Brasil TecPar

# Desafio Programador PHP

O teste é formado por 3 partes.

1. A criação de uma rota que encontra um hash, de certo formato, para uma certa string fornecida como input.
2. A criação de um comando que consulta a rota criada e armazena os resultados na base de dados.
3. Criação de rota que retorne os resultados que foram gravados.

## 1) Rota para geração do hash

A rota deve receber uma string como parâmetro e, para essa string, calcular um hash md5. Para o cálculo do hash a string informada deve ser concatenada com uma string aleatória de 8 dígitos (será chamada de key). O retorno da rota deve conter somente o hash encontrado, a key que gerou o hash e o número de tentativas necessárias para encontrar a key.

Para que o hash seja considerado uma solução para a string de entrada, ele deve começar com 4 dígitos zero. Ou seja, o hash deve ser prefixado com 0000.

As keys candidatas a serem uma solução devem ser concatenadas com a string de entrada até que um hash com esse padrão seja encontrado. Em pseudocódigo a geração do hash deve ser feita concatenado como exemplificado abaixo. Os textos da string e da key devem ser concatenados sem nenhum caractere entre eles e na ordem exemplificada.

*md5( concat( <string>, <key> ) )*

Além da implementação descrita acima, deve haver um limite no número de requisições aceitas de um mesmo IP a 10 requisições a cada 1 minuto. Ou seja, se dentro de uma janela de 1 minuto forem feitas 11 requisições, o response da décima primeira requisição deve retornar código 429 e a mensagem *“Too Many Attempts“*.

Pontos chave

- Encontro da key que gere hash iniciando com 4 zeros para certa string informada.
- Controle do número máximo de requisições aceitas em 1 minuto pela rota.

## 2) Comando para consulta da rota

Deve ser criado um command Symfony para consultar a rota descrita acima. Esse deve ter a assinatura conforme mostrado abaixo. Com 2 parâmetros de entrada

1. a string inicial que será enviada na primeira requisição;
2. o número de requisições que devem ser feitas em sequência.

```
avato:test string --requests=100
```

O comando começa fazendo a requisição enviando a string informada como parâmetro na chamada do comando. Por exemplo, considerando a chamada do comando exemplificada abaixo, a string inicial será a palavra “Ávato”. Ao enviar essa string, o retorno será o hash encontrado, a key que gerou o hash e o número

de tentativas realizadas para achar a chave. Essas informações devem ser gravadas em uma tabela de resultados junto com o número da tentativa (qual a requisição na sequência sendo executada dentro do comando), a palavra usada para gerar o hash e um identificador da execução (chamado identificador do batch). Essa informação deve ser a data e hora que o comando começou a executar e deve ser gravada em todas as linhas criadas na tabela pela execução, conforme exemplificado na tabela abaixo. Será usado para identificar a qual execução certo bloco pertence.

Na próxima interação a string enviada para a rota deve ser o hash que foi retornado na requisição anterior. Esse processo deve se repetir x vezes, sendo x o valor informado no parâmetro requests, descrito acima. A execução de todas as requisições deve respeitar os limites da rota criada e deve ser otimizada para que todas as requisições sejam feitas no menor tempo possível.

Considerando o exemplo abaixo, 20 requisições serão realizadas e 20 linhas devem ser gravadas na base de dados, sendo que cada linha começa com o hash da linha de cima, exceto a primeira linha que terá como input a string informada na chamada do comando.

```
avato:test “Ávato” --requests=20
```

Para o comando exemplificado acima um exemplo de solução é mostrado na tabela abaixo. Somente 7 linhas são mostradas, mas o processo se repetiria para as 20 linhas que foram solicitadas na chamada do comando.

| Batch | Número do bloco | String de entrada | Chave encontrada | Hash gerado | Número de tentativas |
|---|---|---|---|---|---|
| 2021-06-01 22:47:12 | 1 | Ávato | Ns3Qw6Ni | 0000e5a78737e88a6065b37aa698bb4d | 6915 |
| 2021-06-01 22:47:12 | 2 | 0000e5a78737e88a6065b37aa698bb4d | tMw8dIas | 0000a93f9f287a03912c8a61d79ffb23 | 12306 |
| 2021-06-01 22:47:12 | 3 | 0000a93f9f287a03912c8a61d79ffb23 | CBsHVmpl | 0000e4c93fbdc09d46361dafc0b5a00c | 462343 |
| 2021-06-01 22:47:12 | 4 | 0000e4c93fbdc09d46361dafc0b5a00c | tnWLjAeX | 00003794acd6639aa65e1be105a568e6 | 209564 |
| 2021-06-01 22:47:12 | 5 | 00003794acd6639aa65e1be105a568e6 | SUUxlWdG | 0000e350aea8329b93a7346e77cb47b3 | 89035 |
| 2021-06-01 22:47:12 | 6 | 0000e350aea8329b93a7346e77cb47b3 | CWHLtOLL | 0000f06cc0271af767f1976526a0a4ec | 28650 |
| 2021-06-01 22:47:12 | 7 | 0000f06cc0271af767f1976526a0a4ec | sMVxU8w6 | 000037181279df3b128f3f6722adf62f | 42819 |

*nome de colunas somente como exemplo, não precisam ser exatamente iguais*

Pontos chave

- Buscar e armazenar resultados obtidos.
- Respeitar o limite de requisições da rota.
- Aguardar o menor tempo possível para realização de todas requisições solicitadas.

Brasil TecPar

## 3) Rota de retorno dos resultados

Criar rota para retorno dos resultados que foram gravados. A rota deve:

- Retornar os resultados de forma paginada;
- Ter o filtro por “Número de tentativas” podendo filtrar por resultados que tiveram menos de *x* tentativas.
- Não devem ser retornados todos os campos da tabela, somente as informações nas colunas batch, “Número do bloco”, “String de entrada” e “Chave encontrada”.

## Regras

- Utilizar Symfony. Priorizando as funções disponíveis que facilitem a resolução do problema;
- Ter instruções de como consultar a rota criada;
- Ter instruções de como executar o command criado;
- Explicação da solução apresentada e principais decisões tomadas na implementação;
- Entregar o resultado pelo GitHub em repositório privado liberando acesso para cati@brasiltecpar.com.br. Quando a implementação estiver finalizada, notificar neste mesmo e-mail conclusão do teste.

## O que será avaliado

- Cumprimento dos requisitos apresentados na descrição;
- Organização do código;
- Uso dos recursos Symfony disponíveis para solução do problema;
- Simplicidade da solução implementada;
- Para teste da implementação o comando será executado pelo menos com 5, 10 e 100 requisições;
- Não há necessidade de criação de interfaces web.