Challenge dos Malocas

Feito por mim: O DIO HAHAHAHA (É o Nicolas mesmo)

Link do meu GitHub: https://github.com/NickRMD

Introdução:

Neste documento irá conter um challenge (desafio, em português) para fazer nos tempos livres.

Condições:

1. Se já conhecer uma linguagem de programação, pode fazer na mesma, mas se quiser aprender uma nova ou estiver começando: Faça na que achar mais fácil, ou se realmente quiser aceitar o desafio, fazer numa linguagem de mais baixo nível (Como C, C++, Rust ou mesmo Go/Golang).
2. Faça na IDE que mais se adequa! Recomendadas as IDEs: VSCODE e JetBrains IntelliJ.
3. Faça o fork do repositório no GitHub anexado e no fim, faça o pedido de pull request referenciando ao fork feito.
4. De maneira detalhada, faça os commits baseando-se completamente na maneira com que acha mais descritiva, sendo usando commits por diretório/diretórios, por cada arquivo ou mesmo fazendo inteiro, mas lembre-se de fazer bons commits, que mesmo concisos, tragam uma boa explicação na mudança de código.
5. Será desqualificado o usuário que for descoberto usando alguma inteligência artificial para fazer trechos completos do código, será permitido o uso de inteligências artificiais somente para fazer partes pequenas ou mesmo médias do código, ou mesmo para correção do mesmo.
6. Utilizar-se de boas práticas, como métodos ágeis (não dependendo completamente, mas recomendado), uso do Trello para dividir as tarefas em partes menores (o link do Trello ou imagens do mesmo dará pontos) e etc.
7. Faça com sua liberdade. Perguntas como: “Qual banco de dados devo utilizar?” ou “Qual framework é recomendado?” deverão ser respondidas por você mesmo.
8. Ao final, anexar documento detalhando um pouco as ferramentas utilizadas, métodos utilizados e outros, como linguagem(ns) utilizada(s) ou framework(s) utilizado(s).
9. O uso de versionamento de API será levado em conta, use o caminho v1 para denotar a versão da API, assim a mudança da API no futuro poderá ser mais tranquila. Ao invés de mudar endpoints completamente, apenas mudar a versão. Exemplo: O endpoint /v1/user existe, mas numa versão futura pode ser colocado o endpoint /v2/user em uma versão que não oferece retrocompatiblidade com a versão anterior, sem problemas de transição.

Challenge dos Malocas

Detalhes:

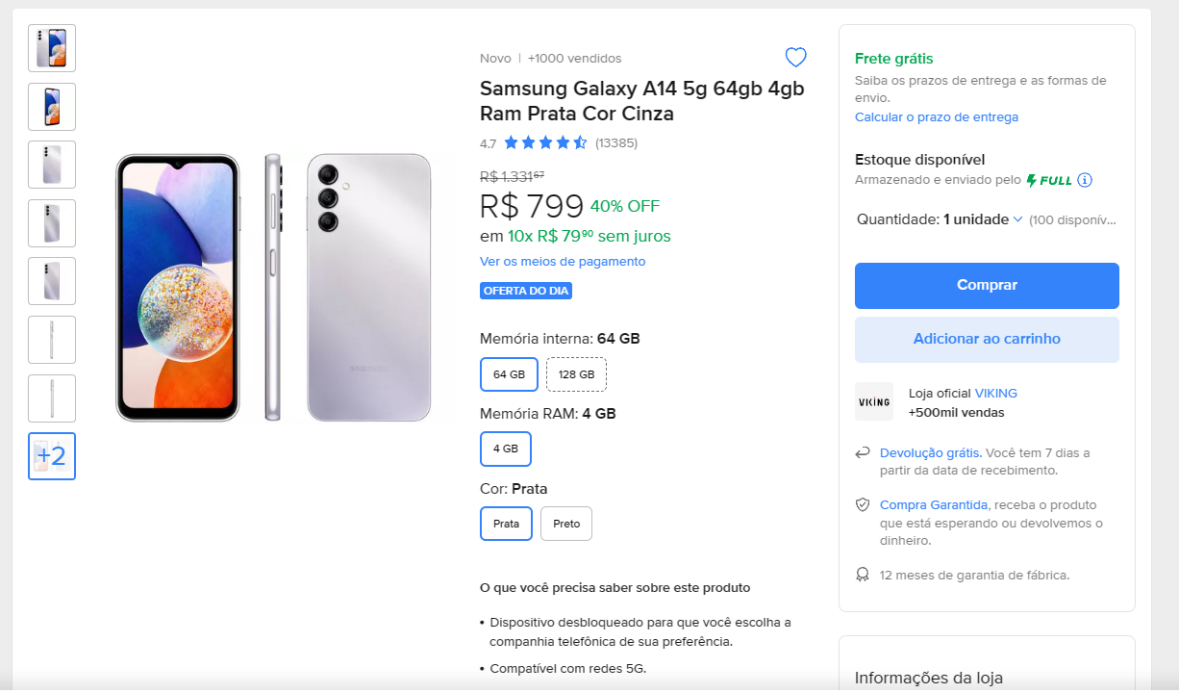
Faça um site utilizando-se de APIs próprias para funcionamento. O site será de uma loja personalizada cujo nome poderá ser escolhido individualmente, ela terá 6 seções principais:

1. Página inicial com lista randomizada de produtos, cada produto terá no máximo 5 tags, contendo no mínimo uma tag para cada produto. No topo irá existir uma barra de pesquisa que terá como propósito procurar tanto tags quanto nome de produtos, como por exemplo: O usuário pesquisa “Geladeira” o resultado será de geladeiras (nome do produto), e se o usuário procurar “Cozinha” irá aparecer coisas para cozinha, como geladeiras, mesas, microondas e etc. (Pesquisa por tags).
2. A segunda é exatamente a seção de pesquisa, que deverá conter filtros de **pelo menos:** faixa de preço, condição (novo ou usado), categorias relacionadas (baseando-se na pesquisa feita pelo usuário) e condição de frete (famoso botão do frete grátis).

Exemplo retirado do site https://www.kabum.com.br/

3. Seção do produto que será composta por imagens do produto, nome, avaliações do produto, preço, caixa para entrada de CEP, loja que vende o produto, condição de frete, descrição do produto, e produtos

relacionados. Todos serão funcionais, pegando as informações de uma API, o preço do frete não precisa ser calculado, afinal depende de fatores externos e muitas vezes de serviços pagos, então poderá ser gerada de maneira aleatória/randomizada.



Exemplo retirado do site https://www.mercadolivre.com.br/

4. Seção de login e registro de usuário e dashboard do mesmo, mostrando produtos comprados, avaliações feitas, opções de segurança (***->exemplos<-***: mudança de senha e adição de login em duas etapas).
5. Seção de produtos comprados, mostrando produtos comprados, quando selecionado o produto deverá abrir um modal, mostrando as informações do produto comprado, preço pago na época que o produto foi comprado (**NA ÉPOCA, NÃO PREÇO ATUAL**) e outras informações do produto.
6. Seção de carrinho e checkout, deverá conter produtos que foram adicionados no carrinho, quantidade adicionada e preço total (incluso frete). O usuário não poderá ser passado para próxima tela sem estar logado ou ter adicionado informações de envio (endereço, cep, etc.). A próxima tela será de checkout, que deverá conter preço total dos produtos, preço total do frete, e preço total (produtos+frete), deverá conter tela para colocar dados de cartão de crédito, ao final tudo isso será enviado ao banco de dados (dados de cartão preferencialmente encriptados).

De resto fica a critério individual, se terá mais ou menos funções, mas lembre-se que **fazer bloat de funções não ajudará na pontuação**, apenas funções bem feitas serão creditadas, desde que detalhadas.

DOCUMENTAÇÃO

A documentação será feita de modo que:

- Detalhe a API: mostrando como cada parte funciona, tipo de status que pode retornar (404, 500 e etc.) além de outras informações relevantes, exemplo: o formato (ex: JSON), caminhos relativos do endpoint (ex: /v1/user) e outros.
- Tente comentar o código, para fazer uma documentação interna das funções, classes e outros.
- Tente fazer esquemas básicos de interação, exemplificando: se o usuário fizer isso, acontece aquilo (se eu aperto o interruptor, a luz liga, se faço de novo, a luz desliga).

De resto fica a critério do participante.

ATENÇÃO:

Funções feitas à parte **DEVERÃO** ser documentadas e descritas separadamente.

LICENCIAMENTO

TODOS os projetos deverão ser disponibilizados em licenças opensource. Para padronizar, será usada a licença MIT em todos os projetos. A mesma será incluída no repositório que será dado fork pelo participante.

PONTUAÇÃO

A pontuação e suas modalidades terão os seguintes critérios:

- Cada modalidade de pontuação seguirá o mesmo padrão! Sendo assim:
    - Cada modalidade valerá 10 pontos.
    - Os pontos no final serão divididos na seguinte equação: SP/NM. Sendo SP a soma dos pontos e NM o número de modalidades.
    - Cada pontuação terá um número único de sub modalidades, algumas poderão ser compartilhadas com outras modalidades. O cálculo será o mesmo das modalidades. SP/NSM. Ou seja, soma dos pontos e número de sub modalidades.
- Ao final, a relação de pontuação de cada participante será mostrada no readme do repositório que será dado merge, ou seja, será mostrado a pontuação tirada para cada modalidade, e a nota final. Tudo isso numa tabela.
- O usuário com maior pontuação terá seu fork dado merge ao repositório principal, mostrando a foto e link de redes sociais (se requisitado) do participante vencedor.

As modalidades serão as seguintes:

- Usabilidade
    - Acessibilidade.
    - Experiência de usuário (UX).
    - Responsividade.
- Boas práticas
    - Manutenibilidade.
    - Métodos ágeis.
- Performance
    - Tempo de carregamento.
    - Tempo de acesso a API.
    - Otimização de código.
- Velocidade de desenvolvimento
    - Utilização de frameworks e bibliotecas (de preferência sem bloats).
    - Métodos ágeis.
- Segurança.
    - Encriptação de dados de usuário.
    - Uso de HTTPS (se possível).
    - Uso de tokens.
    - Proteção contra Injeção de Código (XSS, SQL Injection e etc.).
    - Gerenciamento de Sessão (incluí uso de tokens).
- Documentação
    - Manutenibilidade.
    - Documentação de API.
    - Documentação in-code (comentários).
    - Documentação de interação básica.

Indicação de faixa de pontuações

Pontuações acima de 9 -> Ótima

Pontuações entre 7 e 9 -> Boa

Pontuações entre 5 e 7 -> Média

Pontuações entre 3 e 5 -> Ruim

Pontuações entre 1 e 3 -> Péssima

Pontuações entre 0 e 1 -> Terrível.

Qualquer pergunta, mande aqui:

nicolas@fragmenta.org

Link do GitHub

Repositório | Challenge dos Malocas

QUE OS DESAFIOS COMECEM!