Spring Data

//Elasticsearch

IT BOARDING

BOOTCAMP

Índice

IT BOARDING

BOOTCAMP

Consultas

IT BOARDING

BOOTCAMP

Tipos de búsqueda en ES.

En Elasticsearch tenemos dos tipos de búsquedas:

- Búsquedas con puntuación ó queries.
- Búsquedas sin puntuación ó filters.

Queries son aquellas pensadas para búsquedas sobre frases: Miden cuánto se parece la frase buscada a la frase almacenada

Filters son para búsquedas binarias (si/no). Son cacheadas en memoria. Utilizan campos que no son texto.

{Sintaxis de Queries}

```
GET índice/tipo/_search
{
      "query":{
            <tipoClausula>:{<campoConsultado>:<valor>}
      }
}
```

Consulta Query para ejecutar en Kibana contra Elasticsearch. Pensada para búsquedas exactas o coincidente de frases. Mide cuánto se parece una frase buscada a la almacenada.

{Sintaxis de filters}

```
GET índice/tipo/_search
{
   "query":{
     "bool":{"filter":{
       <tipoClausula>:{<campoConsultado>:<valor>}}
     }}
}
```

Consulta query que incluye un filter dentro de ella. Utiliza GET ya que es una consulta al endpoint proporcionado por Kibana. Consultas binarias donde puede o no encontrarse el valor.

Cláusulas comunes en queries.

Campo igual a valor: term

- Texto contiene a palabra (token): match
- Texto contiene palabras en orden dado: match_phrase
- Campo en un rango {gt,gte,lt,lte}

Búsquedas compuestas: bool

- Operador OR <> should (condición1 || condición2)
- Operador AND <> must (condición1 && condición2)

Agregaciones (aggs, como media, avg)

Elasticsearch almacena las frases en tokens. Ej: “Buenos Aires Ciudad”. Tokens: [“Buenos”, “Aires”, “Ciudad”]

Spring Data

//Elasticsearch

IT BOARDING

BOOTCAMP

Consultas Spring Data ES

Para hacer consultas en Spring Data Elasticsearch podemos usar la anotación **@Query** dentro de nuestros repositories:

@Query: Anotación para indicar una consulta personalizada.

Consulta para determinar si hay autores con el nombre pasado como argumento.

```
@Query("{\"bool\": {\"must\": [{\"match\": {\"autores.nombre\": \"?0\"}}]}}")
Page<Articulo> findByAuthorsNameUsingCustomQuery(String nombre, Pageable pageable);
```

Consultas con el nombre del método

También podemos crear consultas con el nombre del método. En el siguiente ejemplo vemos la misma consulta pero definimos la query con el nombre del método:

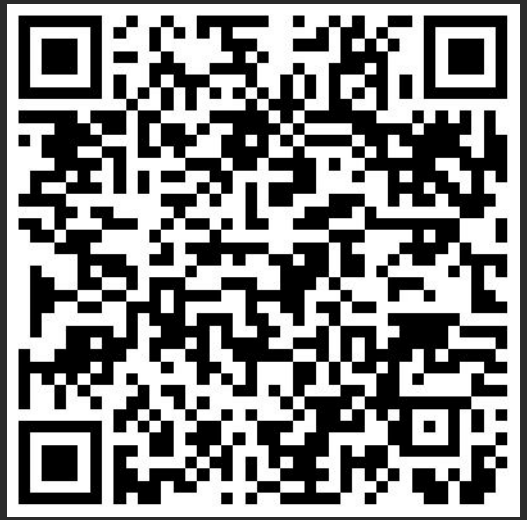
```
Page<Articulo> findByAutoresNombre(String nombre, Pageable pageable);
```

Consultas con el nombre del método

Podemos ver algunos ejemplos en la siguiente tabla:

Method	Query
`findByNameAndPrice`	`{ "query" : { "bool" : { "must" : [ { "query_string" : { "query" : "?", "fields" : [ "name" ] } }, { "query_string" : { "query" : "?", "fields" : [ "price" ] } } ] } }}`
`findByNameOrPrice`	`{ "query" : { "bool" : { "should" : [ { "query_string" : { "query" : "?", "fields" : [ "name" ] } }, { "query_string" : { "query" : "?", "fields" : [ "price" ] } } ] } }}`
`findByName`	`{ "query" : { "bool" : { "must" : [ { "query_string" : { "query" : "?", "fields" : [ "name" ] } } ] } }}`
`findByNameNot`	`{ "query" : { "bool" : { "must_not" : [ { "query_string" : { "query" : "?", "fields" : [ "name" ] } } ] } }}`

Los invitamos a completar la siguiente encuesta sobre el módulo Storage Implementación.
¡Es muy muy importante para nosotros contar con su feedback!
Solamente les tomará unos minutos completarla :)



Link a la encuesta

Gracias

IT BOARDING

BOOTCAMP