Desafio de Software

Aqui na Amazing Devs procuramos entender qual é sua bagagem de conhecimento e como aplicar suas habilidades em projetos reais. Valorizamos muito a qualidade das entregas.

Requisitos

Deverá ser desenvolvido um sistema de e-commerce simplificado, onde teremos apenas duas telas disponíveis, a de listagem de produtos e a do carrinho de compras.

Tela de listagem de produtos

Funcionalidades

- botão de adicionar no carrinho
- paginação

Considerações

- **não é necessário criar um cadastro de produtos**, pode ser usado a lista de produtos disponível no seguinte link (precisa estar logado no github): https://github.com/AmazingDevs/code-challenge/blob/asset/products-data.json?raw=true

Tela de carrinho de compras simplificada

Funcionalidades

- alterar a quantidade de itens
- remover produto do carrinho
- confirmar pedido

Considerações

- quando o usuário clicar no botão de confirmar pedido, o sistema deve conseguir efetuar as seguintes tarefas:
    - efetuar o pagamento
    - baixar o estoque dos produtos. Não pode haver estoque negativo.
    - alterar o status do pedido para "aguardando envio"

  Se um dos processos anteriores não puder ser realizado, o usuário deve receber uma mensagem de erro na tela e os dados devem ser **revertidos**.
- caso o usuário saia da página e volte para a listagem de produtos, o carrinho deve **continuar preenchido**

- ao remover ou alterar a quantidade de produtos, o sistema deve recalcular o valor total do pedido
- **pagamentos são rejeitados** para pedidos com valor acima de **5000,00.** O processo que efetua o pagamento deve ser um componente Fake, ou seja, não realizar nenhuma ação, como acessar serviços externos. Deve ser apenas uma função que sempre retorna sucesso, exceto para valores acima de 5000,00.

Instruções

- **pelo menos dois microserviços**, de preferência em pastas/solutions separadas.
- o projeto precisa executar usando **docker compose**
- todo o código e documentação devem ser **escritos em inglês**
- coloque no **README** tudo que considerar relevante, como executar o projeto e demais informações (vide seção "Como revisamos")
- deve ser realizado em sua linguagem de programação de preferência, utilizando qualquer banco de dados e sistema de filas que achar necessário
- sinta-se livre para usar qualquer biblioteca adicional, qualquer framework, tanto de front-end quanto de backend, considerando a vaga que está buscando preencher
- procure evitar modelos anêmicos: https://www.martinfowler.com/bliki/AnemicDomainModel.html
- faça um repositório privado em seu Github e nomeie como amz-code-challenge-[seu_usuário]. Ex: amz-code-challenge-joao123
- **dê acesso** no repositório ao e-mail **codechallenge@getamazingdevs.com**
- **não faça squash** de seus commits, gostamos de acompanhar a evolução do projeto

Escolha uma das seguintes trilhas técnicas que melhor se adapte ao seu conjunto de habilidades:

Full-stack: inclua um front-end e um back-end.
Back-end: inclua um front-end mínimo, sem muita dedicação a elementos visuais, por outro lado, foque em ter um backend excelente.
Front-end: inclua um back-end mínimo. Concentre-se em tornar a interface o mais polida possível.

Como revisamos

Sua inscrição será analisada por um de nossos arquitetos/engenheiros. Levamos em consideração seu nível de experiência.

Valorizamos a qualidade sobre a conclusão das funcionalidades. Não há problema em deixar as coisas de lado, desde que você descreve elas no README do seu projeto. O objetivo deste exemplo de código é nos ajudar a identificar o que você considera código pronto para produção. Você deve considerar este código pronto para revisão final com seu colega, ou seja, esta seria a última etapa antes de implantar em produção.

Os aspectos do seu código que avaliaremos incluem:

Arquitetura: quão limpa é a separação entre o front-end e o back-end? Qual modelo de arquitetura foi seguido? Se foi aplicado SOLID, KISS, Design Patterns, etc, e como foi aplicado.
Clareza: o README explica de forma clara e concisa os problemas encontrados e as soluções aplicadas? As vantagens e desvantagens da solução são explicadas?
Correto: o aplicativo faz o que foi solicitado? Se estiver faltando alguma coisa, o README explica por que está faltando?
Qualidade do código: o código é simples, fácil de entender e de fácil manutenção? Existem code smells ou red flags? O código é orientado a objetos e segue princípios como o princípio da responsabilidade única? O estilo de codificação é consistente com as diretrizes da linguagem? É consistente em toda a base de código?
Segurança: há alguma vulnerabilidade óbvia?
Testes: quão completos são os testes automatizados? Eles serão difíceis de mudar se os requisitos do aplicativo mudarem? Existem alguns testes unitários e de integração? Não estamos procurando uma cobertura completa (dada a restrição de tempo), mas apenas tentando ter uma ideia de suas habilidades de teste.
UX: a interface web é compreensível e agradável de usar? A API é intuitiva?
Escolhas técnicas: as escolhas de bibliotecas, bancos de dados, arquitetura etc. parecem apropriadas para a aplicação escolhida?
Ponto de bônus (esses itens são opcionais):

Escalabilidade: as escolhas técnicas serão bem dimensionadas? Se não, há uma discussão sobre essas opções no README?
Prontidão para produção: o código inclui monitoramento? Possui registro de log? tratamento de erros adequado?

Dúvidas?

Para qualquer dúvida durante o processo, mande um e-mail para **codechallenge@getamazingdevs.com** ou entre em contato com um de nossos recrutadores.