UNIVERSIDAD DON BOSCO

FACULTAD DE INGENIERÍA

DESARROLLO DE APLICACIONES CON

SOFTWARE PROPIETARIO

(DPS404)

CICLO II 2025

TEMA GENERAL

Desarrollo de página web para “Creaciones Normita”

DOCENTE:

Ing. Kevin Jiménez

NOMBRE	CARNÉ	ROL
Álvarez Sánchez, Jessica Paola	AS241238	Backend
Hernández Guerrero, Guillermo Antonio	HG243075	Frontend
Meléndez Avilés, Yassir Mauricio	MA243080	Backend
Peña Bustillo, Bryan Alexis	PB243032	Líder de Proyecto
Rodríguez Benítez, Giselle Esmeralda	RB243017	Frontend

AULA VIRTUAL, 15 DE AGOSTO DE 2025

¿Qué es ASP.NET Core y cómo evolucionó desde ASP.NET clásico

ASP.MET Core es un framework de desarrollo web de código abierto y multiplataformas, pensado para poder crear aplicaciones modernas como sitios web, APIs y micro servidores. A diferencia de su versión anterior, el ASP.NET clásico, que solo funcionaba en Windows y dependía del .NET Framework completo, ASP.NET Core marca un cambio muy grande.

La evolución de ASP.NET clásico a Core se enfocó en hacerlo un poco más modular, rápido y flexible. En el caso del ASP.NET clásico, este trabajaba con la arquitectura "Web Forms", que buscaba poder imitar la experiencia de desarrollo de escritorio en la web, pero eso terminaba generando proyectos muy pesados y difíciles de mantener. En cambio, ASP.NET Core fue reescrito desde cero con una arquitectura basada en middleware lo que significa que cada parte de la aplicación se construye como un conjunto de componentes que procesan las solicitudes HTTP, lo que lo hace mucha más liviano y fácil de personalizar.

1. Principales características y ventajas de usar ASP.NET Core para el desarrollo web.
    - **Multiplataforma:** Nos permite ejecutar en Windows, macOS y Linux.
    - **Código abierto:** Su código fuente está mas disponible en GitHub, donde cualquiera puede aportar mejoras.
    - **Alto rendimiento:** Gracias a su arquitectura modular y optimizada, es mas rápido que ASP.NET clásico.
    - **Modularidad:** Solo puede incluir los componentes necesarios, lo que reduce así la sobre carga y el tamaño de la aplicación.
    - **Desarrollo en la nube:** Está mas optimizado para la implementación de la nube y la creación de microservicios.
    - **Compatibilidad:** Se íntegra perfectamente con .NET, lo que nos permite el acceso a una vasta biblioteca de paquetes.
    - **Unificado:** Combina los módulos de desarrollo de ASP.NET MVC y Web API en un solo framework.

2. Arquitectura de una aplicación ASP.NET Core: MVC, Razor Pages, Blazor.

La arquitectura de una aplicación ASP.NET Core tiene más flexibilidad y admite varios modelos de desarrollo, siendo así los más comunes:

- **MVC (Modelo-Vista-Controlador):** Este patrón divide la aplicación en tres componentes interconectados. El Modelo que maneja la lógica de datos, la Vista se encarga de las interfaces de usuario y el Controlador actúa como intermediario, procesando así las solicitudes y actualizando el modelo o la vista según sea necesario. Es ideal para aplicaciones web complejas y APIs.
- **Razor Pages:** Es un modelo un poco mas simple que MVC, diseñado para escenarios donde solo necesitemos una arquitectura centrada en la página. Cada pagina tiene su propio código que maneja las solicitudes, lo que simplifica así el desarrollo de las aplicaciones web pequeñas y medianas.
- **Blazor:** Este nos permite crear interfaces de usuarios interactivas del lado del cliente utilizando C# en lugar de JavaScript. Con Blazor Server, el código se ejecuta en el servidor y la UI se actualiza a través de SignalR.

3. Comparación de usos con otras tecnologías web (Node.js, Django, Laravel, Spring Boot).

- **ASP.NET Core vs Node.js**

  Node.js usa JavaScript (o TypeScript) tanto en el cliente como en el servidor, mientras que ASP.NET Core trabaja con C#. Esto marca la diferencia: C# es tipado estático, lo que ayuda a reducir errores y mantener proyectos grandes. Node.js destaca por su rapidez en apps en tiempo real y APIs gracias a su modelo no bloqueante, pero ASP.NET Core suele superarlo en rendimiento y escalabilidad en el lado del servidor.

- **ASP.NET Core vs Django**

  Django (Python) es famoso por traer todo incluido desde el inicio (ORM, panel admin, plantillas, etc.), lo que acelera el desarrollo. ASP.NET Core es más modular: eliges qué usar y qué no. Aquí influye el lenguaje: C# es tipado fuerte, Python es dinámico, y esa diferencia se nota en proyectos grandes.

- **ASP.NET Core vs Laravel**

  Laravel (PHP) se destaca por su sintaxis sencilla y la comunidad que lo respalda. ASP.NET Core ofrece algo parecido, pero con la ventaja del tipado estático y el soporte de Microsoft. Aunque PHP ha mejorado bastante, C# y .NET suelen ser mejor vistos para proyectos empresariales grandes por su rendimiento y estabilidad.

- **ASP.NET Core vs Spring Boot**

  Spring Boot (Java) y ASP.NET Core juegan en la misma liga: aplicaciones grandes, microservicios y APIs. La diferencia está en el ecosistema y el lenguaje. Java tiene mucha base de código y portabilidad; C# aprovecha mejor las herramientas de Microsoft y ofrece muy buen rendimiento. La elección depende más de qué conozca tu equipo (Java o C#) y la infraestructura que ya tenga la empresa.

5. Herramientas necesarias para desarrollar con ASP.NET Core

Para el desarrollo con ASP.NET Core existen diversas herramientas que facilitan el trabajo del programador. Visual Studio es la más utilizada en entornos empresariales porque integra plantillas, depuración avanzada, pruebas unitarias y conectores directos con Azure. Por otro lado, Visual Studio Code es una opción ligera y multiplataforma, que mediante extensiones como C# Dev Kit permite programar en Windows, Linux y macOS con gran flexibilidad. El .NET SDK resulta indispensable, pues contiene los compiladores y librerías necesarias para compilar y ejecutar aplicaciones.

Un IDE o Editor de Código:

Visual Studio:

Un IDE completo para Windows que ofrece herramientas integradas para el desarrollo de aplicaciones .NET, incluyendo ASP.NET Core.

En cuanto a bases de datos, SQL Server es la alternativa más común dentro del ecosistema Microsoft, aunque la compatibilidad de ASP.NET Core con PostgreSQL, MySQL y MongoDB amplía las posibilidades. Además, en proyectos modernos se utilizan herramientas de DevOps como Docker y Azure DevOps, que permiten aplicar metodologías de integración y entrega continua (CI/CD). Este conjunto de utilidades convierte a ASP.NET Core en una plataforma robusta para desarrollar aplicaciones escalables y seguras.

6. Posibles desventajas o retos al trabajar con ASP.NET Core

Si bien ASP.NET Core ofrece ventajas importantes, también presenta ciertos retos. Uno de los principales es la curva de aprendizaje, ya que requiere comprender conceptos avanzados como middleware, inyección de dependencias y patrones de arquitectura. Para quienes migran desde tecnologías como PHP o frameworks de JavaScript, esta transición puede resultar compleja. Otro desafío es el despliegue, que a menudo necesita servidores configurados con .NET Runtime o el uso de contenedores, lo cual demanda mayor inversión en comparación con el hosting compartido de Laravel o Django.

Otro aspecto para considerar es el ritmo acelerado de actualizaciones de Microsoft. Aunque esto garantiza mejoras en seguridad y rendimiento, también obliga a capacitar constantemente a los equipos y adaptar los proyectos a nuevas versiones. Finalmente, aunque la comunidad de .NET ha crecido significativamente, en comparación con Node.js (NPM) o Python (PyPI), el ecosistema de paquetes disponibles todavía es más reducido. Esto puede representar un obstáculo cuando se buscan soluciones listas para problemas muy específicos, lo que a veces obliga a desarrollar componentes desde cero. En conclusión, ASP.NET Core es una plataforma poderosa, pero requiere planificación, recursos técnicos y disposición para enfrentar su constante evolución.

7. Importancia de aprender C# y ASP.NET Core en el contexto laboral actual

El aprendizaje de C# y ASP.NET Core es más que una decisión técnica: constituye una estrategia profesional en un mercado laboral altamente competitivo. C# se ha consolidado como uno de los lenguajes de programación más utilizados en aplicaciones empresariales. Según la encuesta de Stack Overflow (2024), se ubica dentro del top 10 de lenguajes más empleados a nivel mundial, lo que evidencia su vigencia.

ASP.NET Core, por su parte, se ha convertido en el framework principal de Microsoft para el desarrollo web, con soporte nativo para arquitecturas modernas como APIs RESTful, microservicios y aplicaciones en la nube. Su estrecha integración con Microsoft Azure le da una ventaja significativa frente a otras tecnologías, pues permite desplegar sistemas escalables con facilidad. Esto ha llevado a que compañías de sectores estratégicos como la banca, telecomunicaciones, educación y gobierno lo adopten en sus proyectos de gran escala.

En el caso de El Salvador y Latinoamérica, muchas instituciones públicas y privadas utilizan el ecosistema Microsoft, lo cual incrementa la demanda de profesionales capacitados en C# y ASP.NET Core. A nivel global, esta tecnología abre oportunidades de trabajo remoto, ya que consultoras internacionales contratan desarrolladores para mantener sistemas robustos bajo este framework. De este modo, aprender C# y ASP.NET Core no solo asegura empleabilidad en el presente, sino que también se proyecta como una inversión a largo plazo en la carrera de cualquier desarrollador.

	Descripción	Ejemplos
Evolución	De ASP.NET clásico (solo Windows) → Core (multiplataforma, modular, middleware).	Una app creada en **ASP.NET clásico** solo corría en Windows/IIS. Con **ASP.NET Core**, la misma app puede ejecutarse en **Linux con Docker**.
Características	• Multiplataforma (Windows, Linux, macOS) • Código abierto (GitHub) • Alto rendimiento (Kestrel) • Integración con la nube (Azure)	Un sistema de **e-commerce** hecho en ASP.NET Core puede correr en Linux, usar GitHub para control de versiones y desplegarse en Azure con alta velocidad gracias a Kestrel.
Arquitectura	• MVC (aplicaciones grandes) • Razor Pages (más simple) • Blazor (UI con C# en lugar de JS)	**MVC**: un sistema bancario con múltiples módulos. **Razor Pages**: un blog personal con formularios. **Blazor**: un panel administrativo con componentes interactivos usando solo C#.
herramientas	• IDEs: Visual Studio, VS Code • SDK: .NET SDK • Bases de datos: SQL Server, PostgreSQL, MySQL, MongoDB • DevOps: Docker, Kubernetes, Azure DevOps	Desarrollar en **Visual Studio**, compilar con **.NET SDK**, conectar a **SQL Server** y desplegar en **Docker + Kubernetes** en la nube.
Retos	• Curva de aprendizaje • Despliegue complejo • Ciclo acelerado de actualizaciones	Un principiante puede confundirse con **middleware y dependencias**. Además, una app en

	• Ecosistema más cerrado que Node.js/Python	producción puede fallar tras una **actualización de .NET Core** si no se da mantenimiento.
Importancia laboral	• Alta demanda en banca, gobierno y empresas con Azure • Uso extendido en El Salvador y Latinoamérica	**Banco Agrícola (SV)** utiliza .NET para sistemas internos. Un desarrollador salvadoreño puede trabajar en proyectos remotos de **EE.UU.** que usan Azure y ASP.NET Core.

Referencias

What is .NET? An open-source developer platform | .NET. (2025). Microsoft. https://dotnet.microsoft.com/en-us/learn/dotnet/what-is-dotnet

wadepickett. (2025). *ASP.NET documentation*. Microsoft.com. https://learn.microsoft.com/es-es/aspnet/core/?view=aspnetcore-9.0

dotnet. (2025, July 8). *GitHub - dotnet/aspnetcore: ASP.NET Core is a cross-platform .NET framework for building modern cloud-based web applications on Windows, Mac, or Linux.* GitHub. https://github.com/dotnet/aspnetcore

Patel, R. (2024, October 4). *Exploring .NET 8 and ASP.NET Core: Features, Advantages, and Comparison with ASP.NET Framework*. Medium. https://medium.com/@ravipatel.it/exploring-net-8-and-asp-net-core-features-advantages-and-comparison-with-asp-net-framework-a6b5743fc7be Mic

Microsoft. (2025, enero 10). ASP.NET Core fundamentals. Microsoft Learn. https://learn.microsoft.com/en-us/aspnet/core/fundamentals/?view=aspnetcore-9.0

Microsoft. (2025, junio 18). Desarrollo de aplicaciones ASP.NET Core MVC. Microsoft Learn. https://learn.microsoft.com/es-es/dotnet/architecture/modern-web-apps-azure/develop-asp-net-core-mvc-apps