Blazor

UI framework para .Net

Índice

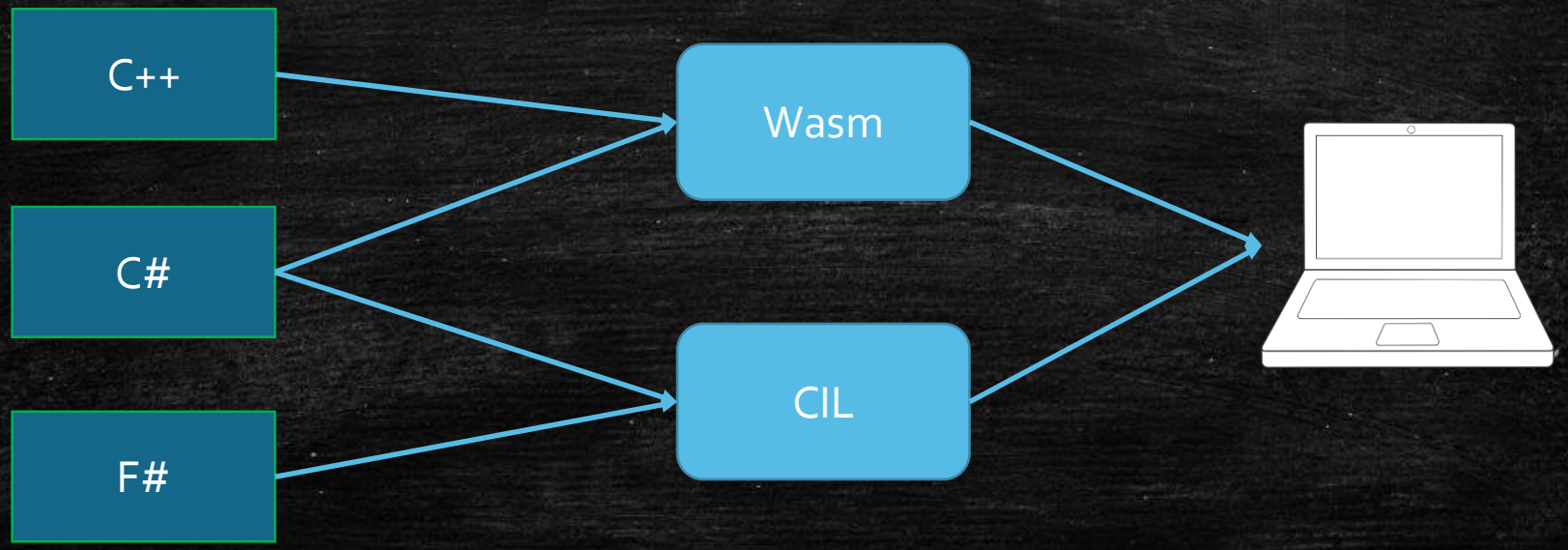
- Introdução
- WebAssembly
- Blazor WebAssembly
- Server Side Blazor
- Vantagens e Desvantagens

Introdução

- Framework para desenvolver Single Page Applications(SPA)
- O nome é uma combinação de Browser com Razor (framework da Microsoft para criar views HTML)
- Tem como objectivo possibilitar escrever código do lado do cliente em C# em vez de Javascript
- Duas vertentes de funcionamento: Blazor WebAssembly(ainda em Beta) e Server-Side(pronto para produção)
- Projecto open source

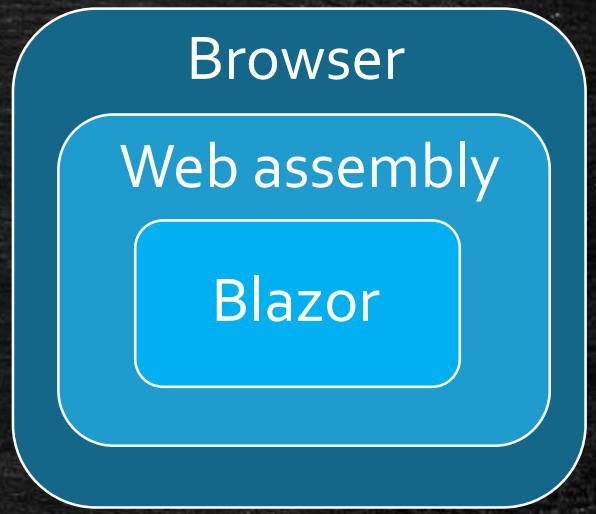
WebAssembly

- Formato para instruções binária que podem ser compreendidas por qualquer entidade configurada para tal (browser, máquina, etc)
- Tem um comportamento análogo ao CIL, ou seja, dado um determinado código fonte este é convertido para um formato intermédio que é depois convertido para binário.



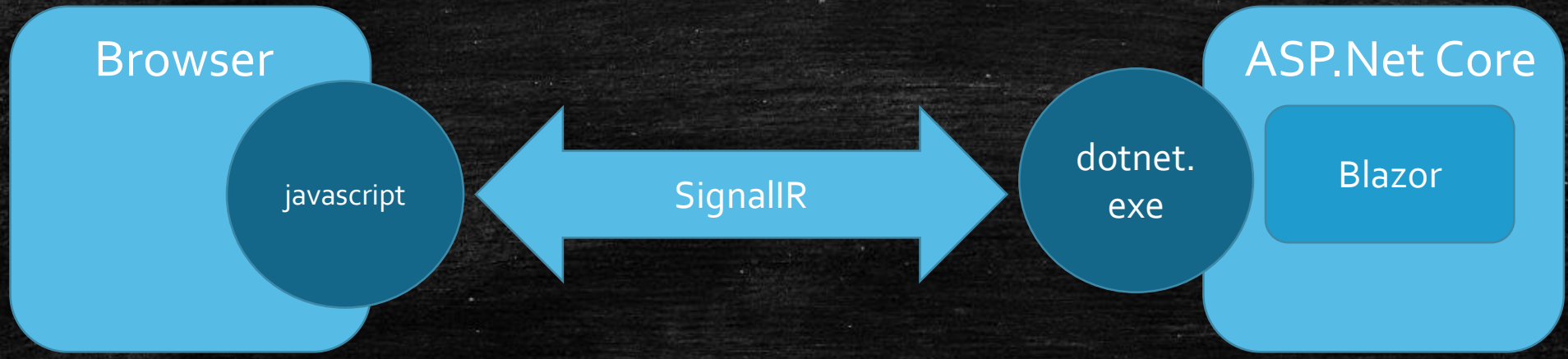
Blazor WebAssembly

- Permite correr código C# do lado do cliente
- Permite correr DLL's .Net no browser sem qualquer plugin
- Aplicações Blazor WebAssembly correm directamente no browser recorrendo a um runtime .Net customizado para poder tirar Partido do WebAssembly.
- É feito o download deste runtime com a applicação, o assembly da mesma e as suas dependências.
- Os componente da UI não são renderizados directamente no DOM mas numa representação do mesmo chamada RenderTree. Quando é detectada uma alteração/evento é calculada uma diferença que é então aplicada ao DOM



Server Side

- A lógica de cliente pode estar a correr no lado do servidor e recorrendo à framework de notificação em tempo real SignalIR, o cliente pode comunicar com o servidor obtendo as alterações que serão reflectidas no DOM.
- Segundo este modelo a aplicação é executada no servidor num ambiente .Net Core e todas as interações(alterações na UI, gestão de eventos ou chamadas a código Javascript) são geridas recorrendo a uma ligação SignalIR



Vantagens

WebAssembly

- WebAssembly corre no cliente, pelo uma aplicação ser publicada recorrendo a ficheiros estáticos.
- Blazor WebAssembly pode funcionar em modo offline. Quando a ligação ao servidor é perdida, a aplicação do lado do cliente pode continuar a funcionar, mas não tendo forma de obter nova informação
- Reduzida carga do lado do servidor

Server-Side

- Páginas HTML são pré-renderizadas no lado do servidor, levando a uma inicio rápido.
- Pode correr em browsers mais velhos, dado que não há requisites referentes ao WebAssembly

Desvantagens

WebAssembly

- É necessário fazer download de todas as dependências, levando a um primeira utilização mais lenta
- Blazor Wasm não suporta(por enquanto) multiplas threads, pelo que o processamento ocorre na thread responsável pela UI, sendo que as chamadas ao servidores são feitas assincronamente.
- Blazor Wasm apenas funciona nos browsers mais recentes.

Server-Side

- Escalabilidade pode ser um problema pois o servidor tem de gerir múltiplas ligações SignalIR e alterações de estado.
- Pode haver má experiência de utilizador: havendo latência alta e utilização lenta caso haja muitos eventos a serem lançados, dado que implica constantes chamadas ao servidor.
- Não existe possibilidade de funcionar em modo offline.