Objetivo

1. O objetivo desta atividade é para avaliarmos suas habilidades em codificação e o que você valoriza no desenvolvimento de software. Gostamos de inovação e novas ideias.

A atribuição

Construa uma página web (aplicação web) de listagem que exiba uma listagem dos produtos oferecidos pela *Gubee tecnologia.*

Os dados necessários para serem consumidos são conforme o JSON no final do documento.

Você deve exibir:

- Nome do produto.
- Descrição simples.
- Mercado alvo.
- Tecnologias utilizadas no produto.

Adicionar novos campos é encorajado, mas não mandatório.

Devem existir opções de filtro pelas respectivas tecnologias e mercado alvo. Quando mais de uma tecnologia selecionada, somente retornar os produtos que possuem pelo menos uma das tecnologias, Ex: Se o usuário selecionar Java e Oracle deve listar os produtos que contenham **Java**, **Java e Oracle** e somente **Oracle**. Você está livre para criar o mecanismo de filtro da forma que desejar.

NÃO usar frameworks como: springboot, quarkus ou micronaut.

DEVE ser feito utilizando Jakarta EE apresentando conceitos fundamentais da tecnologia.

NÃO deve ser feito uso de JPA, utilizar persistência com jdbc procurando definir uma forma abstrata de persistência.

Utilize o json em anexo como exemplo de dados, crie um banco de dados local,podendo sem um banco em memória

Faça suposições necessárias para implementação.

Requisitos:

- Deve ser feito utilizando com no mínimo a versão do Java **11**, pode ser utilizando Kotlin.
- Toda logica deve ser testável unitariamente com **junit**.
    - **Não serão avaliados desafios sem testes unitários.**
- Código limpo e boas praticas de OO.
- Aplicar os princípios SOLID.

Desejável:

- Separação entre backend e frontend.
- Arquitetura em camadas.
- Uso da API de Stream e Lambda.
- Endpoints RestFul.

- O objetivo do testes e avaliar principalmente suas aptidões com desenvolvimento e sua preocupação com a qualidade do desenvolvido, sendo assim não existe a necessidade de focar em layout e aparência, faça um layout simples.

Entregáveis

Disponibilizar o fonte do projeto no GITHUB para que possamos analisar.

- Lembre-se, você será avaliado e questionado sobre o que fez, então faça sozinho com o conhecimento que possui.
- **JSON Exemplo:**

2. Como segunda parte do desafio crie uma anotação **@Transaction** simulando um contexto transacional de uma app com acesso a banco de dados, se um método possuir esta anotação deve imprimir antes da chamada "*Iniciando execução do método $metodo.classe*" e no final "*Finalizando execução do método $metodo.classe com $sucesso | $erro*" deve ser executado a partir de um método main.

3. Explique o que é um userprincipal e para que este é usado no dentro de uma app java.

4. Faça um diagrama de classes do código a seguir:

```
package br.com.gubee.ml.domain.ad.port.api.usecase;

import java.util.Random;
import java.util.UUID;
import java.util.concurrent.Executors;
import java.util.concurrent.ScheduledExecutorService;
import java.util.concurrent.TimeUnit;
import java.util.concurrent.atomic.AtomicInteger;

public interface UseCaseNotification {
   void notifyEveryHour(String customerId, PresenterNotification presenter);

   @FunctionalInterface
   interface PresenterNotification {
      void notification(String message);
   }

   class PoolingUseCaseNotification implements UseCaseNotification {

      @Override
      public void notifyEveryHour(String customerId, PresenterNotification presenter) {
         System.out.println("processando regra de negocio");
               presenter.notification(String.format("mensagem a ser enviada para %s: %s",
customerId, new Random().nextInt()));
      }
   }


   static void main(String[] args) {
                                                ScheduledExecutorService controller =
Executors.newSingleThreadScheduledExecutor();
      var notificationUseCase = new PoolingUseCaseNotification();
        PresenterNotification emailPresenter = (message) -> System.out.printf("email %s",
message);
       PresenterNotification whatsAppPresenter = (message) -> System.out.printf("whatApp
%s", message);
          PresenterNotification smsPresenter = (message) -> System.out.printf("sms %s",
message);
             PresenterNotification[] notifications = {emailPresenter, whatsAppPresenter,
smsPresenter};
      controller.scheduleAtFixedRate(() -> {
         var nextPos = Math.abs(new Random().nextInt()) % 3;
                           notificationUseCase.notifyEveryHour(UUID.randomUUID().toString(),
notifications[nextPos]);
         System.out.println();
      }, 1, 1, TimeUnit.SECONDS);
   }
}
```