BASE DE DATOS 2 - 2021

Trabajo de Promoción - ElasticSearch

GRUPO 16

Román Enrique 9959/3 caos.enr@gmail.com

Nahuel Morena 16290/1 nahuel1morena1@gmail.com

Daniela Ruiz 13045/6 danyparedesruiz@gmail.com

Índice:

1. Ayudante guía durante la cursada:

LUCIANO BENITEZ

2. Base del Trabajo:

Tomamos como base el TP4, implementado en SpringData JPA bajo el framework Springframework y con una base de datos relacional MySql.
El TP4 consiste en una capa de servicios, repositorios y 2 juegos de test.

3. Trabajo a realizar:

Cambiar la base de datos relacional por una base de datos NoSql como ElasticSearch orientada a documentos JSON, manteniendo las funcionalidades implementadas en la capa de servicios del TP4. Como parte de la evaluación vamos a seguir usando los test provistos por la cátedra para asegurar que se mantengan dichas funcionalidades.

4. Pasos realizados:

4.1. Limpieza del proyecto base:

Borrar las dependencias maven utilizadas para la persistencia en la base relacional.

Borrar anotaciones en las clases del modelo de datos.

Borrar clases y métodos específicos de la configuración para la base relacional.

4.2. Configuración del nuevo proyecto:

Agregar nuevas dependencias:

- https://mvnrepository.com/artifact/org.springframework.data/spring-data-elasticsearch/4.2.1
- https://mvnrepository.com/artifact/org.springframework/spring-web/5.3.7
- https://mvnrepository.com/artifact/org.elasticsearch/elasticsearch/7.12.1

Configurar Spring para que trabaje con ElasticSearch

- Definición de un cliente básico, en nuestro caso un RestHighLevelClient.
- Definición de un ElasticsearchOperations, en nuestro caso ElasticsearchRestTemplate.

```
@Override
        @Bean
        public RestHighLevelClient elasticsearchClient(){
                ClientConfiguration clientConfiguration =
ClientConfiguration.builder().connectedTo("localhost:9200").build();
                return RestClients.create(clientConfiguration).rest();
        }

        @Bean
        public ElasticsearchOperations elasticsearchTemplate() throws
Exception{
                return new ElasticsearchRestTemplate(elasticsearchClient());
        }
```

La clase configuración SpringDataElasticsearchConfiguration.java contiene estos cambios.

Anotar las clases del modelo de datos:

@Document en reemplazo de la anotación @Table, para representar a un documento en elasticsearch.

Parámetro index=”nombreIndice” para especificar que utilizaremos el índice nombreIndice para el documento.

@Field en reemplazo de la anotación @Column, para representar a las columnas o atributos.

FieldType para especificar el tipo de dato del atributo.

Configurar los repositorios SpringData para elasticSearch

Los repositorios anteriormente extendían a CrudRepository, ahora pasarán a extender a la interface ElasticSearchRepository.

Además el ID es cambiado de LONG a STRING para evitar conflictos en elasticSearch.

5. Problemas y soluciones encontradas:

- Incompatibilidad de versión springFramework utilizado en el TP4 con las utilizadas en la implementación de SpringData. Esto lo solucionamos siguiendo la tabla de compatibilidad que se encuentra en la guía oficial de SpringData.
- SpringData no soporta la última versión de ElasticSearch por lo tanto hay que instalar la versión específica que se encuentra en la tabla comentada anteriormente.
- Conflicto con datos basura en los test, esto se debió a que elasticSearch no soporta transiciones por lo tanto después de que se ejecutaba un test quedaban datos basura utilizado por el mismo que interfieren con los test que le seguían. Aprovechando que springData nos provee varios métodos ya implementados utilizamos los de borrado “delete” y “deleteAll” para mantener un consistencia en los datos. Además retiramos todas las anotaciones relacionadas con las transacciones.
- Problema con la búsqueda exacta de atributos String, esto sucedía porque elasticSearch analiza y indexa los string/text para facilitar la posterior búsqueda de los mismo, con la configuración inicial y con la consulta implementada en el método findByName(String name) de springdata los devolvía los string similares además de los que incluía al string pasado por parámetro. Esto lo solucionamos marcando al atributo como una “keyword” para que elasticSearch lo analice como una palabra completa y utilizando la anotación @Query("{\"match_phrase\": {\"name\": \"?0\"}}") con una consulta específica.
- Un caso similar al anterior, nos encontramos con la persistencia de las fechas (Date) se analizan de manera similar al caso anterior que si no se tiene cuidado puede dar resultados poco exactos. En este caso implementamos atributo “actual” booleano extra que representa que un ProductOnSale esta vigente o no. Fue la solución más rápida y sencilla que encontramos para evitar estancarnos con el manejo de fechas que no aportaba demasiado en este TP.
- SpringData repositories no soluciona o da soporte a todas las consultas que necesitamos para cumplir con las funcionalidades. No llegamos a solucionar este problema por falta de tiempo y problemas de recursos que faciliten las pruebas para la comprensión del entorno de trabajo nuevo y muy diferente a lo que estábamos acostumbrados.

  La posible solución es realizar consultas DSL o nativas utilizando directamente el cliente. Además elasticSearch da soporte a consultas SQL limitadas pasados como string, esta solución la evitamos considerando que era un escape fácil al problema. Estas soluciones deberían ir en implementaciones custom de un repositorio.

6. Conclusiones:

El TP fue todo un desafío desde el punto de vista que es algo muy diferente a la orientación que nos llevan en la facultad hacia las bases relacionales. Además del uso considerado de recursos computacionales de las herramientas en cuestión que complicaron las pruebas para comprenderlas mejor.

7. Lenguajes y herramientas utilizadas:

7.1. IDE Eclipse última versión.

7.2. Java 8.

7.3. Maven integrada a la última versión de Eclipse.

7.4. SpringFramework v 5.3.7

7.5. SpringData ElasticSearch 4.2.1

7.5.1. Elasticsearch v 7.12.1

7.5.2. SpringWeb v 5.3.7

7.6. JUnit v 4.13

7.6.1. JUnit Jupiter v 5.0.0

7.7. ElasticSearch v 7.12

7.7.1. Kibana v 7.12

8. Bibliografía utilizada:

8.1. Teorías de la cátedra

8.2. Tabla de compatibilidad - https://docs.spring.io/spring-data/elasticsearch/docs/current/reference/html/#preface.versions

8.3. Documento de referencia ElasticSearch 7.12 - https://www.elastic.co/guide/en/elasticsearch/reference/7.12/index.html

8.4. Documento de referencia Kibana 7.12 - https://www.elastic.co/guide/en/kibana/7.12/index.html

8.5. SpringData ElasticSearch - https://docs.spring.io/spring-data/elasticsearch/docs/current/reference/html/#preface

8.6. Buscador de dependencias Maven:

8.6.1. https://mvnrepository.com/artifact/org.springframework.data/spring-data-elasticsearch/4.2.1

8.6.2. https://mvnrepository.com/artifact/org.springframework/spring-web/5.3.7

8.6.3. https://mvnrepository.com/artifact/org.elasticsearch/elasticsearch/7.12.1

8.7. Guías y ejemplos:

8.7.1. https://stackoverflow.com/

8.7.2. https://www.elastic.co/es/blog/