## Teste para desenvolvedor .net

O teste abaixo visa avaliar o perfil técnico dos candidatos a desenvolvedor backend. Leia atentamente as instruções abaixo e responDa as questões da melhor forma possível.

* O teste será composto de três perguntas técnicas e uma avaliação prática.
* Responda as perguntas com suas palavras, com seu entendimento sobre o tema.
* Publique o teste prático com as respostas em um repositório no git-hub e nos envie.

## Perguntas:

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

## 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 dominio e suas depêndencias (Bounded Context).

## 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 É um PADRÃO que 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 sincrona 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 CONSISTENCIA EVENTUAL.

## comunicação assincrona 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 TOLERANCIA A FALHAS são MAIORES. GERALMEnte PROCESSOS COMPLEXOS ATRAVÉS DE MICROSERVICES. POR EXEMPLO, UM PEDIDO nUM SISTEMA de e-ECOMMERCE.

## Teste prático:

## regra de negócio:

## 

## criar um microservices que, através de um http post efetue uma operação de debito (origem) e credito (destino) nas contas correntes.

## entidades: contacorrente, lancamentos (voce pode incrementar com outras entidades se achar necessário)

## Parâmetros de entrada:

## conta origem

## conta destino

## valor

## Parâmetros de saída:

## http status code

## 

## informações adicionais:

## o método “post” devera receber os parametros no body da requisição em formato json

## UTILIZE Domain Driven Design

## serão avaliados critérios de arquitetura como separação de responsabilidade, clean code, segurança e testes

## tecnologias que você pode utilizar .net core 2.X, c#, xunits (testes)

## no término do projeto, publique o código em um repositório no git-hub

## REPOSITÓRIO: <https://github.com/jctelheiro/Desafio-Net>

## CoNSIDERAçÕES DO PROJETO no GITHUB:

## A solution foi criada levando em consideração o mínimo valor do produto, para atender a necessidade dos requisitos propostos utilizando o conceito de API para Microservice, DDD, CleanCode, SOLID.

## Utilizei segurança mínima na API através de secret-key que está configurada no arquivo: appsettings.Development.json (a chave secreta é 'teste')

## Utilizei uma implementação de CQRS (parcial) não utilizei as Queries

## O swagger foi configurado para interação se desejável.

## Implementei um repositório in memory para fins de simplicidade

## No DDD tentei demonstrar os conceitos técnicod de AggregateRoot, Entity e ValueObject, neste caso não utilizei os domain events.

## Melhorias:

## Poderia implementar eventsourcing com domain events, mas o tempo não foi suficiente e acho que para o teste prático deve atender.

## Não implementei um gateway de api e nem health check pelos motivos anteriores.

## Os testes unitários do DomainModel estão cobertos plenamente,

## e coloquei somente um teste no controller.

## Ao executar o projeto em modo 'Development' a extensão 'SeedWorkLoadRepositoryExtension' é executada e duas contas correntes serão criadas para auxiliar.