**Universidad Privada del Valle**

**Facultad de INFORMATICA Y ELECTRONICA**

**Evaluación**

**INGENIERIA EN sistemas**

**blazor un nuevo paradigma para el desarrollo web iteractivas con asp.net**

**PROGRAMACION web iii**

**Grupo “D”**

**Integrantes: Denilson Asis Saavedra Mamani**

**Alejandro Javier Chipana Ramirez**

**Docente: Ing. Limber Sanchez Mendoza**

ÍNDICE

[1. MARCO TEORICO 1](#_Toc178104270)

[1.1. RESUMEN 1](#_Toc178104271)

[2. INTRODUCCION 1](#_Toc178104272)

[2.1. Contexto general 1](#_Toc178104273)

[2.2. Justificación 1](#_Toc178104274)

[2.3. Objetivos 2](#_Toc178104275)

[3. Desarrollo Teórico 2](#_Toc178104276)

[3.1. Descripción de la tecnología 2](#_Toc178104277)

[3.2. Ventajas 2](#_Toc178104278)

[3.3. Desventajas 2](#_Toc178104279)

[3.4. Comparación con otras tecnologías 3](#_Toc178104280)

[3.5. Revisión de antecedentes 3](#_Toc178104281)

[4. Discusión 3](#_Toc178104282)

[4.1. Análisis crítico 3](#_Toc178104283)

[4.2. Posibles aplicaciones 4](#_Toc178104284)

[4.3. Desafíos y futuras tendencias 4](#_Toc178104285)

[4.4. Desafíos y futuras tendencias 4](#_Toc178104286)

[5. Conclusiones 4](#_Toc178104287)

[5.1. Resumen de los puntos clave 4](#_Toc178104288)

[5.2. Importancia de la tecnología 5](#_Toc178104289)

[5.3. Sugerencias para investigaciones futuras 5](#_Toc178104290)

[6. Referencias 6](#_Toc178104291)

**La Paz 24 septiembre de 2024**

# MARCO TEORICO

## ****RESUMEN****

Blazor es un framework dentro del ecosistema de ASP.NET Core que permite desarrollar aplicaciones web interactivas utilizando C# en lugar de JavaScript, brindando a los desarrolladores una opción para construir aplicaciones tanto en el lado del cliente (WebAssembly) como en el servidor. Este artículo analiza la relevancia de Blazor en el desarrollo web moderno, destacando sus características, ventajas, y limitaciones. Además, se explora cómo Blazor se compara con tecnologías tradicionales de frontend como Angular y React, y su impacto en la experiencia del desarrollador. El objetivo es proporcionar un análisis detallado sobre cómo Blazor puede transformar el desarrollo web, facilitando la integración del código del frontend y backend en una sola base tecnológica. Finalmente, se identifican los principales desafíos que enfrenta Blazor en términos de rendimiento y ecosistema, y se destacan posibles mejoras y aplicaciones futuras.

# INTRODUCCION

Para implementar un Executor en Kotlin, debes importar las clases correspondientes y crear un ThreadPoolExecutor o usar un ScheduledExecutorService para ejecutar tareas recurrentes o programadas

## Contexto general

ASP.NET Core es un framework de código abierto y multiplataforma para construir aplicaciones web modernas, incluidas aplicaciones en la nube, APIs, y microservicios. Su capacidad de rendimiento, seguridad, y flexibilidad lo posiciona como una de las tecnologías más utilizadas en el desarrollo web. En este ecosistema, **Blazor** destaca como un framework revolucionario, ya que permite a los desarrolladores escribir aplicaciones web interactivas usando C# en lugar de JavaScript.

## Justificación

Blazor es relevante en el ecosistema de ASP.NET Core porque aborda una necesidad clave de los desarrolladores de .NET: la posibilidad de crear aplicaciones web modernas sin tener que dominar JavaScript, manteniendo el desarrollo full-stack en un solo lenguaje. Su integración dentro de ASP.NET Core simplifica la creación de aplicaciones, facilitando el uso de la misma infraestructura para el frontend y el backend.

## Objetivos

El propósito de este artículo es explorar las características de Blazor, sus ventajas y desventajas, y cómo se posiciona frente a otras tecnologías en el desarrollo web moderno. Se examinarán sus posibles aplicaciones y los desafíos que enfrenta.

# ****Desarrollo Teórico****

## Descripción de la tecnología

Blazor, por ejemplo, es una tecnología innovadora dentro de ASP.NET Core que permite a los desarrolladores construir aplicaciones web interactivas utilizando C# en lugar de JavaScript. Esto es posible gracias a WebAssembly, un formato binario ejecutado en el navegador que permite que las aplicaciones web alcancen un rendimiento casi nativo. Blazor se divide en dos modelos principales: Blazor Server y Blazor WebAssembly. Mientras que el primero ejecuta la lógica en el servidor y comunica las actualizaciones al cliente mediante SignalR, el segundo ejecuta el código directamente en el navegador del usuario, eliminando la necesidad de servidores adicionales para ciertas tareas. En ambos casos, la interfaz de usuario se define mediante componentes de Razor, lo que facilita la reutilización y la organización del código.

## Ventajas

* **Unificación de lenguajes**: Todo el desarrollo puede realizarse en C#, eliminando la necesidad de usar JavaScript para el frontend.
* **Reutilización de código**: Permite compartir lógica entre frontend y backend, simplificando la arquitectura del proyecto.
* **Integración nativa con ASP.NET Core**: La tecnología se integra fácilmente con las capacidades de ASP.NET Core, como la autenticación, el manejo de datos, y la inyección de dependencias.

## Desventajas

* **Tiempos de carga inicial**: Blazor WebAssembly puede ser lento en dispositivos con poca capacidad debido a la carga de la aplicación completa en el navegador.
* **Madurez del ecosistema**: A diferencia de frameworks como React o Angular, Blazor aún está en desarrollo y no cuenta con la misma cantidad de bibliotecas o recursos.

## Comparación con otras tecnologías

En comparación con frameworks como Angular o React, Blazor ofrece la ventaja de usar C# en lugar de JavaScript, lo que puede ser atractivo para los desarrolladores .NET que prefieren mantenerse dentro del ecosistema de Microsoft. Sin embargo, frameworks como Angular y React tienen comunidades más amplias y un mayor número de recursos, lo que puede facilitar el soporte y la resolución de problemas para nuevos desarrolladores. También se debe considerar que JavaScript sigue siendo el estándar de facto para el desarrollo web frontend, lo que significa que frameworks como Angular o React pueden estar mejor integrados en entornos de desarrollo donde se espera la interoperabilidad con otras tecnologías JavaScript.

## Revisión de antecedentes

En los últimos años, han surgido varios estudios sobre la adopción de Blazor dentro del desarrollo de aplicaciones empresariales y proyectos personales. Investigaciones indican que Blazor está ganando popularidad entre los desarrolladores .NET debido a su capacidad para crear aplicaciones interactivas utilizando un único lenguaje de programación en todo el stack. Además, la documentación oficial de Microsoft ha hecho hincapié en la capacidad de Blazor para reducir la brecha entre el desarrollo backend y frontend, lo que facilita la creación de aplicaciones con lógica compartida entre el servidor y el cliente.

# Discusión

## Análisis crítico

La relevancia de Blazor en el desarrollo web actual radica en su capacidad para simplificar el ciclo de desarrollo al permitir que los desarrolladores usen un solo lenguaje en lugar de separar el backend (C#) del frontend (JavaScript). Esto no solo mejora la productividad, sino que también permite a los equipos reducir los errores y facilitar el mantenimiento de código a largo plazo. La integración de Blazor en el ecosistema ASP.NET Core significa que puede beneficiarse de la seguridad, escalabilidad y robustez del framework general, lo que lo convierte en una opción sólida para proyectos de gran envergadura.

## Posibles aplicaciones

Blazor es ideal para aplicaciones empresariales que ya usan el ecosistema .NET, como sistemas de gestión, intranets corporativas, y aplicaciones de comercio electrónico que requieren interfaces ricas e interactivas. Además, en proyectos donde la comunicación en tiempo real es crítica, Blazor Server puede aprovecharse junto a **SignalR**.

## Desafíos y futuras tendencias

Blazor es especialmente útil en proyectos que requieren una integración completa entre el backend y el frontend, como sistemas de gestión empresarial, aplicaciones de comercio electrónico, dashboards interactivos, y aplicaciones de tiempo real. Las aplicaciones de línea de negocio (LOB) también pueden beneficiarse enormemente de la capacidad de Blazor para mantener un rendimiento sólido mientras permite actualizaciones en tiempo real sin recargar la página.

## Desafíos y futuras tendencias

Un desafío importante para Blazor WebAssembly es la optimización de rendimiento, especialmente en dispositivos móviles o con hardware limitado. Sin embargo, con los avances en WebAssembly y mejoras constantes por parte de la comunidad de .NET, es probable que veamos una adopción más amplia de esta tecnología en los próximos años. En el futuro, Blazor podría desempeñar un papel aún más significativo en la unificación de las experiencias de desarrollo, lo que lo posicionaría como una herramienta clave para aplicaciones interactivas de alta demanda.

# ****Conclusiones****

## Resumen de los puntos clave

Blazor se presenta como una tecnología disruptiva en el mundo del desarrollo web moderno, integrando C# tanto en el frontend como en el backend. Su principal ventaja radica en que permite a los desarrolladores de .NET trabajar en aplicaciones web sin tener que recurrir al uso de JavaScript para la lógica del lado del cliente, unificando así todo el ecosistema de desarrollo bajo un único lenguaje. La posibilidad de elegir entre dos modelos (Blazor Server y Blazor WebAssembly) le proporciona flexibilidad para adaptarse a distintos escenarios según las necesidades del proyecto, como la ejecución en el servidor o la capacidad de operar en el navegador sin dependencias del servidor.

No obstante, Blazor enfrenta ciertos desafíos, entre ellos el rendimiento en comparación con frameworks más maduros basados en JavaScript, como React o Angular. También, la limitación en cuanto al ecosistema de bibliotecas y herramientas de terceros, aunque esto está mejorando con el tiempo. A pesar de estas limitaciones, la posibilidad de reutilizar el código y la facilidad de integración dentro de ASP.NET Core lo convierten en una de las opciones más atractivas para los desarrolladores que ya tienen experiencia en .NET y desean incursionar en el desarrollo web sin cambiar de lenguaje o aprender nuevas tecnologías.

## Importancia de la tecnología

Blazor destaca por permitir a los desarrolladores de .NET aprovechar sus conocimientos en C# para crear aplicaciones web modernas, reduciendo la dependencia de tecnologías externas como JavaScript. Su integración con ASP.NET Core refuerza su potencial para aplicaciones empresariales.

## Sugerencias para investigaciones futuras

Futuros estudios podrían enfocarse en el análisis de rendimiento de Blazor WebAssembly frente a otras tecnologías basadas en WebAssembly, así como en la creación de herramientas y bibliotecas que aumenten la flexibilidad de este framework en proyectos más complejos.

Otro enfoque interesante para investigaciones futuras sería la creación de herramientas y bibliotecas adicionales que amplíen las capacidades de Blazor en proyectos más grandes y personalizados. El ecosistema de JavaScript cuenta con una vasta cantidad de herramientas y recursos que Blazor aún no iguala. La creación de bibliotecas avanzadas para gráficos, visualización de datos o gestión de estados dentro de Blazor podría mejorar su flexibilidad y posicionarlo como un competidor más robusto frente a los grandes frameworks de JavaScript.

# Referencias

* Microsoft Docs. (2023). Introduction to ASP.NET Core. Microsoft Documentation.  
  <https://learn.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core>  
  Descripción general de ASP.NET Core y su ecosistema.
* Fowler, M. (2022). Blazor: Building Web Applications in .NET. Manning Publications.  
  Explica en detalle cómo crear aplicaciones web interactivas con Blazor.
* Microsoft Docs. (2023). Blazor Overview.  
  <https://learn.microsoft.com/en-us/aspnet/core/blazor>  
  Información oficial sobre el funcionamiento y arquitectura de Blazor, tanto Server como WebAssembly.
* Holzner, S. (2021). ASP.NET Core 5 For Beginners. Apress.  
  Una guía práctica para principiantes que cubre ASP.NET Core 5, incluyendo Blazor y las APIs RESTful.
* Pro ASP.NET Core 6: Develop Cloud-Ready Web Applications Using MVC, Blazor, and Razor Pages. (2022). Apress.  
  Análisis avanzado de ASP.NET Core 6, enfocándose en Blazor y su integración con tecnologías modernas.
* Knutson, D. (2020). Blazor in Action. Manning Publications.  
  Expone un enfoque detallado para desarrollar aplicaciones interactivas con Blazor, incluyendo ejemplos prácticos.
* Esposito, D. (2022). Modern Web Development with ASP.NET Core and Blazor. Packt Publishing.  
  Describe el desarrollo web moderno con ASP.NET Core y cómo Blazor está cambiando el paradigma del desarrollo frontend.
* Microsoft Docs. (2023). Entity Framework Core Overview.  
  <https://learn.microsoft.com/en-us/ef/core/>  
  Introducción a Entity Framework Core y su integración con ASP.NET Core.
* Galloway, J., Wilson, B., & Roth, J. (2021). Professional ASP.NET Core 5. Wrox.  
  Cubre el uso avanzado de ASP.NET Core y los mejores patrones de diseño para el desarrollo de aplicaciones web.