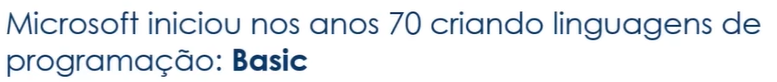
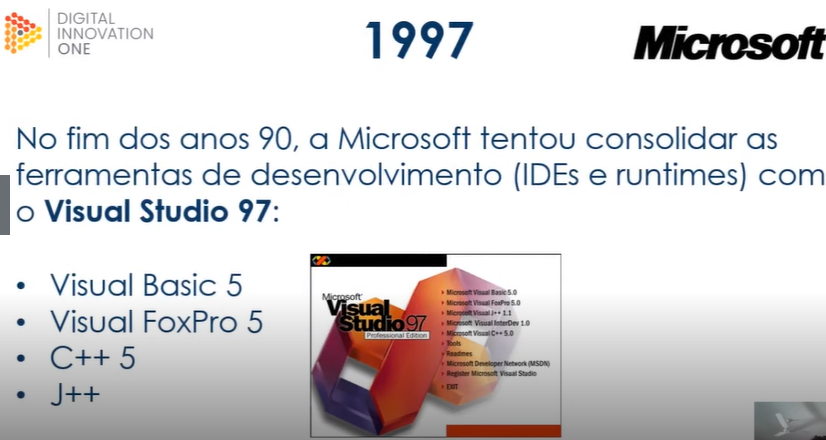
**CONHECENDO O .NET**

**Passado**



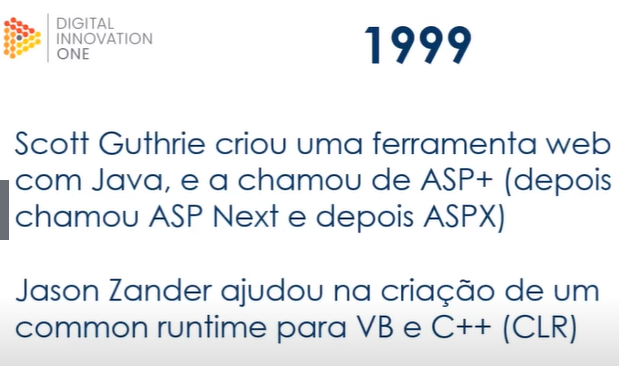


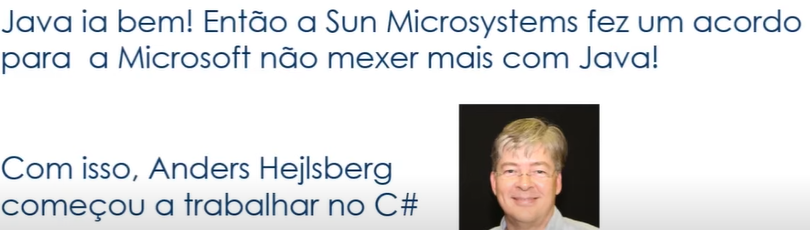
Nessa época o VS tinha várias linguagens, mas cada uma trabalhava com seu runtime próprio.



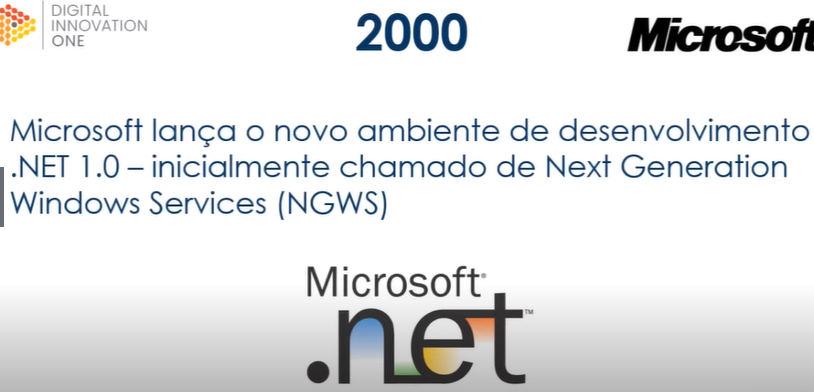
Nesse pacote já começa a se falar em um runtime consolidado.







Essa plataforma foi criada pensando no Windows, pois a Microsoft já tinha aqui seus sistemas operacionais Windows. Os runtimes geravam a linguagem de máquina para plataforma Windows.



Esse senhor não trabalhava para a Microsoft e resolver criar esse projeto MONO a partir da observação ou análise do funcionamento do .NET uma vez que ele não tinha acesso ao código fonte do Sistema. A ideia aqui era criar um paralelo que fossem open source e multiplataforma.

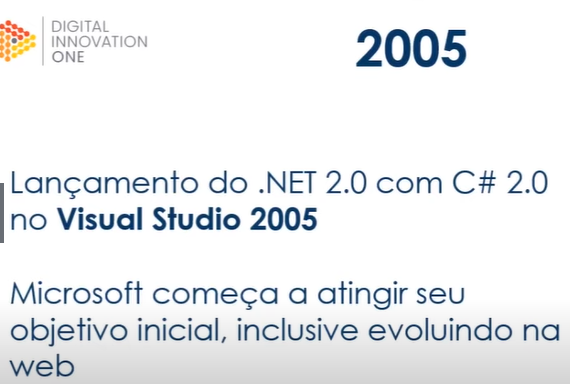




Nessa versão ocorreram diversas melhorias na CLR, inclusive com a versão CLR2. Nessa imagem podemos ver diversas linguagens todas com seus compiladores próprios e que geram uma linguagem intermediária denominada CIL, essa linguagem é interpretada pelo CLR que irá converter para a linguagem de máquina. Essa linguagem intermediária sendo interpretada pelo CLR foi o que permitiu a Microsoft ter várias linguagend em sua plataforma .NET.



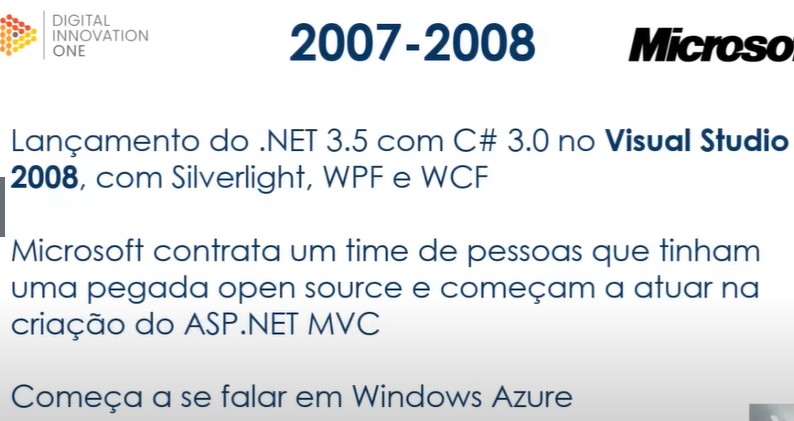
Nesse lançamento já estava melhor consolidado a parte do ASP Net que é a frente web do .NET. Era uma plataforma mais difundida.



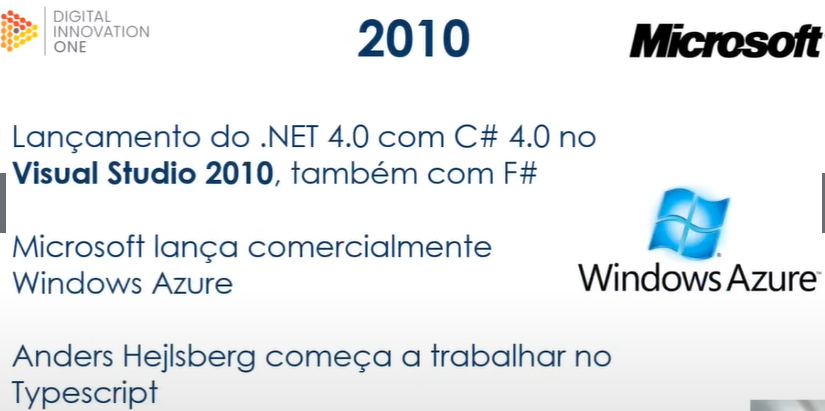
WPF: É uma versão mais moderna para criação de aplicação Desktop.

WCF: É para serviços WEB.

O Windows Azure começa a ser falado como sendo um sistema operacional para aplicação em cloud (nuvem).



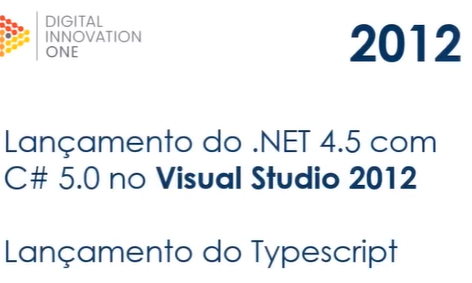
C# é orientada a objeto, enquanto que o F# a uma linguagem a funcional. Nesse momento ainda não existe o produto typescript.



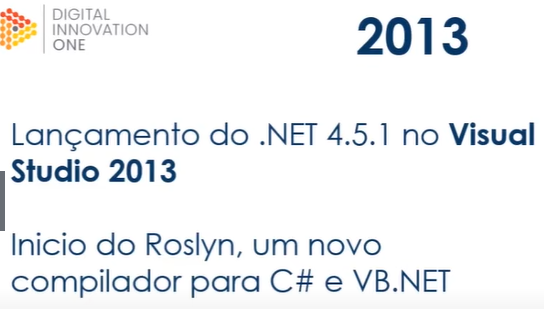
O senhor Miguel do projeto MONO, usando esse projeto MONO conseguiram desenvolver aplicativo para rodar em Android e IOS desenvolvendo em C#. Nesse ponto o senhor Miguel ainda não faz parte da equipe Microsoft, e todo seu desenvolvimento é pela sua empresa.



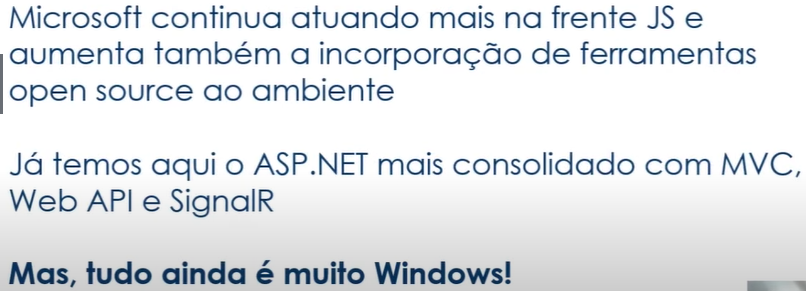
Nesse lançamento já temos o produto typescript. Fato histórico o CEO dessa época era totalmente voltado a ambiente Windows, com declarações polemicas sobre Linux.



Esse Roslyn é um compilador para C# e VB.NET, desenvolvido em C#.



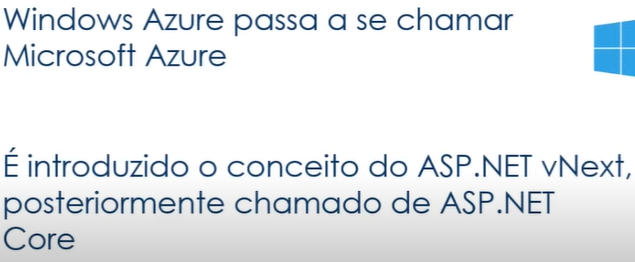
Ainda em 2013, a Microsoft continua atuando nesse frente Java Script já tinha criado o Typescript. O ASP.NET MVC é um novo Framework Web alternativo ao ASP.Net Webforms, ele era mais simples, mas direto e com menos acoplamento ao mundo Windows. O Web API que é voltado para serviço Web. O SignaIR que é um Framework para ter comunicação real time como exemplo CHAT de conversas instantâneas.



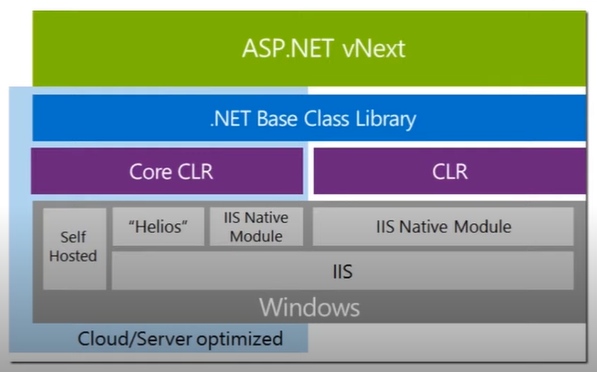
Com seu novo CEO a Microsoft deixa de ser uma empresa focada em Sistema Operacional e passa a ser uma empresa focada em tecnologia para desenvolvimento em Cloud.



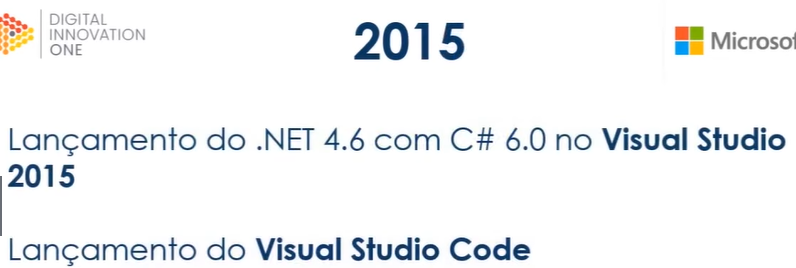
Ainda em 2014 o Windows Azure deixa de ser apenas um Sistema Operacional para ser uma plataforma de serviços para Cloud.



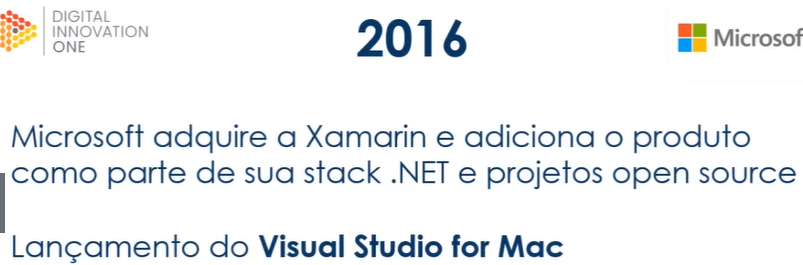
A ideia com o ASP.NET Core é que ele teria uma biblioteca base do .NET que seria compartilhada tanto pelo CLR padrão quando pelo novo Core CLR que trataria dessas novas funcionalidades mais focado em Cloud. Lembrando que até esse momento ainda se falava muito em Windows e muito pouco em multiplataforma.



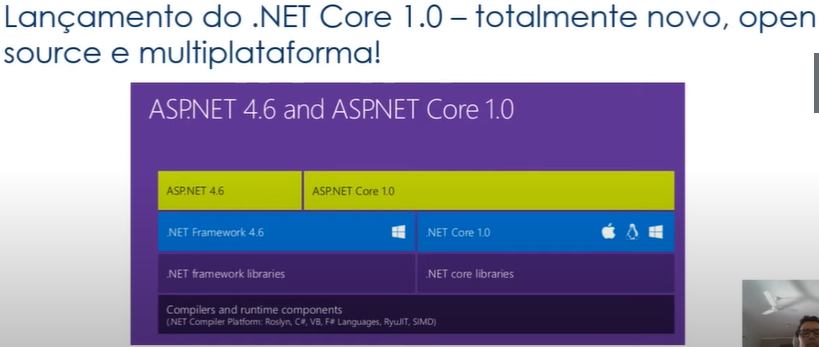
Microsoft continua evoluindo no open source. O VS code é um editor de código muito poderoso, você pode adicionar diversos plug-ins, pode desenvolver em .NET, Java, para Front End e muitas outras possibilidades. Ele é Open Source e multiplataforma ou seja já roda além do Windows, no Linux e MacOS.



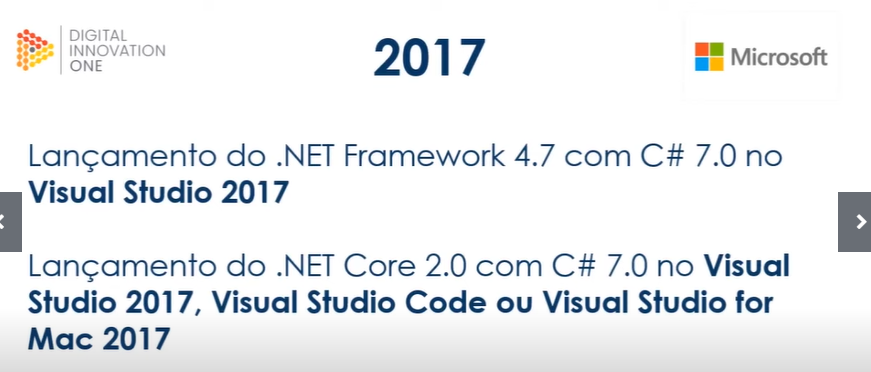
Microsoft compra a Xamarin do Sr. Miguel de Icaza havia criado em 2011. O Visual Studio só rodava em Windows, com esse novo lançamento passa a ter uma Visual Studio for Mac, não com todas as funcionalidades da versão para Windows. Com ele você pode desenvolver aplicativos para celulares IOS, uma vez que era necessário o MAC OS para desenvolver para IOS, uma amarração da Apple.



Ainda em 2016, surge o .NET Core 1.0 é um novo Framework, criado do zero, totalmente open source e multiplataforma. Houve uma mudança muito grande para todos que trabalhavam ou acompanhavam os lançamentos da Microsoft.

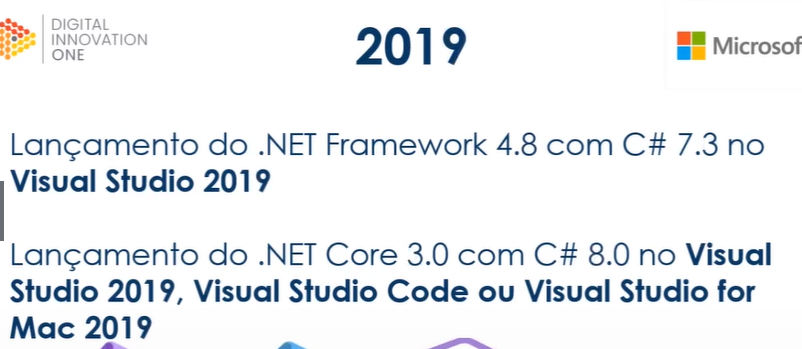


O .NET Framework somente funciona em Visual Studio e voltando apenas para Windows. Enquanto que o .Net Core pode usar com VS do Windows, Visual Studio Code no Linux e Visual Studio for Mac para Mac OS.

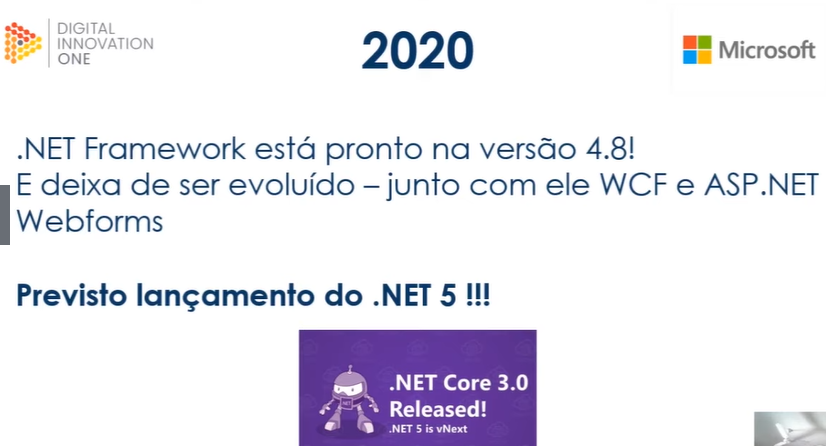


**Presente – Aula foi grava em 2019 então o presente começa daqui pra frente.**

Até antes desse momento os Framework vinham sendo atualizados com a mesma versão do C#, a partir de 2019 houve uma mudança. Isso ocorreu segundo a Microsoft, pois o .NET ficou pronto, ou seja, atingiu a sua total maturidade na versão 4.8. Com isso o produto não será mais evoluído e juntamente com ele o WCF e ASP.NET Webforms, deixando ambos de existir no .Net Core.

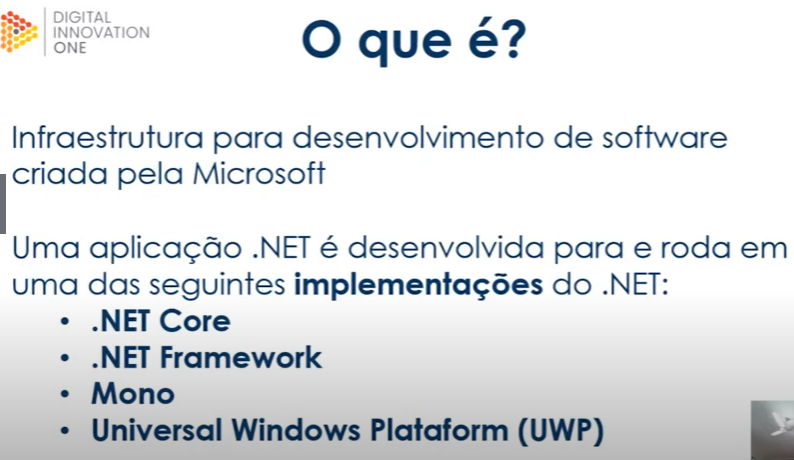


**Futuro**

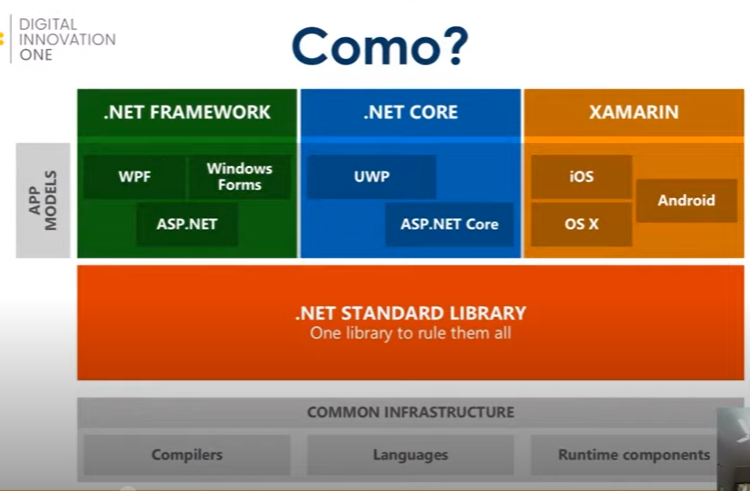


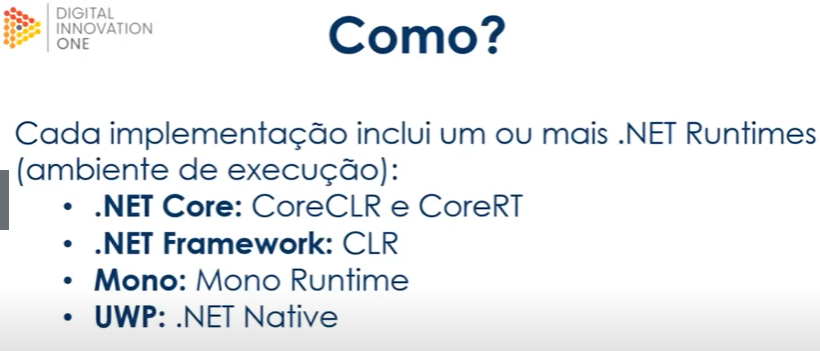
**Como e aonde usar .NET**

Sendo o UWP para internet das coisas e Xbox por exemplo.



Como o .NET pode comunicar com outras versões, ou três implementações como a .NET Framework, a Xamarin e .NET Core. Existe um “contrato” denominado “.NET Standard, basicamente ele diz quais funcionalidades tem em uma das versões que vc encontrará na outra.





A Microsoft desenvolve e suporta essas três linguagens:

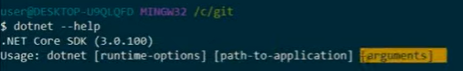
* C# é a principal, orientada a objeto.
* F# que a funcional em outras aulas foi dita quer era para Mac.
* VB existe muitas aplicações mantida em VB NET.





**Conhecendo a CLI**

* Dotnet –help: Comando que mostra a documentação de todos os comandos que podem ser executados via prompt.
* Dotnet –version: mostra a versão que está instalada.
* A estrutura de comandos é .NET “dotnet [runtime-options] [path-to-application] [arguments]”



Outra forma também com .NET seria “dotnet [sdk-options] [command-options] [arguments]”

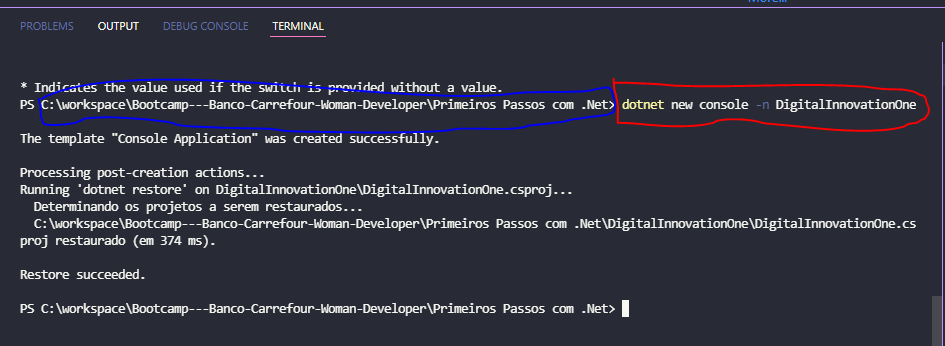


* Dotnet –info: Mostra todas as versões instaladas na estação, traz informação do SO.
* NUGET: É um gerenciador de pacotes para .NET, eu criei uma ferramenta ou um trecho e código que poderia ser aproveitado em outros projetos e quero tornar ele público, ou posso subir também um nuget privado. Existem outras gerenciadores de pacotes.
* Publish: para gerar as .dll para subir no servidor IIS e subir o seu site, por exemplo.
* Restore para restaurar os pacotes do nuget.
* Run ele build e já roda a sua aplicação.
* Test: Projeto de teste por exemplo que testa sua aplicação.
* Tool: Comando novo que permite que instale e gerencia as ferramentas para .NET.

**Criando uma aplicação Console**

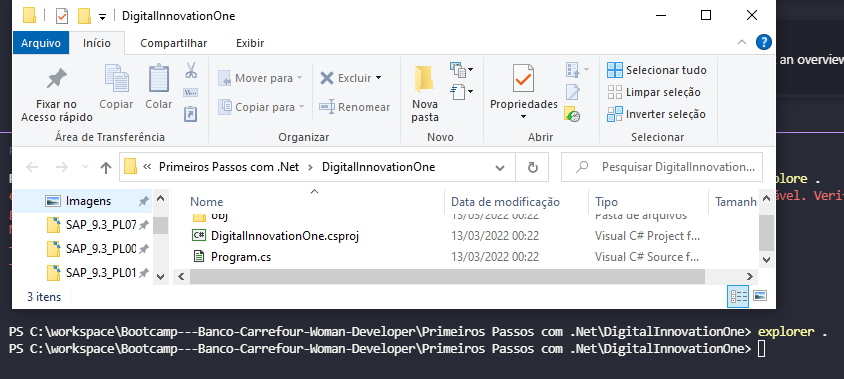
No terminal de console que poderia ser via CMD do Windows, mas usei por dentro do VS Code, posicionei na pasta desejada e dei o comando para criar um projeto novo. Nesse caso o projeto era um console, mas poderia ser qualquer outro tipo, para selecionar outro tipo de projeto bastava dar o comando.

* dotnet net -h
* dotnet new console –n nome\_do\_projeto

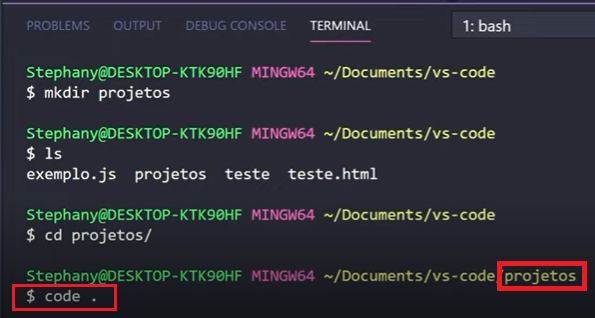


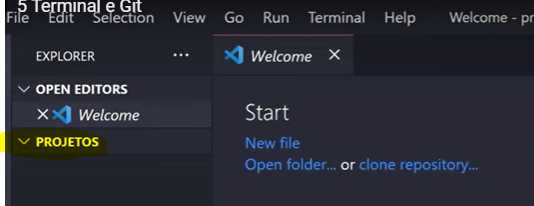
O comando para abrir a pasta via Windows é

* explorer .

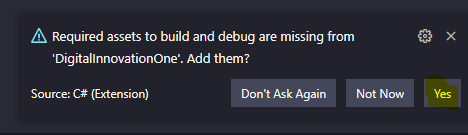


* O comando “CODE .” abre o vscode na pasta que vc estiver, veja o exemplo estou na pasta projeto ao executar o comando abriu o vs code na pasta projeto.





Ao abrir um projeto você pode receber uma alerta pedindo para adicionar uns arquivos build e debug. Pode clicar em YES.



Foi criado duas pastas. Onde a pasta bin são os binários da minha aplicação



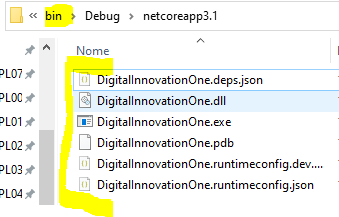
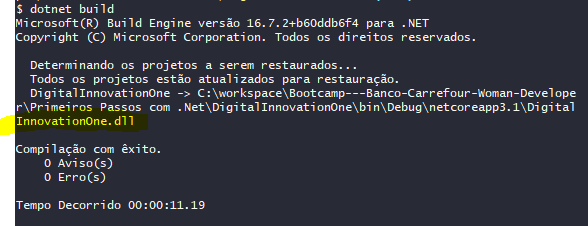
Quando eu executo os comandos

* dotnet build

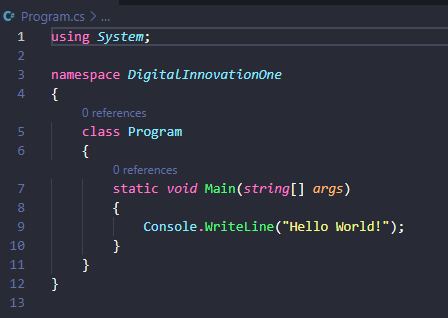
ou

* dotnet run que executa o build antes

ele faz todo o processo de compilação e transforma o meu código fonte em uma linguagem intermediária e o CLR irá criar os binários. Ele informa que foi gerado a .dll e o .exe

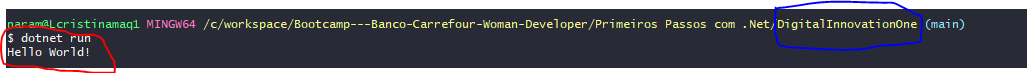
 

Para executar minha aplicação que escreve a mensagem “Ola mundo”.

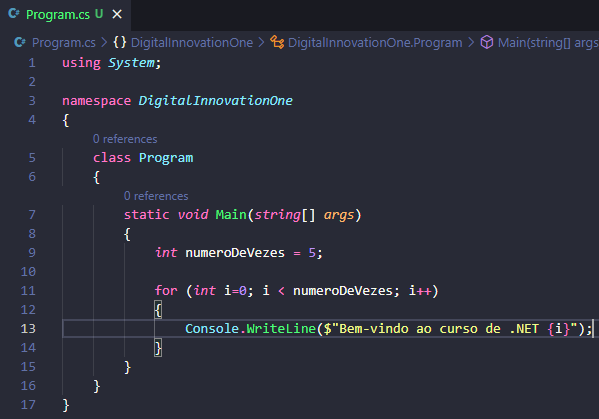
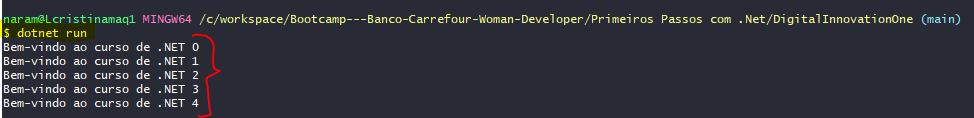


Dentro da pasta do projeto execute o comando

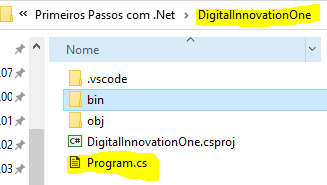
* dotnet run



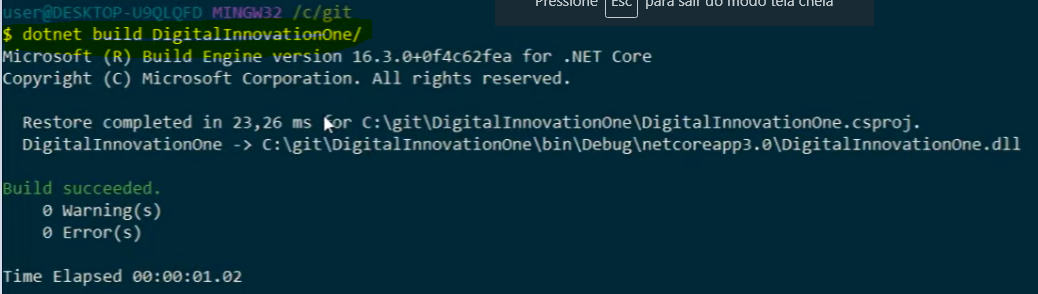
Novo programa e nova execução.

Lembrando que você deve executar os comandos RUN e BUILD na mesma pasta que esteja seu projeto, ou seja, seu arquivo “.CS” que é a extensão C#.



Caso não esteja no mesmo nível de pasta e queira executar um BUILD basta passar a pasta que está o arquivo do projeto ou .cs



Caso queira rodar o comando RUN é necessário passar o parâmetro “-p” e então colocar a pasta que está o arquivo do projeto.

