

**Netshoes**

**Prova Java**

**OLÁ**

Caro candidato, você está participando do processo seletivo Netshoes. Abaixo você encontrará algumas perguntas relacionadas ao perfil de *Software Developer Java*.

Que força esteja com você, e boa sorte!

**1) Expor o serviço de Campanha, seguindo as regras de CRUD (Create, Read, Update e Delete). (15 pontos)**

**\*Leia todo o enunciado**

Eu, como usuário, quero administrar os dados da Campanha e fornecer mecanismos (APIs) para INCLUIR, CONSULTAR, ATUALIZAR, DELETAR as campanhas. Para tanto, os critérios de aceite dessa história são:

* As campanhas deverão ser cadastradas de forma que o serviço retorne essas campanhas seguindo a estrutura abaixo:
  + Nome Da Campanha;
  + ID do Time do Coração;
  + Data de Vigência;
* O Sistema não deverá retornar campanhas que estão com a data de vigência vencidas;
* No cadastramento de uma nova campanha, deve-se verificar se já existe uma campanha ativa para aquele período (vigência), caso exista uma campanha ou N campanhas associadas naquele período, o sistema deverá somar um dia no término da vigência de cada campanha já existente. Caso a data final da vigência seja igual a outra campanha, deverá ser acrescido um dia a mais de forma que as campanhas não tenham a mesma data de término de vigência. Por fim, efetuar o cadastramento da nova campanha:
  + Exemplo:
    - Campanha 1 : inicio dia 01/10/2017 a 03/10/2017;
    - Campanha 2: inicio dia 01/10/2017 a 02/10/2017;
    - Cadastrando Campanha 3: inicio 01/10/2017 a 03/10/2017;
    - -> Sistema:
      * Campanha 2 : 01/10/2017 a 03/10/2017 (porém a data bate com a campanha 1 e a 3, somando mais 1 dia)
        + Campanha 2 : 01/10/2017 a 04/10/2017
      * Campanha 1: 01/10/2017 a 04/10/2017 (bate com a data da campanha 2, somando mais 1 dia)
        + Campanha 1: 01/10/2017 a 05/10/2017
      * Incluindo campanha 3 : 01/10/2017 a 03/10/2017
* As campanhas deveram ser controladas por um ID único;
* No caso de uma nas campanhas já existentes, o sistema deverá ser capas de fornecer recursos para avisar outros sistemas que houve alteração nas campanhas existentes.

**O que se espera para as questões 1 e 2 - dicas e direcionamentos:**

* Os serviços devem receber e responder JSON;
* Uso de algum Banco de Dados será um diferencial, mas você pode salvar em memória; Lembrando que o projeto, quanto mais autônomo, melhor;
* Pense em como documentar os cenários desenvolvidos (Testes sempre são uma boa forma de documentar);
* Ao finalizar o desenvolvimento você deve compartilhar o código pelo Github ou Bitbucket;
* Fique à vontade para entrar em contato e tirar dúvidas;
* A deleção deve ser feita pelo ID;
* Juntamente com o Código, deve-se documentar a estratégia utilizada para a criação da aplicação, a arquitetura utilizada e os padrões. A documentação pode ser feita via GIT/Bitbucket;
* Caso utilize o Git/Bitbucket, não esqueça de criar o .gitignore
* Não utilizar o nome da Netshoes nos projetos ou packages da prova.
* Alguns requisitos não funcionais devem ser previstos:
  + O Serviço será acessado de forma acentuada, ou seja, a previsão é que o serviço receba 100 requisições por segundo;
  + A aplicação deverá prever falhas de integração entre as APIs, não deixando o cliente sem nenhuma resposta;

**2) Expor um serviço de Sócio Torcedor seguindo a especificação abaixo (15 pontos)**

**\*Leia todo o enunciado**

Eu, como usuário, quero me cadastrar informando meu e-mail e o meu Time do Coração através de uma API que me permite participar de algumas campanhas. Para tanto, os critérios de aceite dessa história são:

* Dado um E-mail que já existe, informar que o cadastro já foi efetuado, porém, caso o cliente não tenha nenhuma campanha associada o serviço deverá enviar as novas campanhas como resposta;
* O Cadastro deve ser composto de:
  + Nome Completo;
  + E-mail;
  + Data de Nascimento;
  + Meu Time do Coração;
* O Cliente não pode ter mais de um cadastro ativo;
* Ao efetuar o cadastro, o serviço deverá repassar uma lista de campanhas o qual está associado ao time do coração:
  + O Consumo das listas das campanhas deve ser feita via Serviço exposto conforme descrito no exercício anterior;
  + O Cadastramento das campanhas deverá ser feito via Serviço (API, conforme descrito no exercício anterior)
* A associação do cadastro do Cliente juntamente com as Campanhas deverá ocorrer em um segundo passo, utilizando a API construída no próximo exercício);

**O que se espera para esse exercício - dicas e direcionamentos:**

* Os serviços devem receber e responder JSON;
* Faça o uso de “Mocks” principalmente nos testes;
* Pense em como documentar os cenários desenvolvidos (Testes sempre são uma boa forma de documentar);
* Ao finalizar o desenvolvimento você deve compartilhar o código pelo Github ou Bitbucket;
* Fique à vontade para entrar em contato e tirar dúvidas;
* Juntamente com o Código, deve-se documentar a estratégia utilizada para a criação da aplicação, a arquitetura utilizada e os padrões. A documentação pode ser feita via GIT/Bitbucket e respondido por email com o link.
* Em caso de uso do Git/Bitbucket não esqueça de criar o .gitignore.
* Não utilizar o nome da **Netshoes** nos projetos ou *“packages”* da prova;
* Não precisa ser construído nenhuma tela para os cadastros;
* Alguns requisitos não funcionais devem ser previstos:
  + O Serviço será acessado de forma acentuada, ou seja, a previsão é que o serviço receba 100 requisições por segundo;
  + A aplicação deverá prever falhas de integração entre as APIs, não deixando o cliente sem nenhuma resposta;

**3) Dada uma stream, encontre o primeiro caractere Vogal, após uma consoante, onde a mesma é antecessora a uma vogal e que não se repita no resto da stream. O termino da leitura da stream deve ser garantido através do método hasNext(), ou seja, retorna falso para o termino da leitura da stream. Voce tera acesso a leitura da stream através dos métodos de interface fornecidos ao termino do enunciado. (15 pontos)**

***\*Leia todo o enunciado***

**Premissas:**

Uma chamada para hasNext() ir retornar se a stream ainda contem caracteres para processar.

Uma chamada para getNext() ir retornar o proximo caractere a ser processado na stream.

Não será possível reiniciar o fluxo da leitura da stream.

Não poderá ser utilizado nenhum framework Java, apenas código nativo.

Exemplo:

Input:  aAbBABac**afe**

Output: e

No exemplo, ‘e’ é o primeiro caractere Vogal da stream que não se repete após a primeira Consoante ‘f’o qual tem uma vogal ‘a’ como antecessora.

*Segue o exemplo da interface em Java:*

*public interface Stream{*

*public char getNext();*

*public boolean hasNext();*

*}*

*public static char firstChar(Stream input) {*

*}*

**O que se espera como resultado - Dicas e direcionamentos:**

* Tente criar sua implementação pensando em performance;
* Efetuar casos de teste para diversos cenários e uma boa pratica;
* Documentar o código seguindo as boas práticas de mercado;
* Os códigos mais simples tendem a serem mais elegantes;
* Em caso de não localização do caractere, o sistema deve informar uma mensagem amigável;
* Ao finalizar o desenvolvimento você deve compartilhar o código pelo Github ou Bitbucket;
* Caso utilize o Git/Bitbucket, não esqueça de criar o .gitignore
* Não utilizar o nome da **Netshoes** nos projetos ou “packages” da prova.

1. **O que é Deadlock? Detalhe um pouco sobre o caso e como você poderia resolver isso. (5 pontos)**

DeadLock ocorre quando 2 recursos ficam bloqueados aguardando o termino da execução um do outro.

Ocorre com mais frequencia em banco de dados com alto volume de acesso para leitura e escrita.

**O que se espera como resposta - Dicas e direcionamentos:**

* Detalhe sua linha de raciocínio;
* Elabore um plano de entendimento, por exemplo, lista, de forma a elencar os passos;
* Não copie conteúdo da internet, responda com suas palavras.

A Resposta poderá ser respondida por e-mail, ou, pode-se criar um README no Github/Bitbucket.

1. **Uma das grandes inclusões no Java 8 foi a API Stream. Com ela podemos fazer diversas operações de loop, filtros, maps, etc. Porém, existe uma variação bem interessante do Stream que é ParallelStreams. Descreva com suas palavras quando qual é a diferença entre os dois e quando devemos utilizar cada um deles. (5 pontos)**

ParallelStream é utilizado para dividir o processamento de filtros, map,Coletores em threads separadas com o intuído de melhorar a performance, porém devido ao overhead de controle das threads alguns casos pode não ser uma boa escolha.

Para sistemas que utilizam servidor de aplicação onde atende a diversas requisições, utilizar ParallelStream pode ter efeitos negativos.

**O que se espera como resposta - Dicas e direcionamentos:**

* Detalhe sua linha de raciocínio;
* Elabore um plano de entendimento, por exemplo, lista, de forma a elencar os passos;
* Não copie conteúdo da internet, responda com suas palavras.

A Resposta poderá ser respondida por e-mail, ou, pode-se criar um README no Github/Bitbucket.