## Teste para desenvolvedor .net

## Explique com suas palavras o que é domain driven design e sua importância na estratégia de desenvolvimento de software.

DDD (Domain Driven Design) é uma abordagem para desenvolvimento de sistemas que visa controlar a complexidade através de práticas onde um domínio do problema deve ser atacado independente de tecnologia onde a linguagem entre os membros do negócio e os desenvolvedores deve ser única e focada nos termos de negócio. a importância para o desenvolvimento de software é enorme, pois controla, simplifica, e possibilita que o sistema tenha um ciclo de vida mais longo e econômico. atualmente vêm ganhando destaque devido ao uso de microservices, pois ambos se ajudam quando visam controlar o tamanho do domínio e suas dependências.

## Explique com suas palavras o que é e como funciona uma arquitetura baseada em microservices. explique ganhos com este modelo e desafios em sua implementação.

Uma arquitetura baseada em microservices está em destaque no momento atual do mercado e visa criar serviços independentes de qualquer outro com total isolamento geralmente decompondo sistemas baseados em monolitos. Os ganhos podem ser imensos devido ao alto grau de desacomplamento que a arquitetura oferece, cada microservice pode utilizar uma tecnologia que melhor se aplica a necessidade e ao cenário proposto, facilita a escalabidade horizontal e ganhos de performance, aumenta a tolerancia a falhas, mas toda escolha tem algumas desvantagens como aumenta a complexidade da infra-estrutura para monitorar os serviços e geralamente containers onde são executadas as aplicações, as integrações geralmente são através de mensageria onde os eventos são propagados para integração entre os mesmos e a consistencia eventual e o controle de processos através de saga ou process manager são desafiadores.

## explique qual a diferença entre comunicação sincrona e assincrona e qual o melhor cenário para utilizar uma ou outra.

Comunicação síncrona no caso de serviços se dá através do padrão request / response onde o requisitante só avança quando a resposta é enviada. padrão das apis rest (http). geralmente se aplica em consultas nas apis e processos onde não há opção para consistência eventual, já a comunicação assíncrona se dá através de padrões como publish / subscribe, webhook entre outros onde o requisitante envia a solicitação e em algum momento futuro poderá receber a resposta. Geralmente aplicado em processos mais complexos onde o tempo de resposta pode ser mais alto, a consistência eventual é uma realidade, a escalabilidade e tolerância a falhas são maiores. geralmente processos complexos através de microservices. por exemplo, um pedido num sistema de e-ecommerce.