04

O campo _all e sua relação com as buscas

Fomos apresentados a API `_search` nos exercícios do primeiro capítulo. Apenas para lembrarmos o que fizemos, utilizamos os seguintes comandos:

```
GET /catalogo/pessoas/_search
{}
```

e

```
GET /catalogo/pessoas/_search?q=futebol
{}
```

Agora que lembramos o que havíamos feito, precisamos entender bem o que aconteceu. Quando utilizamos a API `_search`, basicamente estamos executando uma busca em um índice ou em um índice e um tipo. Vale destacar que caso o tipo seja omitido, a consulta será executada em todos os tipos existentes no índice. Mas como essa busca funciona? Sabemos que procurar o valor em todos os campos não será nada eficiente.

Buscas com campo _all

Quando adicionamos documentos a um índice, o ElasticSeach combina os valores de todos os campo em um único campo chamado *_all*. Pense em pegar todos os valores, convertê-los para *string* e concatená-los. Por exemplo, para o documento:

```
{
    "nome" : "João Silva",
    "interesses" : ["futebol", "música", "literatura"],
    "cidade" : "São Paulo",
    "formação" : "Letras",
    "estado" : "SP",
    "país" : "Brasil"
}
```

Teríamos algo como:

`"João Silva futebol música literatura São Paulo Letras SP Brasil"`.

Caso o nome do campo não seja informado explicitamente para a API `_search` junto ao parâmetro *q*, a busca será executada utilizando o `campo _all`. Logo, `_search?q=futebol` é equivalente a `_search?q=_all:futebol`.

##Buscas com parâmetros

Caso queiramos nos limitar ao campo interesses, bastaria fazer:

```
_search?q=interesses:futebol
```

Repare a sintaxe `campo:termo` na busca. Se quisermos buscar com mais de um campo, basta usar o `&` :

```
_search?q=interesses:futebol&cidade:rio
```

##Definindo a quantidade de resultados

Outro detalhe importante é em relação a quantidade de documentos retornados. Por padrão, o ElasticSearch retorna até 10 documentos. Como temos menos documentos em nosso índice, temos a impressão que todos os resultados são sempre retornados.

Assim como em bancos de dados, podemos indicar tanto quando registros queremos retornar como o ponto de início:

```
GET /catalogo/pessoas/_search?size=50
```

Para indicar o primeiro basta fazer como no exemplo a seguir:

```
GET /catalogo/pessoas/_search?size=50&from=10
```

** Importante: Paginação deve ser usada apenas para pequenos volumes de dados, como alguns poucos milhares. Para volumes maiores, devemos utilizar a abordagem da API *scroll*. Veja mais detalhes no final deste capítulo. ***

Devemos também entender a resposta recebida quando usamos APIs de busca, como por exemplo a API `_search` que utilizamos até então. Veja que para as consultas executadas até o momento, recebemos resultados com o seguinte conteúdo:

```
{
  "took" : 59,
  "timed_out" : false,
  "_shards" : {
    "total" : 5,
    "successful" : 5,
    "failed" : 0
  },
  "hits" : {
    "total" : 6,
    "max_score" : 1.0,
    "hits" : [ … resultados ...]
```

Vamos ao significados:

- `took` : duração da consulta em milissegundos.
- `time_out` : valor booleano indicando se a consulta foi abortada por limite de tempo ou não.
- `_shards.total` : número de shards envolvidas na busca.
- `_shards.successful` : número de shards que não apresentaram falha durante a busca.
- `_shards.failed` : número de shards que apresentaram falha durante a busca. Note que o resultado pode não apresentar documentos presentes nas shards que falharam.
- `hits.total` : quantidade de documentos encontrados.

- `hits.max_score` : valor máximo de *score* entre os documentos encontrados. O valor 1.0 significa que o valor da busca foi encontrado totalmente.
- `hits.hits` : os documentos encontrados.

O que aprendemos?

- O campo `_all` e como ele é utilizado.
- Podemos especificar o campo na busca usando `campo:termo` .
- ElasticSearch retorna 10 resultados por padrão.
- Entender as estatísticas que são parte da resposta de requisições de busca, como tempo de resposta e hits.