FRONT END

Desafio – “Eu revendedor ‘O Boticário’ quero ter benefícios de acordo com o meu volume de vendas”.

1. PROBLEMA/OPORTUNIDADE

O Boticário tem várias soluções para ajudar seus revendedores(as) a gerir suas finanças e alavancar suas vendas. Também existem iniciativas para impulsionar as operações de vendas como metas gameficadas e desconto em grandes quantidades de compras.

Agora queremos criar mais uma solução, e é aí que você entra com seu talento ;)

A oportunidade proposta é criar um sistema de Cashback, onde o valor será disponibilizado como crédito para a próxima compra da revendedora no Boticário;

Cashback quer dizer “dinheiro de volta”, e funciona de forma simples: o revendedor faz uma compra e seu benefício vem com a devolução de parte do dinheiro gasto.

Sendo assim o Boticário quer disponibilizar um sistema para seus revendedores(as) cadastrarem suas compras e acompanhar o retorno de cashback de cada um.

Vamos lá?

Requisitos do Teste:

- Tela de cadastro de um novo revendedor(a) solicitando Nome completo, CPF, e- mail e senha;
- Tela de login para informar e-mail e senha;
- Tela de cadastro de compras onde deverá ser informado o código, valor e data;
- Tela de listagem das compras cadastradas exibindo as informações de: ***código da compra***, ***valor***, ***data***, ***% de cashback aplicado***, ***valor do cashback*** e ***status do cadastro***;
- O status do cadastro poderá ser “Em validação”, “Reprovado” e “Aprovado”;
- Tela para exibir o valor de cashback acumulado até o momento, esta informação virá de uma das APIs do Boticário, que é um outro sistema que agrupa e consolida todas as vendas do revendedor(a);

OBS: *Não obrigatoriamente cada requisito descrito acima precisa estar em uma tela, seja criativo e monte a(s) tela(s) como entender melhor. Os dados podem ser mockados de um backend “fake” seja ele localhost ou remoto, não é necessário implementar uma api com toda lógica.*

Requisitos técnicos obrigatórios:

- Utilize um destes frameworks: **Angular**, **React** ou **Vue.js**
- Você pode utilizar um framework de UI. Exemplo: **Bootstrap**, **Material-UI**, **Bulma**, etc.
- Design Responsivo (manter a integridade das informações e layout nos principais breakpoints; Mobile e Desktop).
- Integração com uma API 'fake' (mockado ou remoto).
- Respeitar boas práticas de código e arquitetura.
- Teste Unitário (Jasmine, Jest).

Diferenciais (opcional)**:**

- Testes Automatizados (E2E).
- Utilização de boas práticas do Git.
- State Management (Redux, NGRX, MobX, etc).
- Webcomponents cross-platform.
- Offline Cache com Service Worker.
- Autenticação JWT

DICA: Você pode utilizar serviços de **Fake Online REST API** como https://jsonplaceholder.typicode.com e https://gorest.co.in para nos mostrar como lida em consumir API's :)