**ENTREGA 3**

**Report item 4**

**Diseño y Pruebas**



**Grupo 20:**

**José Ángel Domínguez Espinaco**

**Daniel Lozano Portillo**

**José Joaquín Rodríguez Pérez**

**María Ruiz Gutiérrez**

**Miguel Ternero Algarín**

**Laura Vera Recacha**

Índice:

Introducción 3

Implementacion 3

Resultados 7

Bibliografia: 8

### Introducción

En este documento se explica detalladamente como se llevó a cabo la implementación de la utilidad Hibernate Search con JPA. También se muestran los resultados y algunas partes del código implementado para su funcionamiento adecuado.

### Implementación

Lo primero que hay que hacer es añadir la dependencia en el archivo pom.xml

**<dependency>**

**<groupId>**org.hibernate**</groupId>**

**<artifactId>**hibernate-search-orm**</artifactId>**

**<version>**5.8.1.Final**</version>**

**</dependency>**

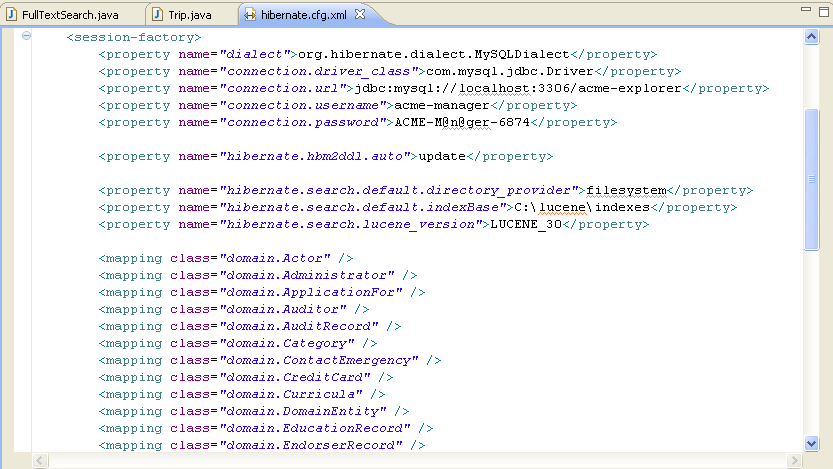
Si está utilizando Hibernate a través de JPA, se agregó estas mismas propiedades a persistence.xml. El siguiente paso es crear el archivo hibernate.cfg.xml en *src/MAIN/resources* y agregar tanto las clases de su dominio como la referencia a la base de datos.

**<property** name="hibernate.search.default.directory\_provider"

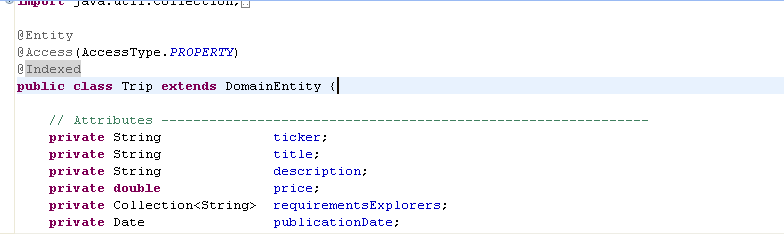
value="filesystem"**/>**

**<property** name="hibernate.search.default.indexBase"

value="/var/lucene/indexes"**/>**



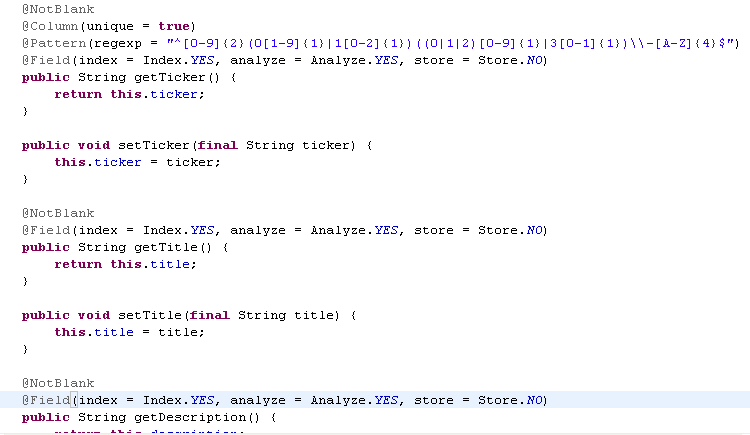
Una vez realizado esto añadimos a la clase Trip @Indexed ya que es la clase que se va Indexar.



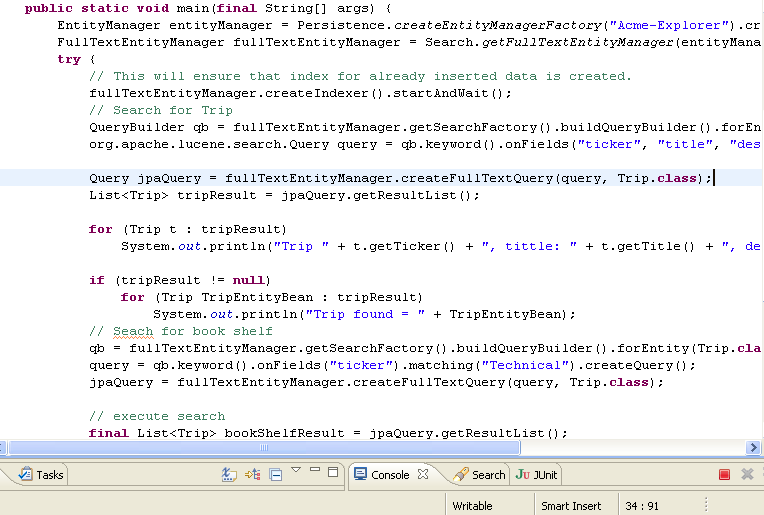
A continuación, debe añadir la anotación @Field en los atributos que desee que sean buscados. En nuestro caso serán ticker,title, y description.

@Field(index = Index.YES, analyze = Analyze.NO, store = Store.YES)

El índice de parámetro = Index.YES asegurará que el texto será indexado, mientras que analyse = Analyze.YES asegura que el texto será analizado usando el analizador Lucene predeterminado. El tercer parámetro, store = Store.NO, garantiza que los datos reales no se almacenarán en el índice. Que estos datos estén almacenados en el índice o no, no tiene nada que ver con la capacidad de buscarlos: el beneficio de almacenarlos es la capacidad de recuperarlos mediante proyecciones



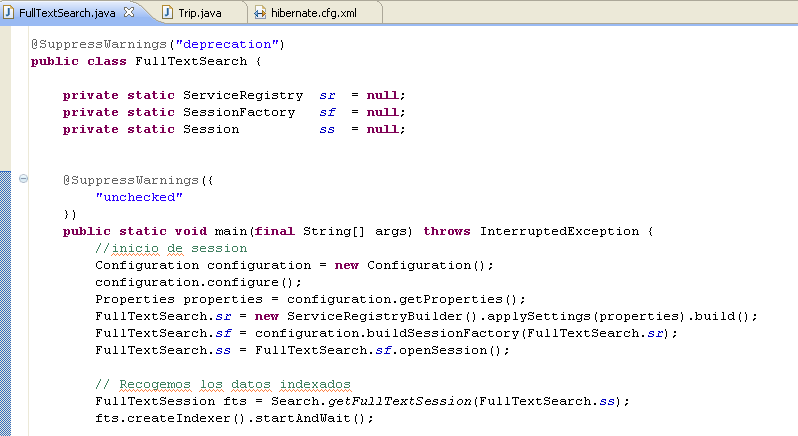
Ya solo nos faltaría crear la Clase para que todo esto funcione.



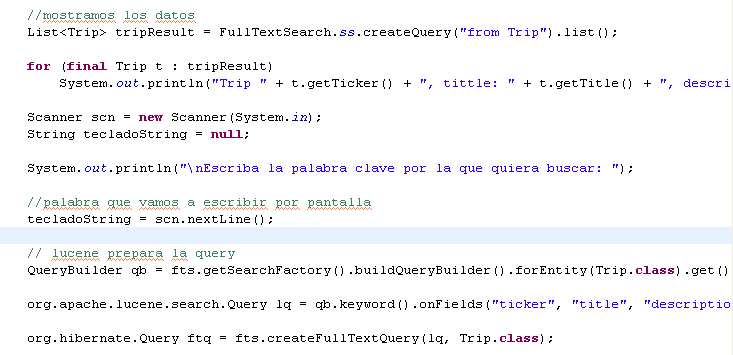
Cuando ya estaba todo listo para funcionar, al ejecutar hibernate, mostro un error de compatibilidad con la versión, por lo que procedimos a hacerlo con la versión 4.5.0 que encontramos que era compatible con nuestra versión de JPA e hibernate.

Finalmente, la versión más nueva no usaba la misma forma para indexar los datos ya populados en la base de datos con lo que se tuvo que cambiar. Y este fue el resultado:

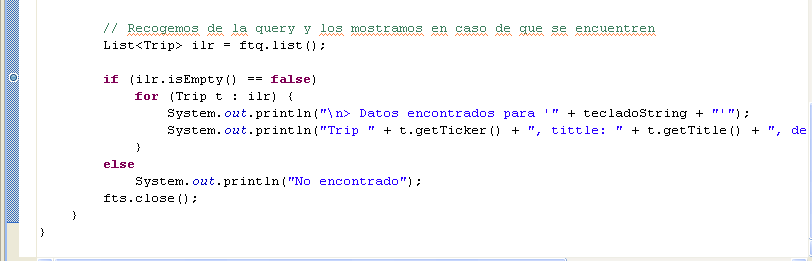
En esta primera imagen se muestra el inicio de sesión y como se recogen los datos indexados.



En la siguiente como se muestran los datos de las Trip indexadas y se pide por pantalla una palabra para la búsqueda y con lucene se busca mediante una query de lucene.

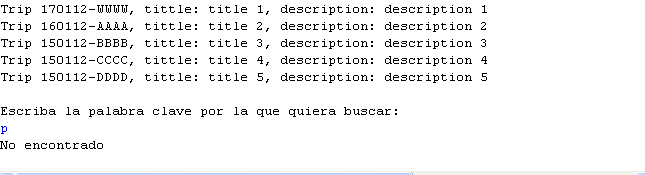


Por último se muestran los datos almacenados en la lista si es que los hubiese.

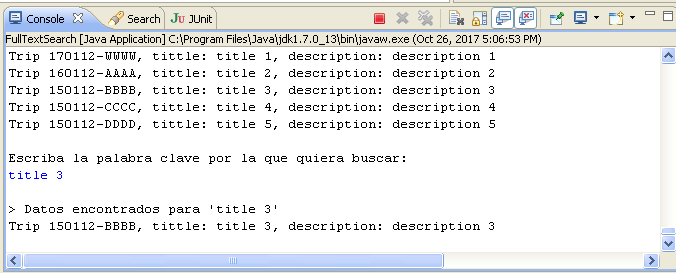


### Resultados

Intentamos buscar una palabra no indexada



Buscando una palabra indexada



Una vez realizada una búsqueda deberá volver a ejecutar FulltextSearch.java para volver a realizar otra. Tenga en cuenta que al volver a ejecutar mostrara algún warning debido a que ya están indexados algunos datos.

### Bibliografía:

<http://hibernate.org/search/documentation/getting-started/>

<https://mprabhat.me/2012/09/30/full-text-search-with-hibernate-search-4-1-lucene-and-jpa/>