Recuperación de Información en Web Semántica
Crawler: ¿Qué hay de comer?

Adrián Blanco Blanco
adrian.blanco@udc.es

Brais Freire
brais.freire@udc.es

1 Introducción

En esta memoria relataremos cómo hemos obtenido la solución para el problema que planteamos. Este se basa en situaciones típicas donde un grupo de personas no saben donde ir a comer, pero tienen ganas de distintos tipos de platos. Se introducen estos en nuestra aplicación y nos devolverá qué restaurantes cumplen con estos requisitos. Para ello, nos basaremos en la web de reparto de comida *JustEat*, donde podemos hacer búsquedas por zonas, pero no por restaurante o por plato concreto. Elegimos esta página por ser la que nos ofrece una información más clara y homogénea de la carta de los lugares de comida de la ciudad. Para conseguir la información implementaremos un plugin de *Apache Nutch* y se guardará y organizará la información con *hbase* y *Solr*. Después esto se mostrará en una web para hacer más cómoda

su utilización.

Nos centraremos en la zona de A Coruña, y la información a guardar para cada restaurante será el *nombre, tipo, dirección, logo, lista de platos con sus precios y posiblemente una descripción*

2 Herramientas

Vamos a ver un poco más en profundidad las distintas herramientas que utilizamos para este proyecto.

- **Apache Nutch 2.3.1:** Web Crawler Open Source escalable donde el almacenamiento es independiente del mismo, por lo que ofrece gran flexibilidad según las necesidades de cada uno. Parte de su fuerza viene en la utilización de plugins configurables y desarrollados desde cero mediante la implementación de librerías específicas. Puede utilizarse en una máquina solamente, pero podría utilizarse en varias con la ayuda de *Hadoop*. Lo utilizaremos para obtener direcciones de restaurantes y para crear dos plugins para parsear y obtener información de las páginas.
- **Solr 4.8.1:** Herramienta para el indexado, replicado y balanceado de búsquedas sobre la información, mostrandola en un formato más comprensible y fácilmente manejable para enseñar al usuario según los requisitos pedidos.
- **HBase 0.98.19-hadoop2:** Base de datos open source, no relacional y distribuida que sirve como soporte para guardar los datos obtenidos con las otras herramientas. Ofrece un servicio de clustering que hace posible la lectura y escritura de Big Data.

3 Configuración

Vamos a ver, paso a paso, cómo configurar un entorno desde cero para poder ejecutar nuestro crawler. Empezaremos configurando el archivo **apache-nutch-2.3.1/conf/nutch-site.xml**, donde pondremos el siguiente texto:

```
<configuration>
   <property>
      <name>storage.data.store.class</name>
      <value>org.apache.gora.hbase.store.HBaseStore</value>
      <description>Default class for storing data</description>
   </property>

   <property>
      <name>http.agent.name</name>
      <value>My Nutch Spider</value>
   </property>
</configuration>
```

Con esto especificaremos el tipo de sistema para la base de datos y le daremos un nombre a la *araña* que crawleará las webs. Lo siguiente es descomentar las dos líneas que nos indican para hbase[1]. Una vez hecho esto, podremos compilar nutch ejecutando **ant runtime** desde la carpeta raiz del mismo.

Para configurar hbase, lo primero será descargarlo y descomprimirlo:

```
wget http://archive.apache.org/dist/hbase/hbase-0.94.14/hbase-0.94.14.tar.gz
tar xzvf hbase-0.94.14.tar.gz
```

Ahora solo tendremos que abrir el fichero **conf/hbase-site.xml** e introducir el siguiente texto:

[1] En nuestro caso, 116 y 117

```
<configuration>
  <property>
    <name>hbase.rootdir</name>
    <value>file:///home/brais/crawler/hbasedata/</value>
  </property>
  <property>
    <name>hbase.zookeeper.property.dataDir</name>
    <value>/home/brais/crawler/zookeeper/</value>
  </property>
</configuration>
```

Ahora vamos a configurar los campos que obtendremos e indexaremos de la web, para ello añadimos el siguiente código entre las etiquetas de <*fields*> del archivo **apache-nutch-2.3.1/runtime/local/conf/schema.xml**:

```
<fields>
  [...]
  <field name="name" type="string" stored="true" indexed="true"/>
  <field name="address" type="string" stored="true" indexed="true"/>
  <field name="city" type="string" stored="true" indexed="true"/>
  <field name="typeRest" type="string" stored="true" indexed="true"/>
  <field name="imgUrl" type="string" stored="true" indexed="false"/>
  <field name="dishes" type="string" stored="true" indexed="true"/>
  <field name="prizes" type="string" stored="true" indexed="true"/>
</fields>
```

Una vez copiados, sobreescribimos el archivo **solr-4.8.1/example/solr/collection1/conf/schema.xml**. A continuación, añadimos el siguiente código a **apache-nutch-2.3.1/runtime/local/conf/solrindex-mapping.xml**:

```
<fields>
  [...]
  <!-- Our fields -->
  <field dest="name" source="name" />
  <field dest="address" source="address"/>
  <field dest="typeRest" source="typeRest"/>
  <field dest="imgUrl" source="imgUrl"/>
  <field dest="dishes" source="dishes"/>
  <field dest="city" source="city"/>
  <field dest="prizes" source="prizes"/>
</fields>
```

Ahora ya tendremos solr y hbase listos, por lo que vamos a terminar con Nutch. Primero, en la ruta **apache-nutch-2.3.1/runtime/local** creamos la carpeta *urls*, y dentro de ella, un archivo llamado seed.txt, donde pondremos las urls iniciales, que son estas:

```
https://www.just-eat.es/area/15001-a-coru%c3%b1a
https://www.just-eat.es/area/15002-a-coru%c3%b1a
https://www.just-eat.es/area/15003-a-coru%c3%b1a
https://www.just-eat.es/area/15004-a-coru%c3%b1a
https://www.just-eat.es/area/15005-a-coru%c3%b1a
https://www.just-eat.es/area/15006-a-coru%c3%b1a
https://www.just-eat.es/area/15007-a-coru%c3%b1a
https://www.just-eat.es/area/15008-a-coru%c3%b1a
https://www.just-eat.es/area/15009-a-coru%c3%b1a
https://www.just-eat.es/area/15010-a-coru%c3%b1a
https://www.just-eat.es/area/15011-a-coru%c3%b1a

https://www.just-eat.es/area/27001-lugo
https://www.just-eat.es/area/27002-lugo
https://www.just-eat.es/area/27003-lugo
https://www.just-eat.es/area/27004-lugo

https://www.just-eat.es/area/32001-ourense
https://www.just-eat.es/area/32002-ourense
https://www.just-eat.es/area/32003-ourense
https://www.just-eat.es/area/32004-ourense
https://www.just-eat.es/area/32005-ourense

https://www.just-eat.es/area/36001-pontevedra
https://www.just-eat.es/area/36002-pontevedra
https://www.just-eat.es/area/36003-pontevedra
```

```
https://www.just-eat.es/area/36004-pontevedra
https://www.just-eat.es/area/36005-pontevedra
```

Estas son las principales áreas de cada una de las capitales de provincia gallegas. A partir de aqui, se sacarán las distintas url's que contenga cada una para obtener la de los distintos restaurantes de cada zona. Además, tendremos que aplicar un filtro a las urls para que estas no se alejen de la página y no indexe las que sabemos que no nos interesan, como los menús, assets, pie de página, y los menús de los laterales donde se clasifican los restaurantes por tipos. Al eliminarlas, conseguiremos reducir de manera importante el tiempo de ejecución, al tener que parsear muchas menos cosas. Añadiremos esto al final del archivo **apache-nutch-2.3.1/runtime/local/conf/regex-urlfilter.txt**

```
-^https://www.just-eat.es/assets/.*
-^https://www.just-eat.es/cookiespolicy
-^https://www.just-eat.es/faq
-^https://www.just-eat.es/blog/.*
-^https://www.just-eat.es/blog
-^https://www.just-eat.es/contact
-^https://www.just-eat.es/adomicilio/.*
-^https://www.just-eat.es/privacy-policy

-^https://www.just-eat.es/area/15001-a-coru%c3%b1a/.*
-^https://www.just-eat.es/area/15002-a-coru%c3%b1a/.*
-^https://www.just-eat.es/area/15003-a-coru%c3%b1a/.*
-^https://www.just-eat.es/area/15004-a-coru%c3%b1a/.*
-^https://www.just-eat.es/area/15005-a-coru%c3%b1a/.*
-^https://www.just-eat.es/area/15006-a-coru%c3%b1a/.*
-^https://www.just-eat.es/area/15007-a-coru%c3%b1a/.*
-^https://www.just-eat.es/area/15008-a-coru%c3%b1a/.*
-^https://www.just-eat.es/area/15009-a-coru%c3%b1a/.*
-^https://www.just-eat.es/area/15010-a-coru%c3%b1a/.*
-^https://www.just-eat.es/area/15011-a-coru%c3%b1a/.*

-^https://www.just-eat.es/area/27001-lugo/.*
-^https://www.just-eat.es/area/27002-lugo/.*
-^https://www.just-eat.es/area/27003-lugo/.*
-^https://www.just-eat.es/area/27004-lugo/.*

-^https://www.just-eat.es/area/32001-ourense/.*
-^https://www.just-eat.es/area/32002-ourense/.*
-^https://www.just-eat.es/area/32003-ourense/.*
-^https://www.just-eat.es/area/32004-ourense/.*
-^https://www.just-eat.es/area/32005-ourense/.*

-^https://www.just-eat.es/area/36001-pontevedra/.*
-^https://www.just-eat.es/area/36002-pontevedra/.*
-^https://www.just-eat.es/area/36003-pontevedra/.*
-^https://www.just-eat.es/area/36004-pontevedra/.*
-^https://www.just-eat.es/area/36005-pontevedra/.*

# accept anything else
+^https://www.just-eat.es/.*
```

Para poder utilizar la aplicación web con jquery, fue necesario modificar Solr para evitar el error "Access-Control-Allow-Origin" . Para ello, deberemos añadir en **solr-4.8.1/example/solr-webapp/webapp/WEB-INF/web.xml** lo siguiente:

```
<filter>
 <filter-name>cross-origin</filter-name>
  <filter-class>org.eclipse.jetty.servlets.CrossOriginFilter</filter-class>
  <init-param>
    <param-name>allowedOrigins</param-name>
    <param-value>*</param-value>
  </init-param>
  <init-param>
    <param-name>allowedMethods</param-name>
    <param-value>*</param-value>
  </init-param>
  <init-param>
    <param-name>allowedHeaders</param-name>
    <param-value>*</param-value>
  </init-param>
```

```
</filter>
<filter-mapping>
  <filter-name>cross-origin</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
```

Esto actuará de filtro en las conexiones, permitiendo conexiones del tipo GET para nuestra página sin error.

4 Implementación de Plugins

Lo primero que haremos, será decirle a Nutch que queremos que trabaje con los dos plugins que vamos a hacer. Para ello, modificaremos el campo *plugin.includes* del archivo runtime/local/conf/nutch-site.xml, poniendo el siguiente:

```
<property>
   <name> plugin.includes </name>
   <value> protocol-http|urlfilter-regex|parse-(html|tika)|index-(basic|anchor)|indexer-
       solr|urlnormalizer-(pass|regex|basic)|scoring-opic|parse-justeat|index-justeat|
       nutch-extensionpoints|protocol-httpclient</value>
</property>
```

En nuestro caso hemos creado dos,*parse-justeat*, el cual recibe la información de un restaurante y saca la información relevante del mismo, *index-justeat* para indexar la información obtenida por el parser y poder mandarla a solr. Primero explicaremos como crear la estructura que rodea al plugin con el ejemplo del indexer, ya que el del parser se haría de manera análoga, para después explicar ambos plugins.

4.1 Configuración del plugin

Comenzaremos creando una carpeta con el nombre del plugin en **apache-nutch-2.3.1/src/plugin**, y dentro de ella los siguientes tres archivos:

build.xml

```
<project name="index-justeat" default="jar-core">
  <import file="../build-plugin.xml"/>
</project>
```

ivy.xml

```
<ivy-module version="1.0">
  <info organisation="org.apache.nutch" module="${ant.project.name}">
    <license name="Apache 2.0"/>
    <ivyauthor name="Apache Nutch Team" url="http://nutch.apache.org"/>
    <description>
        Apache Nutch
    </description>
  </info>

  <configurations>
    <include file="../../..//ivy/ivy-configurations.xml"/>
  </configurations>

  <publications>
    <!--get the artifact from our module name-->
    <artifact conf="master"/>
  </publications>

  <dependencies>
  </dependencies>

</ivy-module>
```

plugin.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<plugin
  id="index-justeat"
  name="JustEat Indexing Filter"
  version="1.0.0"
  provider-name="nutch.org">
  <runtime>
    <library name="index-justeat.jar">
      <export name="*"/>
    </library>
  </runtime>
  <requires>
    <import plugin="nutch-extensionpoints"/>
  </requires>
  <extension id="org.apache.nutch.indexer.justeat"
      name="Nutch JustEat Indexing Filter"
      point="org.apache.nutch.indexer.IndexingFilter">
    <implementation id="JustEatIndexer"
      class="org.apache.nutch.indexer.justeat.JustEatIndexer"/>
  </extension>
</plugin>
```

Para terminar la configuración, añadiremos las siguientes lineas en **apache-nutch-2.3.1/src/plugin**.

```
<target name="deploy">
   [...]
   <ant dir="parse-justeat" target="deploy"/>
   <ant dir="index-justeat" target="deploy"/>
</target>
```

4.2 Implementación

4.2.1 parse-justeat

Dentro de su carpeta, creamos la siguiente ristra, **src/java/org/apache/nutch/parse/justeat/JustEatParser.java**. Dentro de la misma tendremos tres funciones importantes, que se llaman en este orden:

1. **filter:** Implementación de *ParseFilter*. Recibe el contenido de las distintas páginas para pasarlas a la siguiente función. Si resulta tener el contenido esperado, osea, es un restaurante, se mandan al *pushIntoMetadata* para que llegue al index-justeat.
2. **getRestaurant:** Recibe el contenido de una página y recorre sus etiquetas para ver si alguna contiene un nombre igual al que buscamos. Por ejemplo, para obtener el nombre del restaurante buscamos la etiqueta *h1* con un nombre de clase *infoTextBlock-item-title*.
3. **pushIntoMetadata:** Para cerrar, recibimos los atributos del restaurante, y los ponemos en la *metadata* de la página, para que pueda ser indexada posteriormente por el otro plugin.

4.2.2 index-justeat

En esta ocasión, crearemos **src/java/org/apache/nutch/indexer/justeat/JustEatIndexer.java**, donde implementaremos *IndexingFilter*. Para ello solo tendremos una función *filter*, la cual extrae el metadata de la página para prepararlo en un *NutchDocument* para que solr sea capaz de interpretarlo.

5 Interfaz Web

Para hacer esto accesible a un usuario medio, vamos a preparar una interfaz web que lea los datos de Solr. Tendremos una caja de texto donde el usuario pondrá los distintos platos que quiere que contenga

el restaurante al que comer. Además, tiene un desplegable donde puede elegir si quiere buscar en todos los restaurantes o filtrar por una ciudad específica.

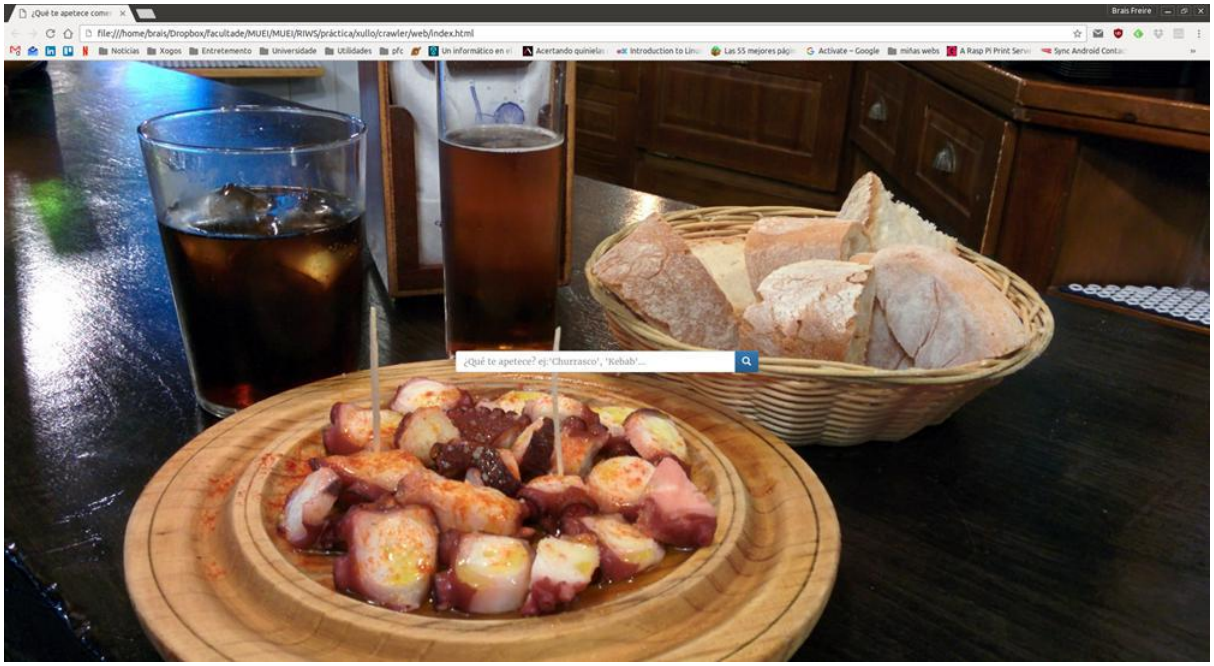

Figure 1: Pantalla inicial de la web

La consulta se realiza poniendo, en la caja de texto, los nombres de los platos separados por AND. Por ejemplo:

```
Pulpo AND Churrasco
```

O, simplemente, un único plato:

```
Ensalada
```

Una vez dentro, tendremos los distintos restaurantes que cumplen los requisitos, donde podremos ver un botón que posibilita ver el menú que ofrecen a través de un modal junto a sus precios. Además, podemos acceder directamente a la página de JustEat por si prefiere hacer el pedido a domicilio y ahorrar en los gastos de envío evitando tener que pedir a más de un sitio.

También podemos filtrar la búsqueda por tipo de restaurante, pudiendo así afinar todavía más la misma y adaptar las preferencias del usuario al local que más le apetece.

5.1 ¿Cómo está hecho?

La aplicación web está hecha sobre html+css+jquery. Consta de 2 páginas unicamente: index.html y results.html.

Index.html

Desde index.html se introduce el plato a buscar, que se envía a results.html a traves de GET. También se valida el formulario para evitar enviar datos vacíos.

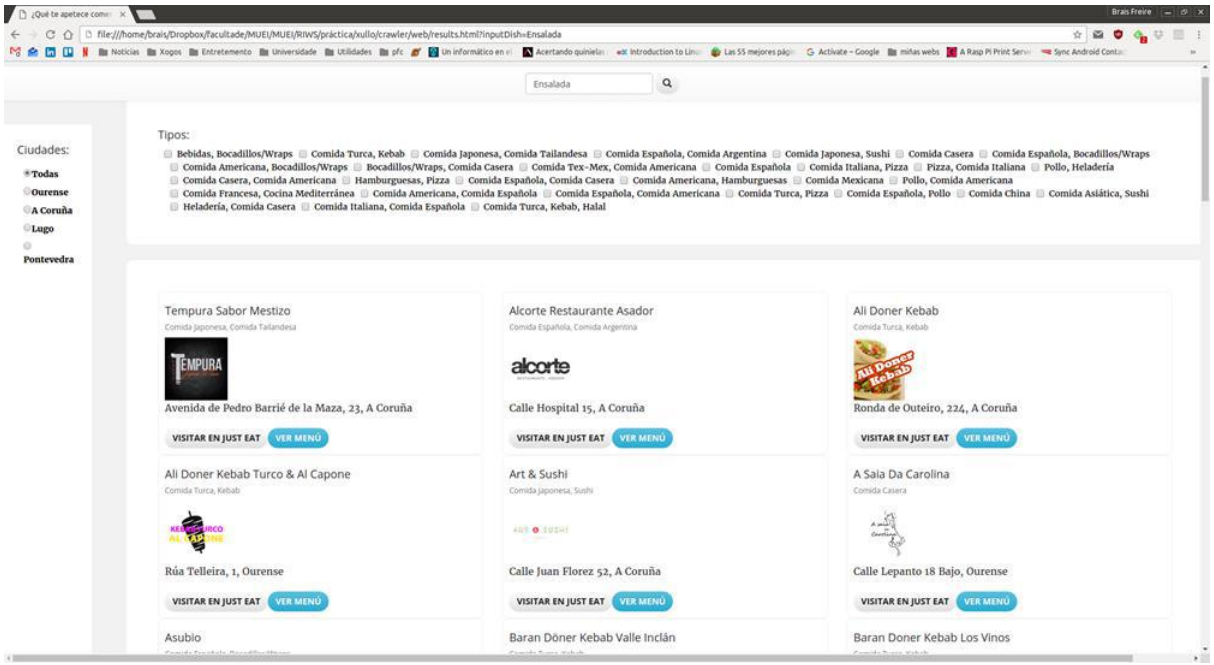

Figure 2: Búsqueda por platos

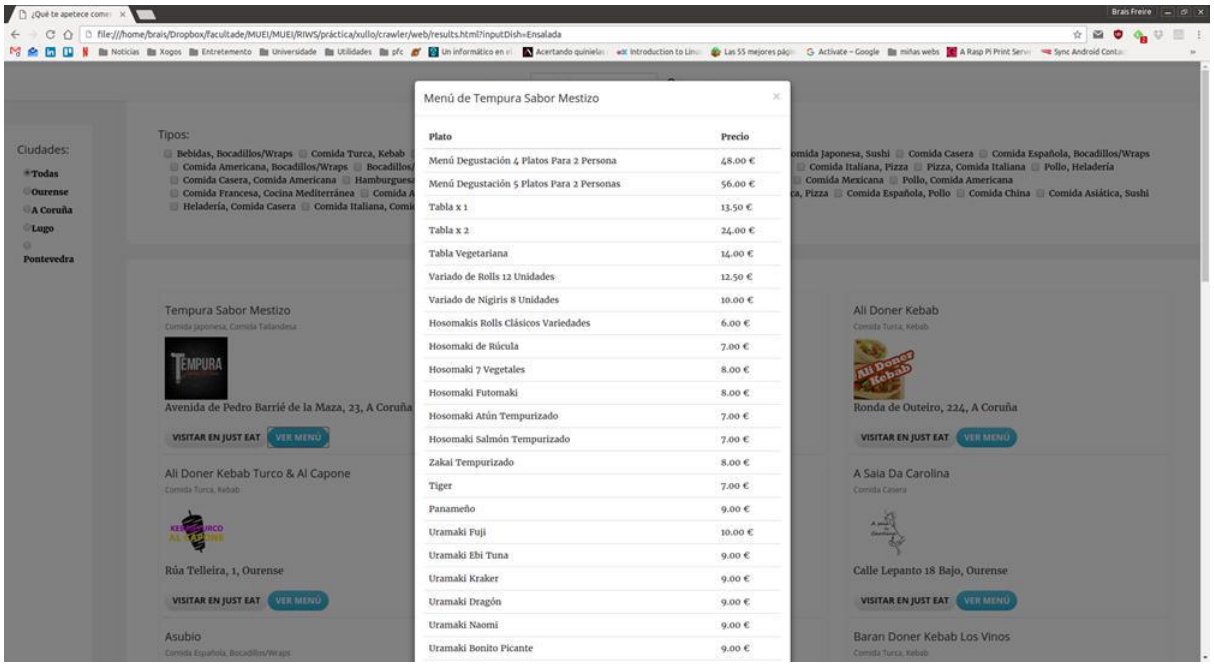

Figure 3: Ejemplo de menú de un restaurante

Results.html

En esta página es donde se realiza practicamente todo. Consta de 4 partes: una zona superior con otro componente de texto para buscar, inmediatamente debajo tenemos la zona donde se muestran los distintos tipos de restaurantes, a la izquierda el filtro por ciudades y, por último, el área donde se muestran los resultados.

El core de esta página es un archivo llamado search.js. Aquí se construyen las distintas url al endpoint de solr. Al cargar la página por primera vez, se generan los filtros (tanto de restaurantes como de ciudades) y se imprimen a la página. Esto se hace en la función llamada printMenus().

```
function printMenus(){
  //Getting the diferent cities of the restaurants in Solr

  $.getJSON(cities, function(json){ //Connecting to Solr and getting data

  }).done(function(json){
    $('#loadingDivC').hide();
    var cities = parseElements(json.response.docs); //We transform items for usability
    $('<label class="checkbox"><input class="box" type="radio" name="city" value="All"
        onclick="filterCity(this);" checked="checked">Todas</label>').appendTo('#
        cityCheckboxes');
    for (var i=0; i<cities.length; i++){
      $('<label class="checkbox"><input class="box" type="radio" name="city" value="'+
          cities[i]+'" onclick="filterCity(this);">'+cities[i]+'</label>').appendTo('#
          cityCheckboxes');
    }
  });

  //Getting the diferent types of the restaurants in Solr
  $.getJSON(typeRestaurants, function(json){

  }).done(function(json){
    $('#loadingDivT').hide();
    var types = parseTypeElements(json.response.docs);

    for (var i=0; i<types.length; i++){
      $('<label class="checkbox-inline"><input type="checkbox" onclick="filterType(this)
          ;" value="'+types[i]+'">'+types[i]+'</label>').appendTo('#typeCheckboxes');
    }
  });
}
```

Cabe destacar que hemos escogido radiobuttons para el filtro de las ciudades porque entendemos que interesa buscar en un lugar concreto. No se trata de una elección del tipo "comer en Coruña o en Lugo".

También se produce una primera conexión para obtener los restaurantes del plato buscado utilizando la función connectSolr(). Cada vez que se selecciona un filtro, se produce una actualización del área de resultados, llamando de nuevo a la función connectSolr(), que recibe un plato como parámetro.

```
function connectSolr(dish){ //Connecting the Solr endpoints
  var connect =solrAPI+dish+"&wt=json&indent=true&rows=50"

  if (city != null){ //City Filter
    if (city.length > 0){
      if (city != "All"){
        connect = connect+"&fq=city%3A*"+city;
      }
    }
  } else {
  }

  if (filterTypeRest.length>0){ //Type Restaurant Filter
    for (var i=0; i<filterTypeRest.length; i++){
      var filterRest;
      var res = filterTypeRest[i].split(" ");
      for (var j=0; j<res.length; j++){
```

```
18          if (j==0){
19            filterRest=res[j];
20          } else {
21            filterRest=filterRest+"+"+res[j];
22          }
23        }
24
25        if (i==0){
26          connect = connect+"&fq=typeRest%3A%22"+filterRest+"%22";
27        } else {
28          connect = connect+"+OR+typeRest%3A%22"+filterRest+"%22";
29        }
30      }
31    }
32
33    $('#inner').empty();  //Clear previous results
34
35    $.getJSON( connect, function(json){  //Getting new data
36        //console.log(json);
37    })
38      .done(function(json){
39        $('#loadingDiv').hide();
40
41        //Parsing restaurants to show the result in the page
42        console.log(json.response.docs)
43        var restaurants = parseRestaurants(json.response.docs);
44        if (restaurants.length > 0){
45          printRestaurants(restaurants);
46        }else{
47          $('<div class="alert alert-warning" role="alert"><center><strong>Lo sentimos!</
                strong> No disponemos de restaurantes que cumplan el criterio de busqueda :(
                </center></div>').appendTo('#inner');
48        }
49      })
50      .fail(function (){
51         $('#loadingDiv').hide();
52         $('<div class="alert alert-warning" role="alert"><center><strong>Lo sentimos!</
               strong> Se ha perdido la conexion con Solr </center></div>').appendTo('#inner'
               );
53      });
54  }
```

De esta función, cabe destacar varias cosas. La generación de las URL de endpoints se crean de la siguiente manera. Tenemos generadas, en variables globales, 3 urls que son:

```
1  var cities = 'http://localhost:8983/solr/collection1/select?q=*%3A*&wt=json&group=true&
       group.field=city&group.main=true&rows=500' //distintas cidades
2  var typeRestaurants = 'http://localhost:8983/solr/collection1/select?q=*%3A*&wt=json&
       group=true&group.field=typeRest&group.main=true&rows=500' //distintos tipos de
       restaurante
3  var solrAPI = "http://localhost:8983/solr/collection1/select?q=";
```

Las dos primeras se utilizan para recoger las distintas ciudades y tipos de restaurante que están almacenadas, haciendo uso del group.main y group.field. Se utilizan para generar los filtros que va a tener la página. La última de ellas es la base para generar consultas. A partir de la misma se generan las distintas consultas del usuario. Esto se hace de la siguiente manera:

```
1  var connect =solrAPI+dish+"&wt=json&indent=true&rows=50"
2    if (city != null){ //City Filter
3      if (city.length > 0){
4        if (city != "All"){
5          connect = connect+"&fq=city%3A*"+city;
6        }
7      }
8
9    if (filterTypeRest.length>0){ //Type Restaurant Filter
10     for (var i=0; i<filterTypeRest.length; i++){
11       var filterRest;
12       var res = filterTypeRest[i].split(" ");
13       for (var j=0; j<res.length; j++){
14         if (j==0){
15           filterRest=res[j];
```

```
        } else {
          filterRest=filterRest+"+"+res[j];
        }
      }

      if (i==0){
        connect = connect+"&fq=typeRest%3A%22"+filterRest+"%22";
      } else {
        connect = connect+"+OR+typeRest%3A%22"+filterRest+"%22";
      }
    }
  }
```

6 Conclusiones y Trabajo Futuro

Como hemos visto, hemos conseguido *darle la vuelta* a JustEat, ofreciendo una nueva idea de negocio. Ofrecemos más formas de búsqueda, como por platos o restaurantes, donde en JustEat no se permite, dejando solo buscar por código postal. Además, y donde creemos que puede estar parte de la ventaja competitiva, ofrecemos un buscador para que un grupo de personas sea capaz de encontrar un restaurante a su medida, e incluso permitimos poder hacer el pedido en la propia página de JustEat, consiguiendo así un ahorro en posibles gastos de envío por tener que llamar a dos sitios distintos. También nos puede servir como una ayuda a encontrar nuevos sitios para visitar.

Aún así, este trabajo es solo un punto de partida. Muchos de los restaurantes de la ciudad no se ofertan en esta página, por lo que tendríamos que buscar más páginas que nos ofrezcan este tipo de servicio. La elección sigue siendo buena, no encontramos otra página que nos dé tanta información como JustEat, pero aún así, depender de un único proveedor no es bueno. Una forma *sencilla* de atajar esto puede ser abrir una forma de que los dueños de un restaurante fueran capaces de mandar sus datos a la página.

En el ámbito económico, podríamos ofertar *restaurantes patrocinados*, donde con búsquedas específicas, aparezcan siempre arriba. Por ejemplo, un restaurante italiano puede querer patrocinar su local y cree que su punto fuerte son los raviolis, por lo que puede pagar una suscripción para que, en caso de que el usuario busque este plato, aparezca siempre en los primeros puestos.

7 Anexo I: Manual de usuario

Para hacer más cómoda la ejecución de la aplicación, hemos creado una serie de scripts que automatizarán gran parte del trabajo. Comenzaremos iniciando solr y hbase con el siguiente script, que utilizaremos cada vez que reiniciemos el ordenador:

```
#!/bin/bash

cd solr-4.8.1/example/
java -Dsolr.solr.home=solr -jar start.jar &
cd ../..
sleep 5

printf "Loading HBASE...\n"
hbase-0.98.19-hadoop2/bin/start-hbase.sh &
```

Para crawlear, utilizaremos el siguiente, donde decidiremos si inyectar nuevas urls si pulsamos *s*. Para obtener los restaurantes, ejecutaremos dos veces este script, inyectando solo la primera vez.

```
#!/bin/bash

cd apache-nutch-2.3.1/runtime/local/

```

```
read -n1 -r -p "Press s to inject..." key
if [ "$key" = 's' ]; then
  bin/nutch inject urls
fi
bin/nutch generate -topN 3
bin/nutch fetch -all
bin/nutch parse -all
bin/nutch updatedb -all
```

Una vez obtenido el contenido buscado, lo subiremos a solr con el siguiente comando:

```
#!/bin/bash

cd apache-nutch-2.3.1/runtime/local/
bin/nutch solrindex http://localhost:8983/solr/ -all
```

Como un script auxiliar, también hemos utilizado este para eliminar el contenido guardado en solr y en hbase.

```
#!/bin/bash

cd "solr-4.8.1/example/exampledocs"
java -Ddata=args -jar post.jar "<delete><query>*:*</query></delete>"

cd ../../..
rm -r zookeeper
mkdir zookeeper
rm -r hbasedata
mkdir hbasedata
```

Para visualizar los datos, abriremos un navegador e iremos a http://localhost:8983/solr/, donde entraremos en el panel de control. Si después elegimos *collection1* en *core selection*, podemos elegir la opción *Query* para hacer consultas sobre los datos.