**Desafio**

**1 Objetivo**

O objetivo deste documento é definir as regras para implementação do desafio no intuito de testar os conhecimentos do candidato, analisar a qualidade do código desenvolvido e apoiar no enquadramento entre júnior, pleno e sênior.

**2 Tecnologia**

A aplicação deve ser construída observando-se as seguintes diretrizes tecnológicas:

Backend

* PHP;

Frontend

* HTML;
* CSS;
* Javascript

Banco de dados:

* Mysql

**3 Diretrizes**

O backend deve ser construído utilizando-se das boas práticas do desenvolvimento de aplicações PHP, observando-se padrões de projeto quando necessário, simplicidade da codificação, padrões arquiteturais, raciocínio lógico, etc.

Quanto ao frontend, não é necessário elaborar telas complexas. Não há necessidade de menu, ou seja, pode abrir a aplicação diretamente na tela de listagem. Você tem liberdade total no desenvolvimento da solução, lembrando apenas que estamos focando nas funcionalidades da aplicação e não em User interface.

Você não precisa concluir o desafio caso o tempo não possibilite. Apenas sugerimos que as funcionalidades entregues estejam completas, ou seja, se você conseguiu concluir apenas um sistema, que o mesmo funcione perfeitamente. Além disso, melhor duas funcionalidades executando corretamente, do que quatro que não funcionam.

A implementação pode ser feita em qualquer plataforma, AWS, Azure, Linode, etc, quaisquer custos decorrentes da implementação desses sistemas, até o limite de R$ 100,00, serão reembolsados mediante comprovação de que o candidato realizou o teste.

É permitido ao usuário a utilização de qualquer framework ou tecnologia que facilite ou mesmo implemente exatamente o que está sendo requisitado no desafio. Por exemplo, Laravel Queues (<https://laravel.com/docs/5.5/queues>).

**4 Conclusão do Desafio**

Para conclusão do desafio, esperamos receber:

* URL de acesso para o repositório do projeto hospedado no GitHub, GitLab, ou solução semelhante, dê preferência, com múltiplos commits para avaliarmos o fluxo de trabalho executado pelo candidato.
* Instruções para acesso ao sistema na plataforma cloud escolhida pelo candidato: AWS, Azure, GCP, Linode, DigitalOcean, etc.
* Documento respondendo os seguintes questionamentos:
* Qual o tempo aproximado gasto por você no desenvolvimento do software? (Não deve ser considerado o tempo gasto com ambiente)
* Quais as dificuldades que você enfrentou no desafio?

Dúvidas no decorrer do desafio e os artefatos de conclusão citados a cima podem ser encaminhados para o e-mail e csaraiva@widepay.com.

**Caso de Uso – Validação de URLs**

**5 Contextualização**

A Interwebs Corp. faz o rastreamento de websites sob demanda. Seus clientes podem acessar a aplicação web para cadastrar as URLs que desejam rastrear. Ao cadastrar uma nova URL o cliente apenas recebe uma confirmação de que a URL foi cadastrada com sucesso, além de poder visualizá-la na sua lista de URLs cadastradas. Dentro de poucos instantes, o robô da Interwebs, que nada mais é do que um script executado separadamente, irá acessar todas as URLs cadastradas, de todos os clientes. O robô irá armazenar o código de status HTTP e o corpo da resposta, de forma que o cliente saiba quando sua URL foi acessada, qual foi o status code retornado, bem como ter a possibilidade de visualizar o corpo do HTML retornado.

**6 Etapa 1**

O candidato deve implementar um sistema que permita um usuário autenticado cadastrar URLs e visualizar se o conteúdo referente aquela URL já foi baixado pelo segundo sistema (veja a etapa 2), com a possibilidade de visualizar esse conteúdo.

**6.1 Regras:**

* Somente usuários cadastrados e autenticados podem cadastrar URLs e visualizar o resultado das URLs previamente cadastradas.
* O formulário de cadastro de URL deve ter uma validação simples, para que a string informada no campo tenha o formato de uma URL.
* O painel de visualização das URLs deve ter um mecanismo de refresh (estilo ajax sem recarregar a página toda) para acompanhar atualizações de status das URLs.
* Para as URLs que já tiverem uma resposta, o sistema deve possibilitar a exibição do resultado (corpo da resposta HTTP).

**6.2 Opcionais**

Sugerimos algumas implementações e tecnologias que não são obrigatórias, mas havendo tempo e caso queira fazê-los, pode agregar valor à solução e avaliação da mesma:

* Mensagens de cadastro de URL Ok ou de url inválida. Outras mensagens que achar conveniente;
* Mensagens de credenciais inválidas no login;
* Opção de editar e excluir URL;
* Para URLs com resultado SEM SUCESSO pode-se desativar a visualização dos detalhes do resultado (corpo da resposta HTTP) ou pode presentar uma página de retorno com o erro 404.
* Pode ser utilizado teste unitário, TDD ou BDD no desenvolvimento realizado.
* Pode ser utilizando alguma tecnologia de frontend como Bootstrap, ExtJS, React, VueJS;
* Outros adicionais à escolha do candidato;

**7 Etapa 2**

O candidato deve implementar um sistema que armazene a resposta das requisições HTTP feitas para cada URL cadastrada pelo sistema da primeira etapa. Para cada requisição a resposta deve ser armazenada de forma que o sistema 1 tenha acesso, bem como o 'status code' da resposta.

**7.1 Regras**

* Esse sistema não é acessível pelo usuário final, não tendo qualquer tipo de painel de gerenciamento.
* O candidato pode implementar da maneira que julgar necessário um sistema que leia constantemente as URLs cadastradas pelo sistema da etapa 1, faça uma requisição HTTP à URL e armazene a resposta e o status code onde fiquem acessíveis pelo sistema da etapa 1.

**7.2 Opcionais**

Sugerimos algumas implementações e tecnologias que não são obrigatórias, mas havendo tempo e caso queira fazê-los, pode agregar valor à solução e avaliação da mesma:

* Utilização de Log;
* O sistema desenvolvido na etapa 2 pode ser executado automaticamente a cada X periodo;
* O sistema 2 pode ser implementado como um daemon, lançando dois processos filhos para efetivamente executar o trabalho de acessar as URLs que estão na fila e armazenar o resultado desse acesso.
* Outros adicionais à escolha do candidato;