CENTRO UNIVERSITÁRIO GERALDO DI BIASE

BRUNO SEIXAS
LUIS GULHERME RAMPASO
LUIZ FERNANDO ZAMBOTI
MIRIÃ ODETE
VINICIUS MOTA

PESQUISA SOBRE A ESTRUTURA BLAZOR

Volta Redonda
2024

BRUNO SEIXAS
LUIS GULHERME RAMPASO
LUIZ FERNANDO ZAMBOTI
MIRIÃ ODETE
VINICIUS MOTA

PESQUISA SOBRE A ESTRUTURA BLAZOR

Pesquisa apresentada ao curso de Sistemas de informação, CENTRO UNIVERSITÁRIO GERALDO DI BIASE.

Professor: Rosenclever Gazoni

Volta Redonda
2024

Resumo

Este projeto de pesquisa foca no estudo e aplicação do Blazor, um framework de desenvolvimento web que permite a criação de aplicações interativas usando a linguagem C#. O Blazor permite a execução de código C# diretamente no navegador, graças ao suporte do WebAssembly em todos os navegadores modernos. Este projeto visa explorar as capacidades do Blazor para criar interfaces de usuário avançadas, compartilhar a lógica do aplicativo entre o servidor e o cliente, e renderizar a interface do usuário como HTML e CSS. Além disso, o projeto também investigará o uso de componentes Blazor, que são classes .NET que permitem uma lógica de renderização da interface do usuário flexível, manipulação de eventos do usuário, e a possibilidade de serem aninhados, reutilizados e distribuídos. Através deste projeto, pretendemos demonstrar as vantagens do uso do Blazor em comparação com outras tecnologias de desenvolvimento web, como a sintaxe de marcação de modelo Razor baseada em C#. O objetivo final é desenvolver uma aplicação web interativa robusta usando o Blazor, demonstrando assim seu potencial e eficácia como uma ferramenta de desenvolvimento web.

Sumario

Introdução

O desenvolvimento web moderno está evoluindo com o Blazor, um framework que transforma a maneira como criamos aplicações web interativas. Utilizando a linguagem C# e a plataforma .NET, o Blazor permite aos desenvolvedores construir UIs ricas e dinâmicas, executar lógica de aplicativo de forma compartilhada e beneficiar-se das vantagens do ecossistema .NET. Com a capacidade de escrever código C# em vez de JavaScript, o Blazor simplifica o desenvolvimento e oferece uma alternativa poderosa para frameworks tradicionais.

O que é Blazor

Blazor é um framework da Microsoft para a criação de aplicações web interativas utilizando C# em vez de JavaScript. Assim como Django no Python, o Blazor facilita a construção de aplicações completas, mas o faz usando a plataforma .NET e a linguagem C#.

Funcionalidades do Blazor

1- Criação de UIs interativas com C#

- Com Blazor, é possível criar interfaces de usuário avançadas utilizando C# graças ao suporte ao WebAssembly (WASM). O WebAssembly permite que os navegadores executem código binário compilado (como DLLs .NET), possibilitando a execução de código C# no cliente. Além disso, o Blazor permite a interação com JavaScript através de JavaScript Interop, caso seja necessário.

2- Compartilhamento de lógica entre servidor e cliente

- Uma das grandes vantagens do Blazor é a capacidade de compartilhar a lógica de negócios entre o cliente e o servidor, utilizando a mesma base de código na plataforma .NET. Isso simplifica o desenvolvimento e a manutenção de aplicações.

3- Renderização de interface como HTML e CSS

- O Blazor renderiza a interface do usuário utilizando HTML e CSS, garantindo compatibilidade com todos os navegadores modernos, incluindo navegadores móveis. Isso assegura que as aplicações Blazor possam ser executadas em diversas plataformas sem problemas.

4- Modalidades de hospedagem

- **Blazor WebAssembly (WASM):** Executa diretamente no navegador usando WebAssembly. Isso permite que o código C# seja baixado e executado no cliente.

- **Blazor Server:** Executa no servidor e usa SignalR para comunicação em tempo real com o cliente. A interface do usuário é atualizada via conexões SignalR, proporcionando uma experiência de aplicação de página única (SPA).

Características dos componentes no Blazor

1- Baseado em componentes

- Os aplicativos Blazor são compostos de componentes. Esses componentes são classes .NET que podem ser encapsuladas em assemblies .NET, facilitando a modularização e reutilização de código.

2- Definição de lógica de renderização

- Cada componente define sua lógica de renderização, permitindo criar interfaces de usuário dinâmicas e flexíveis.

3- Manipulação de eventos do usuário

- Os componentes podem manipular eventos do usuário, como cliques de botões, entradas de texto, etc.de maneira eficiente.

4- Aninhamento e reutilização

- Componentes podem ser aninhados dentro de outros componentes e reutilizados em diferentes partes da aplicação, promovendo a modularidade.

5- Compartilhamento e distribuição

- Componentes podem ser compartilhados e distribuídos como bibliotecas de classes Razor ou pacotes NuGet, facilitando a integração e a reutilização em múltiplos projetos.

6- Ferramentas de desenvolvimento

- **Blazor CLI**: Ferramenta de linha de comando para criar e gerenciar projetos Blazor.

- **Visual Studio e Visual Studio Code:** Suporte completo para desenvolvimento, depuração e publicação de aplicações Blazor.

- **Live Reload:** Funcionalidade que permite ver mudanças em tempo real sem a necessidade de recompilar a aplicação.

Diferenças entre Blazor e Razor

Razor: Razor é uma sintaxe de marcação de modelo baseada em C#. Você pode escrever (principalmente) marcação HTML padrão e jogar algum C# nela para lidar com coisas como exibir dados e executar lógica de visualização. O caso de uso comum para Razor na última década tem sido alimentar as Visualizações do ASP.NET MVC. Quando você escreve uma Visualização MVC (a parte V do MVC), normalmente usa a sintaxe Razor para vincular dados à sua marcação. Além da simples vinculação de modelo, o Razor também permite que você use expressões condicionais, como instruções if. No núcleo, o Razor permite que você pegue a marcação HTML padrão e a misture com dados vindos de outro lugar.

Blazor: Blazor é um framework que aproveita os componentes Razor para produzir HTML dinâmico. A maior diferença entre Razor e Blazor é que Razor é uma linguagem de marcação com C#, enquanto Blazor é o framework que permite que você execute código

C# e use o mecanismo de visualização Razor no navegador. Blazor é destinado a combinar ideias da pilha Razor .NET atual com a arquitetura de framework SPA moderna. A criação de código é destinada a ser flexível e incentiva layouts baseados em componentes. Blazor é esperado para ser rápido, porque webAssembly é rápido. Ele compila para bytecode que é executado diretamente pelo carregador wasm do navegador.

Vantagens do Blazor

O Blazor, um framework inovador para desenvolvimento web, oferece diversas vantagens que o tornam uma escolha atraente para os desenvolvedores. Vamos explorar essas vantagens em detalhes:

- Simplificação do Desenvolvimento: Em vez de precisar dominar duas linguagens diferentes (por exemplo, C# para o back-end e JavaScript para o front-end), os desenvolvedores podem focar em uma única linguagem: o C#. Isso reduz a complexidade e facilita o trabalho, especialmente para equipes menores.
- Reutilização de Código: Com o Blazor, partes do código podem ser reutilizadas tanto no servidor quanto no cliente. Isso evita a duplicação de esforço e melhora a manutenção do código ao longo do ciclo de vida do projeto.
- Execução Nativa: O Blazor WebAssembly é uma linguagem de baixo nível que pode ser executada quase na velocidade de código nativo. Em comparação, o Javascript é uma linguagem interpretada e, portanto, mais lenta.
- Eficiência: O Blazor WebAssembly é projetado para ser eficiente em termos de tempo de execução e uso de memória. Isso resulta em aplicações mais rápidas e responsivas para os usuários.
- Sem Necessidade de Plugins: As aplicações Blazor WebAssembly funcionam em qualquer navegador moderno sem a necessidade de plugins adicionais. Isso simplifica a experiência do usuário e reduz a carga de manutenção para os desenvolvedores.
- Compatibilidade: Qualquer usuário com um navegador moderno pode acessar a aplicação Blazor, sem precisar instalar plugins ou extensões. Isso aumenta a acessibilidade e amplia a base de usuários potenciais.
- Segurança Aprimorada: Menos dependências de plugins significam menos vulnerabilidades potenciais. Eliminar plugins reduz os vetores de ataque, melhorando a segurança geral da aplicação.

Em resumo, o Blazor oferece uma abordagem unificada, eficiente e segura para o desenvolvimento web, tornando-o uma escolha promissora para projetos modernos.

Desvantagens do Blazor

O Blazor, apesar de suas vantagens, também apresenta algumas desvantagens que os desenvolvedores devem considerar. Vamos explorar esses pontos:

- Tamanho da Aplicação Inicial (Blazor WebAssembly): As aplicações Blazor WebAssembly tendem a ter um tamanho inicial maior devido ao runtime do .NET que é baixado junto com a aplicação. Isso pode resultar em tempos de carregamento mais longos, especialmente em conexões de internet mais lentas.
- Suporte de Navegador e Compatibilidade: Embora o WebAssembly seja suportado pela maioria dos navegadores modernos, pode não ser totalmente compatível com navegadores mais antigos ou versões específicas. Isso pode limitar a audiência da aplicação ou exigir soluções alternativas para garantir a compatibilidade.
- Performance: O Blazor Server pode mitigar algumas questões de desempenho ao fazer o processamento no servidor, mas isso pode introduzir latência devido à comunicação constante entre o cliente e o servidor. Para aplicações em tempo real ou com alta interatividade, essa latência pode ser uma limitação. Por outro lado, o Blazor WebAssembly pode ter limitações de desempenho em comparação com aplicações JavaScript nativas devido ao overhead do runtime .NET e à execução em um ambiente sandbox.
- SEO (Search Engine Optimization): Aplicações SPAs (Single Page Applications), incluindo aquelas desenvolvidas com Blazor, podem enfrentar desafios em termos de SEO. O conteúdo é frequentemente carregado dinamicamente via JavaScript, o que pode ser menos eficaz para indexação por motores de busca. Embora o Blazor Server possa mitigar parte desse problema, ainda pode não ser tão eficiente quanto aplicações renderizadas no servidor.
- Ecossistema e Bibliotecas: O ecossistema do Blazor está crescendo, mas ainda é relativamente novo em comparação com frameworks JavaScript estabelecidos, como React, Angular e Vue. Isso pode resultar em menos bibliotecas, componentes de terceiros e recursos disponíveis para os desenvolvedores.

- Aprendizado e Adoção: Para desenvolvedores com experiência principalmente em desenvolvimento web tradicional (HTML, CSS, JavaScript), aprender Blazor e C# pode ter uma curva de aprendizado. Além disso, a adoção de uma nova tecnologia pode exigir investimento significativo em treinamento e adaptação.

Em resumo, embora o Blazor ofereça vantagens notáveis, é importante considerar essas desvantagens ao escolher a tecnologia para um projeto específico.

Conclusão

O Blazor, como um framework emergente para desenvolvimento web, apresenta uma série de vantagens e desafios que os desenvolvedores devem considerar cuidadosamente ao escolher a tecnologia para seus projetos. Nesta conclusão, abordaremos os principais pontos discutidos anteriormente e destacaremos as implicações práticas do uso do Blazor.

Vantagens do Blazor:

- Simplificação do Desenvolvimento: A unificação da linguagem (C#) para o back-end e o front-end simplifica o fluxo de trabalho dos desenvolvedores. Equipes menores podem se beneficiar significativamente dessa abordagem, reduzindo a curva de aprendizado e melhorando a produtividade.
- Reutilização de Código: Compartilhar lógica de aplicativo entre o servidor e o cliente é uma vantagem substancial. Evitar a duplicação de esforço economiza tempo e facilita a manutenção do código.
- Execução Nativa e Eficiência: O Blazor WebAssembly, com sua execução quase nativa, oferece desempenho comparável a código nativo. Isso supera a interpretação do JavaScript e resulta em aplicações mais rápidas e responsivas.
- Compatibilidade e Segurança: A ausência de plugins adicionais nas aplicações Blazor WebAssembly melhora a segurança. A compatibilidade com navegadores modernos amplia a base de usuários potenciais.

Desafios do Blazor:

- Tamanho Inicial da Aplicação (Blazor WebAssembly): O tamanho inicial maior das aplicações WebAssembly pode afetar os tempos de carregamento. Estratégias de otimização, como carregamento assíncrono, são essenciais para mitigar esse problema.
- Performance e SEO: O Blazor Server pode introduzir latência devido à comunicação constante entre cliente e servidor. A indexação por motores de busca em aplicações SPAs requer atenção especial.
- Ecossistema e Curva de Aprendizado: O ecossistema do Blazor ainda está em crescimento, com menos bibliotecas e recursos disponíveis em comparação com frameworks JavaScript estabelecidos. A adoção do Blazor pode exigir investimento em treinamento e adaptação.

Considerações Finais:

O Blazor oferece uma alternativa promissora para o desenvolvimento web, especialmente para aqueles familiarizados com a plataforma .NET. Ao pesar as vantagens e desafios, os desenvolvedores devem avaliar cuidadosamente as necessidades específicas de seus projetos. A escolha entre Blazor Server e Blazor WebAssembly dependerá do contexto, dos requisitos de desempenho e da equipe de desenvolvimento. Com uma abordagem estratégica, o Blazor pode ser uma ferramenta poderosa para criar aplicações web modernas e eficientes.

Referências Bibliográficas

1. Antlia. “Razor e Blazor: os principais componentes do .NET MAUI.” Disponível em: [Razor e Blazor: os principais componentes do .NET MAUI](https://antlia.com.br/artigos/razor-e-blazor-os-principais)
2. MICROSOFT. Blazor | Criar aplicativos Web cliente com C# | .NET disponivel em (https://dotnet.microsoft.com/pt-br/apps/aspnet/web-apps/blazor)
3. Wladimilson M. Nascimento. NET Core 3.0 - Razor Components | Blog da TreinaWeb disponivel em : (https://www.treinaweb.com.br/blog/net-core-3-0-razor-components/)

4. MICROSOFT Estrutura do projeto do aplicativo Blazor do ASP.NET Core | Microsoft Learn disponivel em : (https://learn.microsoft.com/pt-br/aspnet/core/blazor/project-structure?view=aspnetcore-8.0)