


O Minicurso de hoje é focado nas habilidades necessárias para o Hackathon. Também haverá suporte para quem quiser utilizar o Visual Studio Code.

1 – GIT

Vamos começar o segundo dia do Minicurso com o uso do github.

1 – Para aqueles que não possuem uma conta no github, [clique aqui e crie uma](#).

2 – Acesse o endereço: <https://github.com/italoaguiar/Minicurso-Asp.net-Core>

3 – Procure o botão  e clique nele. Isso cria uma cópia do projeto para a sua conta para que você possa trabalhar nela e posteriormente enviar suas alterações para o repositório de origem.

4 – Um popup se abre quando você faz parte de mais de uma organização. Neste caso escolha seu nome.

5 – Clique em  e copie o endereço escrito no campo que aparece: 

6 – No próximo passo é necessário ter o git instalado em seu computador. Caso você não possua, clique [aqui e baixe o instalador](#).

6 – Abra o cmd, powershell ou algum terminal de sua preferência e navegue para a pasta que você deseja colocar o projeto para trabalhar. No cmd você pode utilizar `cd nomeDaPasta` para entrar em um diretório, `cd ..` para voltar para o diretório anterior e `mkdir nomeDaNovaPasta` para criar uma nova pasta.

7 – Digite o comando abaixo e pressione enter:

```
Microsoft Windows [versão 10.0.18362.418]
(c) 2019 Microsoft Corporation. Todos os direitos reservados.

C:\Users\italo>cd Documents
C:\Users\italo\Documents>mkdir Github
C:\Users\italo\Documents>cd Github
C:\Users\italo\Documents\Github>git clone endereçoCopiadoNoPasso5
C:\Users\italo\Documents\Github>git checkout -b minhaBranch
```

Entre no explorer e abra a pasta que você clonou o projeto e veja os arquivos que foram baixados.

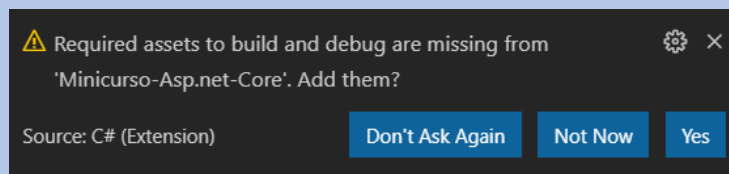
2 – Uma API RESTFUL

Abra o arquivo DeliveryWebApp.sln no Visual Studio Community



1 – Abra a pasta do projeto que você clonou no Visual Studio Code

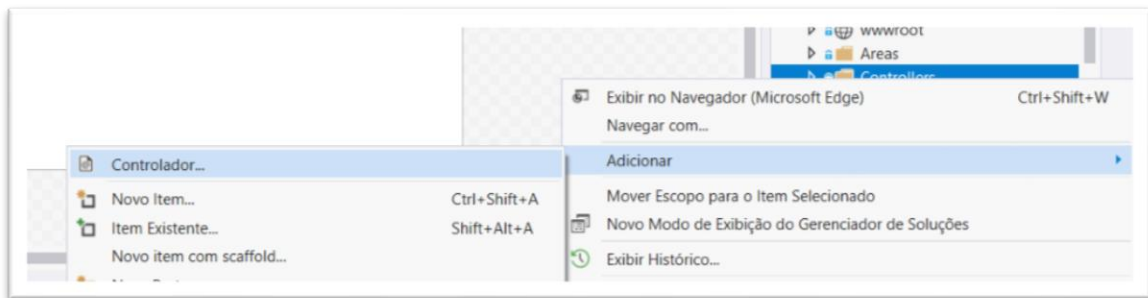
2 – Possivelmente o Visual Studio Code pode sugerir que você instale algum recurso adicional para acessar alguma funcionalidade. Neste caso, clique em sim (Yes).



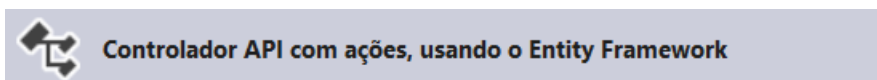
3 – Abra um novo terminal através do menu Terminal/New Terminal e execute o comando **dotnet restore**.

```
PS C:\Users\italo\source\repos\Minicurso-Asp.net-Core> cd DeliveryWebApp
PS C:\Users\italo\source\repos\Minicurso-Asp.net-Core\DeliveryWebApp> dotnet restore
```

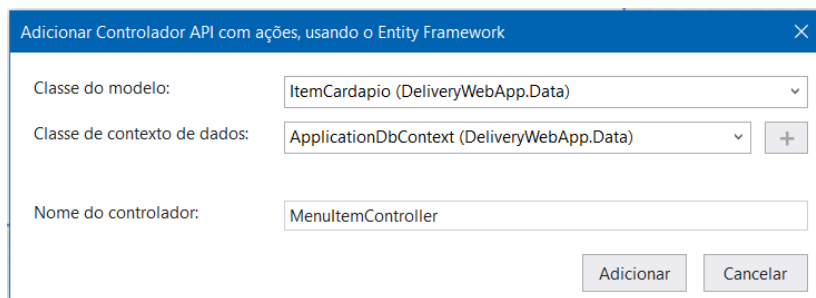
Vamos agora criar um novo controller. Para isso Clique com o botão direito sobre a pasta **Controllers** e selecione **Adicionar/Controlador...**



Selecione a opção abaixo e clique em adicionar:



Preencha o formulário da seguinte forma e clique em adicionar:



Execute o projeto clicando em 

Abra o navegador e acesse <https://localhost:44330/api/menuitem> . Observe que o número da porta pode variar de acordo com o seu projeto.

1 – No terminal do VS Code, execute o comando a seguir:

```
ps> dotnet aspnet-codegenerator --project . controller -name MenuItemController -m ItemCardapio -api -outDir Controllers -dc ApplicationDbContext
```

4 – Para executar o projeto insira o comando **dotnet run** e em seguida abra o endereço <https://localhost:5001/api/menuitem> no seu navegador.

Vamos abrir o arquivo **MenuItemController.cs** na pasta **Controllers**. Vamos observar o primeiro método desta classe:

```
// GET: api/MenuItem
[HttpGet]
public async Task<ActionResult<IEnumerable<ItemCardapio>>> GetItemsCardapio()
{
    return await _context.ItemsCardapio.ToListAsync();
}
```

O atributo [HttpGet] informa que esta requisição é do tipo GET. Este atributo pode ser alterado para [HttpPost], [HttpPut], [HttpDelete] ,etc. Dentro deste método temos um acesso ao banco de dados. Da forma escrita, todos os itens do cardápio serão retornados pela API quando esse método for chamado.

Vamos filtrar um pouco estes resultados. Reescreva o método da forma abaixo:

```
// GET: api/MenuItem
[HttpGet]
public async Task<ActionResult<IEnumerable<ItemCardapio>>> GetItemsCardapio()
{
    return await _context.ItemsCardapio
        .Where(x=> x.Nome.Contains("a") && x.Preco < 20) //itens cujo nome contem 'a' e preço < 20
        .Take(5) //limita o numero de resultados a 5
        .ToListAsync();
}
```

Execute o projeto e acesse o endereço `api/menuitem` para visualizar os resultados filtrados.

Os demais métodos desta classe abordam a criação de novos itens, alteração de itens existentes ou a exclusão permanente de itens do banco de dados. Dica: Você pode baixar o software [PostMan](#) para testar cada um destes métodos e compreender seu funcionamento.

3 – GIT (Segunda Parte)

Agora que fizemos várias modificações no nosso projeto, vamos salvar essas modificações no nosso repositório do github.

No prompt cmd ou no terminal do Visual Studio code, digite os comandos a seguir:

```
Microsoft Windows [versão 10.0.18362.418]

(c) 2019 Microsoft Corporation. Todos os direitos reservados.

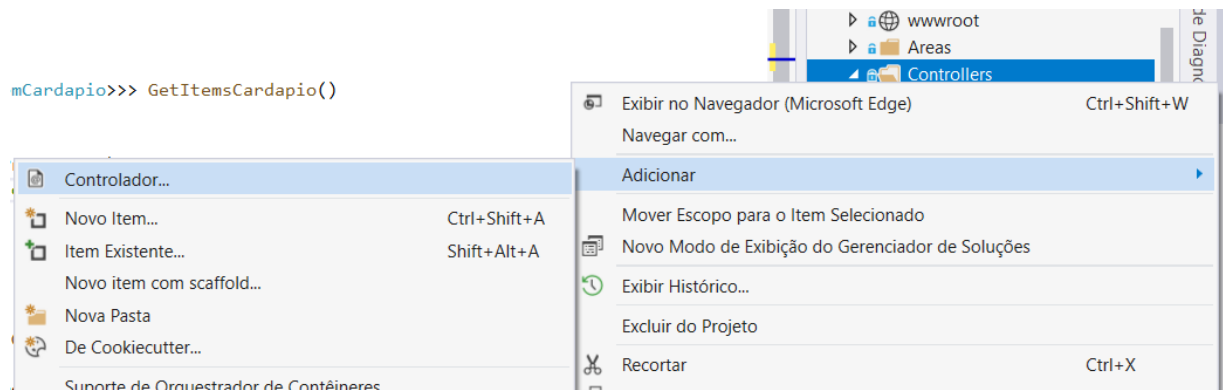
> git add *
> git commit -m "Descrição das minhas alterações"
> git push origin minhaBranch
```

Se você nunca utilizou estes comandos antes, ao executar o último comando, uma janela de login será aberta para que você se autentique antes de enviar suas alterações.

4 – Uma API personalizada

Nas etapas anteriores nós criamos uma API Rest para uma de nossas tabelas em nosso banco de dados. Mas também podemos criar uma API para funções específicas independente de banco de dados.

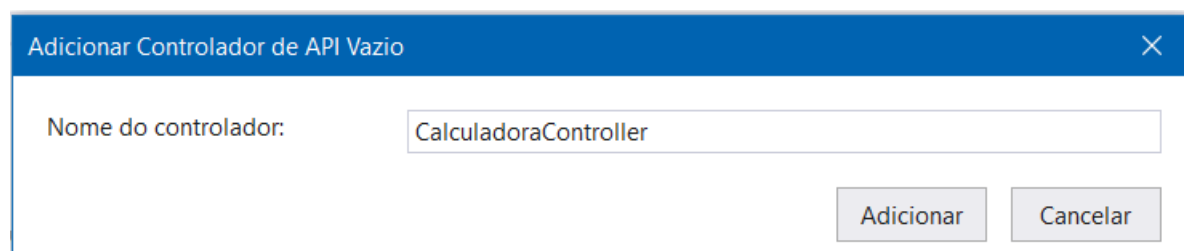
Vamos criar um novo controlador. Clique com o botão direito sobre a pasta Controllers e selecione Adicionar/Controlador...



Escolha e clique em adicionar:



Atribua o nome abaixo e clique em adicionar:



1 – No terminal do VS Code, execute o comando a seguir:

```
ps> dotnet aspnet-codegenerator --project . controller -name  
CalculadoraController -api -outDir Controllers
```

Vamos abrir o arquivo **CalculadoraController.cs** na pasta **Controllers**.

Dentro da classe **CalculadoraController** adicione o seguinte método:

```
[HttpGet("{a}/{b}")]  
public ActionResult<double> Soma(double a, double b)  
{  
    return a + b;  
}
```

Execute o projeto, abra o navegador e acesse <https://localhost:5001/api/calculadora/1.333/1.557>

Se tudo correr bem, a resposta será 2.8899999999999997.

Vamos adicionar um novo método conforme a imagem a seguir:

```
[HttpGet("ParesEntre/{a}/{b}")]
public ActionResult<List<int>> ParesEntre(int a, int b)
{
    List<int> list = new List<int>();

    for(int i = a; i < b; i++)
    {
        if(i%2 == 0)
        {
            list.Add(i);
        }
    }

    return list;
}
```

Execute o projeto, abra o navegador e digite: <https://localhost:5001/api/calculadora/ParesEntre/1/50>

Se tudo correr bem, a resposta será:

[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48]

5 – GIT (Terceira Parte)


Envie suas alterações para seu repositório:

```
Microsoft Windows [versão 10.0.18362.418]
(c) 2019 Microsoft Corporation. Todos os direitos reservados.

> git add *
> git commit -m "Descrição das minhas novas alterações"
> git push origin minhaBranch
```

Agora, se você acessar a página de seu repositório, você verá um novo botão:



Clicando neste botão será aberto um formulário onde você deverá descrever um resumo das suas alterações. Ao terminar clique em 

Muito provavelmente teremos algum conflito. Sua tarefa é descobrir como resolver este conflito e finalmente enviar seu Pull Request.