Práctica 2:
Linked Data, Big Data y Web Social

Introducción y Objetivos

El objetivo de la práctica es la gestión de datos de la Web Social utilizando tecnologías de Linked Data y de Big Data, todos ellos conceptos importantes en el ámbito de los Sistemas de Información Distribuidos. Se pretende:

- Saber utilizar una API para manejar consultas en SPARQL.
- Saber acceder a un punto de acceso SPARQL (*endpoint*) para responder consultas.
- Conocer la estructura de almacenamiento de datos de la DBpedia.
- Saber utilizar una API para recuperar datos de Twitter.

Como ejemplo de aplicación, se propone el dominio de futbolistas (por ser tendencia en Twitter con frecuencia), pero los alumnos pueden proponer aplicaciones alternativas de su interés personal.

Ejercicios

1. Desarrollar una aplicación Java que, dada una serie de cadenas de caracteres, escriba cuáles de ellos corresponden a futbolistas. Opcionalmente, se puede mostrar el club actual del jugador y el estadio donde el equipo tiene su sede.

La información necesaria se obtendrá de la DBpedia en inglés a través de una consulta SPARQL. Puesto que la DBpedia se actualiza con poca frecuencia (e.g., según sus datos Neymar se *quedó* en el Barça), usaremos el punto de acceso de DBpedia-live (http://dbpedia-live.openlinksw.com/sparql), de actualizaciones mucho más frecuentes (ver http://downloads.dbpedia.org/live/changesets). Por ejemplo, comparemos las páginas de Neymar en la DBpedia (http://dbpedia.org/page/Neymar) y en la DBpedia-live (http://live.dbpedia.org/page/Neymar). Para una entrada como "Neymar_Jr", se debería detectar que existe una redirección al recurso "Neymar" y mostrar la salida correcta.

En concreto, se utilizará la biblioteca Apache Jena para interactuar con el punto de acceso: descargar Apache Jena http://jena.apache.org/download y añadir las bibliotecas `.jar` en la carpeta `lib` a nuestro proyecto. Para encontrar las propiedades RDF relevantes para la consulta SPARQL, consultar la página en DBPedia de varios futbolistas.

2. Extender la aplicación anterior para considerar las tendencias (*trending topics*) en Twitter a nivel nacional como entrada. Para acceder a los datos de Twitter, usaremos la biblioteca Twitter4j, que puede descargarse de http://twitter4j.org y se deberá añadir el fichero `twitter4j-core-4.0.7.jar` (o la versión que corresponda) como biblioteca de nuestro proyecto Java.

Además, habrá que configurar crear una cuenta de Twitter (en http://twitter.com) si es que no se tiene, añadiendo el número de teléfono a la cuenta, y registrar nuestra aplicación en la página http://apps.twitter.com.

En la pestaña *"Your Access Token"*, obtener los Access Token y el Access Token Secret. Una vez registrada la aplicación, configurar Twitter4j para usar nuestras claves (*consumerKey*, *consumerSecret*, *accessToken* y *accessTokenSecret*) a través del fichero `twitter4j. properties`, tal y como se explica en la página http://twitter4j.org/en/configuration.html#configuration.

Nota: puede haber ambigüedad al usar la clase `Query`, siendo preciso especificar a qué paquete pertenece: `twitter4j.Query` o bien `org.apache.jena.query.Query`.

Entrega

La entrega se realizará a través de Moodle hasta las 12 horas del **22 de marzo** y consistirá en un fichero `.zip` que incluya al menos

- Las clases `.java` desarrolladas por el alumno,
- un ejecutable `.jar` y
- una brevísima documentación en `pdf` con los nombres de los autores y los aspectos no triviales del trabajo realizado, incluyendo la instalación de las herramientas necesarias.

Por privacidad, asegurarse de que la entrega NO incluye el fichero `twitter4j.properties`.

Apéndice. Plantilla para Jena

```
import java.util.*;
import org.apache.jena.query.*;
import org.apache.jena.rdf.model.*;

public class P2
{

   public static void main(String[] args)
   {
      String tendencias[] = {"Neymar", "Iago_Aspas", "Lionel Messi"};
      Set<String> resultados = new HashSet<String>();
      String sparqlEndpoint = "http://dbpedia-live.openlinksw.com/sparql";

      for (String tendencia : tendencias)
      {
         String selectQuery = ...
         org.apache.jena.query.Query query = QueryFactory.create(selectQuery);
         QueryExecution exec = QueryExecutionFactory.sparqlService(
            sparqlEndpoint, query );
         try {
            ResultSet results = exec.execSelect();
            while ( results.hasNext() )
            {
               QuerySolution qs = results.nextSolution();
               RDFNode n = qs.get(variableName);
               if (! n.isLiteral() )
                  resultados.add(n.toString());
            }
         } finally {
            exec.close();
         }

         String askQuery = ...

         org.apache.jena.query.Query query = QueryFactory.create(askQuery);
         QueryExecution exec = QueryExecutionFactory.sparqlService(
            sparqlEndpoint, query );
         if (exec.execAsk() )
            resultados.add(tendencia);
      }
   }
}
```