Programación Web 3

UNLaM - Tecnicatura en Desarrollo Web

Trabajo Práctico de Investigación

**Net Blazor**

**Integrantes:**

* Leandro Alonso - 41693526
* Joel Escobar - 44107580
* Diego Arredondo - 31829648

Un dibujo de una cara feliz

Descripción generada automáticamente con confianza baja**Blazor** es un **framework** del equipo de **Microsoft ASP.NET** que permite a los desarrolladores escribir aplicaciones interactivas de una sola página (**SPA**) sin **JavaScript**.

El nombre está inspirando en **Browser + Razor** (que es un motor de vistas en **ASP.NET**).

Se podría decir que en simples palabras es **C#** en el área del Front-End/Cliente. Esto era algo que antes no era posible. Con este framework podemos escribir con el mismo lenguaje tanto en el Back-End como en el Front-End, haciendo que se desarrolle más rápido y con menos errores, haciéndolo Full Stack.

Una imagen de ejemplo:



En el ejemplo anterior tenemos un contador que se va a ir incrementando a cada pulsación del botón.

Los componentes **Blazor** se implementan en archivos**.cshtml** usando una combinación de **C#** y **HTML**.  
La interfaz de usuario de un componente se define usando **HTML** mientras que la lógica de representación dinámica como bucles, condicionales, expresiones, etc. se añaden usando una sintaxis de **C#** incrustada.

Texto

Descripción generada automáticamenteEn el bloque **@functions**, podemos definir todas las propiedades que se utilizan en la vista, y los métodos se vinculan con el control como un evento. Cuando se compila una aplicación Blazor, el HTML y la lógica de renderizado de C# se convierten en una clase componente, y el nombre de la clase generada coincide con el nombre del archivo.

Tiene la opción de ser ejecutado en el navegador del cliente o de ejecutarse en el servidor:

* **Blazor Server Side**
* **Blazor Web Assembly**

Ambas opciones son conceptualmente diferentes pero para nosotros solo es cuestión de configuración.

En **Blazer Server Side**, habrá un **BlazorUp** se ejecutará completamente en el **servidor** dentro de un **ASP.NET Web Side** que a si vez corre en el **.NET Runtime**. La primera vez que se navegue en el Browser, una conexión será establecida con SignalR que es una tecnología de Microsoft que permite que cualquier actualización sea resuelta por el servidor en tiempo real.

Esta aplicación del lado del servidor que mantendrá todo el tiempo en memoria, lo que se creará una instancia por usuario, dando una mayor velocidad al no tener que descargar casi ningún archivo.

En **Blazor WebAssembly** se ejecuta íntegramente en el lado del **cliente**, siendo soportado por todos los navegadores modernos.

Esta aplicación del lador del cliente que mantendrá en el browser, haciendo que la aplicación sea descargada por completo junto a cualquier otro archivo estático, haciendo que la primera carga sea lenta, pero por la memoria en chache las próximas se efectuaran con mayor velocidad. Con esto hará que cada acción que el usuario haga será resultado directamente en el cliente. Esto hace que no se efectué ninguna llamada al servidor, y también tiene como ventaja la escalabilidad ya que cada usuario ejecuta la aplicación en su navegador mientras que en Blazor Server cobra en el servidor, lo cual es un reto para la **escalabilidad**. Según Microsoft un buen servidor web puede ejecutar una aplicación de Blazor Server hasta 20 mil usuarios activos simultáneos sin ninguna degradación visible en el rendimiento.

Otra ventaja y gran diferencia del **WebAssembly** es que puede ser usado en modo offline como una **AWP** (Aplicación Web Progresiva). En cambio, con Blazor Server no puede hacerlo ya que requiere conexión permanente al servidor.

Luego está la facil combinación de WebAssembly con otras tecnologías del Back-End como **Node** **JS** o **GO**, mientras que Blazor Server está atado a ASP.NET Core. También Assembly puede ser publicado como archivos estáticos por lo que puede tener la aplicacion hosteada de manera gratuita, en cambio Blazor Server siempre necesita servidor, aumentando los costos. Lo bueno de ambas es que usan el mismo tipo de código, por lo tanto, se aprenden ambos a la vez.

**En la actualidad,** Microsoft plantea lanzar las ediciones **Blazor PWA** y **Blazor Hybrid**. El primero es compatible con [aplicaciones web progresivas](https://en.wikipedia.org/wiki/Progressive_web_app) (PWA). Este último es un marco nativo de la plataforma (a diferencia de un marco web), pero aún representa la interfaz de usuario utilizando tecnologías web (por ejemplo, [HTML](https://en.wikipedia.org/wiki/HTML) y [CSS](https://en.wikipedia.org/wiki/CSS)).

También se ha considerado un tercero, **Blazor Native**, un marco nativo de la plataforma que representa una interfaz de usuario nativa de la plataforma, pero no ha llegado a la etapa de planificación.

Con el lanzamiento de **.NET 5**, Blazor ha dejado de funcionar en Internet Explorer y la versión heredada de Microsoft Edge.

Blazor ofrece un competidor directo a los SPA de JavaScript como Angular, Vue y React, ya que, usando WebAssembly es capaz de escribir lógica UI usando C# y no JavaScript. Pero, aunque tiene potencial de acabar con la dependencia de JavaScript, este no va a desaparecer, porque todavía hay muchas cosas que WebAssembly aun no puede hacer, entonces para ayudar se usa la abstracción IJSRuntime, lo cual hace llamadas a funciones JavaScript. Incluso podemos hacer que funciones JS hagan llamadas al código C#.

Un ejemplo para llamar JS desde C# seria:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

O bien al revés, llamar a C# desde JS:

*Blazor:*

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

*JS:*

*Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente*

¿Entonces… Sustituira Blazor a JS?

Blazor es una alternativa, no puede hacer todo lo que JS hace, si Blazor no soporta una característica, puede ser necesaria una cantidad de código en JS para lograr cumplir el objetivo. Si algún día lo suplanta estará por verse, hasta ahora solo hay una alternativa.

En conclusión, Blazor es un Framework que deja escribir código C# en el Front-End con sus dos opciones:

**Blazor WebAssembly**; siendo un competidor directo a los frameworks como JavaScript. No necesita el uso de .NET en el servidor, pudiendo tener backend en Node, PHP o Rails, porque WebAssembly compila archivos estáticos.

**Blazor Server**; se puede usar para muchos escenarios, recomendados donde hay poca latencia y pocos problemas de conectividad es en aplicaciones de ittranet o aplicación de cara al publico de baja demanda

Referencias/Bibliografía

* [Blazor: Pasado, Presente y Futuro - Lautaro Carro](https://lautarocarro.blog/blazor-pasado-presente-y-futuro/)
* [▷ ¿Qué es Blazor? Ventajas y desventajas (elcamino.dev)](https://elcamino.dev/que-es-blazor/)
* [| ASP.NET Core Blazor Aprendizaje de Microsoft](https://learn.microsoft.com/en-us/aspnet/core/blazor/?view=aspnetcore-6.0)
* [Blazor - Wikipedia](https://en.wikipedia.org/wiki/Blazor)
* [Blazor: Qué es y características | OpenWebinars](https://openwebinars.net/blog/blazor-que-es-y-caracteristicas/)