Atividade Bruno

Luiza Fernanda RA: 324117266

1. A Entrega Padrão: o Crie uma requisição do tipo GET. o Tente buscar o CEP da nossa faculdade (ou um famoso, como o da Sé em SP: 01001000). o Pergunta: Qual foi o Status Code retornado? O JSON veio completo? Resposta: 200OK, Sim, veio completo.

2. O Caso do CEP Inexistente: o Tente fazer um GET para um CEP que não existe (ex: 99999999). o Desafio: Observe o JSON retornado. O ViaCEP retorna um erro 404 ou ele retorna um JSON com um campo "erro": true? Por que você acha que eles escolheram essa estratégia? Resposta: Retorna um Jason com um campo erro true. Para manter o padrão da API.

3. Mudando o Formato (O Camaleão): o O ViaCEP é incrível porque ele aceita outros formatos além do JSON. Tente mudar o final da URL de /json/ para /xml/. o Pergunta: O que mudou na visualização dos dados? Qual formato parece mais fácil de ler no C#? Mudou de Jason para xml. Jason é o formato mais fácil de ler.

Passo 3: O Relatório do Engenheiro Após realizar os testes, responda: 1. Verbo HTTP: Qual método você usou? GET

Por que não usamos POST para consultar um CEP? Não usamos o POST porque estamos consultando dados.

2. Análise de Dados: Copie o JSON do seu CEP e identifique: o Qual é a Chave (Key) que guarda o nome da cidade? Localidade

Qual é a Chave que guarda o nome da rua? Logradouro

3. Integração C#: Se você fosse criar uma classe em C# para receber esses dados, como ficaria a sua propriedade para a chave localidade? Use o que aprendemos sobre [JsonPropertyName].

Resposta: using System.Text.Json.Serialization;

```
public class CepResponse
{
    [JsonPropertyName("cep")]
    public string Cep { get; set; }

    [JsonPropertyName("logradouro")]
    public string Logradouro { get; set; }
```

```
    [JsonPropertyName("bairro")]
    public string Bairro { get; set; }

    [JsonPropertyName("localidade")]
    public string Cidade { get; set; }

    [JsonPropertyName("uf")]
    public string Uf { get; set; }
}
```