ATIVIDADES TEÓRICAS

# Perguntas teóricas de introdução

1. **Em quais linguagens o C# foi inspirado?**

**O C# teve como principais inspirações o C++ e o Java.**

1. **Inicialmente o C# foi criado para qual finalidade?**

**O C# foi criado com a finalidade de facilitar a portabilidade ao flexibilizar o desenvolvimento de aplicativos. Para isso, o C# possibilita a criação das soluções para serem executadas sobre a plataforma .NET Framework. Dessa maneira, o desenvolvedor não cria solução para um dispositivo específico, mas para a plataforma.**

1. **Quais os principais motivos para a Microsoft ter migrado para o Core?**

**O principal motivo foi adicionar as seguintes características e torna-lo o sucessor do .Net framework. Primeiramente, a possibilidade de plataforma cruzada entre diferentes sistemas operacionais para uma mesma solução. O .Net Core possui código-fonte aberto, se tornando um software livre. Ele implementa funcionalidades modernas como programação assíncrona e recursos de alto desempenho. Além de fornecer compatibilidade com contêineres do Docker disponibilizando uma implantação flexível.**

1. **Cite as principais diferenças entre .Net Full Framework e .Net Core.**

**O .Net Framework é recomendado quando:**

**- O aplicativo atualmente já utiliza do .Net framework;**

**- O aplicativo utiliza de bibliotecas, tecnologias ou componentes não disponíveis para o .Net Core**

**O .Net Core é recomendado quando:**

**- O aplicativo necessita de plataforma cruzada**

**- O aplicativo apresenta micro serviços**

**- O aplicativo pretende ser aplicado em contêineres do Docker**

**- O aplicativo requer alto desempenho**

## ****PERGUNTAS DE POO****

1. **O que é POO?**

A programação orientada a objetos é um paradigma de programação baseado no conceito de classes e objetos. A ideia principal consiste em trazer uma aproximação da estrutura do código com objetos tangíveis ou conceitos do mundo real.

1. **O que é polimorfismo?**

O polimorfismo é a presença de implementações diferentes de um mesmo método presentes em classes derivadas de uma mesma superclasse. Esta superclasse é uma classe generalizada e as classes derivadas dela são classes especializadas que possuem comportamentos diferentes para o método generalizado. Desta maneira, ao utilizar da classe generalizada podemos chamar o método sem especificar a classe especializada.

1. **O que é abstração?**

A abstração é a interpretação classificada de uma entidade. Para isso, deve ser definido sua identidade com o nome de classe, suas características com as propriedades e suas ações com implementações de métodos.

1. **O que é encapsulamento?**

O encapsulamento de atributos e métodos é a limitação do escopo para apenas as classes ou objetos correspondentes. Desta maneira, classes ou objetos alheios não possuem a visibilidade desses componentes para leitura ou modificação, oferecendo segurança ao comportamento previsto dos elementos.

1. **Quando usar uma classe abstrata e quando devo usar uma interface?**

Interfaces apresentam apenas declarações de métodos, sem corpo e exigem que classes declarem seu comportamento ao implementarem a interface. Múltiplas interfaces podem ser implementadas por uma única classe.

Classes abstratas podem possuir atributos e métodos com um comportamento definido. Mas uma classe só pode implementar uma única classe abstrata.

1. **O que faz as interfaces IDisposable, IComparable, ICloneable e IEnumerable?**

IDisposable fornece um mecanismo para liberar recursos não gerenciados.

IComparable define um método de comparação de tipo específico generalizado que implementa uma classe ou um tipo de valor para solicitar ou classificar suas instâncias.

ICloneable dá suporte à clonagem, que cria uma nova instância de uma classe com o mesmo valor de uma instância existente.

IEnumerable expõe um enumerador que dá suporte a uma iteração simples em uma coleção não genérica.

1. **Existe herança múltipla (de classes) em C#?**

Não, essa estratégia não é permitida no C# devido a complicações acarretadas pelo uso de herança múltipla. Apesar disso, o C# possui herança transitiva, isto é, que permite herança através de hierarquia sequencial, disponibilizando os membros herdados da classe avô e da classe pai para a classe neto. Outra alternativa é a utilização de interfaces implementadas.