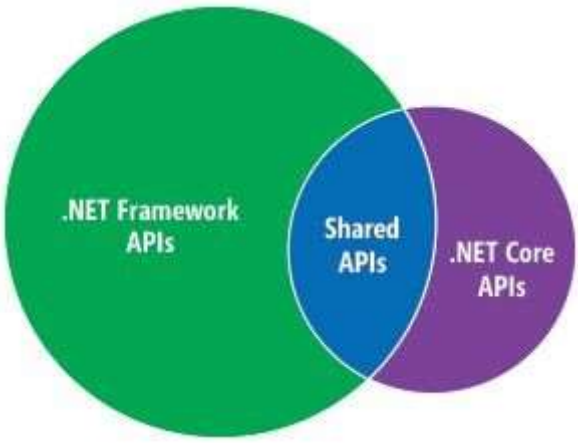
INTRODUCCIÓN A .NET

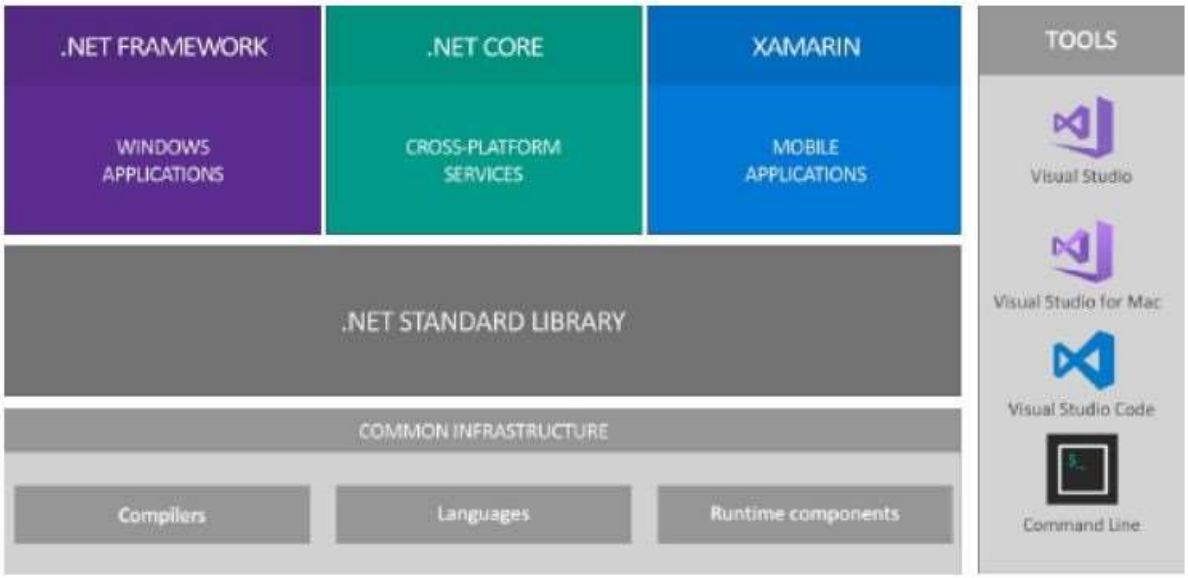
Una aplicación de .NET se desarrolla y se ejecuta en una o varias implementaciones de .NET. Las implementaciones de .NET incluyen estos componentes: **.NET Framework**, **.NET Core** y **Mono**. Hay una especificación de API (Application Programming Interface, que es un conjunto de librerías de código), común a todas las implementaciones de .NET que se denomina **.NET Standard**. Veremos una breve introducción a cada uno de estos conceptos.

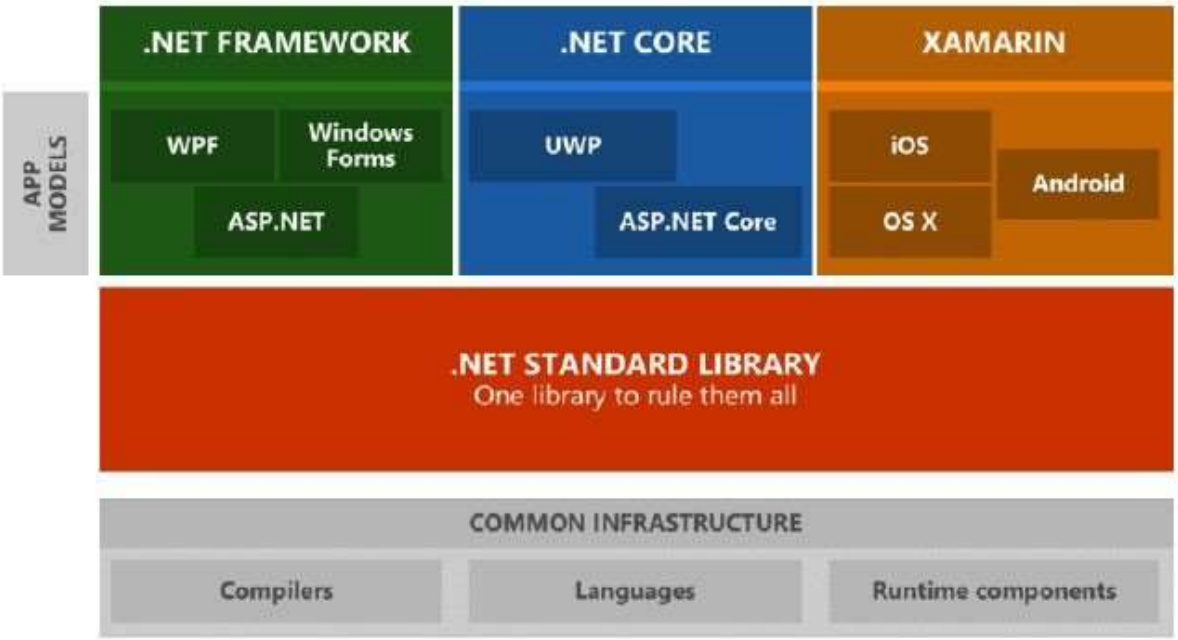


.NET Standard

.NET Standard es un conjunto de API que se implementa mediante la biblioteca de **clases base** de una implementación de .NET. Más formalmente, es una especificación de API de .NET que constituye un **conjunto uniforme de contratos** contra los que se compila el código. Estos contratos se implementan en cada implementación de .NET. Esto permite la portabilidad entre diferentes implementaciones de .NET, de forma que el código se puede ejecutar en cualquier parte; por ejemplo, esto logra que la implementación .NET Core sea multiplataforma.

.NET Standard es también una plataforma de destino. Si el código tiene como destino una versión de .NET Standard, se puede ejecutar en cualquier implementación de .NET que sea compatible con esa versión de .NET Standard.





Implementaciones de .NET

Cada implementación de .NET incluye los siguientes componentes, luego profundizaremos conceptualmente en algunos de ellos:

- Uno o varios entornos de ejecución, por ejemplo:
    - **CLR** para .NET Framework (Common Language Runtime) o **CoreCLR** y **CoreRT** para .NET Core (.NET Core Common Language Runtime y .NET Core Runtime)

- Una biblioteca de clases que implementa .NET Standard y puede implementar API adicionales, por ejemplo:
    - Biblioteca de clases base de .NET Framework
    - Biblioteca de clases base de .NET Core (.NET Core Base Class Llibrary)

- Opcionalmente, uno o varios marcos de trabajo de la aplicación, por ejemplo:
    - ASP.NET para .NET Framework y para .NET Core o Windows Forms para .NET Framework o Windows Presentation Foundation (WPF) para .NET Framework.

- Opcionalmente, herramientas de desarrollo. Algunas herramientas de desarrollo se comparten entre varias implementaciones.

Hay cuatro implementaciones principales de .NET que Microsoft desarrolla y mantiene activamente: .NET Core, .NET Framework, Mono y UWP.

.NET Core

.NET Core es una implementación multiplataforma de .NET diseñada para aplicaciones instaladas tanto en servidores propios como servidores en la nube a gran escala. .NET Core puede ejecutarse en entornos Windows, macOS y Linux. Implementa .NET Standard, de forma que cualquier código que tenga como destino .NET Standard se pueda ejecutar en .NET Core. ASP.NET Core se ejecuta en .NET Core.

.NET Framework

.NET Framework es la implementación original de .NET que existe desde 2002. Es el mismo entorno que los desarrolladores de .NET han usado siempre. Las versiones 4.5 y posteriores implementan .NET Standard, de forma que el código que tiene como destino .NET Standard se puede ejecutar en esas versiones de .NET Framework. Contiene API específicas de Windows adicionales, como la API para el desarrollo de aplicaciones de escritorio con Windows Forms y WPF. .NET Framework está optimizado para crear aplicaciones de escritorio de Windows.

Mono

Mono es una implementación de .NET que se usa principalmente cuando se requiere un entorno de ejecución a pequeña escala. Es el entorno de ejecución que se usa para aplicaciones de **Xamarin** en los sistemas operativos para pequeños dispositivos **Android**, **Mac**, **iOS**, **tvOS** y **watchOS**. Mono también se puede usar para desarrollar juegos creados con el motor de **Unity**.

Admite todas las versiones de .NET Standard publicadas actualmente.

Históricamente, Mono implementaba mucha funcionalidad de la API de .NET Framework y emulaba algunas de las funciones más populares en Unix. A veces, se usa para ejecutar aplicaciones de .NET que se basan en estas capacidades en Unix.

Mono se suele usar con un compilador Just-In-Time, pero también incluye un compilador estático completo (Ahead Of Time compilation) que se usa en plataformas como iOS.

Plataforma universal de Windows (UWP)

UWP (Universal Windows Platform) es una implementación de .NET que se usa para compilar aplicaciones Windows modernas y táctiles y software para Internet de las cosas (**IoT**). Se ha diseñado para unificar los diferentes tipos de dispositivos de destino, incluidos PCs, tablets, phablets, teléfonos móviles e incluso la consola Xbox. UWP proporciona muchos servicios, como una tienda de aplicaciones centralizada, un entorno de ejecución (AppContainer) y un conjunto de API de Windows para usar en lugar de Win32 (WinRT).Las aplicaciones pueden escribirse en C++, C#, VB.NET y JavaScript. Al usar C# y VB.NET, usa la API de .NET Core.

Entornos de tiempo de ejecución .NET

Un entorno de ejecución (Runtime) es el entorno de ejecución de un programa administrado. El sistema operativo forma parte del entorno de ejecución de una aplicación dado que una aplicación se ejecuta en un sistema operativo, pero aparte está el entorno de ejecución .NET. Estos son algunos ejemplos de los entornos de ejecución .NET:

- Common Language Runtime (**CLR**) para .NET Framework
- Core Common Language Runtime (**CoreCLR**) para .NET Core
- **.NET Native** para la Plataforma universal de Windows
- El entorno de ejecución **Mono** para **Xamarin.iOS**, **Xamarin.Android**, **Xamarin.Mac** y el marco de escritorio de **Mono**

Herramientas de .NET e infraestructura común

Tiene acceso a un amplio conjunto de herramientas y componentes de infraestructura que funcionan con todas las implementaciones de .NET. Se incluyen, entre otros:

- Los lenguajes .NET y sus compiladores/intérpretes (C#, F#, VB.Net, R, Javascript, Phyton, TypeScript)
- El sistema de proyectos de .NET (basado en archivos *.csproj*, *.vbproj* y *.fsproj*)
- MSBuild, el motor de compilación usado para compilar proyectos
- NuGet, administrador de paquetes de Microsoft para .NET (también disponible desde https://www.nuget.org/)
- Herramientas de organización de compilación de código abierto, como CAKE y FAKE

INTRODUCCIÓN A .NET FRAMEWORK

¿Qué es .NET Framework?

.NET Framework es un entorno de ejecución administrado que proporciona diversos servicios a las aplicaciones en ejecución. Consta de dos componentes principales: **Common**

Language Runtime (CLR), que es el motor de ejecución que controla las aplicaciones en ejecución, y la **biblioteca de clases de .NET Framework**, que proporciona una biblioteca de código probado y reutilizable al que pueden llamar los desarrolladores desde sus propias aplicaciones. Los servicios que ofrece .NET Framework a las aplicaciones en ejecución son los siguientes:

- **Administración de la memoria**. En muchos lenguajes de programación, los programadores son responsables de asignar y liberar memoria y de administrar la vida útil de los objetos. En las aplicaciones de .NET Framework, CLR proporciona estos servicios en nombre de la aplicación.

- **Sistema de tipos comunes**. En los lenguajes de programación tradicionales, el compilador define los tipos básicos, lo que complica la interoperabilidad entre lenguajes. En .NET Framework, los tipos básicos los define el sistema de tipos de .NET y son comunes a todos los lenguajes que tienen como destino .NET. Esto provee un "sistema unificado de tipos de dato".

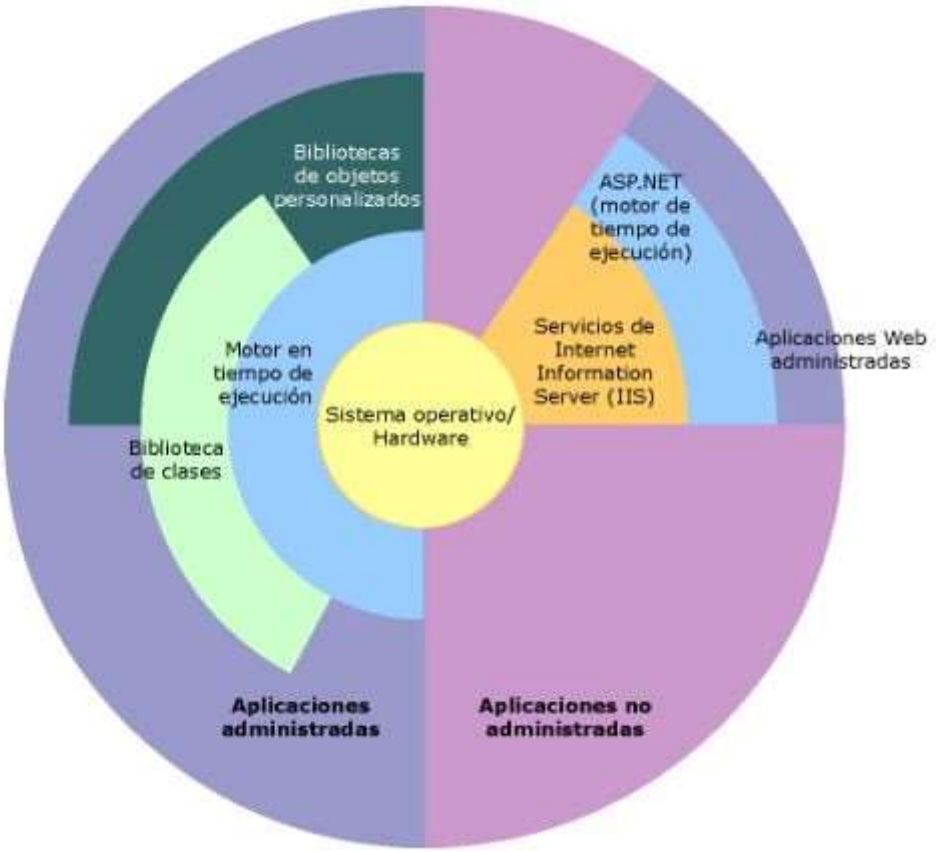
- **Biblioteca de clases extensa**. En lugar de tener que escribir cantidades extensas de código para controlar operaciones comunes de programación de bajo nivel, los programadores pueden usar una biblioteca de tipos accesible en todo momento y sus miembros desde la biblioteca de clases de .NET Framework.

- **Marcos y tecnologías de desarrollo**. .NET Framework incluye bibliotecas para determinadas áreas de desarrollo de aplicaciones, como ASP.NET para aplicaciones web, ADO.NET para el acceso a los datos y Windows Communication Foundation para las aplicaciones orientadas a servicios.

- **Interoperabilidad de lenguajes**. Los compiladores de lenguajes cuya plataforma de destino es .NET Framework emiten un código intermedio denominado Lenguaje intermedio común (CIL – Common Intermediate Language), que, a su vez, se compila en tiempo de ejecución a través de Common Language Runtime. Con esta característica, las rutinas escritas en un lenguaje están accesibles a otros lenguajes, y los programadores pueden centrarse en crear aplicaciones en su lenguaje preferido dado que luego todo compila a código común.

- **Compatibilidad de versiones**. Con raras excepciones, las aplicaciones que se desarrollan con una versión determinada de .NET Framework se pueden ejecutar sin modificaciones en una versión posterior.
- **Ejecución en paralelo**. .NET Framework ayuda a resolver conflictos entre versiones y permite que varias versiones de Common Language Runtime coexistan en el mismo equipo. Esto significa que también pueden coexistir varias versiones de las aplicaciones, y que una aplicación se puede ejecutar en la versión de .NET Framework con la que se compiló.

- **Compatibilidad con múltiples versiones** (multi-targeting). Al establecer .NET Standard como destino, los desarrolladores pueden crear bibliotecas de clases que funcionan en varias plataformas de .NET Framework compatibles con esa versión del estándar. Por ejemplo, las bibliotecas que tienen .NET Standard 2.0 como destino pueden usarlas las aplicaciones que tienen como destino .NET Framework 4.6.1, .NET Core 2.0 y UWP 10.0.16299.



NET Framework consta de dos componentes principales: Common Language Runtime y la biblioteca de clases de .NET Framework. Common Language Runtime es el fundamento de .NET Framework.

Common Language Runtime

El motor en tiempo de ejecución se puede considerar como un agente que administra el código en tiempo de ejecución y proporciona servicios centrales, como la administración de memoria, la administración de subprocesos y la comunicación remota, al tiempo que aplica una seguridad estricta a los tipos y otras formas de especificación del código que promueven su seguridad y solidez. De hecho, el concepto de administración de código es un principio fundamental del motor en tiempo de ejecución. El código destinado al motor en tiempo de ejecución se denomina **código administrado**, a diferencia del resto de código, que se conoce como **código no administrado**.

Biblioteca de Clases de Net Framework

La biblioteca de clases es una colección completa orientada a objetos de tipos reutilizables que se pueden emplear para desarrollar aplicaciones que abarcan desde las tradicionales que tienen interfaz gráfica de usuario (GUI), o de línea de comandos y hasta las aplicaciones basadas en las innovaciones más recientes.

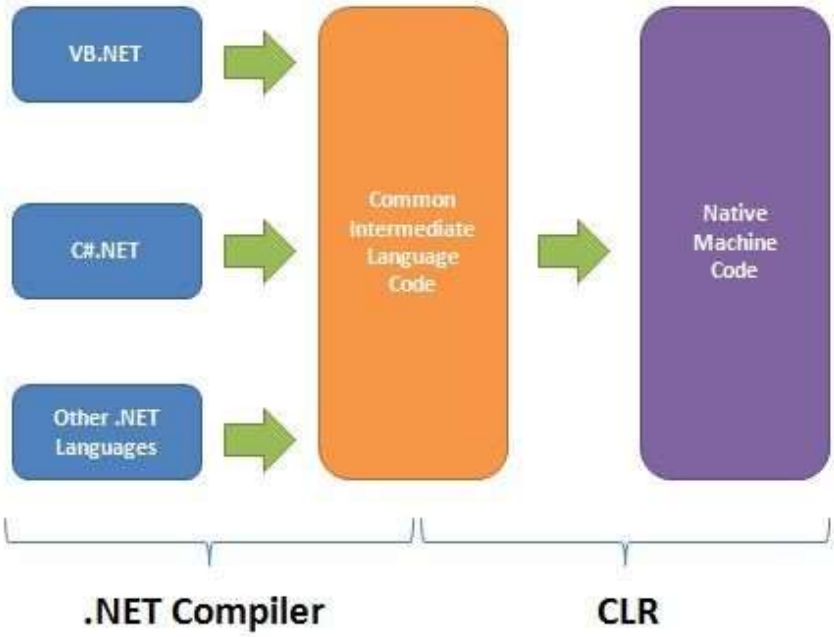
Compilador “Just in Time” y Microsoft Intermediate Language (MSIL)

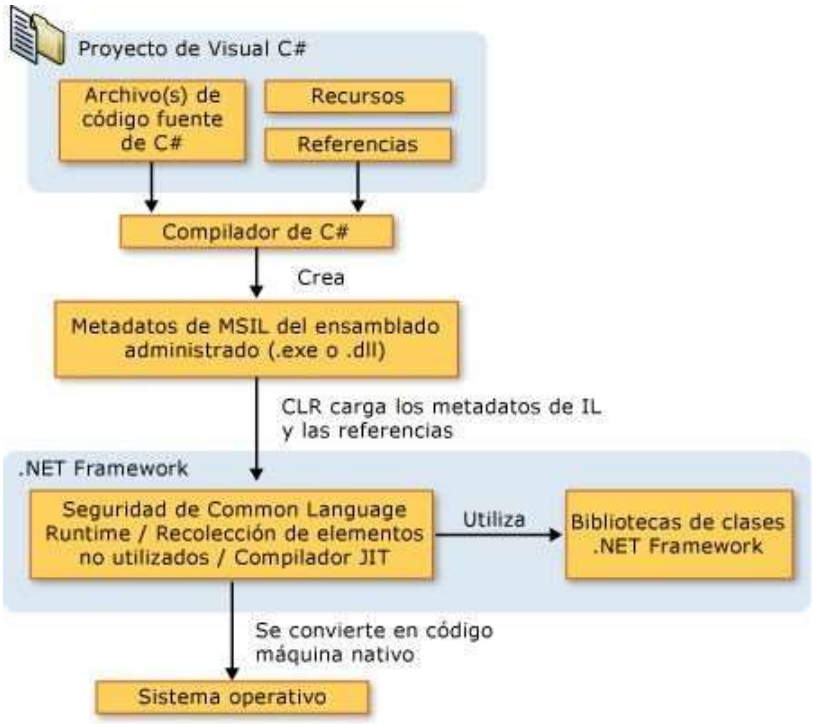
El **JIT** (Just in time) es otro programa de .net framework, es el encargado de convertir el código intermedio **MSIL**, en código de nuestro sistema. MSIL cuya sigla significa Microsoft Intermediate Language es un lenguaje intermedio, no es el lenguaje de máquina nativo (assembler) que luego se termina de compilar en la computadora/dispositivo destino donde se ejecute la aplicación.

Cuando se compila a **código administrado**, el compilador convierte el **código fuente** en Lengua intermedio de Microsoft (MSIL), que es un conjunto de instrucciones independiente de la CPU que se pueden 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 de la aplicación, se debe convertir el MSIL a código específico de la CPU, normalmente mediante un compilador Just-In-Time (JIT). Common Language Runtime 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 metadatos. Los metadatos describen 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 en tiempo de ejecución utiliza en tiempo de ejecución. El lenguaje intermedio de Microsoft (MSIL) y los metadatos 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 metadatos, 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 en tiempo de ejecución localiza y extrae los metadatos del archivo cuando son necesarios durante la ejecución.

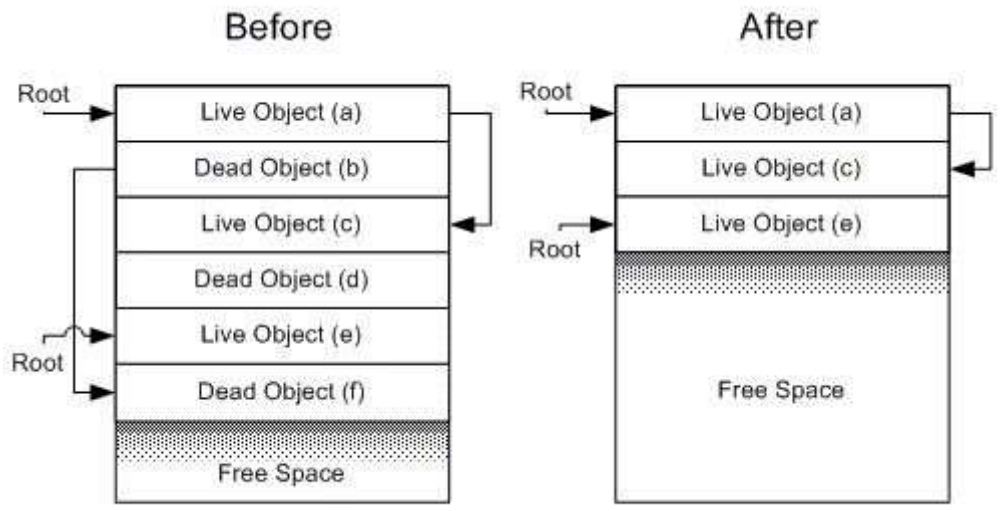




Garbage Collector - GC o recolector de basura

El objetivo del GC es proporcionar una capa de abstracción para los desarrolladores en cuestiones de manejo de memoria. Esto introduce una gran ventaja sobre otros lenguajes de programación en los que el desarrollador se tiene ocupar por completo de esta tarea, es decir de liberar la memoria cuando variables objeto ya no se utilizan en el programa. Esto lo realiza

óptimamente el GC. Escribir manualmente código que maneje correctamente la memoria en todas las situaciones es complejo, y las posibilidades de introducir errores graves en la aplicación son múltiples, por ejemplo, si no estuviera GC podría provocar: corrupción del heap, corrupción del stack, pérdida de memoria, fragmentación de memoria etc.



INTRODUCCIÓN A .NET CORE

¿Qué es .NET Core?

.NET Core es una plataforma de desarrollo de uso general de código abierto de cuyo mantenimiento se encargan Microsoft y la comunidad .NET en GitHub. Es multiplataforma, admite Windows, macOS y Linux y puede usarse para compilar aplicaciones en la nube y de Internet de las cosas (Internet of Things).

.NET Core tiene las siguientes características:

- **Multiplataforma:** se ejecuta en los sistemas operativos Windows, macOS y Linux.
- **Coherente en todas las arquitecturas:** se ejecuta el código con el mismo comportamiento en varias arquitecturas, como x64, x86 y ARM.
- **Herramientas de línea de comandos:** incluye herramientas de línea de comandos fáciles de usar que pueden utilizarse para el desarrollo local y en escenarios de integración continua.

- **Implementación flexible:** se puede incluir en la aplicación o se puede instalar de forma paralela a nivel de usuario o de equipo. Se puede usar con contenedores de Docker.
- **Compatible:** .NET Core es compatible con .NET Framework, Xamarin y Mono, mediante .NET Standard.
- **Código abierto:** la plataforma .NET Core es de código abierto, con licencias de MIT y Apache 2. .NET Core es un proyecto de .NET Foundation.
- **Compatible con Microsoft:** .NET Core incluye compatibilidad con Microsoft, como se indica en.NET Core Support (Compatibilidad de .NET Core).

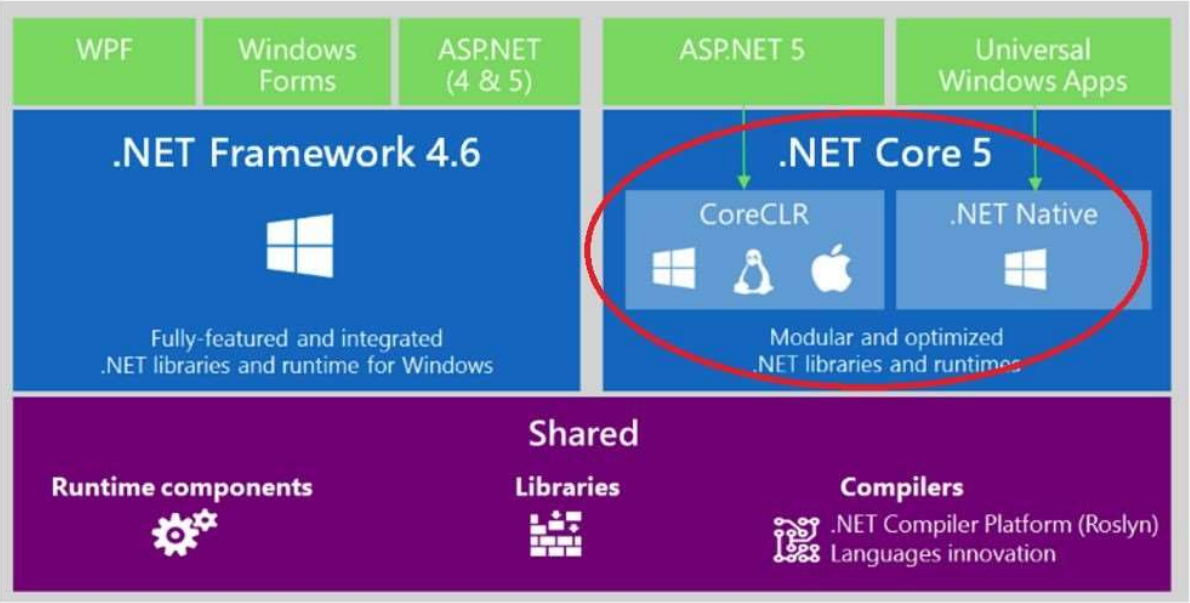
Lenguajes

Los lenguajes C#, Visual Basic y F# pueden usarse para escribir aplicaciones y bibliotecas para .NET Core. Estos lenguajes se integran o se pueden integrar en los editores de texto y entornos de desarrollo integrado que se prefieran, como Visual Studio, Visual Studio Code, Sublime Text y Vim. Esta integración se proporciona, en parte, gracias a la buena gente de los proyectos OmniSharp y Ionide.

Marcos de trabajo

Se han creado varias plataformas en .NET Core:

- ASP.NET Core
- Plataforma universal de Windows (UWP) de Windows 10
- Tizen



Composición

.NET Core consta de las siguientes partes:

- El runtime de .NET Core, que proporciona un sistema de tipos, la carga de ensamblados, un colector de elementos no utilizados, interoperabilidad nativa y otros servicios básicos. Las bibliotecas de .NET Core Framework, que proporcionan tipos de datos primitivos, tipos de composición de aplicaciones y utilidades fundamentales.
- El runtime de ASP.NET, que proporciona una plataforma en la que se pueden crear modernas aplicaciones conectadas a Internet y basadas en la nube, como aplicaciones web, aplicaciones de IoT y back-end móviles.
- Las herramientas de CLI de .NET Core y compiladores de lenguaje (Roslyn y F#) que habilitan la experiencia de desarrollador de .NET Core.
- La herramienta dotnet, que se usa para iniciar aplicaciones .NET Core y herramientas de CLI. Selecciona el entorno de tiempo de ejecución y lo hospeda, proporciona una directiva de carga de ensamblados e inicia aplicaciones y herramientas.

Estos componentes se distribuyen de las siguientes formas:

- Runtime de .NET Core: incluye el runtime y las bibliotecas de la plataforma de .NET Core.
- Runtime de ASP.NET Core: incluye el runtime y las bibliotecas de la plataforma de ASP.NET Core y .NET Core.

- SDK de .NET Core: incluye las herramientas de CLI de .NET, el runtime de ASP.NET Core y el runtime y la plataforma de .NET Core.

Código abierto

.NET Core es código abierto (licencia MIT) y fue presentado a .NET Foundation por Microsoft en 2014. Ahora es uno de los proyectos más activos de .NET Foundation. Todos los individuos y organizaciones pueden adoptarlo libremente, con cualquier fin: personal, académico o comercial. Varias empresas usan .NET Core como parte de aplicaciones, herramientas, nuevas plataformas y servicios de hosting. Algunas de estas empresas realizan contribuciones significativas a .NET Core en GitHub y proporcionan una guía sobre la dirección del producto como parte del Technical Steering Group de .NET Foundation.

Para obtener información general y conceptual sobre .NET Framework, se sugiere consultar el siguiente link Glosario de .NET.