Qué es .NET Core | OpenWebinars

Qué es .Net Core

.NET Core es la plataforma de desarrollo de Microsoft más moderna, de código fuente abierto, multiplataforma y de alto rendimiento para la creación de todo tipo de aplicaciones.

Fue liberada en el año 2016, y es el resultado de múltiples esfuerzos para hacer más ágil el desarrollo en **.NET** puesto que el **.NET Framework**, depende totalmente del sistema operativo Windows.

.NET Core, a diferencia del **.NET Framework**, no tiene este tipo de dependencia del sistema Windows y es modular, usando el sistema de paquetes **NuGet**, gracias al cual, recibiremos las diversas actualizaciones de **.NET Core**, a diferencia de **.NET Framework** que se actualiza a través de Windows Update.

Arquitectura de .NET Core

Esta arquitectura modular permite que podamos incluir únicamente lo necesario que requieren nuestras aplicaciones, haciendo que tengan un peso menor a la hora de ser desplegadas y que sea más sencilla su actualización a través de updates de **NuGet**.

Al ser multiplataforma, no depende de cosas específicas del sistema operativo, como sucede con el **.NET Framework**.

Características de .NET Core

Las principales características de **.NET Core** son las siguientes:

- Es multiplataforma y viene con soporte para su uso con contenedores **Docker.**
- Alto rendimiento. Se ha desarrollado desde cero y se le ha dado una alta importancia a esta característica.
- Asincronía con el uso de async/await. Se ha implementado este patrón en todas las librerías comunes para mejorar el rendimiento en las llamadas I/O.
- Es Open Source.

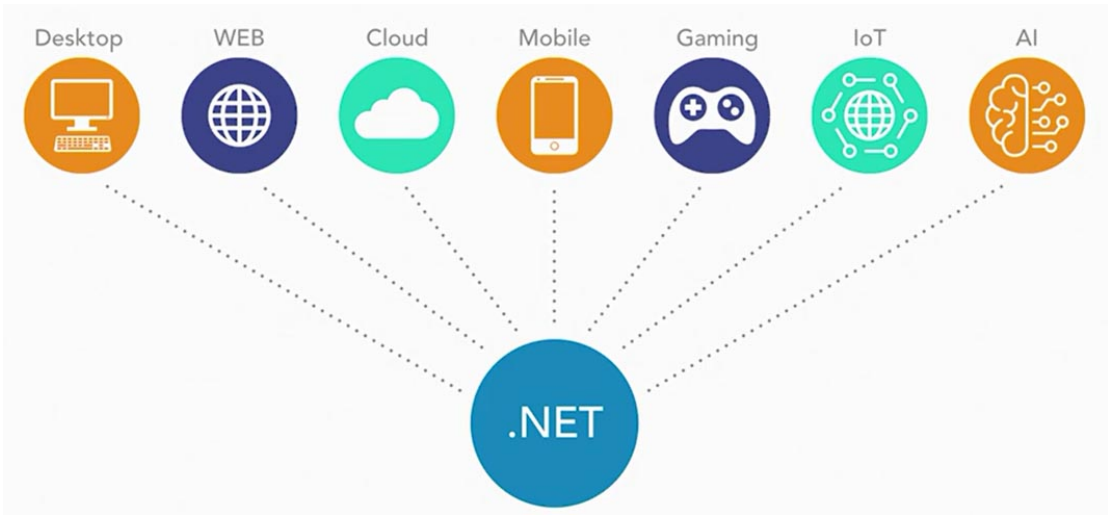
Para qué usar .NET Core

.NET Core nos permite realizar todo tipo de aplicaciones, como aplicaciones **web** que podrás desplegar en **Windows**, **Linux**, **Mac Os**.

Podrás desplegar tus aplicaciones usando contenedores **Docker** en distintas plataformas **Cloud** como **Azure**, **Amazon**, **GCP**.

Puedes usarla para crear aplicaciones de escritorio **UWP** que te permitirá correr tu aplicación en **Windows 10**, **XBOX** y **HoloLens** compartiendo el código y sin tener que reescribir tus bibliotecas.

Se puede usar para Internet Of Things, Inteligencia Artificial, desarrollo de juegos...



Versiones de .NET Core

Las versiones existentes de .NET Core son las siguientes:

Version	Lanzamiento
.NET Core 1.0	06/2016
.NET Core 1.1	11/2016
.NET Core 2.0	08/2017
.NET Core 2.1	05/2018
.NET Core 2.2	12/2018
.NET Core 3.0	09/2019
.NET Core 3.1	12/2019
.NET 5	11/2020

Puede ser que te hagas un lío con **.NET Framework**, **.NET Core**, y **.NET 5** puesto que se está unificando todo y es interesante que lo tengas claro. Así que veamos cómo hemos llegado hasta aquí.

Historia de .NET

En el año 2001 se publican los estándares 334 y 335 en ECMA para el lenguaje de programación C# y el Common Language Runtime (CLR).

En el año 2002, es liberado al mundo entero el **.NET Framework 1**. En ese mismo año, el proyecto Mono comienza. **Mono**, tenía como objetivo ser una implementación de **.NET** en **Linux**.

En el año 2008, Microsoft anuncia el proyecto ASP.NET MVC como un proyecto open source. Durante la conferencia Build, el creador de C#, Anders Hejlsberg, anuncia la liberación del compilador de C#, llamado Roslyn, como un proyecto open source.

En ese mismo año, comienza el proyecto **.NET Core**. Esto fue debido a que Microsoft no tenía una plataforma de desarrollo web robusta y de alto rendimiento que pudiera competir con otras.

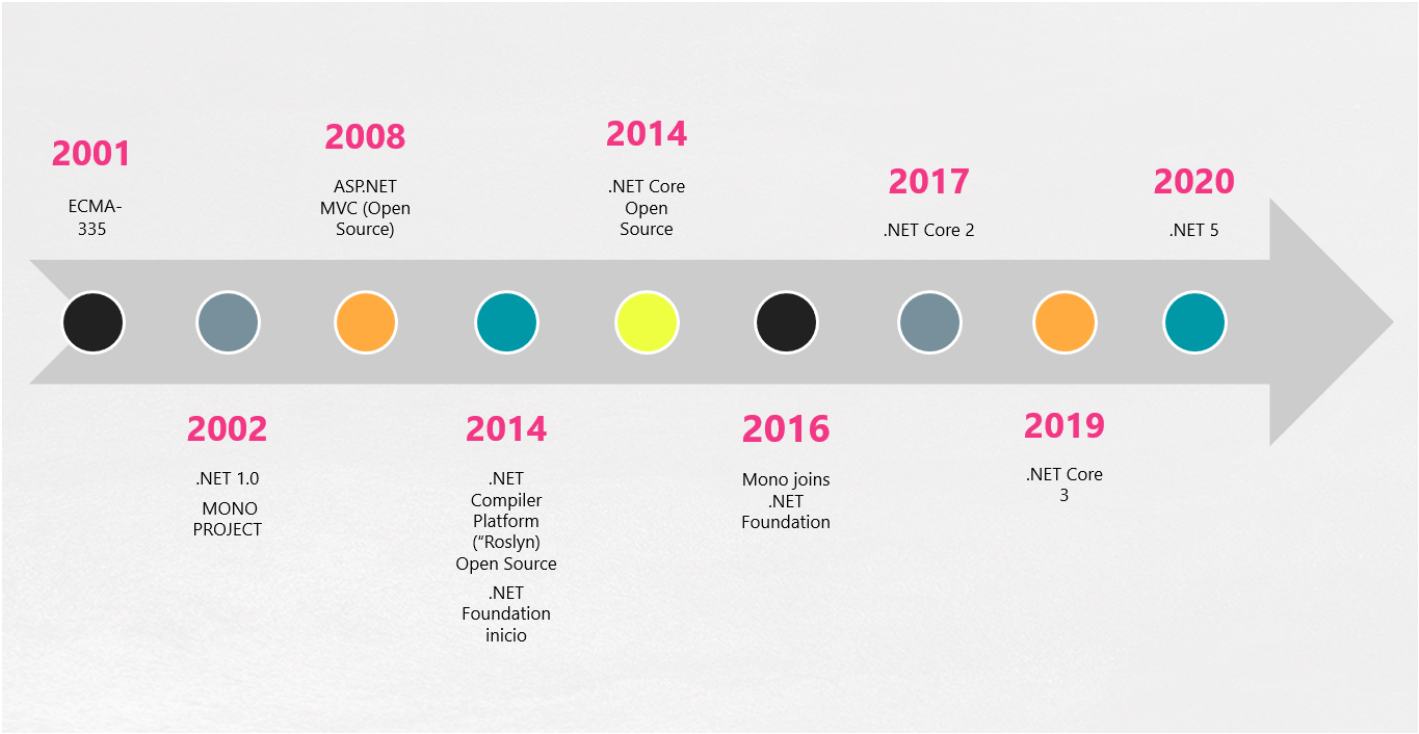
En aquel momento existía, y existe a día de hoy, **ASP.NET** en el **.NET Framework**, pero dicha plataforma siempre se ha visto lastrada negativamente, puesto que el **.NET Framework** un proyecto muy rígido y monolítico.

La empresa tomó la decisión de construir una plataforma de desarrollo web desde cero. Por esto, podemos asegurar que **ASP.NET Core** fue la causa detrás del impulso para crear **.NET Core**.

En 2016 Microsoft adquiere **Xamarin** y el proyecto **Mono** se une a la organización **.NET Foundation**.

En 2017, es liberado **.NET Core 2**.

Desde finales del año 2019 tenemos a nuestra disposición **.NET Core 3.1**, y para finales del año 2020 tendremos **.NET 5** que va a ser la unificación del **.NET Framework**, **.NET Core** y **Mono** en una sola plataforma de desarrollo open source, de alto rendimiento y multiplataforma.

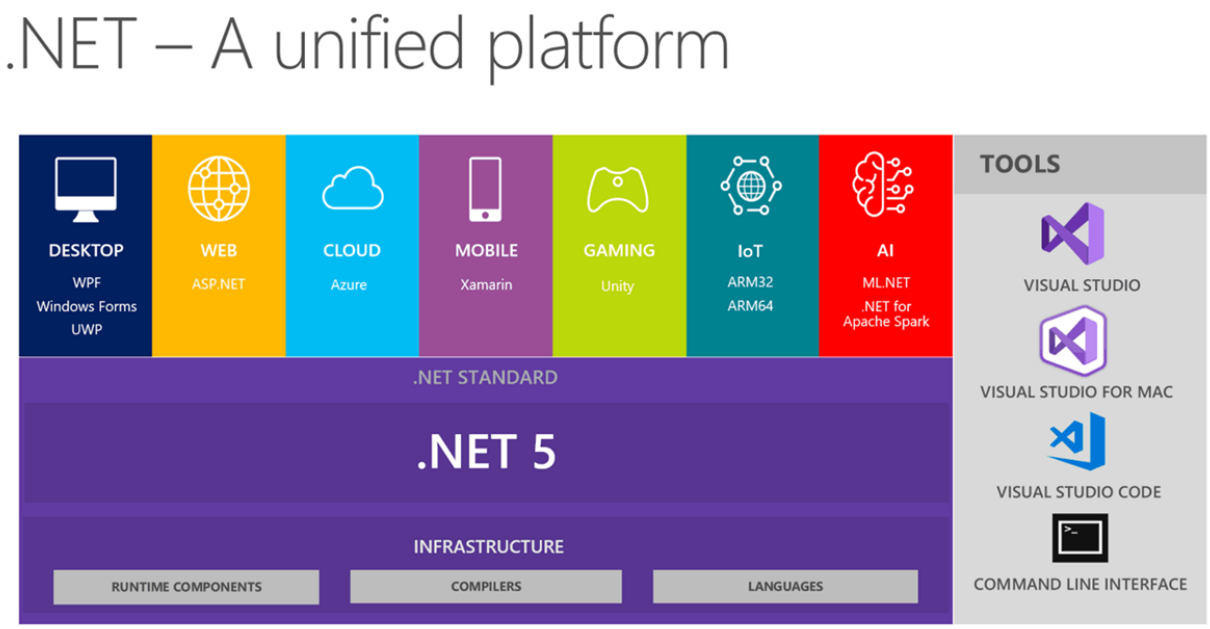


Con todo esto expuesto nos podemos preguntar...

Futuro de .NET y .NET Core

El futuro de .NET es .NET Core.

A través de muchos años, se han creado diversas plataformas **.NET**, como el **.NET Framework**, **.NET Core** y **Mono**, además de otras que, de alguna forma u otra, ayudaron a perfilar lo que es .NET hoy en día, como Silverlight, Windows Phone y el .NET Compact Framework.



Desde que se lanzó .NET Core 3, **no se van a migrar más características de .NET Clásico a .NET Core** (ni a .NET 5). Con lo que las siguientes tecnologías no tienen un equivalente en .NET Core.

Web Forms no tiene su equivalente Core con **ASP.NET Core**. Existe **Blazor**, que podríamos decir que es su sustituto, aunque no hay una migración directa.

Si quieres migrar un servicio **WCF** a **.NET Core** tendrías que utilizar **Web API**, que no sería de forma directa o gRPC, pero requerirá de un esfuerzo en rehacer todo tu código.

Si tienes proyectos creados con **Workflow Foundation**, no hay nada oficial en .NET Core, pero hay una versión Open Source de Workflow Foundation migrada a .NET Core, el problema es que no es una migración hecha por Microsoft y no podemos saber si la comunidad que tiene detrás le dará un soporte suficiente como para arriesgarnos con nuestras aplicaciones.

Dicho esto, no hay una obligación en migrar tus aplicaciones actuales a **.NET Core**, si ya las tienes funcionando correctamente. Son plataformas diferentes, y la migración directa puede llevarte más tiempo del que crees sobre todo si tienes alguna de las tecnologías Legacy anteriormente mencionadas.

Lo que sí que es recomendable es que si tienes que aprender **.NET**, aprendas directamente **.NET Core**, salvo que tengas que mantener aplicaciones Legacy, realmente se parecen mucho, pero a día de hoy si tienes que empezar por una de ellas **lo recomendado** es **.NET Core y C#**.

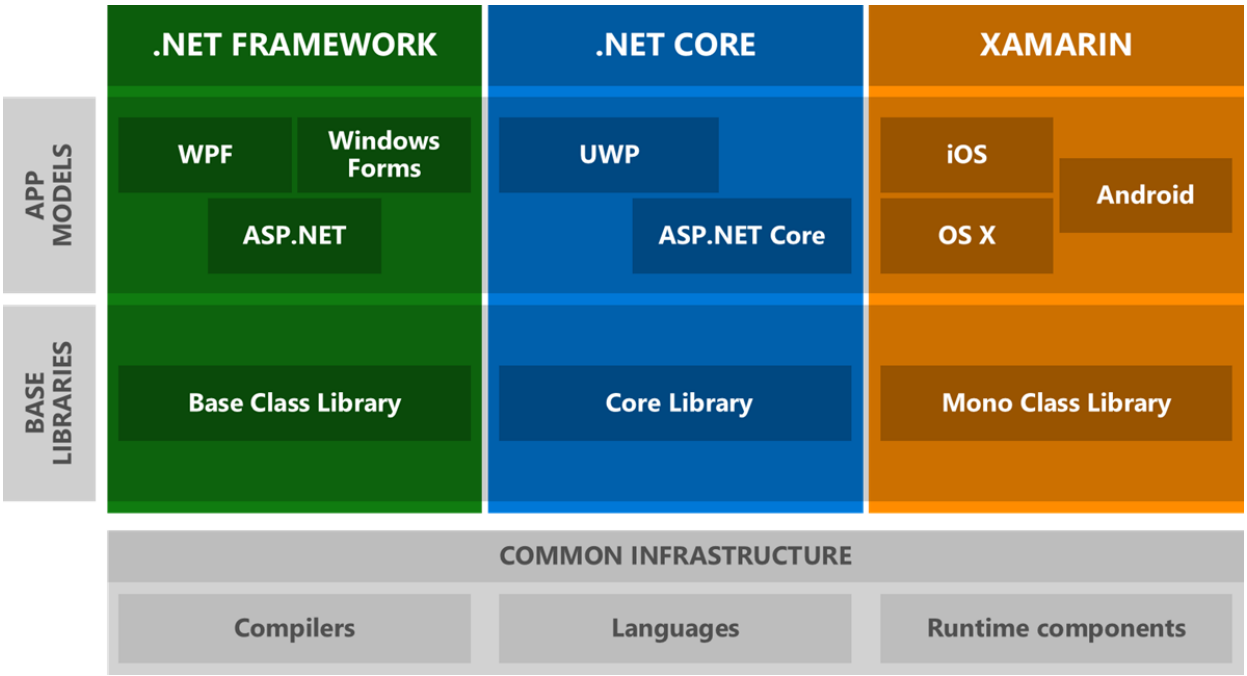
Si ya eres desarrollador de .NET y C#, aprender **.NET Core** no te costará en exceso y merece la pena.

Entonces el futuro es **.NET Core**, que se va a llamar **.NET 5** cuando salga la siguiente versión, pero para comprenderlo todo necesito que entiendas que es .NET Standard.

Qué es .NET Standard

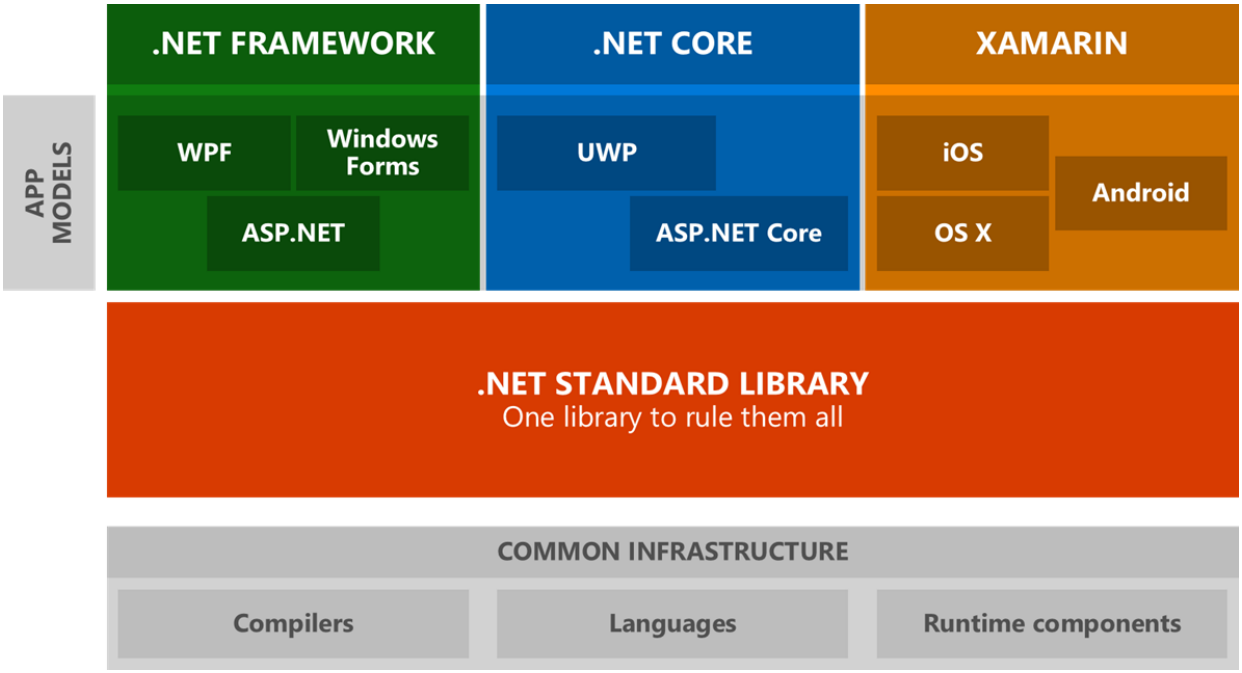
.NET Standard es una nueva especificación que viene a poner orden en el ecosistema de .NET.

Lo que implica es tener una especificación y una serie de reglas comunes para todas las bibliotecas de clases base existentes para este ecosistema.



Esta necesidad es debida a la existente fragmentación de las distintas implementaciones de las bibliotecas de clases base que existen las diferentes plataformas **.NET** (**.NET Framework**, **Mono** usado en **Xamarin**, y **.NET Core**).

Debido a esto, el poder reutilizar código es muy complicado, por lo que el simple hecho de **arreglar un bug** o añadir nuevas funcionalidades, implicaba que se tuviera que hacer en cada una de estas plataformas, con el consiguiente incremento de tiempo de desarrollo y posibilidad de no estar completamente alineadas.



A través de esta especificación, tenemos la seguridad de escribir código multiplataforma, las novedades nos llegarán antes, nuestros conocimientos sirven para varias plataformas y podemos reusar código entre todas ellas.

En **.NET Standard**, las versiones más nuevas incluyen todas las API de sus versiones anteriores. Queda explicado con el siguiente gráfico.

Con todo esto, llegamos a poder explicar que es **.NET 5**

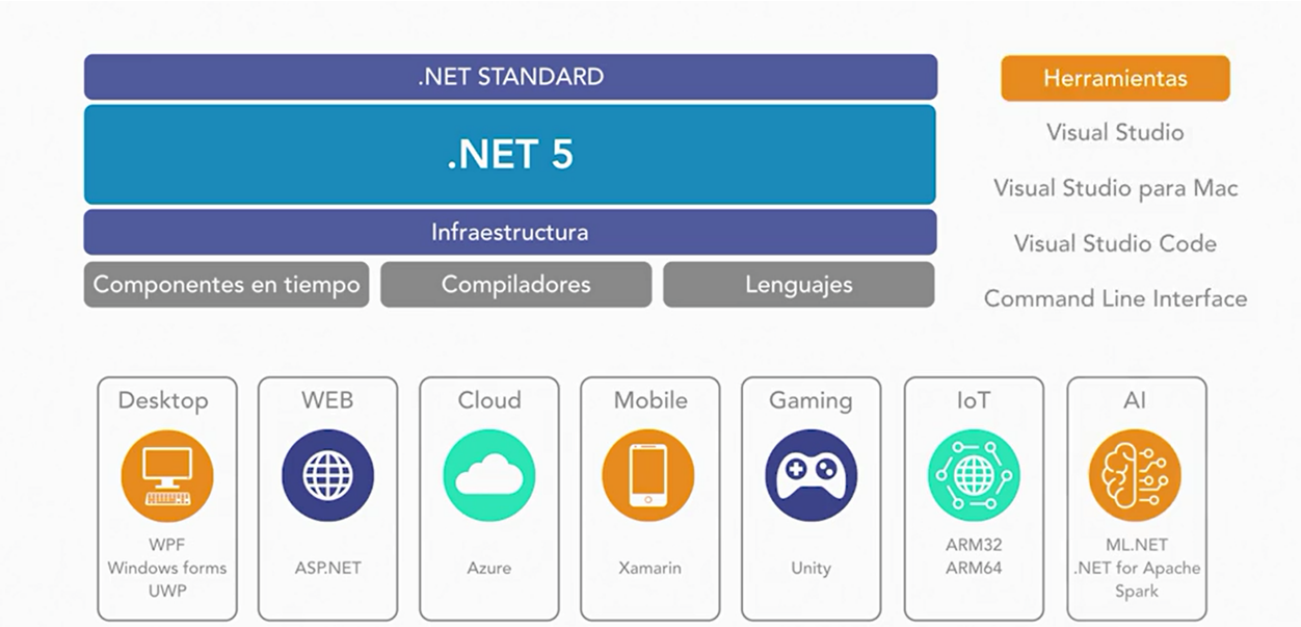
Qué es .NET 5

.NET 5 va a ser la unificación de **.NET Framework**, **.NET Core** y **Mono** en una sola plataforma de desarrollo multiplataforma.

Esta evolución de **.NET Core** contará con una sola biblioteca de clases basada en **.NET Standard**, herramientas unificadas. Además, **.NET** contará con la posibilidad de interoperar con código escrito en otros lenguajes, como **Java** o **Swift**.

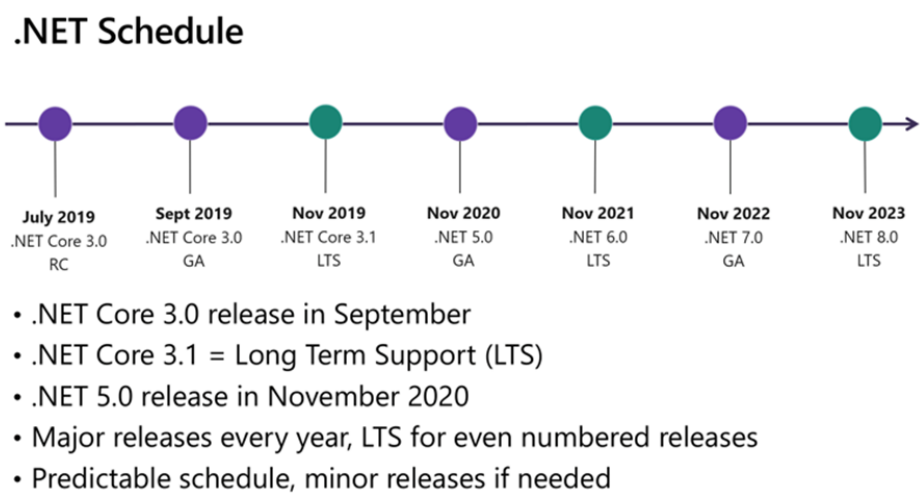
.NET 5 será la plataforma de desarrollo para la construcción de todo tipo de aplicaciones.

Podremos crear aplicaciones multiplataforma como aplicaciones y servicios web, escritorio, juegos, aplicaciones cloud, móviles...



Microsoft ha anunciado un calendario anual de liberaciones de versiones que comienza con **.NET 5**.

Es decir, en el año 2021, tendremos .NET 6, en 2022, .NET 7, en 2023, .NET 8, etc.



Qué lenguajes se pueden usar en .NET Core

- **C#**

Es un lenguaje de programación multiparadigma y muy sofisticado que ha evolucionado en conjunto con la estrategia de .NET.

Tiene nuevas características, como tipos por referencia nullables, rangos, índices, streams asíncronos... es la opción por defecto a la hora de crear aplicaciones en **.NET Core** y de la que más documentación vas a encontrar.

- **F#**

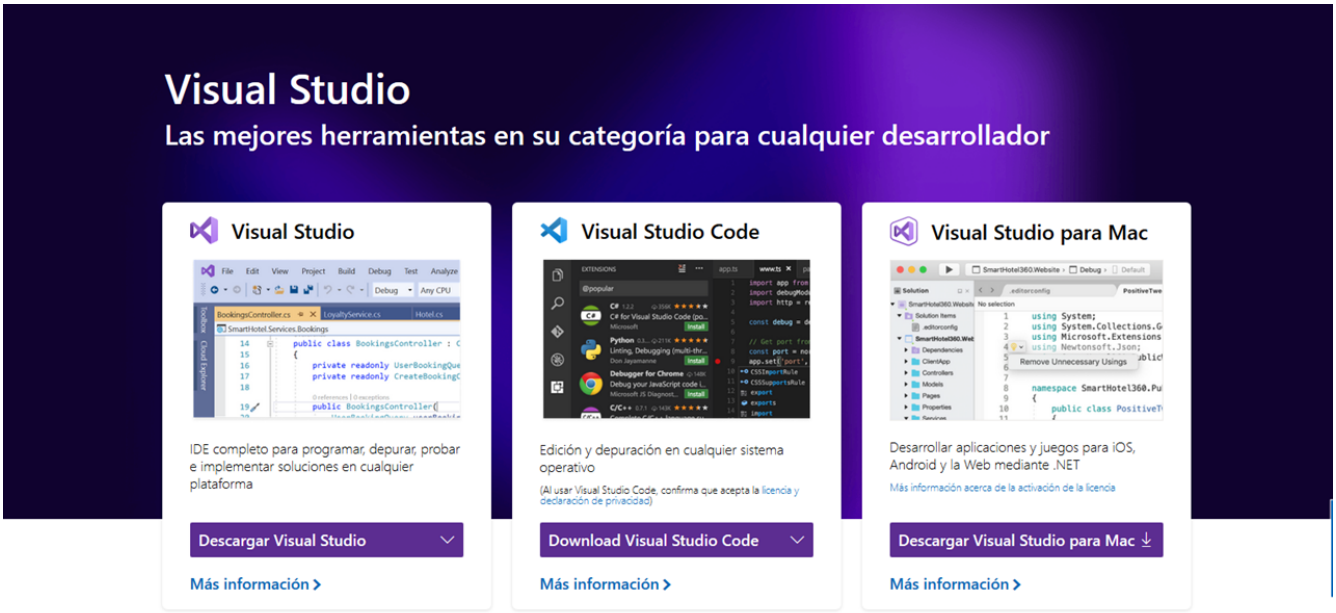
Es un lenguaje funcional que ha ido creciendo desde su creación en el año 2005.

Con características únicas y potentes, puede ser usado para crear cualquier tipo de solución de software y además, cuenta con una comunidad muy activa.

- **Visual Basic**

También aparece como una opción para crear soluciones con .NET, pero te puedo adelantar que Microsoft ha decidido no seguir evolucionando este lenguaje con lo que no va a recibir nuevas características que se vayan incluyendo cuando evolucione la plataforma **.NET Core**, con lo que no puedo recomendar que empieces con este lenguaje si te interesa **estar al día**.

Qué herramientas se pueden usar para trabajar con .NET Core



Visual Studio es el entorno de desarrollo integrado o **IDE** siglas de Integrated Development Envionment en inglés, que por excelencia es usado en el sistema operativo Windows.

Visual Studio Code es otra herramienta de desarrollo que podemos usar, tiene una gran adopción y aceptación, es de código fuente abierto y cuenta con un ecosistema de extensiones y componentes muy grande.

Por otro lado, si eres usuario de **Mac** también existe **Visual Studio for Mac** para dicho sistema operativo.

Sí debes saber que **Visual Studio for Mac** y **Visual Studio** para Windows tienen diferentes plantillas de proyectos, ya que en Visual Studio de Windows puedes crear aplicaciones de **escritorio**, por ejemplo, con **WPF** o **Windows Forms**, mientras que en Visual Studio for Mac, no.

Mejora las habilidades de tus desarrolladores

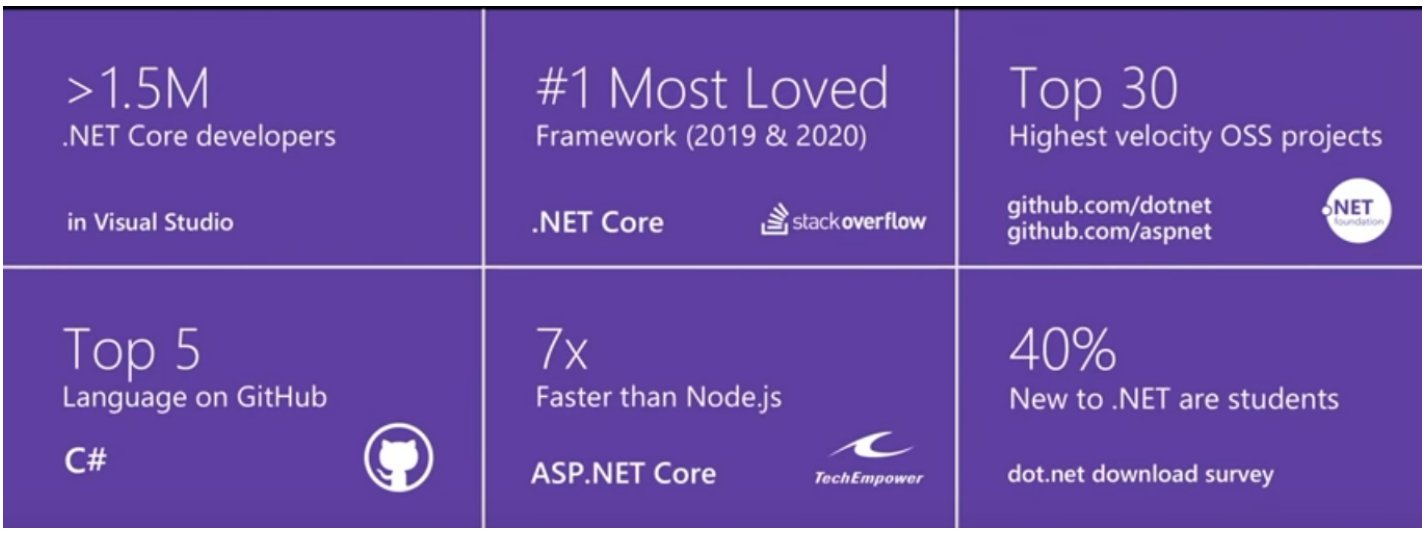
Acelera la formación tecnológica de tus equipos con OpenWebinars. Desarrolla tu estrategia de atracción, fidelización y crecimiento de tus profesionales con el menor esfuerzo.

Solicitar más información

Beneficios de usar .NET Core

.NET Core tiene todas estas ventajas:

- Tus aplicaciones **.NET** pueden ser multiplataforma
- **ASP.NET Core** supera a ASP.NET en .NET Framework en **características** y **rendimiento**
- **.NET Core** es el foco de la **innovación**
- Los ciclos de lanzamiento de **nuevas versiones** son más rápidos
- Está orientado a obtener el mejor **rendimiento**



Por todo lo expuesto anteriormente podemos decir que esa plataforma te va a asegurar **estar a la última** y que todo lo que desarrolles pueda ser **actualizado** y con **soporte** durante mucho tiempo

¿Te ha parecido interesante? Te puede interesar nuestro taller Crea tu Api en C# con .NET Core.