Serasa Consumidor - Teste para analista desenvolvedor

Olá, obrigado pelo interesse em fazer parte da nossa equipe.

O objetivo deste teste é verificar (até certo ponto) suas habilidades de codificação e arquitetura. Para isso você receberá um problema simples onde poderá mostrar suas técnicas de desenvolvimento.

Nós encorajamos você a exagerar um pouco na solução para mostrar do que você é capaz.

Considere um cenário em que você esteja construindo uma aplicação pronta para produção, onde outros desenvolvedores precisarão trabalhar e manter essa aplicação ao longo do tempo.

Você **PODE** e **DEVE** usar bibliotecas de terceiros, usando ou não um framework, você decide. Lembre-se, um desenvolvedor eficaz sabe o que construir e o que reutilizar.

Na entrevista de "code review", esteja preparado para responder algumas perguntas sobre essas bibliotecas e, caso utilize, sobre o framework. Como e por que você as escolheu e com quais outras alternativas você está familiarizado, serão algumas dessas perguntas.

Como este é um processo de "code review", evite adicionar código gerado ao projeto.

Obs: Para realizar esse teste, não crie um repositório público! Esse desafio é compartilhado apenas com pessoas que estamos entrevistando e gostaríamos que permanecesse assim.

Aqui no Serasa Consumidor, nós utilizamos o [Docker](https://www.docker.com/products/docker) para executar as aplicações, por isso, pedimos que você faça o mesmo neste teste. Isso garante que tenhamos um resultado idêntico ao seu quando testarmos sua aplicação.

Para facilitar o teste, disponibilizamos alguns containters que vão lhe ajudar a construir e executar suas aplicações, mas fique à vontade para alterá-los conforme preferir!

Para executá-los é fácil, acesse o diretório `user-api` e execute o comando: `docker-compose -up -d` e em seguida acesse o diretório `order-api` e execute o mesmo comando: `docker-compose -up -d`

Requisitos mínimos para o teste:

- Persistência de dados em banco relacional e não relacional. Pode ser MySQL ou PostgreeSQL e queremos ver você utilizar Elastic Search!
- Camada de cache em memória. Pode ser Redis, Memcached, ou APCU.
- Utilização de um ORM para manipulação dos dados.
- Testes unitários.
- Documentação de setup e do funcionamento das APIs (um Makefile cai muito bem!).

Instruções

- Clone este repositório.
- Crie uma nova branch chamada `dev`
- Desenvolva as aplicações.
- Crie uma "pull request" da branch `dev` para a "branch" `master`. Essa PR deve conter as instruções para executarmos as suas aplicações, as tecnologias que você decidiu usar, por que decidiu utilizá-las e também as decisões que você teve quanto ao design do seu código.

Requisitos das aplicações:

Nós desejamos que você crie 2 aplicações básicas (microserviços) que comuniquem-se entre si.

O primeiro deles deverá ser um cadastro de usuários, contendo os seguintes recursos:

- Listar, exibir, criar, alterar e excluir usuários

Tabela de usuários `user` deverá conter os campos: id, name, cpf, email, phone_number, created_at, updated_at

E o segundo deverá ser um serviço de pedidos, onde este deverá conter o id do usuário que fez o pedido e se comunicar com o serviço de usuários para retornar as informações do mesmo. Esse serviço deverá ter os seguintes recursos:

- Listar, Listar por usuário, exibir, criar, alterar e excluir.

Tabela de pedidos `order` deverá conter os campos: id, user_id, item_description, item_quantity, item_price, total_value, created_at, updated_at

Lembre-se de fazer a comunicação necessária entre os serviços para garantir a consistência de dados.

Essas aplicações também **DEVEM** estar de acordo com os padrões REST e **DEVE** ser disponibilizada uma documentação contendo os endpoints e payloads utilizados nas requisições.

Critérios de avaliação

Dê uma atenção especial aos seguintes aspectos:

- Você **DEVE** usar bibliotecas de terceiros, e pode escolher usar um framework, utilizar não vai ser uma penalidade, mas você vai precisar justificar a sua escolha.

- Suas aplicações **DEVEM** executar em containers Docker.

- Suas aplicações **DEVEM** retornar um JSON válido e **DEVEM** conter os recursos citados anteriormente.

- Você **DEVE** escrever um código testável e demonstrar isso escrevendo testes unitários.

- Você **DEVE** prestar atenção nas melhores práticas para segurança de APIs.

- Você **DEVE** seguir as diretizes de estilo de código.

- Você **NÃO** precisa desenvolver um "frontend" (telas) para esse teste.

Pontos que consideramos um bônus:

- Fazer uso de uma criptografia reversível de dados sensíveis do usuário, como: email, cpf e telefone, antes de persisti-los no banco de dados
- Suas respostas durante o code review
- Sua descrição do que foi feito na sua "pull request"
- Setup da aplicação em apenas um comando ou um script que facilite esse setup
- Outros tipos de testes, como: testes funcionais e de integração
- Histórico do seus commits, com mensagens descritivas do que está sendo desenvolvido.

Boa sorte!