Linx Impulse - Desafio Técnico Desenvolvedor Fullstack

O desafio visa avaliar sua capacidade para implementação de um sistema contendo mais de uma camada, duas de backend e uma de frontend. A aplicação a ser desenvolvida, é um sistema para exibir as vitrines de recomendação em um site, o sistema deverá conter 3 camadas, que serão descritas no decorrer deste documento.

Especificações gerais

- Queremos avaliar aquilo que você conseguir produzir do desafio, então mesmo que você não consiga desenvolver tudo, envie o resultado para que possamos analisar;
- Procure desenvolver primeiro os itens obrigatórios e fechar o escopo do projeto, e só então partir para os opcionais;

Pontos avaliados

- Funcionamento do sistema;
- Utilização de boas práticas na definição das rotas e parâmetros da api;
- Organização do projeto e estilo de código;
- Boas práticas de qualidade e performance do HTML e CSS;
- Otimização do projeto frontend;
- Tecnologias/ferramentas utilizadas no desenvolvimento do projeto;

Envio do projeto

O projeto deve ser criado em um repositório no github, e o link do repositório deve ser enviado para correção.

Escopo do projeto

Api de Catálogo

A função da api de catálogo, é servir os dados do catálogo de produtos da loja, para que esses dados possam ser consumidos através das outras aplicações. Essa api geralmente é consumida por diversas aplicações, e exige-se um tempo de resposta baixo, e alta disponibilidade.

O único caso de uso da api é servir os dados de um produto, recebendo como entrada uma chave (id do produto).

Especificações

Itens obrigatórios

- Você deve implementar um script para importação dos dados do arquivo catalog.json fornecido em anexo para o banco de dados, e uma api REST para servir os dados desses produtos armazenados no banco de dados;
- A api pode ser feita em uma tecnologia/linguagem de sua escolha, e que você julgar que melhor se encaixe na especificação da aplicação;
- O SGBD para armazenamento dos dados também pode ser definido a sua escolha, de acordo com o caso de uso;
- A api deve receber como input um id de produto e responder com os dados do produto requisitado;
    - A api deve suportar dois formatos de resposta:
        - **complete**: retorna todos os campos armazenados no banco de dados;
        - **compact**: retorna apenas os campos *name, price, status* e *categories*;
- Definições de padrões de rota, parâmetros etc fazem parte do desafio;
- Utilização de boas práticas na estrutura do projeto, codificação etc, também fazem parte do desafio;

Itens não obrigatórios (diferenciais)

- Requisições constantes ao SGBD são caras. Como poderíamos definir uma arquitetura que diminua a quantidade de acessos ao banco de dados? Se possível, implemente essa arquitetura planejada no projeto da api;
- Utilização de docker;
- Geração de uma documentação do uso da api;
- Implementação de testes unitários;

API de recomendações

A função da api de recomendações é fornecer a lista de recomendações para serem exibidas nas vitrines de recomendação que serão exibidas nos sites dos nossos clientes.

Essa api funciona comunicando-se com outros microserviços, para obter a lista de recomendações já pré processada pelos nossos algoritmos, e em cima dessa lista, é possível aplicar diversas regras de negócio, validações, além de inteligência de personalização real time, para se exibir as vitrines de recomendações.

Especificações

Itens obrigatórios

- Você deve implementar uma api em **node-js** para servir a lista de recomendações para serem exibidas nas vitrines dentro do site. Uma aplicação frontend se comunicará com essa api para poder renderizar as vitrines;
- No contexto do desafio, teremos apenas duas vitrines de recomendação para serem exibidas, uma de produtos mais populares, e outra de produtos em oferta;
    - A api conterá um único endpoint, que retornará os dados das duas vitrines, contendo o título de cada vitrine, e a lista de produtos a serem exibidos;
    - Um parâmetro *maxProducts* deverá ser informado via *query* para definir o máximo de produtos que serão exibidos em cada uma das vitrines;
        - O menor valor permitido para esse parâmetro é 10, caso um número inferior seja informado, o parâmetro é desconsiderado e serão devolvidos 10 produtos;
- Para se obter a lista de produtos recomendados em cada uma das vitrines, a api de recomendações deve se comunicar com um microserviço responsável por servir a lista de recomendações já pré processada de acordo com o algoritmo.
    - O padrão para obter as recomendações é o seguinte: `https://wishlist.neemu.com/onsite/impulse-core/ranking/<algoritmo>.json`
    - Os identificadores dos algoritmos que devem ser utilizados:
        - Mais populares: `mostpopular`;
        - Produtos em oferta: `pricereduction`;
- A lista de recomendações conterá apenas os ids dos produtos. Para obter os dados completos dos produtos, a api de recomendações deve se comunicar com a api de catálogo, implementada por você no passo anterior;
- A lista de recomendações pode conter produtos que não estão mais disponíveis no catálogo (status diferente de available). A api de recomendações deve tratar esses casos, e excluir esses produtos da sua resposta;

Itens não obrigatórios (diferenciais)

- Tráfego de rede entre microserviços em geral é algo custoso, e que prejudica a performance dos serviços. Como poderíamos resolver esse problema na api de recomendações? Se possível, implemente essa solução;
- Utilização de docker;
- Geração de uma documentação do uso da api;
- Implementação de testes unitários;

Aplicação frontend

A terceira camada é uma aplicação frontend que deve fazer uma chamada para a api de recomendações, e renderizar um carrossel com os produtos da resposta.

Layout

As vitrines devem ser implementadas seguindo o layout abaixo. As imagens também foram enviadas em anexo no email.

Vitrine de mais populares

Vitrine de ofertas

Especificações

Itens obrigatórios

- A aplicação deve ser implementada sem utilização de nenhum framework ou lib;
- O estilo do carrossel deve seguir a imagem anexada;

- A aplicação frontend deve se comunicar com a api de recomendações para obter os dados das vitrines;
- Cada vitrine deverá exibir 16 produtos no carrossel;

Itens não obrigatórios (diferenciais)

- Utilização de ferramentas de build ou bundle no projeto;
- Utilização de algum pré-processador de css;