



Escuela
Politécnica
Superior

Aplicación para el análisis de la reutilización de datos abiertos de investigación



Máster Universitario en Tecnologías
de la Informática

Trabajo Fin de Máster

Autor:

Juan Esquedo Roig

Tutor/es:

Jose Norberto Mazón

Septiembre 2017



Universitat d'Alacant
Universidad de Alicante

Aplicación para el análisis de la reutilización de datos abiertos de investigación

Juan Esquerdo Roig

Septiembre 2017

1. Introducción

Hoy en día tenemos diversas fuentes de datos, de hecho toda empresa o entidad pública (ente en general) es una fuente constante que genera una cantidad ingente de datos diariamente, estos datos a veces son recogidos por las empresas u otras entidades. Y y, en algunos de los casos, publicados en repositorios de datos abiertos para la consulta de cualquier persona.

Esto cobra especial relevancia cuando los datos en cuestión proceden de resultados de investigación, ya que su apertura hace que la comunidad científica pueda reutilizarlos con el fin de avanzar más rápidamente en las investigaciones y publicar trabajos relevantes.

La relevancia de los trabajos de investigación se mide usando diversas medidas bibliométricas [2] como el número de citas, el factor de impacto, el índice h, etc. Además, últimamente han surgido otras medidas alternativas (denominadas *altmetrics*) [1] que abogan por medir el impacto que tienen los trabajos de investigación en las redes sociales.

Sin embargo, esta medición de impacto, tan usada y aceptada para los trabajos científicos (publicaciones en revista, por ejemplo), no ha sido explorada para los conjuntos de datos abiertos de investigación. Precisamente, en este trabajo se plantea el desarrollo de una aplicación que permita (i) obtener los datos abiertos que se publican en repositorios de investigación (ii) calcular diversas medidas de relevancia a partir de ellos y, finalmente, (iii) proporcionar mecanismos para analizar dichos datos.

2. Estado del arte

Los repositorios de datos como su nombre indica son unidades donde se almacenan colecciones de archivos digitales con carácter científicos. Podremos encontrar diferentes repositorios, encontrando repositorios temáticos (sobre una materia en concreto como puede ser la salud). O genéricos, gestionados por una institución en sus labores de investigación.

La existencia de tantos repositorios y la generación de tantos archivos digitales de carácter científico ha obligado a la creación de una herramienta para poder indexar o realizar una búsqueda de los archivos[1].

Los repositorios tienden a seguir el patrón de una primera fase inicial, donde tendremos un buscador por el que podremos buscar por una temática concreta (como por ejemplo IOT), por el autor o incluso el DOI.



Figura 1: Ejemplo de buscador en un repositorio

Una vez realizada la búsqueda, pasaremos a una segunda fase, donde podremos elegir el artículo que queramos o realizar una búsqueda más especializada, filtrando por los parámetros que nos interesen.

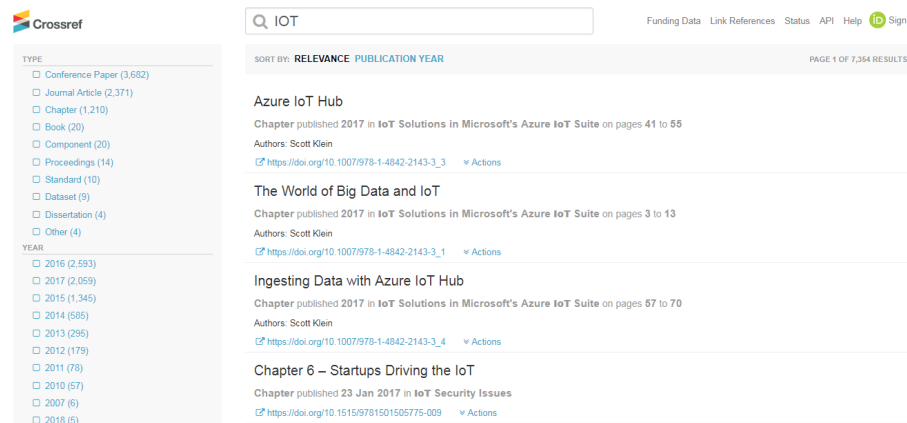


Figura 2: Ejemplo de resultados y opciones de filtrado en un repositorio

Y por último una tercera fase donde elegido el conjunto de datos (o dataset), nos llevara a la página donde podremos descargar el dataset y la visualización de algunos parámetros generales como las fechas, sitio, etc.



Figura 3: Ejemplo de dataset

Cabe destacar que estos repositorios tienen un API que permite obtener de una manera dinámica los datos de interés, en lugar de usar manualmente la interfaz comentada.

Tenemos repositorios como 'r3data' que nos permite ver la visualización de diferentes repositorios de un tema concreto. Pero con este repositorio no conseguimos saber si el repositorio concreto tiene API y la obtención de los datos que nos interesasen del Dataset.

```
▼<list>
  ▼<repository>
    <id>r3d10000002</id>
    <name>Access to Archival Databases</name>
    <link href="/api/v1/repository/r3d10000002" rel="self"/>
  </repository>
  ▼<repository>
    <id>r3d10000004</id>
    <name>Datenbank Gesprochenes Deutsch</name>
    <link href="/api/v1/repository/r3d10000004" rel="self"/>
  </repository>
```

Figura 4: r3data

Tenemos también repositorios como 'Zenodo' que se descartó automáticamente, ya que requiere de autenticación y la instalación de diversos tokens, ya que buscábamos datos abiertos en su totalidad.

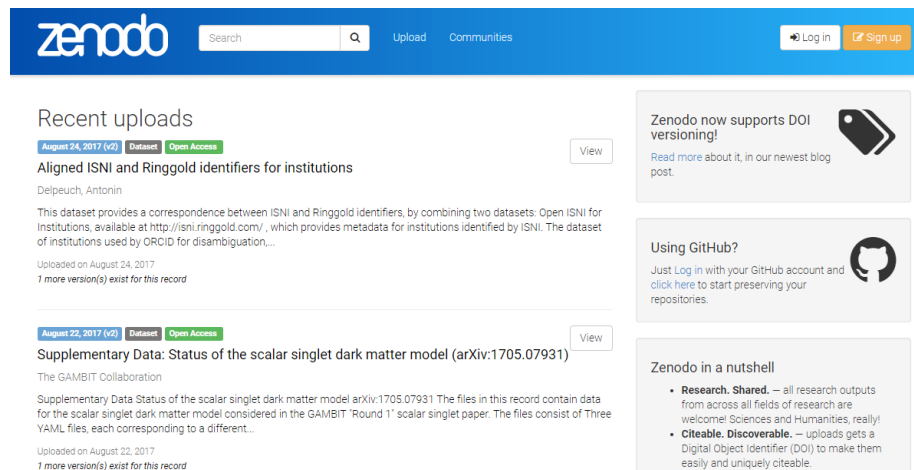


Figura 5: Zenodo

Otra opción es ‘EuropePMC’ donde si podemos obtener los datos, pero el API no permite la obtención de muchos campos que podamos utilizar, y además algunos de ellos, con la llamada al API no se obtienen correctamente.

```

▼<responseWrapper>
  <version>5.2.1</version>
  <hitCount>292347</hitCount>
  <nextCursorMark>AoIIQIqZ/SgzNjkyMDU5Mw==</nextCursorMark>
  ▼<request>
    <query>doi</query>
    <resultType>lite</resultType>
    <synonym>>false</synonym>
    <cursorMark>*</cursorMark>
    <pageSize>25</pageSize>
    <sort/>
  </request>
  ▼<resultList>
    ▼<result>
      <id>28841008</id>
      <source>MED</source>
      <pmid>28841008</pmid>
      <doi>10.1021/jacs.7b05490</doi>
      ▼<title>
        Size Fractionation of Graphene Oxide Nanosheets via Controlled Directional Freezing.
      </title>
      ▼<authorString>
        Geng H, Yao B, Zhou J, Liu K, Bai G, Li W, Song Y, Shi G, Doi M, Wang J.
      </authorString>
      <journalTitle>J Am Chem Soc</journalTitle>
      <pubYear>2017</pubYear>
      <journalIssn>0002-7863; 1520-5126;</journalIssn>
      <pubType>journal article</pubType>
      <isOpenAccess>N</isOpenAccess>
      <inEPMC>N</inEPMC>
      <inPMC>N</inPMC>
      <hasPDF>N</hasPDF>
      <hasBook>N</hasBook>
      <citedByCount>0</citedByCount>
      <hasReferences>N</hasReferences>
      <hasTextMinedTerms>N</hasTextMinedTerms>
      <hasDbCrossReferences>N</hasDbCrossReferences>
      <hasLabsLinks>N</hasLabsLinks>
      <hasTMAccessionNumbers>N</hasTMAccessionNumbers>
    </result>
  </resultList>
</responseWrapper>

```

Figura 6: EuropePMC

Debemos destacar la herramienta PlumX; PlumX es una herramienta para medir el impacto de repositorios, esta herramienta nos permite conseguir más valor de los repositorios, ya que aplica diferentes métricas (Número de veces que se ha visto el abstract, número de veces que se ha abierto el artículo, etc.) Consiguiendo de un solo vistazo, la visualización del artículo y sus métricas, siendo esta la finalidad de nuestro trabajo.

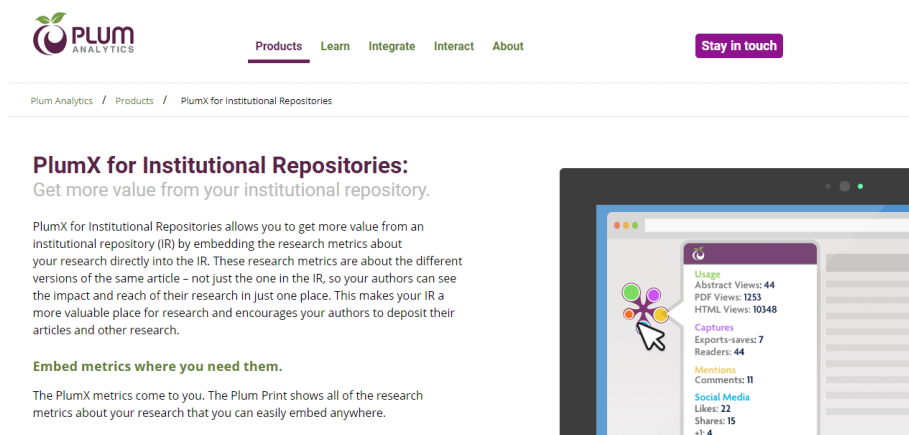


Figura 7: PlumX

El gran inconveniente de esta herramienta es que debemos tener una base de conocimiento con diferentes artículos y además comprender y configurar su aplicación en nuestra base de conocimiento. Además, una diferencia con nuestra propuesta es que PLumX se centra en artículos de investigación, mientras que la propuesta de este trabajo se centra en conjuntos de datos (abiertos) de investigación.

3. Análisis

3.1. Metodología

El desarrollo de un proyecto de esta envergadura engloba ciertas limitaciones en la planificación como definición de requisitos débiles o poco específicos, largos periodos de construcción del proyecto que dificultan la planificación temporal, la necesidad de mucho aprendizaje y la incertidumbre de si ciertas partes van a ser viables o no al inicio. Por todo esto se ha decidido usar como metodología de trabajo una metodología ágil ya que se ajusta con la mayoría de restricciones comentadas.

Dentro de las metodologías ágiles hay varias opciones disponibles como Scrum, xP, ASD, FDD y muchas otras, pero se ha seleccionado FDD (Feature-Driven Development), debido a que el proyecto dispone de muchas funcionalidades distintas y se pueden ir añadiendo al sistema como features (Características de un elemento software) en cada iteración.

Hay que tener en cuenta que aunque se haya usado una metodología ágil, estas suelen estar preparadas para trabajo en equipo, y dado que este trabajo se ha desarrollado de manera individual no se han seguido todas las características de la metodología al pie de la letra.

La metodología es iterativa e incremental y se compone de 5 fases, 3 de ellas iniciales en las que se crea una visión general del proyecto y se obtiene una lista

de funcionalidades ordenadas por prioridad y las dos últimas fases en las que se itera por cada una de las funcionalidades de la lista y en las que se diseña y se implementa esa funcionalidad.

Las reuniones con el cliente (en este caso se entiende como ‘cliente’ a el tutor del TFM) se han realizado en periodos de 1-2 semanas en los que se mostraban los resultados de la funcionalidad programada o avances conseguidos para la iteración, y se recogía el feedback para modificar si fuera necesario y posteriormente se iniciaba la siguiente iteración hasta que el producto estuviera terminado.

Feature Driven Development (FDD)



Figura 8: Feature Driven Development

3.2. Requisitos funcionales

Los requisitos funcionales nos definen las funciones del sistema de software o de sus componentes, donde cada función es un conjunto de entradas, comportamientos y salidas que define una funcionalidad específica que el software debe cumplir. A continuación se listan los requisitos funcionales de este proyecto:

- Debe existir una aplicación web sencilla e intuitiva para el control de todo

el sistema.

- En nuestra aplicación deberemos poder elegir entre una página de inicio, la recolección de información y la muestra de datos mediante gráficos.
- La aplicación debe recoger todos los datasets del repositorio elegido.
- La aplicación deberá conectarse a Twitter para obtener las citas de un dataset concreto.
- La aplicación deberá mostrar comparativas mediante gráficos de los diferentes resultados obtenidos.
- La aplicación funcionará en cualquier dispositivo, ya sea ordenador o dispositivo móvil de una forma fluida.

3.3. Herramientas y tecnologías

En esta sección se muestra las herramientas y tecnologías que se han empleado para la creación del proyecto, así como la justificación de las elecciones.

3.3.1. Lenguaje de programación: Java

Si bien es cierto que la mayoría de páginas web en el mercado están programados en otros lenguajes, se ha de tener en cuenta que Java es un lenguaje flexible y muy útil con la finalidad de realizar un desarrollo software, por ejemplo, para un trabajo de fin de máster como el que nos ocupa. De hecho las ventajas de velocidad y rendimiento que se obtendrían con otro lenguaje no compensan la inversión sobre todo temporal, ya que este proyecto debe entregarse en tiempo finito y se realiza por una única persona.

Finalmente se ha seleccionado Java como lenguaje a usar en el proyecto, debido a la rapidez con la que permite la creación de código y que va a permitir centrarme más en los objetivos del proyecto que en batallas con el código, teniendo en cuenta además que la eficiencia va a ser menor que una compilación en otros lenguajes.

3.3.2. Gestión y construcción de proyectos: Maven

Maven es una herramienta muy común y muy utilizada para este propósito. Con esto obtenemos una estructura del proyecto, para que cualquier desarrollador, sepa encontrar lo que busca, o donde tiene que poner cada cosa.

Maven viene con objetivos predefinidos para realizar ciertas tareas, y además gestiona la inclusión de dependencias de otros módulos y componentes externos, como librerías útiles para nuestro proyecto.

3.3.3. Framework: Spring Boot

Este framework para el desarrollo de aplicaciones. Proporciona una capa de abstracción que permite usar ‘etiquetas’ que nos permiten realizar actividades complejas de una manera sencilla, no obligándonos a implementar muchas de las funcionalidades (como por ejemplo algunos de los accesos a la base de datos) para el desarrollo del proyecto.

Otra de las grandes ventajas por las que se escogió este framework es porque su curva de aprendizaje es bastante baja, en poco tiempo se puede tener un dominio bastante avanzado de Spring, permitiendo así una rápida ejecución del proyecto; una de las cosas que permite esto es la gran cantidad de tutoriales y proyectos de ejemplo que podemos encontrar en la propia página. Además cuenta con una gran comunidad detrás donde la mayoría de las dudas que pueden surgir a los programadores más novatos están resueltas. Sumados todos estos factores obligan a cualquier programador novato-senior a ser un framework a tener muy en cuenta.

3.3.4. Base de datos: Mysql

Debido a la gran cantidad de datos que vamos a manejar, se ha optado por una base de datos multidimensional, para conseguir unos resultados óptimos en consultas a la base de datos. Para ello hemos utilizado el esquema en estrella, que nos permite implementar la funcionalidad de una base de datos multidimensional en una base de datos relacional.

Este esquema es ideal por su sencillez y velocidad para los análisis multidimensionales, obteniendo el mejor rendimiento y velocidad ya que podemos indexar las dimensiones de una forma individualizada.

3.3.5. Testing: JUnit y pruebas manuales

Por último en lo que respecta a nuestro software, hemos realizado una serie de test unitarios, para comprobar el correcto funcionamiento de nuestras clases Java.

Además también hemos comprobado manualmente, que los resultados coincidían, ya sea mirando el número de registros guardados en la base de datos, o comprobando que las métricas para algunos de los artículos se habían calculado correctamente.

4. Cuerpo del trabajo

4.1. Desarrollo del proyecto

En este apartado queremos demostrar la arquitectura de nuestro proyecto. Para ello lo haremos en forma de demostración con algunas capturas de pantalla de lo que ha sido el proyecto. Cabe destacar que para no extendernos mucho, vamos a hacer una demostración resumida de las partes más importantes.

4.2. Base de datos

Se ha diseñado una base de datos multidimensional [3] que permita almacenar dos medidas concretas de cada uno de los conjuntos de datos según su autor y fecha de publicación: una medida es tradicional y consiste en el número de citas recibidas por el conjunto de datos y otra medida es altmetrics y consiste en el número de citas recibidas mediante Twitter. A continuación se define el diseño multidimensional realizado.

Tabla de hechos:

- Citas

Tablas de dimensiones:

- Dataset
- Autor
- Fecha

Métricas:

- CitasCrossref
- CitasTwitter

Jerarquías:

- Elegidas en base a los parámetros que nos facilitaba el API

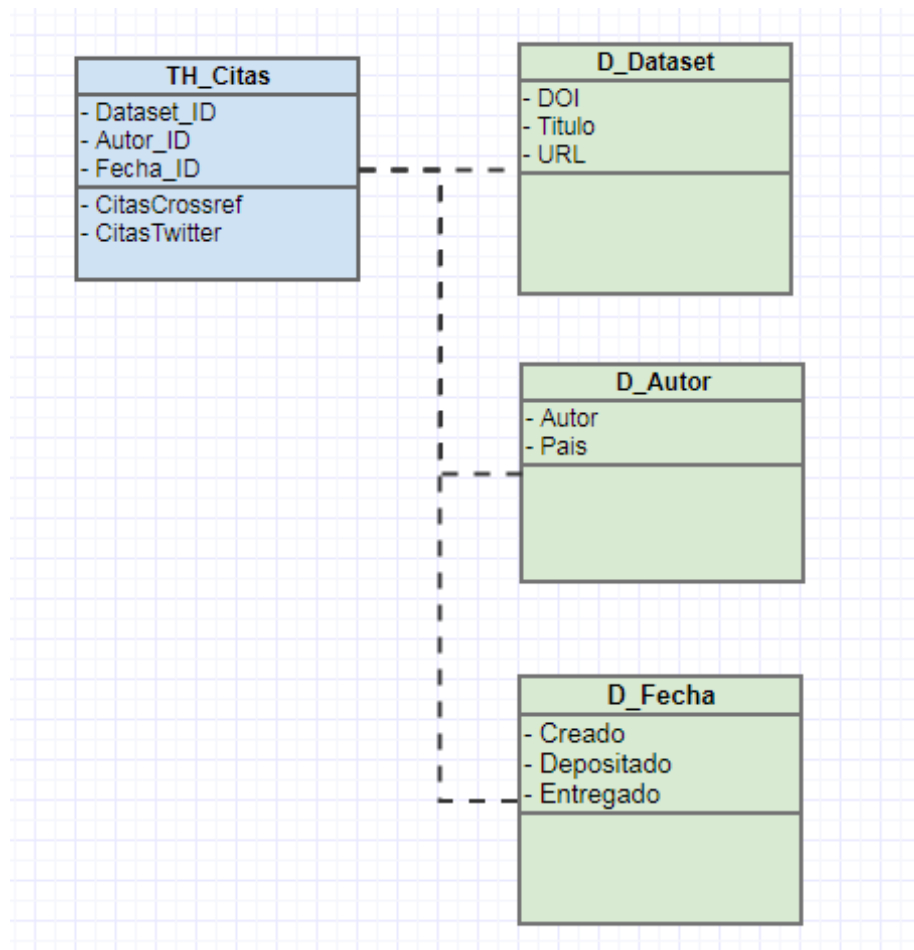


Figura 9: Esquema base de datos multidimensional

4.3. Arquitectura

La arquitectura de forma esquematizada de nuestro proyecto es la siguiente:



Figura 10: Esquema de la arquitectura de nuestra aplicación

Ahora desarrollaremos más extensamente las partes de las que consta nuestro proyecto.

4.4. Recolección de datos

La tarea inicial que realizara nuestra aplicación es la de poblar nuestra base de datos multidimensional. La primera tarea por tanto a realizar es la recolección de todos los datasets de la API de Crossref.

El API de Crossref, es una aplicación REST, por lo que haciendo una llamada HTTP, con los parámetros de entrada correctos, deberá devolvernos un objeto ya sea XML o JSON (en este caso JSON), con los datos que nos interesen.

El API de Crossref cuenta actualmente con 1.500.000 de datasets aproximadamente y cada día subiendo, para la obtención de estos datasets, utilizamos la siguiente llamada:

<https://api.crossref.org/types/dataset/works?rows=1000&offset=1>

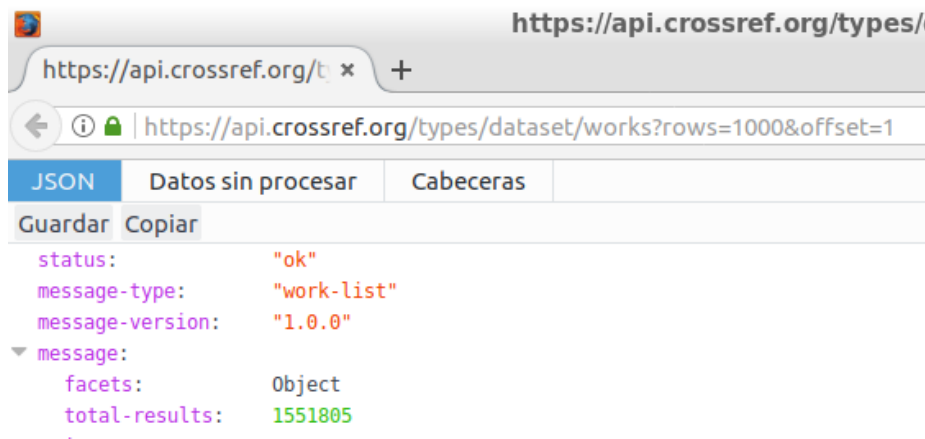


Figura 11: Número de datasets en CrossRef

En un principio, la llamada debería devolvernos todos los datasets almacenados, pero por limitaciones del API CrossRef, como máximo nos devolverá 1000 datasets. Tendremos que por tanto, utilizar un cursor y conseguir todos los datasets de mil en mil. Para ello iremos incrementando el valor del parámetro 'offset' y realizando unas 1500 llamadas aproximadamente para la obtención de todos los dataset.

Para cada llamada recibiremos por tanto un objeto JSON que a su vez, tendrá mil 'hijos' (aparte de cabeceras y datos que no nos interesan para nuestra labor), estos hijos iremos 'despiezando' para obtener los valores de los campos que nos interesan con la librería ObjectMapper, como mostramos brevemente en la siguiente imagen:

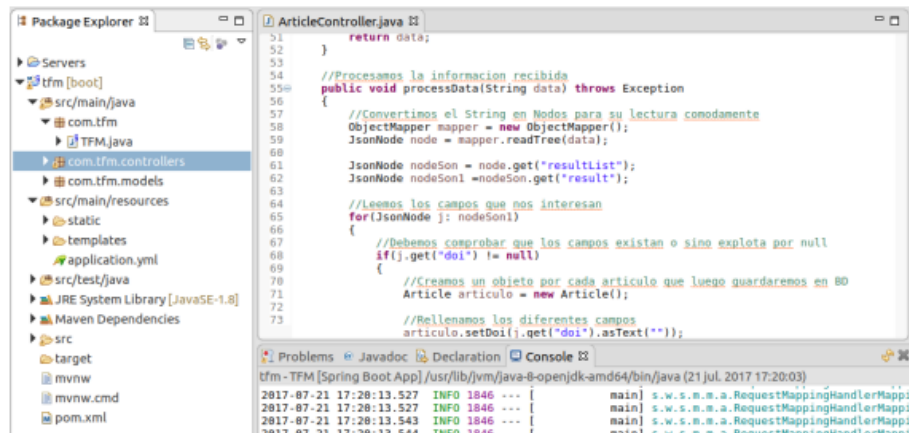


Figura 12: 'Despiece' del objeto JSON recibido

Una vez tenemos todos nuestros datos en nuestras variables, usaremos nuestra variable ‘DOI’ donde tendremos almacenado el ID del documento, para realizar una búsqueda en el API de Twitter.

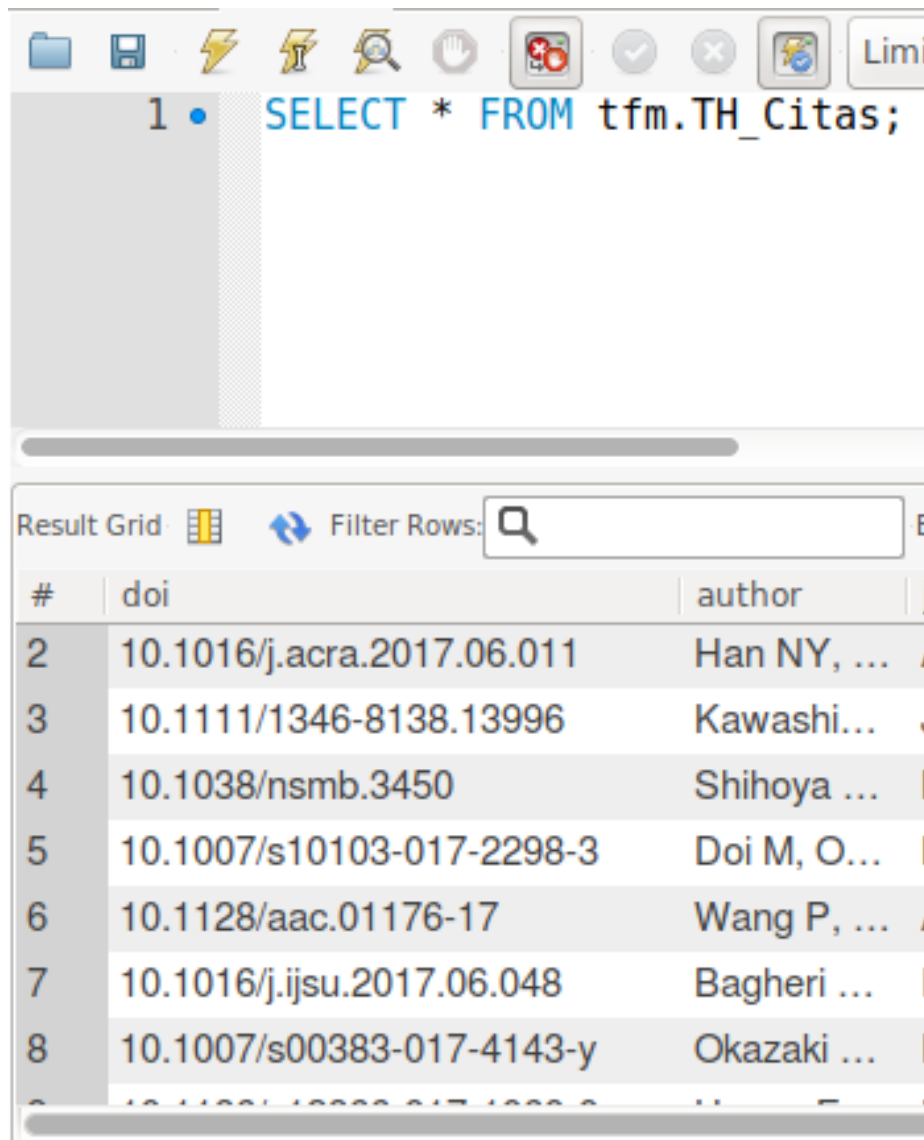
Con el API de Twitter simplemente tendremos que realizar una conexión mediante un usuario registrado en Twitter y realizar la búsqueda del DOI correspondiente (Y así para cada uno de los DOI que hayamos obtenido). Esto nos devolverá un Array de objetos, del que solo tendremos que realizar un count, para así saber el número de elementos que tiene, y por tanto saber el número de veces que ha sido nombrado este DOI concreto.

Una vez tenemos todas nuestras variables con los datos obtenidos, simplemente guardamos en nuestra base de datos, mediante el uso de la librería hibernate de Java, en los atributos correspondientes de cada tabla.

4.5. Visualización

No hemos finalizado aun todas las opciones de las que dispondrá un usuario, pero nos hemos esforzado en realizar una aplicación intuitiva, que cualquier usuario, pueda manejar sin ningún tipo de problema ni ninguna complejidad.

Una vez tenemos creada nuestra base de conocimiento, con todos los documentos guardados en la base de datos, ahora simplemente nos encargaremos de mostrar todos los datos recogidos.



The screenshot shows a database query interface. At the top, there is a toolbar with various icons. Below the toolbar, a SQL query is entered in a text area: `1 • SELECT * FROM tfm.TH_Citas;`. Below the query area, there is a section labeled "Result Grid" with a search filter input. The results are displayed in a table with three columns: "#", "doi", and "author". The table contains eight rows of data, with the first row being the header and the subsequent rows containing specific citation data.

#	doi	author
2	10.1016/j.acra.2017.06.011	Han NY, ...
3	10.1111/1346-8138.13996	Kawashi...
4	10.1038/nsmb.3450	Shihoya ...
5	10.1007/s10103-017-2298-3	Doi M, O...
6	10.1128/aac.01176-17	Wang P, ...
7	10.1016/j.ijsu.2017.06.048	Bagheri ...
8	10.1007/s00383-017-4143-y	Okazaki ...

Figura 13: Registros en la tabla dimensiones Citas

Para ello primero que todo hemos necesitado la librería Thymeleaf, con la que conseguimos transportar todos los datos que tenemos en nuestro servidor (Realizaremos una consulta a nuestra base de datos multidimensional, y luego simplemente guardaremos en variables, que luego en nuestro frontend serán etiquetas con los valores obtenidos) a nuestro cliente.

Una vez tenemos las etiquetas con los valores que queramos, simplemente utilizaremos las librerías de google (en JavaScript), para el montaje de gráficas,

donde simplemente tendremos que especificar los valores que ya hemos obtenido de nuestra base de datos, y las librerías se encargaran de montar las gráficas.

Hemos por tanto dividido nuestra página en tres partes claramente diferenciadas:

La primera parte (parte de arriba) sería un panel de usuario, donde tendría unas opciones generales para el usuario. La segunda parte (parte de la izquierda), sería una parte relacionada con las diferentes opciones que ofrecería nuestra aplicación. Y una tercera parte, donde el usuario vería la información que quiere, dándole más importancia a esta parte, y siendo donde se mostrara todo lo que está ocurriendo.

En la parte izquierda, al pulsar sobre la opción ‘Obtener datos’ guardaremos todos los datasets y el número de menciones en Twitter. Luego al pulsar en mostrar gráficos, podremos obtener la gráfica y datos de ese DOI concreto.

5. Resultados

En este apartado queremos demostrar los resultados de nuestro proyecto. Para ello lo haremos en forma de demostración con algunas capturas de pantalla de los hechos contrastados y lo que verá finalmente el usuario. Cabe destacar que para no extendernos mucho, vamos a hacer una demostración resumida de las partes más importantes.

Buscamos uno de los datasets que menos referencias ha tenido:

```
DOI: "10.3886/ICPSR24588", "type": "dataset", "created": {"date-parts": [[2009, 4, 20]], "date-time": "2009-04-20T12:40:240343000"}, "source": "Crossref", "is-referenced-by-count": 1, "title": ["ABC News/Washington Post Poll, May 2007"], "prefix": "10.3886", "author": [{"name": "ABC News/Washington Post", "id": "10.3886/ICPSR24588"}]
```

Figura 14: Búsqueda en CrossRef

Ese mismo DOI lo buscamos en Twitter:



Figura 15: Búsqueda en Twitter

Ahora comprobamos que los datos que muestra nuestra aplicación son esos, teniendo una sola cita en la API CrossRef.



Figura 16: Resultado en nuestra API

Como podemos comprobar, la aplicación funciona correctamente, pero ahora elegiremos el que más citas tiene en el API CrossRef:

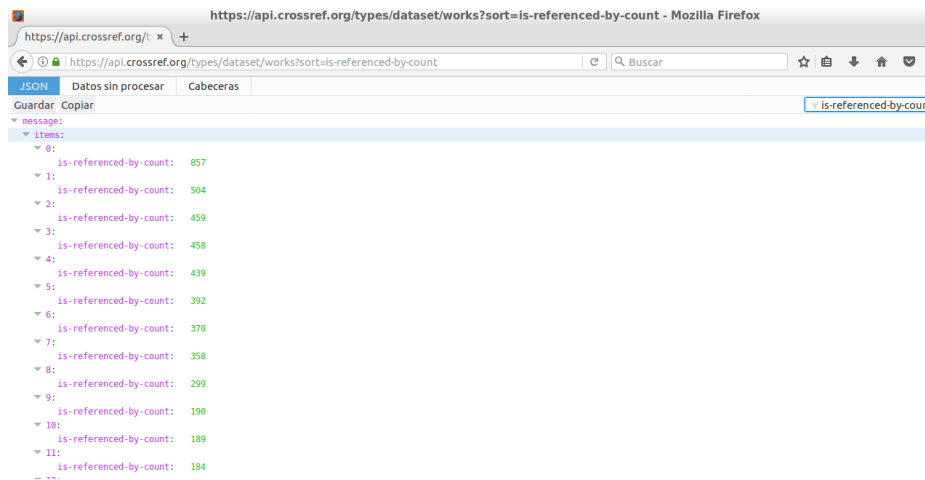


Figura 17: Búsqueda en CrossRef

Como podemos comprobar, el dataset que más citas tiene, tiene 857 citas actualmente, vamos a buscar ahora este dataset en Twitter:

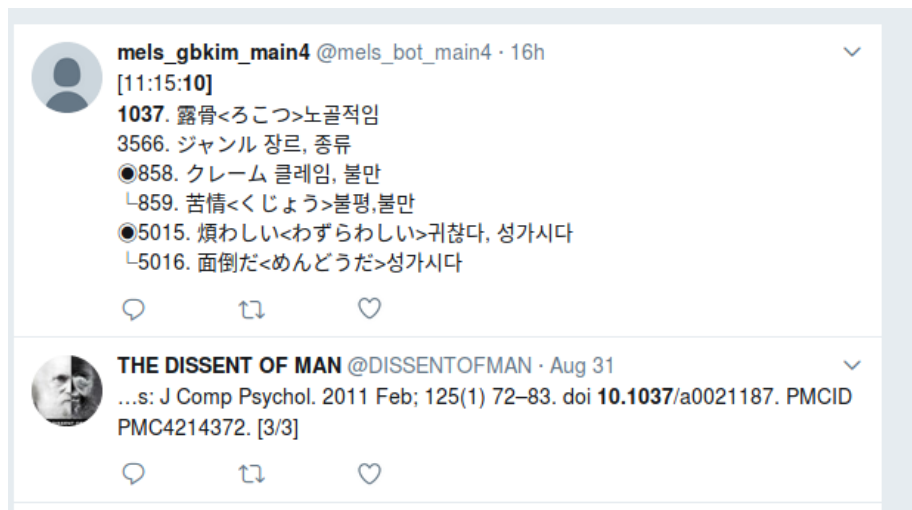


Figura 18: Búsqueda en Twitter

Como podemos comprobar este dataset ha sido nombrado dos veces en Twitter, y esto es lo que nos muestra nuestra aplicación:

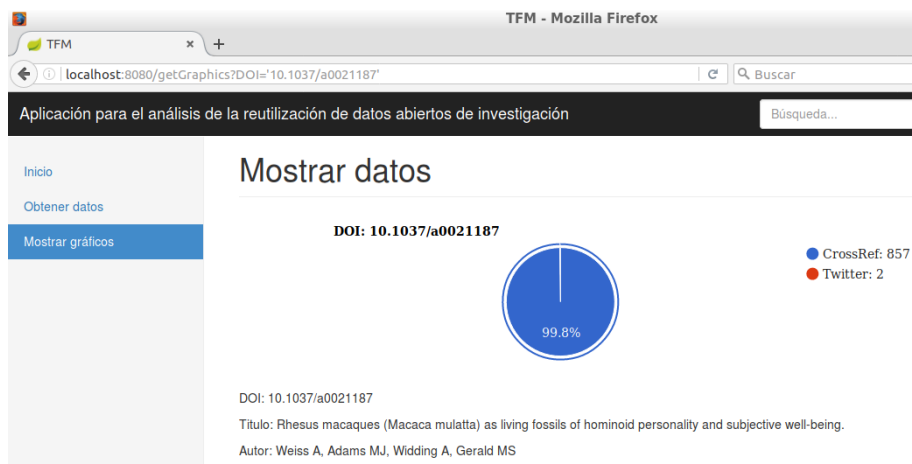


Figura 19: Búsqueda en Twitter

6. Conclusiones

Es complicado determinar la relevancia bibliométrica de un trabajo científico como una publicación, ya que deben extraerse los datos necesarios así como determinar la mejor métrica o conjunto de métricas. Esta problemática se acentúa

cuando hablamos de datos abiertos ya que no existen soluciones que permitan obtener cómo se citan los diversos conjuntos de datos.

En este trabajo se ha planteado el desarrollo de una aplicación que permita obtener información como el impacto de los datasets, tanto en el propio repositorio abierto, como en las redes sociales. Podemos deducir que datasets son los más relevantes o más utilizados, pudiendo por ejemplo realizar una decisión más acertada en lo que respecta a la calidad de un dataset.

Realizando este tipo de estudios demostramos que podemos obtener información valiosa, que posteriormente procesaremos y obtener información que nos será útil (el impacto del dataset) para creer en la veracidad de los datos.

Aunque ahora mismo estos indicadores solo se utilizan como simples experimentos para algunos de los artículos de índole científica, debido mayormente a validez de las fuentes y problemas técnicos, las nuevas métricas intentan paliar las carencias que nos proporcionan las mediciones actuales.

Nuestra aplicación por tanto intenta recoger los datos de una fuente fiable, e intentar paliar estas carencias de validez de fuentes y problemas técnicos; aunque a la única conclusión a la que podemos llegar con este tipo de métricas es que han tenido un gran impacto inicial en el ámbito científico y que recorrerán un largo camino en la comunidad científica.

Referencias

- [1] Todd J. Vision Christine Mayo. *The Location of the Citation: Changing Practices in How Publications Cite Original Data in the Dryad Digital Repository*. Dryad Digital Repository, 2016.
- [2] Evaristo Jiménez Daniel Torres, Álvaro Cabezas. *Altmetrics: nuevos indicadores para la comunicación científica en la Web 2.0*. Comunicar, 2013.
- [3] & Ross M. Kimball, R. *The data warehouse toolkit: the complete guide to dimensional modeling*. John Wiley & Sons, 2011.