**ASP.NET**

**Contexto Histórico:**

* **Asp Clássico**: lançado em 1996, tinha uma semelhança com o PHP e seu código era feito em VBScript.
* **Asp**.**NET WEBFORMS:** Lançado em 2002 junto com o .Net Framework. Tinha a proposta de atrair os desenvolvedores desktop. Passou a utilizar o POO.
* **ASP**.**NET MVC:** Lançado em 2009, foi lançado o padrão MVC.
* **ASP**.**NET CORE**: Anunciado em 2014 as primeiras versões do asp.net core.

**ASP.NET CORE**

Existem três tópicos que fazem com que o **ASP.NET CORE** seja tão especial.

* **Inovação**: Com a nova versão do asp.net core, ele deixou aquela impressão “lenta” da sua versão anterior. Agora, possuindo uma plataforma muito rápida, superando até mesmo alguns concorrentes de mercado.
* **Open Source**: Grupos que trabalham para cada vez mais deixar o .Net Open Source.
* **Multiplataforma**: Estratégias feitas pela Microsoft para fazer com que o **Core** rode em todas em plataformas.

**.NET CLI**

**Command Line Interface**, através dele conseguimos montar projetos pela tela de comando. Com ela, consigo criar qualquer tipo de projeto, arquivo através da linha de comando.

**.NET STANDARD LIBRARY**

É a biblioteca que dá suporte ao **ASP.NET CORE.**

**O .NET STANDARD LIBRARY** é uma especificação que documenta como o .net se comporta. Já os outros frameworks (.NET Core .NET Framework), “colocam” em prática as “exigências”.

**GERENCIANDO VERSÕES**

**Usando a tela de comando(CMD):**

**Tela de comando**:

**dotnet - -info**. (mostra todas as informações do SDK).

**dotnet -h.** (mostra tela de ajuda **COM TODOS OS COMANDOS**).

Ao mostrar todos as versões do projeto instalada no computador, será visualizado diversas versões do RUNTIME. Porém, não tem nenhum problema, não temos nenhum padrão. Definimos um padrão no projeto.

Geralmente essas informações se encontram em **global.json**.

**SEMPRE TER A BOA PRÁTICA DE INFORMAR NO GLOBAL JSON A VERSÃO DO SDK OBTIDA EM “dotnet - -info.”**

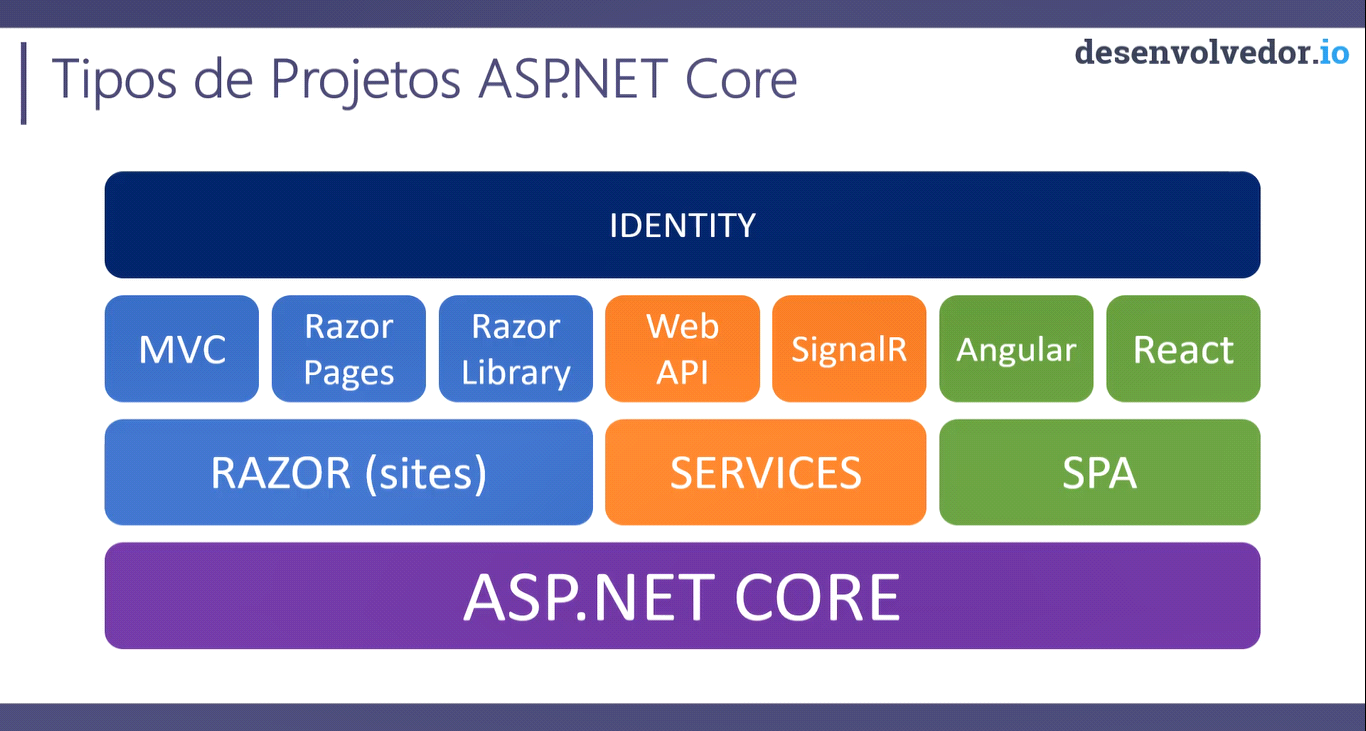
**TIPOS DE PROJETOS ASP.NET CORE**

O **ASP.NET CORE** é uma tecnologia para web oferecendo alguns templates específicos para o desenvolvimento.

O ASP.NET CORE é um framework que proporciona tudo para a criação dos projetos, seja serviços. Ela está subdivida em 3 partes:

* **Razor:** MVC, Razor Pages, Razor Library.
* **Services:** Web API , SignalIR.
* **SPA:** Angular, React.

Por último, não menos importe **IDENTITY:**

* Serve para identificar um usuário, autentificar, delegar atribuições para o usuário. Atendendo todas as tecnologias.

**ESTRUTURA DO PROJETO**

* **Connected Services:** Onde podemos conectar serviços existentes, Azure, Secure Secrets dentre outros.
* **Dependencies:** Parte responsável pelas dependências do projeto dívida em três partes.
* **Analyzers:** dependências de análises de performance do VS.
* **NuGet:** responsável pelas entregas de pacotes do ASP.NET, features.
* **SDK:** suporte à tecnologia.
* **Properties (lauchSetting.**json): O arquivo json é configurar como a sua aplicação será iniciada no VS. Nesses arquivos, conseguimos alterar para ambiente de produção ou desenvolvimento.
* **Wwwroot:** Responsável por ter os arquivos estáticos do programa, sendo eles texto puro, binários, que não necessitem de compilação. (css, js, lib).
* **Appsetting.**json: Configuração específicas para ambientes específicos.
* **Program.**cs: Tem como função dizer como o programa irá funcionar.
* **Startup.**cs: Configuração do modelo mvc, injection dependence e etc.

**ASP.NET POR BAIXO DOS PANOS**

* **Owin (Open Web Interface for .NET):** é uma especificação de como separar o servidor da aplicação. Antigamente em outras versões do .net, o servidor(IES) era junto com a aplicação(ASP.NET). A partir do core, eles foram separados, independente do host, possuindo seu próprio host, customizado.
* **Middleware:** Middleware são componentes de software em uma aplicação ASP.NET Core. Seriam como barramentos que são passados por fluxos lógicos. Ele pode ser trabalhado em conjunto com outros middleware. Exemplo: Asp.NET CORE é um pipeline. Esse pipeline possui diversos fluxos, cada um responsável por uma lógica, ex: MVC Core, Identity, Logging etc.

**INTERFACES**

* **IConfiguration**: responsável por gerenciar as constraints do ASP.NET Core, ele obtém as informações pelas chaves configuradas.
* **IServiceCollection**: É uma coleção de serviços. Essa interface é estendida por classes que implementam Middlewares. Aqui adicionamos itens ao pipeline do asp.net Core.
* **IApplicationBuilder**: É uma interface que implementa os métodos de configuração para cada Middleware adicionado.
* **IHostingEnvironment**: Conseguimos definir os ambientes que estamos rodando e ela se comporta de acordo com esse ambiente. Development and Client.

**CRIANDO UM MIDDLEWARE**

* Tem que ter uma associação com o RequestDelegate;
* Devo fazer a injeção de dependencia do Request;
* E um método invokeAsync, que retorna uma task.
* A ordem importa, pois dependendo de qual middleware for chamado primeiro, ele não pode ter um desempenho correto.