Nesta fase da mentoria, você irá aprender a realizar uma **REST API** com **.NET CORE,** programaremos em c#.

Breve explicação

- **REST API**: É uma arquitetura de desenvolvimento de software baseada em comunicação entre cliente e servidor através de protocolos HTTP. Uma REST API é uma interface que permite que diferentes sistemas se comuniquem entre si, transferindo dados em formato JSON ou XML, por exemplo.

- **C#**: É uma linguagem de programação moderna e orientada a objetos, desenvolvida pela Microsoft. Foi projetada para ser simples, segura, rápida, moderna e multi-paradigma, ou seja, suporta programação funcional, programação orientada a objetos e programação imperativa.

- **.NET Core**: É uma plataforma de desenvolvimento multiplataforma, de código aberto, mantida pela Microsoft. Ela inclui um conjunto de ferramentas, linguagens e bibliotecas para desenvolvimento de aplicativos, tais como o C#. O .NET Core é capaz de executar em diferentes sistemas operacionais, como Windows, macOS e

Linux, permitindo que desenvolvedores criem aplicativos para diferentes plataformas usando a mesma base de código.

O que iremos nos aprofundar?

- **C# .NET Core API**: Para criar uma API em C# .NET Core, é preciso definir os endpoints que permitirão a comunicação entre a API e outros sistemas. Cada endpoint é definido por uma classe que herda da classe ControllerBase e é decorada com atributos que definem o método HTTP e o caminho para o endpoint. A lógica de negócios da aplicação é definida em serviços que são injetados na classe do endpoint por meio da injeção de dependência.

- **Injeção de dependência**: A injeção de dependência é uma técnica que permite que as dependências de um objeto sejam injetadas nele por meio do construtor ou de propriedades públicas. Em C# .NET Core, a injeção de dependência é usada para injetar serviços em outras classes, como os endpoints da API ou as classes de validação.

- **Arquitetura Hexagonal**: A arquitetura hexagonal é um padrão que pode ser implementado em conjunto com o C# .NET Core para criar uma aplicação bem estruturada e fácil de manter. A camada de domínio contém as entidades, objetos de valor e serviços que definem a lógica de negócios da aplicação. A camada de aplicação é responsável por expor as funcionalidades da aplicação por meio da API, enquanto a camada de adaptadores é responsável por implementar as interfaces necessárias para que a aplicação possa se comunicar com o mundo externo, como a API, bancos de dados e outras integrações.

- **Entity Framework**: O Entity Framework permite que os desenvolvedores criem uma representação em código das tabelas de um banco de dados, por meio da criação de classes de modelo. Essas classes podem ser usadas para executar operações de consulta, inserção, atualização e exclusão no banco de dados usando métodos como DbSet.Find(), DbSet.Add(), DbSet.Update() e DbSet.Remove(). O Entity Framework também oferece recursos como migrações para ajudar a gerenciar as alterações no esquema do banco de dados.

- **Fluent Validation**: Para usar o Fluent Validation em um projeto C# .NET Core, é preciso instalar a biblioteca usando o NuGet e criar classes que herdam da classe AbstractValidator<T>. Essas classes são usadas para definir as regras de validação para um determinado objeto de modelo, como por exemplo, se um campo é obrigatório ou se ele precisa ter um valor mínimo ou máximo.

- **Token de autorização**: Para usar tokens de autorização em um projeto C# .NET Core, é preciso configurar um esquema de autenticação e autorização na API. Isso pode ser feito por meio da configuração de políticas de autorização que permitem

restringir o acesso a determinados recursos da API com base nas permissões do usuário autenticado.

- **Swagger**: O Swagger é uma ferramenta de documentação para APIs RESTful que permite que os desenvolvedores visualizem e interajam com a API sem precisar escrever nenhum código. Em C# .NET Core, o Swagger pode ser configurado facilmente usando o pacote Swashbuckle.AspNetCore.

O que preciso para começar?

Primeiro de tudo, precisamos criar o ambiente necessário para prosseguirmos com o desenvolvimento, como também iremos utilizar **Banco de dados,** o ambiente de banco de dados deve estar funcionando também!**.**

Vá no no repositório da mentoria clicando aqui, e verifique monte o ambiente que está dentro da pasta **ApiC#**, depois de montado, desenvolva os exercícios abaixo, em c# e commit no seu github, após finalizar ou se houver alguma dúvida, contate um dos mentores.

Segue exercícios:

1. Crie um programa que leia um número inteiro do usuário e imprima se ele é par ou ímpar.

2. Crie uma classe chamada Pessoa que tenha como atributos nome e idade. Adicione um método chamado Apresentar que imprime uma mensagem contendo o nome e a idade da pessoa.

3. Crie uma classe chamada Calculadora que tenha os métodos Soma, Subtracao, Multiplicacao e Divisao. Cada um desses métodos deve receber dois números como parâmetros e retornar o resultado da operação correspondente.

4. Crie uma aplicação console que leia o nome e a idade de várias pessoas do usuário, armazene essas informações em um array e, em seguida, imprima na tela todas as pessoas que têm idade igual ou superior a 18 anos.

5. Crie um programa que receba uma frase do usuário e retorne a mesma frase invertida.

6. Crie uma classe chamada Retângulo que tenha como atributos largura e altura. Adicione um método chamado Área que retorna a área do retângulo (largura x altura) e outro método chamado Perímetro que retorna o perímetro do retângulo (2 x largura + 2 x altura).

7. Crie um programa que leia um número inteiro do usuário e imprima todos os números primos menores ou iguais a ele.

8. Crie uma classe chamada Conta Bancária que tenha como atributos número da conta, nome do titular e saldo. Adicione métodos para depositar, sacar e transferir dinheiro entre contas.

9. Crie um programa que leia um número inteiro do usuário e imprima o fatorial desse número.