Questões relacionadas a C#

1. **Orientação a Objetos:**

• **Explique o conceito de herança múltipla e como C# aborda esse cenário.**

R.: A capacidade de uma mesma classe herdar membros como Métodos, Propriedade, Campos e outras funcionalidade de mais de uma classe. O C# não possui esse mecanismo, mas há formas de emular esse comportamento através da implementação de Interfaces e Extensões.

**• Explique o polimorfismo em C# e forneça um exemplo prático de como ele pode ser implementado.**

R.: É capacidade de classes derivadas especializar os comportamentos de membros herdados de uma superclasse, de forma a substituir o complementar a funcionalidade dessa rotina na superclasse.

Um exemplo prático seria duas classes de Envio de E-Mail que herdam de uma classe mãe com método abstrato de envio. A classe mãe encapsula o básico da instancia do objeto de Mensagem, populando Assunto, Destino e Mensagem Básica. Uma das classes especializa o método adicionado formato HTML, e novas informações a Mensagem, enquanto a outra classe adiciona Anexos ao objeto de Mensagem.

Todas as classes filhas executam o Envio herdado da classe mãe.

1. **SOLID:**

**• Descreva o princípio da Responsabilidade Única (SRP) e como ele se aplica em um contexto de desenvolvimento C#.**

R.: É o principio que postula que uma classe deve ser especializada em apenas um assunto do software, de forma que a implementação funcionalidades orientadas a apenas uma mesma razão para serem modificadas.

**• Como o princípio da inversão de dependência (DIP) pode ser aplicado em um projeto C# e como isso beneficia a manutenção do código?**

R.: É um princípio de arquitetura que visa desacoplar módulos de implementações em relação a módulos de abstração, de forma que os módulos de abstração não incorporar ou influencia em nada nos módulos de implementação. No C# tal princípio pode ser implementado por uso de Interfaces ou Classes Abstradas, encapsulando a implementação em classes que implementem ou herdem essas.

**3. Entity Framework (EF):**

• **Como o Entity Framework gerencia o mapeamento de objetos para o banco de dados e vice-versa?**

R.: Através de marcações de Atributos ou implementações implícitas através da Classe ModelBuilder o EF mapeia tabelas relacionais, colunas e restrições de chave estrangeira do banco de dados para classes C# denominadas de entidades. Espelhando o relacionamento dos objetos de banco de dados em uma estrutura Orientada a Objetos, que através do encapsulamento das classes ADO.Net permite a realizações das operações SQL traduzidas de instruções C#/Linq.

• **Como otimizar consultas no Entity Framework para garantir um desempenho eficiente em grandes conjuntos de dados?**

R.: Analisar as consultas resultantes da tradução da linguagem Linq para SQL, de forma que se preciso, refatorar a mitigando a utilização de estruturas desnecessárias e execuções obsoletas.

**4. WebSockets:**

•**Explique o papel dos WebSockets em uma aplicação C# e como eles se comparam às solicitações HTTP tradicionais.**

R.: WebSockets permitem a comunicação continua com o software desenvolvido em C#, de forma a trafegar streams dos dados em tempo real se a necessidade do fechamento da conexão. Diferente de requisições HTTP, como por exemplos as implementadas em APIs REST, onde a conexão é aberta e fechada ao final de um ciclo atrelado a execução da rotina solicitada.

• **Quais são as principais considerações de segurança ao implementar uma comunicação baseada em WebSockets em uma aplicação C#?**

R.:

* 1. Não devemos implementar WebSocckets em um contexto de um ambiente misto, em uma aplicação que utilize HTTPS, ou vice-versa.
  2. Sempre optar por conexão segura (WSS) ao invés de uma conexão não segura com o protocolo WS.
  3. Implementar autenticação e autorização do WebSocket.
  4. Restringir acesso entre origens (CORS).
  5. Implementar rotinas de tratamento contra Cross-Site Scripting (XSS)
  6. Permitir execuções de solicitações do tipo WebScockets Request

**5. Arquitetura:**

• **Descreva a diferença entre arquitetura monolítica e arquitetura de microsserviços.**

R.: Arquitetura monolítica é o modelo tradicional do programa de software, que é construído como unidade unificada e é autossuficiente e independente de outros aplicativos.

Já a arquitetura distribuída em Microsserviços implementa uma série softwares (services) autônomos entre si, separando as funcionalidades em processos menores que funcionam independentes uns dos outros, mas contribuem para o todo do sistema.

• **Como você escolheria entre a arquitetura de microsserviços e a arquitetura monolítica ao projetar.**

R.: Minha escolha sempre é pautada na necessidade do escopo do projeto e os recursos tecnológicos disponíveis. Há cenários em que uma arquitetura monolítica é a melhor escolha, por exemplo quando o software precisa ter funcionalidades espelhadas em modelos relacionais e com ciclos de vida interdependentes. Já arquitetura distribuída em microsserviços é melhor opção para sistemas principalmente de acesso web e multiplataforma que precisam de funcionalidades com ciclos de vida independentes umas das outras para manter a aplicação em funcionamento.

**Repositório do teste prático:**

https://github.com/hiltonsrn/PostManager.git