Teste para Engenheiro de QA (Quality Assurance) - Pleno

✪ Objetivo do Desafio

Avaliar a capacidade do candidato em automatizar testes, validar APIs, garantir a qualidade do software e realizar testes de performance.

👉 Descrição do Desafio

O candidato deverá:

1. **Criar testes automatizados de interface (UI)** para um formulário de cadastro.
2. **Criar testes automatizados de API** utilizando uma API mock.
3. **Criar testes de performance** simulando múltiplos acessos simultâneos.

✪ 1. Testes de UI (Interface) - Cypress ou Selenium

O candidato deve escrever um teste automatizado para um formulário de cadastro de usuário, validando os seguintes requisitos:

- **Campos obrigatórios**: O formulário não deve permitir envio sem preencher todos os campos.
- **Senha forte**: A senha deve ter mínimo 8 caracteres, 1 letra maiúscula e 1 número.
- **Confirmação de e-mail**: O e-mail digitado no campo "Confirmação de E-mail" deve ser igual ao e-mail principal.

✨ Exemplos de Cenários de Teste:

- Preencher o formulário corretamente e enviar → Deve exibir mensagem de sucesso.
- Deixar campos obrigatórios vazios → Deve exibir mensagens de erro.
- Digitar uma senha fraca (exemplo: "12345") → Deve exibir erro de validação.
- Digitar e-mails diferentes nos campos de "E-mail" e "Confirmação de E-mail" → Deve exibir erro.

✪ Ferramentas Permitidas:

- **Cypress** (preferível para testes modernos de UI).
- **Selenium + WebDriver** (se o candidato tiver experiência com Selenium).

✪ 2. Testes de API (Postman ou Jest)

O candidato deve criar testes automatizados para validar uma API REST, garantindo que:

- **Respostas corretas**: Testar requisições GET e POST.
- **Respostas HTTP adequadas**: Testar status 200, 400 e 500.
- **Validação da estrutura JSON**: O retorno da API deve conter os campos esperados.

✨ API Mock para os Testes

- **Opção 1**: Utilizar a API mock abaixo:
  - https://jsonplaceholder.typicode.com/users
- **Opção 2**: Criar sua própria API mock utilizando ferramentas como:
  - Mockoon, JSON Server ou Postman Mock Server.

✨ Exemplos de Cenários de Teste:

- Fazer uma requisição GET e validar se os dados retornados estão corretos.
- Enviar um POST sem um campo obrigatório e garantir que a API retorne erro 400.
- Simular um erro no servidor e garantir que ele retorne 500.

✪ Ferramentas Permitidas:

- **Postman** (para testes manuais e automação de collections).
- **Jest + Supertest** (para testes automatizados em Node.js).

✪ 3. Testes de Performance (JMeter ou k6)

O candidato deve realizar um teste de carga simulando 100 usuários simultâneos acessando a API, analisando:

- **Tempo de resposta**: A API consegue responder rapidamente sob carga?
- **Erros de requisição**: Existem falhas quando muitos usuários acessam ao mesmo tempo?
- **Uso de CPU/memória**: O sistema se mantém estável?

✨ Exemplo de Cenário de Teste:

- Configurar um teste de carga para 100 usuários simultâneos acessando a API mock.
- Medir tempo médio de resposta e documentar os resultados.
- Observar se há erros 500 ou falhas de requisição sob carga.

✪ Ferramentas Permitidas:

- **Apache JMeter** (padrão para testes de performance).
- **k6** (alternativa moderna baseada em JavaScript, ideal para integração com CI/CD).

✪ Critérios de Avaliação

- **Clareza e cobertura dos testes**: Código bem estruturado e legível.
- **Automatização bem feita**: Scripts fáceis de manter e entender.
- **Conhecimento em performance e segurança**: Identificação de possíveis gargalos.
- **Boas práticas de QA**: Aplicação de técnicas adequadas para cada tipo de teste.

📢 Entrega

1. **Código no GitHub** (público ou privado).
2. **Relatório com os resultados dos testes** (pode ser um README.md no repositório).

Boa sorte! 🚀