Unidad 5

Blazor

Capitulo 1

Un enfoque moderno para el desarrollo web con .NET

Contenidos

- ¿Qué es Blazor?
- ¿Por qué elegir Blazor?
- Tipos de aplicaciones Blazor
- Arquitectura de una aplicación Blazor
- Demo básica – “Hola Mundo” en Blazor
- Práctica: Laboratorios
- Ventajas y desafíos de Blazor
- Referencias & Material Complementario

¿Qué es Blazor?

- Es un Framework desarrollado por Microsoft.
- Es de código abierto.
- Nos permite desarrollar aplicaciones web de una sola página (SPA) dinámicas y eficientes usando C# y Razor.
- Browser + Razor = Blazor

¿Por qué elegir Blazor?

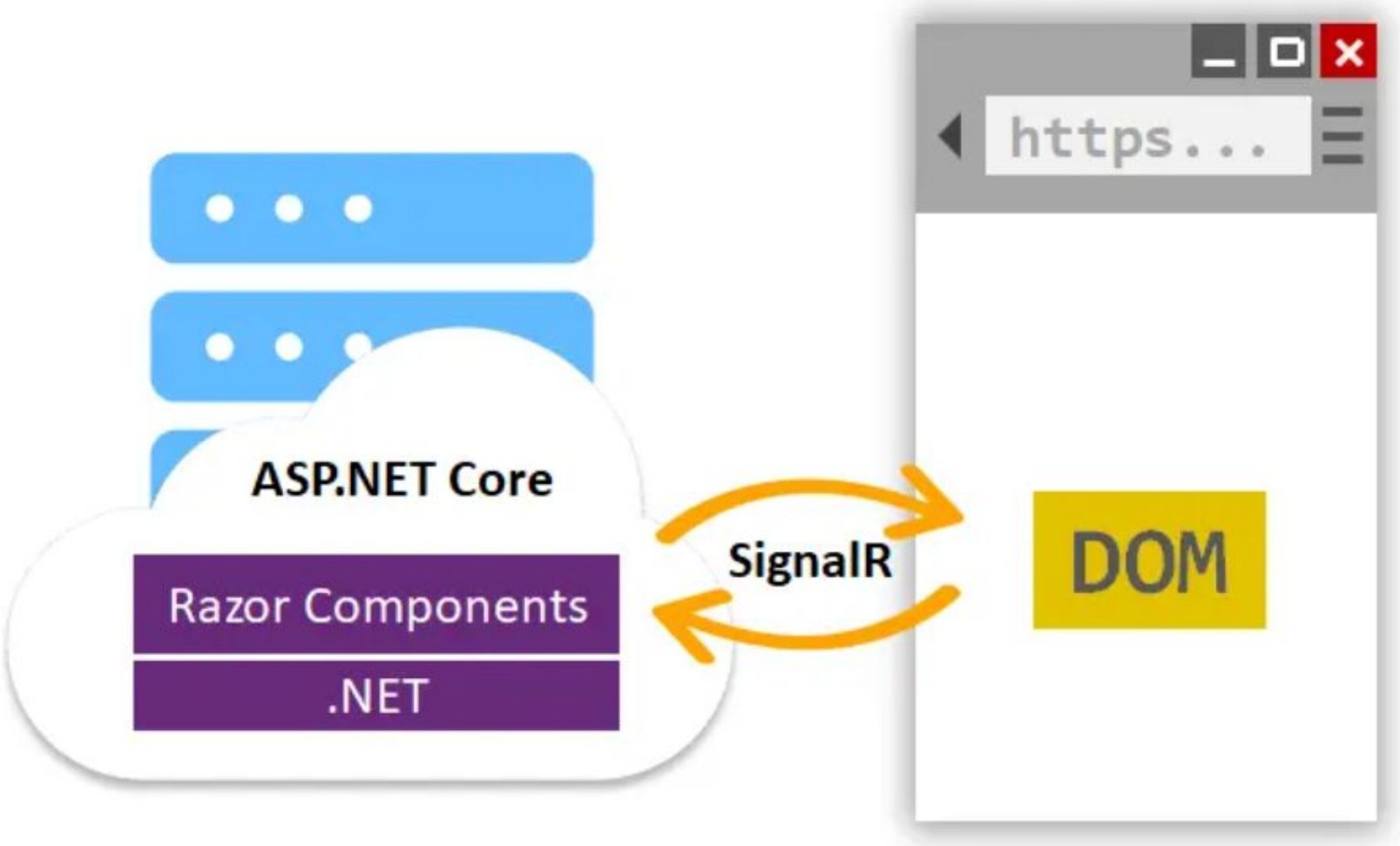
- **Código reutilizable**: Nos permite escribir código en C# que puede ser compartido entre el frontend y el backend.
- **Fácil de aprender**: Si ya conocemos C# y .NET, la curva de aprendizaje es suave.
- **Rendimiento nativo**: Ejecuta código directamente en el navegador usando WebAssembly (en “Blazor WebAssembly”).
- **Soporte completo para .NET**: Usa librerías .NET y aprovecha el ecosistema existente.

Tipos de aplicaciones Blazor

- **Blazor Server**: La lógica se ejecuta en el servidor y el HTML se actualiza en el navegador mediante SignalR.

- **Blazor WebAssembly (WASM)**: La aplicación se ejecuta directamente en el navegador, sin servidor (usa WebAssembly).

Blazor Server



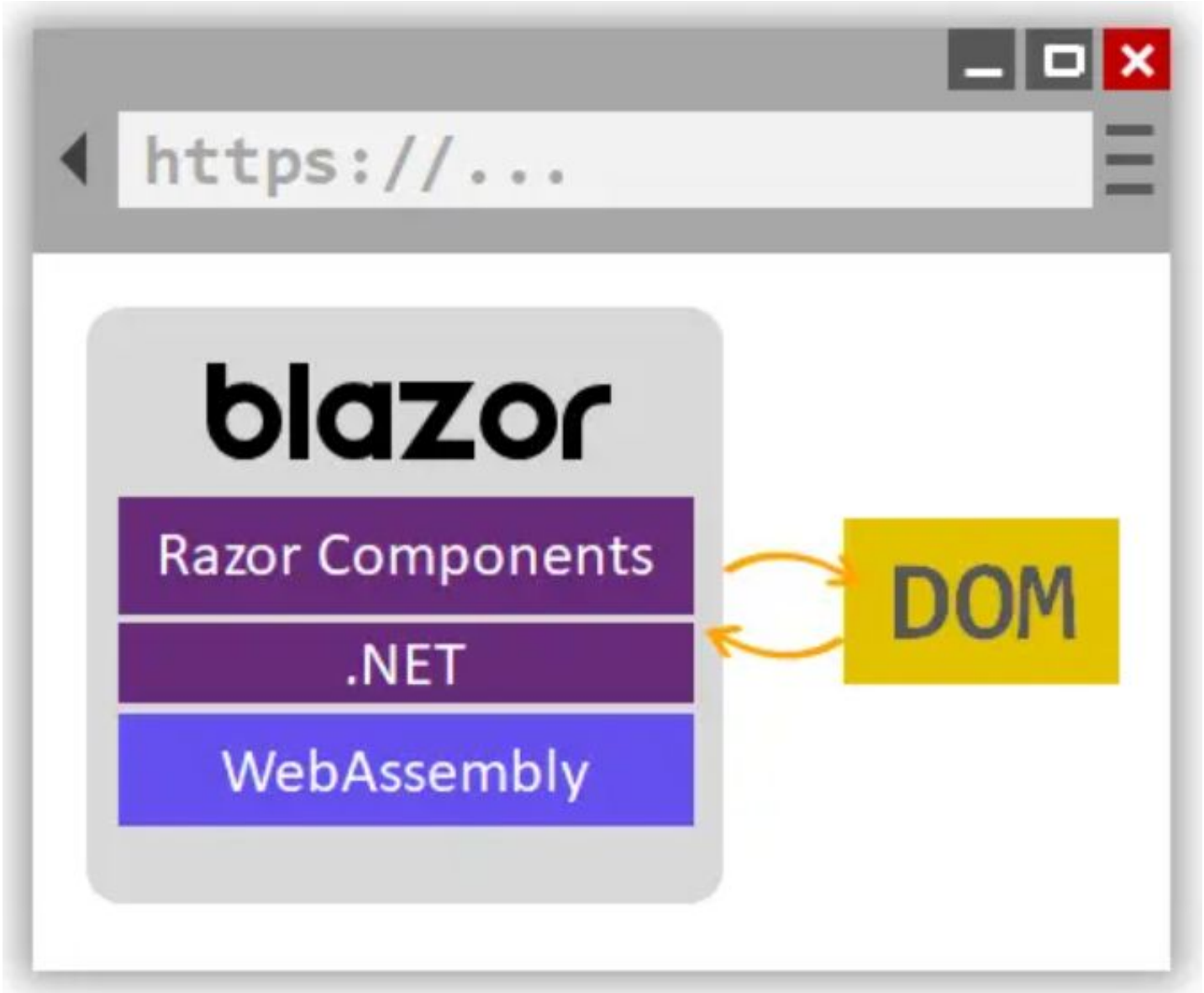
Blazor Server

- **Ejecución en el servidor:** Se ejecuta en el servidor y utiliza SignalR para comunicar cambios de la UI al navegador del cliente.
- **Interacción constante:** Cada vez que el usuario interactúa con la interfaz, esas interacciones se envían al servidor para ser procesadas, y luego las actualizaciones se devuelven al cliente. Esto introduce una ligera latencia.
- **Rendimiento inicial rápido:** La aplicación se carga rápidamente, ya que el navegador solo necesita descargar un pequeño archivo HTML y JavaScript para iniciar la conexión con el servidor.

Blazor Server

- **Estado en el servidor:** El estado de la aplicación se mantiene en el servidor, lo que permite aplicaciones que dependen mucho del servidor sin descargar todo al cliente.
- **Dependencia de la conexión:** Necesita una conexión constante y estable al servidor. Si la conexión se interrumpe, la aplicación deja de funcionar.
- **Escenarios ideales:** Aplicaciones que requieren acceso frecuente a datos del servidor o que no pueden descargar grandes cantidades de código al cliente.

Blazor WebAssembly (WASM)



¿Qué es WebAssembly?

- WebAssembly (Wasm) es un formato de código binario que permite ejecutar código de bajo nivel en navegadores web y otros entornos. proporcionando un rendimiento casi nativo.

- Fue diseñado para complementar JavaScript, permitiendo que lenguajes como C, C++, Rust, y otros, puedan ser compilados y ejecutados en el navegador de manera eficiente y segura.

Blazor WebAssembly (WASM)

- **Ejecución en el cliente:** Todo el código de la aplicación (incluyendo .NET) se descarga y ejecuta en el navegador del usuario. Esto se hace a través de WebAssembly.
- **Modo offline:** Una vez descargada la aplicación, puede funcionar incluso sin conexión a internet, siempre que no dependa de un backend para obtener datos dinámicos.
- **Desempeño inicial:** Puede tener tiempos de carga más largos al inicio, ya que se debe descargar todo el código y las dependencias en el navegador.

Blazor WebAssembly (WASM)

- **Interacciones:** No hay comunicación constante con el servidor una vez que la aplicación está cargada, excepto para llamar APIs u obtener datos. Las interacciones con la UI son rápidas, ya que todo ocurre localmente.
- **Escenarios ideales:** Aplicaciones que requieren más interactividad, que puedan ejecutarse completamente en el cliente y donde la conectividad no sea una limitante constante.

Arquitectura de una aplicación Blazor

- **Componentes**
- **Páginas (Razor Pages)**
- **Rutas**
- **Binding de datos**

Arquitectura de una aplicación Blazor

- **Componentes**: Son los bloques principales de una aplicación Blazor. Escrito en C# y Razor.

```
<h3>Contador</h3>

<p>Cuenta actual: @count</p>

<button @onclick="Incrementar">Incrementar</button>

@code {
    private int count = 0;

    private void Incrementar()
    {
        count++;
    }
}
```

Arquitectura de una aplicación Blazor

- **Páginas:** Son un tipo especial de componente. Se asocian a una ruta.

```
@page "/contador"

<h3>Contador</h3>
<p>Cuenta actual: @count</p>
<button @onclick="Incrementar">Incrementar</button>

@code {
    private int count = 0;

    private void Incrementar()
    {
        count++;
    }
}
```

Arquitectura de una aplicación Blazor

- **Rutas**: Manejo de navegación entre componentes.
  - Directiva @page.
  - Enrutamiento dinámico con parámetros.
  - Múltiples rutas en un solo componente.
  - Enrutamiento con parámetros opcionales (“?”).

Importante: Blazor nos permite implementar una aplicación con el enfoque de una sola página (SPA), lo que significa que cuando los usuarios navegan entre diferentes rutas, el navegador no recarga toda la página; en su lugar, Blazor actualiza solo la parte necesaria.

Arquitectura de una aplicación Blazor

- **Binding de datos**: Sincronización entre la interfaz de usuario y los datos.
  - **One-Way Data Binding**: Enviar datos desde el código hacia la vista.
  - **Two-Way Data Binding**: Enlazar los datos entre la vista y el modelo, ideal para formularios.

```
<h3>Ingresa tu nombre:</h3>
<input @bind="nombre" />

<p>Hola, @nombre</p>

@code {
    private string nombre = "";
}
```

Demo básica – “Hola Mundo” en Blazor

- Crear un proyecto Blazor en Visual Studio.
- Explicación del código inicial:
  - Program.cs,
  - App.razor,
  - MainLayout.razor.
- Cómo se muestran los componentes y la interacción básica.
- Referencia a ejemplo Microsoft:
  - https://dotnet.microsoft.com/es-es/learn/aspnet/blazor-tutorial/intro

Laboratorios

Ventajas y desafíos de Blazor

- **Ventajas**:
  - No necesitas aprender JavaScript.
  - Integración perfecta con el ecosistema .NET.
  - Reutilización de código en cliente y servidor.

- **Desafíos**:
  - Tamaño de descarga en Blazor WebAssembly.
  - Limitaciones del entorno de WebAssembly en el navegador.

Referencias & Documentación Complementaria

Documentación Oficial Microsoft

- Repositorio Oficial:
  - https://github.com/dotnet/blazor

- Videos
  - https://www.youtube.com/playlist?list=PLdo4fOcmZ0oXNZX1Q8rB-5xgTSKR8qA5k

- Documentación Oficial
  - https://learn.microsoft.com/en-gb/aspnet/core/blazor/?view=aspnetcore-8.0&WT.mc_id=dotnet-35129-website

¿Preguntas?