto de MSIL en cualquier arquitectura compatible.

Cuando el compilador produce MSIL, también genera información adicional sobre el código. Esta información describe los tipos que aparecen en el código, incluidas las definiciones de los tipos, las firmas de los miembros de tipos, los miembros a los que se hace referencia en el código y otros datos que el motor de tiempo de ejecución utiliza en tiempo de ejecución.

El IL y los datos adicionales, conocidos como MetaData, se incluyen en un **archivo Ejecutable Portable** (**PE**), que se basa y extiende el PE de Microsoft publicado y el formato Common Object File Format (COFF) utilizado tradicionalmente para contenido ejecutable. Este formato de archivo, que contiene código MSIL o código nativo así como metadata, permite al sistema operativo reconocer imágenes de Common Language Runtime.

La presencia de metadatos junto con el Lenguaje intermedio de Microsoft (MSIL) permite crear códigos autodescriptivos, con lo cual las bibliotecas de tipos y el Lenguaje de definición de interfaces (IDL) son innecesarios.

El motor de tiempo de ejecución localiza y extrae los metadatos del archivo cuando son necesarios durante la ejecución.

#### Ensamblados, Manifiestos y Metadatos

Los componentes de código, manejados y ejecutados por el CLR son conocidos como **ensablados** (assemblies) y conforman la unidad básica funcional de un desarrollo .Net que contienen código CIL, una colección de tipos, recursos y un manifiesto, el cual es un conjunto de metadatos[[8]](#footnote-9) que describe su contenido.



*Figura: Ensamblado (assembly)*

Cada ensamblado (assembly) es almacenado como un archivo .exe or .dll y el .NET Framework los utiliza como unidad fundamental para diferentes propositos, tales como los relativos a seguridad, identificación de tipos, alcance de referencias, versionado y despliegue.

Los metadatos describen las dependencias e información de versionado asociado con el ensamblado que permite la ejecución en paralelo de múltiples versiones diferentes de un mismo ensamblado sin que esto genere conflicto[[9]](#footnote-10).

##### Desensamblador de MSIL

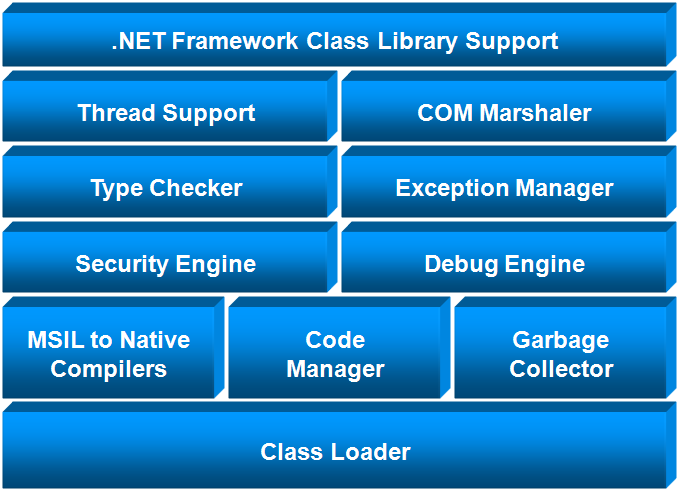
Puede utilizar la utilidad **ildasm.exe**[[10]](#footnote-11) que forma parte del SDK[[11]](#footnote-12) para examinar y ver información del ensamblado.



Si quiere conocer más sobre esta utilidad visite el sitio MSDN “[Entendiendo y utilizando Ensamblados y Espacios de Nombres en .Net](http://msdn.microsoft.com/es-ar/library/ms973231.aspx)”, “[Cómo: Ver el contenido de un ensamblado](http://msdn.microsoft.com/es-ar/library/ceats605.aspx)” o bibliografía sugerida que trata el tema.

#### Componentes del CLR

El CLR es el encargado de gestionar lo que se denomina código administrado, y para ello cuenta con un conjunto de componentes que proveen una serie de servicios automáticos al código que se ejecuta.



*Gráfico*: Componentes principales del CLR

Entre los servicios variados que cuenta se encuentran los siguientes:

* **Cargador de Clases** (Class Loader): permite cargar en memoria las clases.
* **Compilador MSIL a nativo** (MSIL to Native Compilers): transforma código intermedio de alto nivel independiente del hardware que lo ejecuta a código de máquina propio del dispositivo que lo ejecuta.
* **Administrador de Código** (Code Manager): coordina toda la operación de los distintos subsistemas del Common Language Runtime.
* **Recolector de Basura** (Garbage Colector): elimina de memoria objetos no utilizados automáticamente.
* **Motor de Seguridad** (Security Engine): administra la seguridad del código que se ejecuta.
* **Motor de** [**Depuración**](http://es.wikipedia.org/wiki/Depuraci%C3%B3n_de_programas) (Debug Engine): permite hacer un seguimiento de la ejecución del código aún cuando se utilicen lenguajes distintos.
* **Verificador de Tipos** (Type Checker): controla que las variables de la aplicación usen el área de memoria que tienen asignado.
* **Administrador de Excepciones** (Exception Manager): maneja los errores que se producen durante la ejecución del código.
* **Soporte de** [**Multiproceso**](http://es.wikipedia.org/wiki/Multiproceso) ([hilos](http://es.wikipedia.org/wiki/Hilo_de_ejecuci%C3%B3n)) (Thread Support): permite desarrollar aplicaciones que ejecuten código en forma paralela.
* **Empaquetador de COM** (COM Marshaler): coordina la comunicación con los componentes COM para que puedan ser usados por el .NET Framework.
* **Soporte de la Biblioteca de Clases Base** (BCL Support): que incluye soporte para muchas funcionalidades comunes en las aplicaciones.

#### Infraestructura Común de Lenguajes (CLI)

La Infraestructura Común de Lenguajes (**CLI** por sus siglas en inglés “[**Common Language Infrastructure**](http://es.wikipedia.org/wiki/Common_Language_Infrastructure)”) es una especificación estandarizada que describe un entorno virtual para la ejecución de aplicaciones, cuya principal característica es la de permitir que aplicaciones escritas en distintos lenguajes de alto nivel puedan luego ejecutarse en múltiples plataformas tanto de hardware como de software sin necesidad de reescribir o recompilar su código fuente.

La especificación del **CLI** está formada por un **Sistema Común de Tipos**, (**CTS** por sus siglas en inglés de Common Type System), **Metadatos** y **Especificación Común del Lenguaje** (**CLS** por sus siglas en inglés de Common Language Specification).

Debido a la publicación de la norma para la Infraestructura Común de Lenguajes (**CLI**), el desarrollo de lenguajes se facilita, por lo que el marco de trabajo .NET soporta una variada gamas de lenguajes de programación y es posible desarrollar cualquiera de los tipos de aplicaciones soportados por la plataforma con cualquiera de ellos, lo que elimina las diferencias que existían entre lo que era posible hacer con uno u otro lenguaje.

Entre algunos de los lenguajes desarrollados para el marco de trabajo .NET encontramos a C#, Visual Basic .NET, Delphi (Object Pascal), C++, F#, J#, Perl, Python, Fortran, Prolo, Cobol y PowerBuilder.

#### Sistema de Tipos Comunes (CTS: Common Type System)

En un sentido amplio, un **Tipo** de datos define un conjunto de valores y las operaciones que se pueden realizar sobre estos valores.

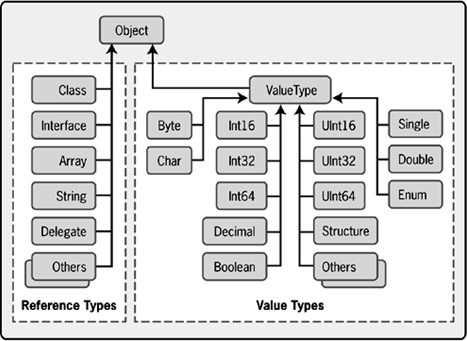
El Sistema de Tipos Común (CTS) es un estándar[[12]](#footnote-13) ([ECMA 355](http://www.ecma-international.org/publications/standards/Ecma-335.htm)) implementado en el .Net Framework, que especifica cómo se declaran, utilizan y administran los tipos en el motor de ejecución (runtime). Es también una parte importante de la compatibilidad en tiempo de ejecución con la integración entre lenguajes.

El sistema de tipos común realiza las **funciones** siguientes entre otras:

* Establece un marco de trabajo que ayuda a permitir la integración entre lenguajes, la seguridad de tipos y un alto rendimiento en la ejecución de código..
* Proporciona un modelo orientado a objetos que admite la implementación completa de muchos lenguajes de programación.
* Define reglas que deben seguir los lenguajes, lo que ayuda a garantizar que los objetos escritos en distintos lenguajes puedan interactuar unos con otros.
* Proporciona una biblioteca que contiene los tipos de datos primitivos (como Boolean, Byte, Char, Int32 y UInt64) utilizados en el desarrollo de aplicaciones.

##### Clasificación de Tipos

El CTS soporta dos categorías generales de tipos, que a su vez se dividen en subcategorías. Todos los tipos derivan del tipo base **System.Object**.



*Gráfico*: Sistema de Tipos Comunes (CTS)

**Tipos de Valor**

Los tipos de (o por) valor contienen directamente sus datos y las instancias de los tipos de valor se almacenan en la pila (stack) o se asignan en línea en una estructura. Los tipos de valor pueden ser **integrados** (o Incorporados, implementados por el motor en tiempo de ejecución), **definidos por el usuario** o **enumeraciones**. Para obtener una lista de los tipos de valor integrados, vea [Biblioteca de clases de .NET Framework](http://msdn.microsoft.com/es-ar/library/hfa3fa08(v=vs.80).aspx).

**Tipos de Referencia**

Los tipos de referencia guardan una referencia a la dirección en memoria del valor y se asignan en un sistema de cola (Heap). Los tipos de referencia pueden ser tipos autodescriptivos, punteros o de interfaz. A su vez los tipos autodescriptivos se dividen en matrices y clases. Los tipos de clase pueden ser definidas por el usuario, tipos de valor a los que se ha aplicado la conversión boxing y delegados.

Las variables que son **tipos de valor** tienen, cada una, su propia copia de los datos y, por lo tanto, las **operaciones** en una variable **no afectan a las demás**. Las variables que son **tipos de referencia** pueden hacer referencia al mismo objeto y, por lo tanto, las **operaciones** en una variable **pueden afectar al mismo objeto** al que hace referencia otra variable.

En el siguiente **ejemplo** se muestra la diferencia entre los tipos de referencia y los tipos de valor.

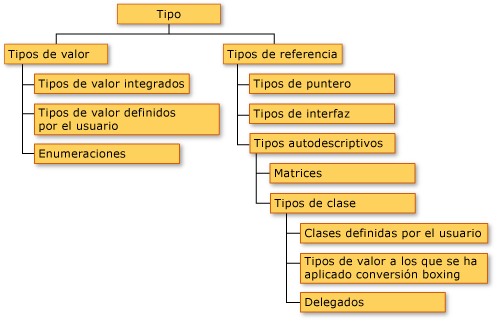
[C#]  
using System;  
class Class1  
{  
public int Value = 0;  
}  
class Test  
{  
 static void Main() {  
 int val1 = 0;  
 int val2 = val1;  
 val2 = 123;  
 Class1 ref1 = new Class1();  
 Class1 ref2 = ref1;  
 ref2.Value = 123;  
 Console.WriteLine("Values: {0}, {1}", val1, val2);  
 Console.WriteLine("Refs: {0}, {1}", ref1.Value, ref2.Value);  
 }  
}

El resultado del código anterior es el siguiente:

Values: 0, 123  
Refs: 123, 123

**Diagrama de Clasificación de Tipos**

El diagrama siguiente ilustra cómo se relacionan las diferentes categorías y subcategorias de los distintos tipos. Tenga en cuenta que las instancias de los tipos pueden ser simplemente tipos de valor o tipos autodescriptivos, aunque haya subcategorías de estos tipos.



*Gráfico*: Categorías y subcategorías del CTS

Para obtener más información sobre cada tipo, vea [tipos de valor](http://msdn.microsoft.com/es-ar/library/34yytbws(v=vs.80).aspx), [enumeraciones](http://msdn.microsoft.com/es-ar/library/6e3t9w21(v=vs.80).aspx), [clases](http://msdn.microsoft.com/es-ar/library/2s9w552e(v=vs.80).aspx), [delegados](http://msdn.microsoft.com/es-ar/library/96b1ayy4(v=vs.80).aspx), [matrices](http://msdn.microsoft.com/es-ar/library/45b76eb2(v=vs.80).aspx), [interfaces](http://msdn.microsoft.com/es-ar/library/f9846y7x(v=vs.80).aspx) y [punteros](http://msdn.microsoft.com/es-ar/library/cxx6f46y(v=vs.80).aspx). También puede consultar en MSDN el tema “[Tipos (Guía de programación de C#)](http://msdn.microsoft.com/es-es/library/ms173104(v=vs.90).aspx)”

Si desea ampliar contenidos sobre este tema puede consultar el “[*Sistema de Tipos Comunes*” en el MSDN](http://msdn.microsoft.com/es-es/library/zcx1eb1e(v=vs.80).aspx), <http://msdn.microsoft.com/es-es/library/zcx1eb1e(v=vs.80).aspx>

#### Especificación Común de Lenguajes (CLS)

Para poder interactuar completamente con otros objetos, sea cual fuere el lenguaje en que se hayan implementado, los objetos deben exponer sólo aquellas características que sean comunes para todos los lenguajes con los que deben interoperar. Por este motivo, se ha definido el **Common Language Specification** (**CLS**), que es un conjunto de características básicas del lenguaje, requeridas por la mayoría de las aplicaciones que establece los requisitos de compatibilidad.

Las reglas de CLS definen un subconjunto del [Sistema de Tipos Comunes](http://msdn.microsoft.com/es-ar/library/zcx1eb1e(v=vs.80).aspx), es decir, todas las reglas que se aplican al sistema de tipos común se aplican también a CLS, salvo que se definan reglas más estrictas en CLS.

*Nota*: puede ampliar su conocimiento sobre el tema [CLS en el sitio MSDN](http://msdn.microsoft.com/en-us/library/12a7a7h3.aspx)

### Base Class Library (BCL)

Este componente esencial del .Net Framework provee un extenso conjunto de soluciones predefinidas y unificadas para necesidades generales básicas y avanzadas de la programación de aplicaciones, totalmente orientada a objetos, organizadas en una jerarquía de clases llamada “**Base Class Library**” (**BCL**)

Las clases unificadas de .NET proporcionan un método coherente de acceso a la funcionalidad de la plataforma. Si aprende a utilizar la biblioteca de clases, observará que todas las tareas siguen la misma arquitectura uniforme. Ya no necesitará aprender ni administrar distintas arquitecturas API para escribir las aplicaciones.

Provee una colección de tipos y código orientado a objetos reusable para la mayoría de las tareas comunes que se encuentran involucradas en el desarrollo de aplicaciones, incluyendo entre otras el acceso y manejo de datos y el desarrollo de formularios Windows, Web y Servicios Web XML.

Como para dar una idea de algunas de las operaciones comunes que ofrece podemos nombrar las siguientes:

* Interacción con los dispositivos periféricos.
* Acceso a una variada gama de fuentes de datos.
* Tareas de I/O
* Manejo de datos (ADO.NET).
* Transmisión y recepción de datos por distintos medios (XML, TCP/IP)
* Administración de componentes Web que corren tanto en el servidor como en el cliente (ASP.NET).
* Cifrado de datos.
* Manejo y administración de excepciones.
* Manejo del sistema de ventanas (Windows Form).
* Herramientas de despliegue de gráficos (GDI+).
* Manipulación de archivos de imágenes.
* Herramientas de seguridad e integración con la seguridad del sistema operativo.
* Manejo de tipos de datos unificado.
* Interacción con otras aplicaciones.
* Manejo de cadenas de caracteres y expresiones regulares.
* Manejo de arreglos de datos y colecciones.
* Operaciones aritméticas.
* Aleatoriedad.
* Manipulación de fechas, zonas horarias y periodos de tiempo.
* Manejo de idiomas y Localización.
* Interacción con el API Win32 o Windows API.
* Administración de memoria.
* Generación de código.
* Auto descripción de código.
* Compilación de código

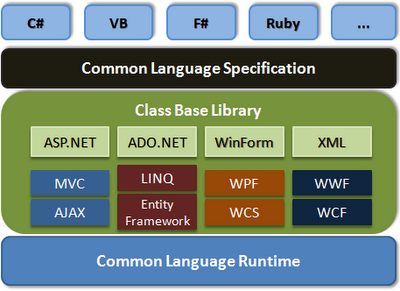
Provee un conjunto amplio de interfaces, además de clases abstractas y concretas. Las clases concretas están disponibles para ser utilizadas, pero también extenderlas, a partir de herencia. Para utilizar la funcionalidad de una interfase, se puede crear una clase que la implemente o derivar una clase de las clases del Framework que ya la implementan.

Estas librerías que son parte fundamental del .Net Framework[[13]](#footnote-14), se encuentran, al igual que este, en constante evolución y actualización con cada nueva versión liberada.

#### Organización de la BCL

La Biblioteca de Clases Base además de seguir un sistema jerárquico de clases se encuentra organizada por medio de **espacios de nombres** (namespaces) que concentra clases de funcionalidad afin.

Existen grupos de tecnologías claves incluidas en la BCL, tales como ASP.NET, Servicios Web XML, Windows Forms, ADO.NET y otras que se muestran en el siguiente gráfico.



*Figura: Arquitectura del .NET Framework.*

**ASP.NET**: permite el desarrollo de páginas Web bajo el modelo Web Forms.

**ASP.NET MVC**: permite el desarrollo web, basado en el patrón Model-View-Controller (esto fue incorporado a partir del SP1 de .NET Framework 3.5).

**AJAX**: permite una interacción asíncrona desde el Browser del cliente, con el servidor Web de nuestra aplicación.

**ADO.NET**: conjunto de clases que exponen servicios de acceso a datos que permiten conectarse a diferentes orígenes tales como base de bases relacionales (motores MS SQL Server, Oracle entre otros o expuestos mediante OLE DB y ODBC), XML y de aplicaciones y recuperar, manipular y actualizar los datos contenidos.

**LINQ** (Language Integrated Query): permite emplear sintaxis de consulta para diferentes tipos de orígenes de datos, ejemplo: colecciones, DataSets, XML.

**Entity Framework**: permite manejar un modelo de objetos para diferentes orígenes de datos, manejando la persistencia de datos (operaciones contra un origen), basado en código .NET.

**WinForm** (Windows Forms): modelo de desarrollo de aplicaciones cliente de escritorio usando Formularios Windows que cuenta con una variedad de controles.

**WPF** (Windows Presentation Foundation): permite la mejora visual de las aplicaciones a través del uso de XAML (eXtensible Application Markup Language), el WPF incorpora como mejora principal el uso de los aceleradores de gráfico (Codecs)

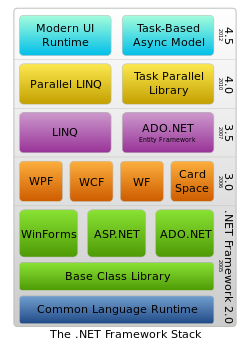
**WCS** (Windows CardSpace) permite el uso de tokens de identificación para aplicaciones que ejecuten bajo Windows XP SP2 ó superior.

**WWF** (Windows Workflow Foundation) permite implementar procesos de negocio basados en Workflows, y podemos integrarlo con otras plataformas como Microsoft Office SharePoint Server 2007 (MOSS).

**WCF** (Windows Communication Foundation) permite implementar servicios para aplicaciones distribuidas, y es la plataforma base actual de Microsoft para la construcción de soluciones basadas en SOA (Services Oriented Architecture).

**XML**: permite manipular y aprovechar funciones del estándar XML orientado principalmente a la interoperabilidad entre los sistemas.

Evaluación de tecnologías incluidas en la BCL según la evolución del .Net Framework



Fuente: [Wikipedia](http://en.wikipedia.org/wiki/File:DotNet.svg)

##### Espacio de Nombres (Namespaces)

Un namespace se utiliza para declarar un ámbito que permite organizar el código y proporciona una forma de crear tipos globalmente únicos.

Dentro de un espacio de nombres, se pueden declarar uno o varios de los siguientes tipos:

* otro espacio de nombres
* clase (class)
* interfaz
* struct
* enum
* delegado (delegate)

Aunque declare explícitamente un espacio de nombres en un archivo de código fuente (de C# por ejemplo), el compilador agrega un espacio de nombres predeterminado. Este espacio de nombres sin denominación, a veces denominado espacio de nombres global, está presente en todos los archivos. Cualquier identificador del espacio de nombres global puede utilizarse también en un espacio de nombres declarado.

De todos modos se recomienda incluir su código siempre dentro de un **namespace** utilizando como **convención** al generarlos la siguiente: *CompanyName*.*TechnologyName*.

Los espacios de nombres disponen implícitamente de un acceso público que no puede modificarse. Para obtener una descripción de los modificadores de acceso que se pueden asignar a los elementos de un espacio de nombres, consulte Modificadores de Acceso. Un espacio de nombres se puede definir en dos o más declaraciones.

El espacio de nombres principal y más básico es el ***System*** donde se encuentran la definición de los tipos de datos nativos del .NET Framework, tales como: Int32, Int64, Boolean, String, etc.

### Soporte a múltiples Lenguajes de Programación

La plataforma .Net soporta[[14]](#footnote-15) una gran variedad de lenguajes, dentro de los que se encuentran los desarrollados por el propio Microsoft, tales como C#[[15]](#footnote-16), Visual Basic .NET, C++, J# y recientemente F#[[16]](#footnote-17) (lenguaje funcional) o los desarrollados por otros fabricantes tales como, Cobol, Phyton, Delphi (Object Pascal), Perl, Python, Fortran, Prolog y PowerBuilder entre otros.

Los lenguajes de programación de la plataforma .NET, utilizan los servicios y características de .NET Framework a través de un conjunto común de clases unificadas, la BCL.

En la mayoría de las situaciones, puede utilizar de manera eficiente todos los lenguajes de programación. Sin embargo, cada lenguaje tiene sus puntos fuertes, y es recomendable comprender las características únicas para cada uno de ellos.

La elección de un lenguaje de programación depende de sus conocimientos del lenguaje y del ámbito de la aplicación que está generando. Las aplicaciones de pequeño tamaño se suelen crear utilizando un único lenguaje, y en otros casos pueden utilizarse más de uno de acuerdo a los requerimientos particulares de la solución.

## 

## 

## Resumen de la Unidad / Capítulo

**IMPORTANTE:** los **contenidos** desarrollados en esta sección **no son completos** , sino que son un mini resumen de conceptos generales de los temas de mayor importancia del capítulo a modo de un repaso general de los mismos una vez completada la lectura del mismo.

**Microsoft .Net** es una plataforma formada por un conjunto de tecnologías que permiten el desarrollo y ejecución de software bajo el .Net Framework.

El **.Net Framework** es el entorno de trabajo de la plataforma .Net, completamente orientado a objetos, cuyos componentes principales son:

* El motor de ejecución Common Language Runtime (CLR)
* La librería de clases Base Class Library (BCL)

El **Common Runtime Language** (**CRL**) es el motor de ejecución común a todos los lenguajes soportados por la plataforma .Net entre los que se destacan C#, Visual Basic .NET, C++, J# y recientemente F#[[17]](#footnote-18) (lenguaje funcional) desarrollados por Microsoft y otros tantos desarrollados por terceros tales, como Cobol, Phyton, Delphi (Object Pascal), Perl, Python, Fortran, Prolog y PowerBuilder entre otros.

Cada lenguaje de programación tiene su propio compilador e intérprete de su sintaxis propia, que debe cumplir las normas del CLS, CLI y CTS, aunque todos, luego al compilar, generan código MSIL.

El Lenguaje Intermedio de la plataforma conocido como **Microsoft Intermediate language** (**MSIL**), es el lenguaje intermedio al que compilan las aplicaciones (Assemblies) .NET. Este lenguaje intermedio es interpretado por el CRL en tiempo de ejecución.

**Common Language Specification** (**CLS**)

Define el conjunto de características que todo lenguaje compatible con .NET debería soportar.

El **Common Language Specification** (**CLS**), es la Especificación Común a todos los Lenguajes .Net, que engloban las pautas que deben cumplir estos para poder ser soportados por la plataforma y que permitir a otras compañías producir lenguajes compatibles con .NET.

Debido a la publicación de la norma para la infraestructura común de lenguajes (CLI por sus siglas en inglés), el desarrollo de lenguajes se facilita, por lo que el marco de trabajo .NET soporta ya más de 20 lenguajes de programación y es posible desarrollar cualquiera de los tipos de aplicaciones soportados en la plataforma con cualquiera de ellos, lo que elimina las diferencias que existían entre lo que era posible hacer con uno u otro lenguaje.

La **Librería** (o biblioteca) **de Clases Base** (**BCL**) de .NET, es el conjunto de clases unificada para toda la plataforma que componen el .NET framework y que permiten realizar casi cualquier tarea de una manera fácil y rápida.

Entre las conjuntos de clases que contiene la BCL se encuentran:

**ADO.NET**, que es la nueva interfaz de bases de datos. No se trata de una evolución de ADO, sino que se trata de una interfaz completamente nueva para el manejo de datos en la aplicación.

**Windows Form** cuenta con un conjunto de clases y métodos que permite el desarrollo de aplicaciones cliente de escritorio.

**ASP.NET**, es la tecnología para el desarrollo de páginas web dinámicas que reemplazó al clásico ASP, completamente integrada dentro del entorno .NET.

**Lenguajes .Net soportados**

Cualquier lenguaje que sea acorde a la Common Language Specification (CLS) puede ejecutarse sobre la CLR. En .NET Framework, Microsoft provee Visual C#, Visual Basic .Net, Visual C++, Visual J# o F#. Terceros proveer nuevos lenguajes tales como implementaciones de Phyton, Pascal, Eiffel, Cobol, Smalltalk y ADA entre otros.

### 

### 

## Auto-evaluación

1. ¿Qué es la plataforma .Net y cómo funciona?
2. ¿Qué es el Framework .Net? ¿Cuales son sus componentes principales y que función y utilidad tiene cada uno?
3. ¿Que es el motor de ejecución CLR de .Net, que función desempeña y de qué manera?
4. ¿De que se trata la Librería de Clases Base (BCL)? Nombre algunas características y componentes de la misma.
5. Explicite y explique las áreas tecnologías de funciones comunes agrupadas en la BCL.
6. ¿De que se trata el Recolector de Basura (Garbage Collector), de que forma parte, cual es su rol (o función) y cómo funciona?
7. ¿De que se trata el CTS? ¿De qué forma está organizado?
8. ¿Cual es la diferencia entre los Tipos por Referencia y por Valor? Citar ejemplos de cada uno.
9. ¿De que se trata la Seguridad de Tipos?
10. ¿De que se trata el MSIL?
11. ¿En qué consiste el proceso de compilación y ejecución del código desarrollado en .Net?
12. ¿A que se refiere el concepto de código manejado o interpretado?¿Cual es la diferencia entre código manejado y no manejado?
13. ¿De que se trata un ensamblado (assembly)? ¿Que relación tiene con MSIL (o CIL) ?
14. ¿A qué nos referimos con Metadata? Nombre algunas características de su contenido.
15. ¿De que se trata la compilación JIT (Just-In-Time)?¿Donde se lleva a cabo y cuando?
16. ¿Explique de qué se trata el CLS y relaciónelo con el CTS?
17. ¿De que se trata el CLI: Common Language Infrastructure? ¿Cuales son las partes principales de la arquitectura CLI?¿Que y quien lo define?¿Nombre un ejemplo equivalente de su implementación?
18. ¿Que es la ECMA-335, ISO/IEC 23271?¿Qué características tiene y cual es su objetivo?¿De ejemplos de su implementación?
19. ¿Qué características deben tener los lenguajes soportados por la plataforma .Net? Nombre algunos ejemplos de lenguajes soportados.
20. ¿Los lenguajes .Net son desarrollados por un único fabricante?¿Cada lenguaje tiene su propio compilador?¿El código generado por el compilador a que lenguaje lo traduce?

## Bibliografía[[18]](#footnote-19)

**IMPORTANTE**: Se sugiere buscar bibliografía adicional a la citada aquí, en el buscador disponible en [Biblioteca UTN Rosario](http://www.frro.utn.edu.ar/biblioteca.php).

BALENA, Francesco “[Programación avanzada con Microsoft Visual Basic.Net.](http://www.frro.utn.edu.ar/biblioteca_libro.php?fac=8&biblio=8&especbiblio=87&ubica=7471&palabra=Microsoft%20Visual%20Basic.Net&orden=&seccion=14&area=40)”, Madrid, McGraw-Hill, 2003.

ISBN: 8448137159

Capítulo 8 “Tipos básicos de .Net framework” y 14 “Ensamblados y appDomains”

Ejemplares Disponibles: [en biblioteca](http://www.frro.utn.edu.ar/biblioteca_libro.php?fac=8&biblio=8&especbiblio=87&ubica=7471&palabra=visual%20basic&orden=&seccion=14&area=40)

FOXALL, James “[Visual C# 2005.](http://www.frro.utn.edu.ar/biblioteca_libro.php?fac=8&biblio=8&especbiblio=5&ubica=8553&palabra=C#&orden=&seccion=14&area=40)”, Madrid, Anaya, 2006.

ISBN: 9788441521216

Capítulo 24 “Visión general de .NET framework”

Disponibilidad: [en biblioteca](http://www.frro.utn.edu.ar/biblioteca_libro.php?fac=8&biblio=8&especbiblio=5&ubica=8553&palabra=C#&orden=&seccion=14&area=40)

HILLAR, Gastón, “[Visual Basic 2005 y .Net 2.0.](http://www.frro.utn.edu.ar/biblioteca_libro.php?fac=8&biblio=8&especbiblio=5&ubica=28614&palabra=C#&orden=&seccion=14&area=40)” Buenos Aires, Hasa, 2007.

ISBN: 9789505282760

Capítulo 1 “Introducción al Entorno de Desarrollo .Net”

Disponibilidad: [en biblioteca](http://www.frro.utn.edu.ar/biblioteca_libro.php?fac=8&biblio=8&especbiblio=5&ubica=8635&palabra=visual%20basic&orden=&seccion=14&area=40)

MACKENZIE, Duncan y SHARKEY, Kent. “[Aprendiendo visual Basic.Net en 21 lecciones avanzadas.](http://www.frro.utn.edu.ar/biblioteca_libro.php?fac=8&biblio=8&especbiblio=87&ubica=7411&palabra=.net&orden=&seccion=14&area=40)” México, Pearson, 2003.

ISBN: 970260379x

Capítulo 8 “Presentación del .NET Framework” y 17 “Uso del .NET Framework”

Disponibilidad: [en biblioteca](http://www.frro.utn.edu.ar/biblioteca_libro.php?fac=8&biblio=8&especbiblio=87&ubica=7411&palabra=visual%20basic&orden=&seccion=14&area=40http://www.frro.utn.edu.ar/biblioteca_libro.php?fac=8&biblio=8&especbiblio=87&ubica=7471&palabra=visual%20basic&orden=&seccion=14&area=40)

SHARP, John. “[Visual C# 2008: paso a paso.](http://www.frro.utn.edu.ar/biblioteca_libro.php?fac=8&biblio=8&especbiblio=5&ubica=8551&palabra=C#&orden=&seccion=14&area=40)” Madrid, Anaya, 2008.

ISBN: 9788441524491

Capítulo 1 “Bienvenido a C#”

Disponibilidad: [en biblioteca](http://www.frro.utn.edu.ar/biblioteca_libro.php?fac=8&biblio=8&especbiblio=5&ubica=8551&palabra=Visual%20C#%202008&orden=&seccion=14&area=40)

**Fuentes Digitales**

+ Microsoft Developer Network (MSDN) On Line: [http://msdn.microsoft.com](http://msdn.microsoft.com/es-es/library/zcx1eb1e(v=vs.80).aspx)

Framework .Net: <http://msdn.microsoft.com/en-us/library/a4t23ktk.aspx>

Versiones del .Net Framework: <http://msdn.microsoft.com/es-es/library/bb822049.aspx>

Common Language Runtime: <http://msdn.microsoft.com/es-ar/library/8bs2ecf4(v=vs.80).aspx>

Metadatos: <http://msdn.microsoft.com/es-es/library/4y7k7c6k(v=vs.80).aspx>

Ensamblados y Espacios de nombres: <http://msdn.microsoft.com/en-us/library/ms973231.aspx>

Lenguajes de programacion .Net: <http://msdn.microsoft.com/es-es/library/aa292164(v=vs.71).aspx>

F#: <http://msdn.microsoft.com/es-es/magazine/cc164244.aspx>

+ Articulo “Introducción a .NET” en sitio web DVJoker

<http://www.devjoker.com/contenidos/Articulos/25/Introducci%C3%B3n-a-NET.aspx>

+ Wikipedia, conceptos genéricos y específicos:

Microsoft .Net: <http://es.wikipedia.org/wiki/Microsoft_.NET>

Framework .Net: <http://en.wikipedia.org/wiki/.NET_Framework>

Common Language Runtime: <http://es.wikipedia.org/wiki/Common_Language_Runtime>

Compilación en tiempo de ejecución: <http://es.wikipedia.org/wiki/Compilaci%C3%B3n_en_tiempo_de_ejecuci%C3%B3n>

**Historial de Versiones**

|  |  |  |  |
| --- | --- | --- | --- |
| **Versión** | **Fecha** | **Autor** | **Detalle** |
| 1.0 | 12/03/2012 | Ezequiel Porta | Versión inicial de la Unidad 1 “Plataforma y Entorno de Desarrollo” |
| 1.1 | 21/03/2012 | Ezequiel Porta | Ultimos agregados y modificaciones y generación de una primer versión publicable, previa consulta y revisión de docentes de la cátedra. |
| 1.2 | 23/07/2013 | Ezequiel Porta | Correcciones menores y agregado de gráfico sobre tecnologías incluidas como evolución del .Net Framework |
|  |  |  |  |

1. En términos generales según el concepto de [Framework en Wikipedia](http://www.google.com/url?q=http%3A%2F%2Fes.wikipedia.org%2Fwiki%2FFramework&sa=D&sntz=1&usg=AFQjCNEgCUV53UczQSASntNZgxKhlznghg), es un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular, que sirve como referencia para enfrentar y resolver problemas de índole similar. [↑](#footnote-ref-2)
2. El .NET Framework se incluye en versiones del sistema operativo Windows Server 2008 y Windows Vista en adelante. De igual manera puede ser instalada en Windows XP, y en la familia de sistemas operativos Windows Server 2003. Una versión "reducida" de .NET Framework está disponible para la plataforma Windows Mobile y Phone, en teléfonos inteligentes. Existen implementaciones en otros sistemas operativos como por ejemplo el implementado por el [Proyecto Mono](http://www.mono-project.com/.NET_Framework_Architecture) para Linux. [↑](#footnote-ref-3)
3. Conocido como [CLR](http://msdn.microsoft.com/en-us/library/8bs2ecf4(v=vs.80).aspx) por sus siglas en ingles, hacen referencia al **motor de ejecución** o “**runtime**” . [↑](#footnote-ref-4)
4. Un [compilador en tiempo de ejecución](http://es.wikipedia.org/wiki/Compilaci%C3%B3n_en_tiempo_de_ejecuci%C3%B3n), es aquel que utiliza una técnica de compilación dinámica para mejorar el rendimiento de sistemas de programación. [↑](#footnote-ref-5)
5. Llamado **IDE** por sus siglas en inglés “Integrated Development Environment”, tal como **Visual Studio .Net** [↑](#footnote-ref-6)
6. Cada lenguaje .Net tiene su propio compilador. [↑](#footnote-ref-7)
7. Considerando el uso del compilador del lenguaje específico por línea de comandos. Por ejemplo el compilador del lenguaje C# es [csc.exe](http://msdn.microsoft.com/es-es/library/78f4aasd.aspx). Si se utiliza un entorno de desarrollo integrado (IDE) este paso y el siguiente lo realiza el mismo entorno de manera automática. [↑](#footnote-ref-8)
8. El concepto de “**metadato**” (o **metadata** en inglés) se refiere a un conjunto de datos que describen o explican otro conjunto de datos. Ver ampliación del concepto de [metadato en wikipedia](http://en.wikipedia.org/wiki/Metadata)  [↑](#footnote-ref-9)
9. En relación al desarrollo con versiones anteriores a .Net donde se generaban grandes inconvenientes con lo que se conocía como “**el infierno de las dll**” [↑](#footnote-ref-10)
10. Si desea saber donde encontrar la utilidad [ILDASM.exe](http://msdn.microsoft.com/es-ar/library/f7dy01k1.aspx) vea esta entrada del [Blog MSDN](http://blogs.msdn.com/b/lucian/archive/2008/11/15/where-are-the-sdk-tools-where-is-ildasm.aspx) [↑](#footnote-ref-11)
11. Un SDK (Software Development Kit - Kit de Desarrollo de Software) es un conjunto de herramientas y programas de desarrollo que permite crear aplicaciones para un determinado paquete de software, estructura de software, plataforma de hardware, sistema de computadora, consulta de videojuego, sistema operativo o similar. Pueden incluir herramientas de debugger, códigos de ejemplos, documentación, y muchas veces un entorno de programación IDE. [↑](#footnote-ref-12)
12. La especificación para el CTS se encuentra en el **estándar** [**ECMA 355**](http://www.ecma-international.org/publications/standards/Ecma-335.htm) que define la arquitectura de la “Infraestructura Común del Lenguaje" ([**CLI**](http://msdn.microsoft.com/en-us/netframework/aa569283): Common Language Infrastructure) y Microsoft. NET es una implementación de la norma. Otro estándar es el ECMA 334 que define la Especificación del Lenguaje **C#**. [↑](#footnote-ref-13)
13. Vea el conjunto de espacios de nombres abarcados por la [Librería del .Net Framework en el sitio MSDN](http://msdn.microsoft.com/en-us/library/gg145045(v=VS.100).aspx) [↑](#footnote-ref-14)
14. Puede ver un detalle de los “[Lenguajes de Progrmacion](http://msdn.microsoft.com/es-es/library/aa292164(v=vs.71).aspx)” .Net en el sitio del MSDN [↑](#footnote-ref-15)
15. Lenguaje totalmente nuevo con la aparición de .Net cuya sintaxis tiene muchas similitudes a C con una plena implementación del Paradigmas Orientado a Objetos (POO). [↑](#footnote-ref-16)
16. [Lenguaje funcional](http://msdn.microsoft.com/es-es/magazine/cc164244.aspx) fue creado por Don Syme del equipo de investigación de Microsoft. [↑](#footnote-ref-17)
17. [Lenguaje funcional](http://msdn.microsoft.com/es-es/magazine/cc164244.aspx) fue creado por Don Syme del equipo de investigación de Microsoft. [↑](#footnote-ref-18)
18. La Bibliografía está organizada en una primer sección de libros impresos, ordenados alfabéticamente por apellido del autor, muchos de ellos disponibles en Biblioteca de UTN Rosario en cuyo caso aparecerá un enlace a los datos de la publicación que figuran en línea en el buscador del sitio web de la biblioteca. Seguido se detallan aquellos recursos digitales ya sean otros libros disponibles en dicho formato que podrá encontrar en el aula virtual o de alguna otra fuente de información como pueden ser los libros en línea del MSDN del sitio oficial de Microsoft o sitios de empresas o profesionales que poseen contenidos de valor. [↑](#footnote-ref-19)