## Teste para desenvolvedor .net

O teste abaixo visa avaliar o perfil técnico dos candidatos a desenvolvedor backend na Superdigital. 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 envie o endereço, (BRUNO.COUTO@superdigital.com.br)

## Perguntas:

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

## Para definir o design orientado por domínio, devemos primeiro estabelecer o que queremos dizer por domínio neste contexto (e em desenvolvimento em geral). A definição de dicionário comum de domínio é: "uma esfera de conhecimento ou atividade." Perfurar um pouco de que, domínio no Reino da engenharia de software comumente se refere à área de assunto em que o aplicativo é destinado a aplicar. Em outras palavras, durante o desenvolvimento de aplicativos, o domínio é a "esfera de conhecimento e atividade em torno da qual a lógica do aplicativo gira".

A importância na estratégia de desenvolvimento de software utilizando o design orientado por domínio seria uma solução adotada com foco na modelagem das entidades principais de negócio usando a linguagem adequada daquele domínio para facilitar a manutenção, extensão e entendimento.

Com essa modelagem de software que segue um conjunto de práticas com objetivo de facilitar a implementação de subsistemas e suas respectivas regras/processos de negócios por contextos dentro de um modelo de domínio (Domain Driven Design). A partir dessa abordagem temos os seguintes componentes guias de implementação:

Domain Model

Contempla as entidades de domínio, estados e comportamentos, lógica de negócio, interfaces para os serviços de domínio e de repositório e estruturas (objetos de valor) que agregam valor as entidades.

Domain Api

Serviço de Domínio que orquestra as operações sobre as entidades de domínio podendo trabalhar com diversas entidades, realizando persistência através de repositórios.

Application Service

Serviço de Aplicação que orquestra ações disparadas pela camada de apresentação e fornece DTOs para a comunicação, compondo adaptadores, interfaces para camadas superiores, suporte para implementação de controle de transações, registro de operações (logs). Também serve como fábrica de serviços para as camadas superiores.

Repository

Realiza a persistência das entidades se comunicando diretamente com o meio de acessos aos dados.

External Service

Serviços externos que realizam a consulta/persistência por meios externos.

## 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.

Microservices é uma nova arquitetura de modelagem e criação de software que se referem a um software dividido em pequenos software ou micro-softwares que têm responsabilidades reduzidas e são independentes.

Com este tipo de modelagem temos ganhos de deploy independente e são escaláveis, cada serviço também possui um limite de módulo firme, que permite inclusive que diferentes serviços sejam escritos em diferentes linguagens de programação e equipes diferentes.

Os desafios em implementar seguindo uma modelagem de microservices seria colocar cada elemento de funcionalidade em um serviço separado e escala distribuindo os serviços entres os servidores, replicando por demanda.

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

Seguindo o modelo de ambiente cliente-servidor as aplicações trabalham através de requisiçõese respostas, o cliente envia uma requisição e o servidor aonde está hospedado a aplicação ou serviço retorna uma resposta que pode ser do tipo Síncrona e Assíncrona.

Requesição síncrona quando enviamos uma requisição, o processo remetente é bloqueado até que ocorra uma resposta, com isso não é possível enviar novas requesições até que a requisição atual seja finalizada, havendo sincronismo entre as requesições. Devemos utilizar requisições síncronas quando tivemos interações sistemáticas uma atrás da outra havendo necessidade de resposta imediata.

Requesição assíncrona não existe sincronismo entre as requesições, com isso podendo enviar diversas requesições em paralelo. Onde cada resposta retorna quando estiver pronta. Devemos utilizar requesições assíncronas quando não houver a necessidade de respostas sistêmicas imediatas.

## 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

## https://github.com/AndersonFRS/Superdigital