UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

DIRECTORIO DE INVESTIGADORES EN CIENCIAS DE LA COMPUTACIÓN

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN COMPUTACIÓN

FELIPE IGNACIO RUBILAR ROJAS

PROFESORES GUÍA
SERGIO OCHOA DELORENZI
DANIEL PEROVICH GEROSA

MIEMBROS DE LA COMISIÓN:
FEDERICO OLMEDO BERÓN
EDGARD PINEDA LEONE

SANTIAGO DE CHILE
2018

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE INGENIERO CIVIL EN COMPUTACIÓN
POR: FELIPE IGNACIO RUBILAR ROJAS
FECHA: 2018
PROF. GUÍA: SERGIO OCHOA DELORENZI
DANIEL PEROVICH GEROSA

DIRECTORIO DE INVESTIGADORES EN CIENCIAS DE LA COMPUTACIÓN

En los últimos años, varias editoriales, empresas y algunas instituciones educativas se han dedicado a recopilar información sobre el estado de la comunidad de investigadores en Ciencias de la Computación, y mantener esa información en repositorios propios. Estas iniciativas están enfocadas principalmente en el procesamiento de los artículos científicos y en la generación de perfiles públicos de investigadores, más que en apoyar la generación de contactos o instancias de colaboración entre los miembros de la comunidad, o bien dar soporte a las necesidades habituales de los investigadores o de las instituciones a las que ellos pertenecen. Si uno desea acceder a uno de estos servicios, y está disponible, entonces debe pagar por ello. Hoy en día este tipo de iniciativas son manejadas como un negocio, más que como un servicio a la comunidad.

Con el objetivo de subsanar esta situación, en este trabajo de memoria se desarrolló una prueba de concepto que permitió determinar la factibilidad técnica, operativa y económica de implementar una iniciativa de carácter gratuita, que brinde un servicio a la comunidad utilizando datos de diversas fuentes de información científica. Particularmente se diseñó e implementó un sistema de recopilación automática de información de investigadores en ciencias de la computación y de sus publicaciones. Este sistema recupera, valida e integra información de distintas fuentes, y la almacena en un repositorio que es consultable a través de una aplicación Web. Si bien los servicios provistos por la aplicación no son suficientes para lograr el objetivo final deseado y solucionar completamente el problema, éstos permitirán validar el sistema como una solución factible y de interés a los usuarios, con potencial de evolución para satisfacer una mayor variedad de tipos de consulta.

Los resultados de las pruebas realizadas con algunos investigadores del área muestran que el sistema es de su interés, corroborando la usabilidad y utilidad de la aplicación de consulta de datos. Además, los evaluadores destacaron el potencial que la información recopilada tiene a la hora de resolver consultas que surgen durante el desarrollo de sus actividades. Si bien existen preocupaciones relacionadas a la escalabilidad del sistema, se proponen posibles soluciones a explorar en una próxima iteración, por lo que se espera que el sistema implementado sea un primer paso hacia la elaboración de una herramienta que pueda ser utilizada efectivamente por la comunidad, con el fin de disminuir el tiempo y el trabajo manual necesario en el desarrollo de las actividades que se busca apoyar.

Tabla de Contenido

Índice de Ilustraciones

Capítulo 1

Introducción

Gracias al fenómeno de la globalización, cada vez es más común ver a investigadores de distintos lugares del mundo reunidos en conferencias, compartiendo conocimientos, comparando resultados y generando redes de colaboración. De estas instancias de colaboración nacen múltiples publicaciones y proyectos, que permiten determinar las áreas de investigación más activas o los grupos de investigación que existen alrededor del mundo. Esta es información útil a la hora de promover el desarrollo y la cooperación en el ámbito de las Ciencias de la Computación e Informática.

Actualmente, el enfoque de los sitios que recopilan esta información (como Google Scholar [9], ISI Web of Science [18] o Mendeley [11]) está centrado principalmente en las publicaciones, más que en los investigadores que trabajan en ellas. Esto dificulta la promoción de la cooperación entre miembros de la comunidad (por ej. para realizar proyectos conjuntos, asistir a defensas de tesis, dictar cursos cortos, dar charlas, etc.), puesto que no existe una forma fácil de identificar un público objetivo o encontrar posibles interesados a partir de áreas de la computación, temáticas abordadas, ubicación geográfica de los investigadores, colaboraciones previas, etc. Adicionalmente, gran parte de esta información está detrás de paywalls que restringen aún más el acceso a ella.

Existen servicios prestados por otras instituciones internacionales (como IEEE [10], ACM [1], Springer [16]) en donde los interesados en difundir información simplemente envían los detalles sobre la información que se desea difundir (por ej. el llamado a presentación de trabajos a una conferencia), dejando en manos de la institución el determinar a quiénes distribuirlas y mediante qué medio. Esto implica un costo para quienes desean difundir dicha información, el cual puede hacerse difícil de justificar para eventos/iniciativas pequeñas o de carácter regional (por ejemplo, latinoamericanas).

Para intentar mejorar esta situación, este trabajo de memoria implementó un sistema de recolección automática de información sobre investigadores del área de las Ciencias de la Computación. Dicho sistema permite explorar las redes de colaboración existentes en torno a temas de investigación o a autores en particular, y será expuesto en forma gratuita a través de un sitio Web, donde los usuarios podrán realizar búsquedas y consultas sobre la información recopilada.

1.1. Justificación de la memoria

Este trabajo de memoria involucró dos componentes (o partes) importantes. La primera parte del trabajo realizado contempló la integración de múltiples fuentes de información con el fin de obtener un repositorio único de artículos y autores. Esto implicó el desarrollo de funcionalidades para recopilar los datos desde cada fuente, y almacenarla en un modelo de datos común. Este sistema además es capaz de:

- Soportar la recuperación de información almacenada en múltiples esquemas y múltiples formatos.

- Evaluar criterios de igualdad entre entidades (personas, publicaciones, etc.) de distintas fuentes, evitando problemas de alcance de nombre, identificadores, etc.

- Tomar decisiones al encontrar información conflictiva en distintas fuentes de datos.

Dado que el sistema se enfoca en representar y mantener vínculos potenciales y efectivos entre investigadores (más que en la información recolectada sobre cada uno de ellos), este trabajo contempla el uso bases de datos no-relacionales, específicamente bases de datos de grafos. Esta decisión se basó en las optimizaciones que éstas BD poseen al realizar consultas de búsqueda de caminos y relaciones entre entidades, lo que les otorga una ventaja comparativa en términos de desempeño con respecto a las BD relacionales.

Una segunda parte de la memoria consistió en la implementación de una aplicación Web (de front-end) a través de la cual se puede consultar la información recopilada y revisar los resultados de manera visual. Este front-end es capaz de realizar consultas sobre la base de datos y de mostrar los detalles de cada entidad. Además, ofrece funcionalidades para encontrar entidades relacionadas con un cierto criterio de búsqueda (por ejemplo, considerando temas de investigación, colaboraciones previas, etc.).

Algunas de las actividades típicas de un ambiente académico que podrían ser apoyadas con este sistema son las siguientes:

- Identificar posibles evaluadores para un artículo o un proyecto científico, considerando la distancia de coautoría entre los autores/proponentes y los posibles evaluadores.

- Identificar investigadores que trabajan en áreas específicas, por ejemplo para la presentación de proyectos conjuntos, para invitarlos a formar parte de un comité evaluador de tesis (de magíster o doctorado), o un comité de programa de una conferencia, entre otros.

- Identificar la productividad científica y/o el nivel de relevancia de un cierto académico, en un período de tiempo definido.

1.2. Objetivos de la memoria

El objetivo principal de este trabajo de memoria es diseñar e implementar un directorio de investigadores en Ciencias de la Computación, que se alimente automáticamente de la información disponible en diversas fuentes de datos. El sistema debe además relacionar y clasificar esta información (por áreas de investigación, temáticas que abordan, etc.), y dejarla a disposición de la comunidad para ser consultada y utilizada con fines académicos o científicos. Este sistema representa una prueba de concepto, a través de la cual se busca determinar la factibilidad técnica y operativa de desarrollar una solución definitiva para resolver los problemas planteados.

Para alcanzar el objetivo general se definieron los siguientes objetivos específicos:

- **Identificar las fuentes de información a utilizar**, teniendo en cuenta diversos puntos como el acceso a dichas fuentes (legalidad, *paywalls*, etc.), su veracidad, el grado de actualización que tienen, y la factibilidad de ser integradas en un modelo de datos común.

- **Definir e implementar el directorio de investigadores**, permitiendo la recolección automática de información desde las múltiples fuentes escogidas, integrando dicha información en un repositorio común. Esto también contempla la resolución de algunos conflictos que pudieran surgir, por ejemplo, por alcance o similitud de nombre.

- **Implementar una aplicación Web (*front-end*)** para la exploración y visualización de los datos recopilados.

1.3. Estructura de la memoria

A continuación se indica la estructura del documento de memoria. El Capítulo 2 muestra los resultados del análisis realizado sobre las fuentes de datos, con el fin de determinar cuáles de ellas utilizar como input en este trabajo. El Capítulo 3 detalla la concepción de la solución, mediante la definición de los principales requisitos de la solución, su arquitectura y el modelo de datos. Debido a que la mayor parte del software creado corre en el back-end, en el Capítulo 4 se presentan el diseño detallado de los componentes, desde la recopilación de información hasta la construcción de las consultas. En el Capítulo 5 se muestran las distintas visualizaciones construidas y las vistas con las que interactúa el usuario. En el Capítulo 6 se describe el proceso que se utilizó para validar el sistema y los resultados obtenidos. Finalmente, en el Capítulo 7 se presentan las conclusiones sobre el trabajo realizado, y el posible trabajo a futuro a ser realizado en una eventual próxima iteración.

Capítulo 2

Marco teórico

El primer paso realizado para poder concebir la solución fue la identificación y análisis de las posibles fuentes de datos a utilizar para la construcción del directorio de investigadores. La cantidad de registros y la disponibilidad de éstos al público general, la existencia de APIs o paywalls, y la completitud de la información obtenida, fueron los principales factores considerados a la hora de decidir qué fuentes utilizar y qué datos extraer de cada una.

A continuación, se presentan las fuentes de información consideradas durante esta fase de investigación, detallando la información disponible en cada una de éstas, los caminos disponibles para la obtención de dicha información, y las conclusiones obtenidas en fase de la investigación.

2.1. DBLP

DBLP [4] es un gran repositorio bibliográfico de artículos relacionados con Ciencias de la Computación. Actualmente contiene registros de más de 3,8 millones de artículos y 1,9 millones de autores [5]. Este repositorio reúne información sobre artículos publicados en los *journals* de mayor importancia en el área, y en las actas (*proceedings*) de múltiples conferencias.

DBLP libera el contenido del repositorio bajo la *Open Data Commons Attribution License* [15], y ofrece múltiples maneras de recuperar la información que esta iniciativa almacena. Entre las alternativas de entrega de información están las siguientes:

- A través de su **sitio web** [4], el cual presenta múltiples opciones de búsqueda y navegación al usuario final.
- Brinda la capacidad de **exportar datos** de un autor o artículo en particular en múltiples formatos (RIS, BibTeX, JSON, XML, entre otros).
- Ofrece una **API** de búsqueda, con endpoints para la búsqueda de autores, publicaciones y *journals* o conferencias.

- Permite obtener un ***dump*** de la información contenida en su repositorio, entregando un único archivo (3GB) con formato XML, el cual es generado/actualizado mensualmente.

Las entradas del repositorio se clasifican en seis categorías: *Article*, *Book*, *InCollection*, *Proceedings*, *InProceedings* y *Author*. Cada categoría tiene un conjunto particular de atributos, los cuales se muestran en los siguientes ejemplos.

```
<article key="persons/LeyCHM11" mdate="2017-06-09">
    <author>Rita Ley</author>
    <author orcid="0000-0002-1163-8988">Markus Casper</author>
    <author>Hugo Hellebrand</author>
    <author>Ralf Merz</author>
    <title>Catchment classification by runoff behaviour with self-organizing maps (SOM)</title>
    <journal>Hydrology and Earth System Sciences</journal>
    <volume>15</volume>
    <pages>2947-2962</pages>
    <year>2011</year>
    <ee>https://doi.org/10.5194/hess-15-2947-2011</ee>
</article>
```

Figura 2.1: Elemento tipo *Article*

Los elementos de tipo *Article* (Figura 2.1) corresponden a artículos de revista, que contienen una lista de autores, el título de la publicación, el nombre del journal y el año de publicación. El resto de los atributos son opcionales. Los elementos de tipo Book (Figura 2.2) corresponden a libros, y contienen una lista de autores/editores, el título del libro y el año de publicación, mientras que el resto de los atributos son opcionales.

```
<book key="books/daglib/0037002" mdate="2017-05-16">
    <editor>Alexandru Baltag</editor>
    <editor>Sonja Smets</editor>
    <title>Johan van Benthem on Logic and Information Dynamics</title>
    <publisher>Springer</publisher>
    <year>2014</year>
    <isbn>978-3-319-06024-8</isbn>
    <isbn>978-3-319-06025-5</isbn>
    <ee>https://doi.org/10.1007/978-3-319-06025-5</ee>
    <url>db/books/collections/BS2014.html</url>
</book>
```

Figura 2.2: Elemento tipo *Book*

```
<incollection key="reference/algo/X08o" mdate="2017-05-16" publtype="encyclopedia entry">
    <title>Atomic Scan.</title>
    <year>2008</year>
    <booktitle>Encyclopedia of Algorithms</booktitle>
    <ee>https://doi.org/10.1007/978-0-387-30162-4_41</ee>
    <crossref>reference/algo/2008</crossref>
    <url>db/reference/algo/algo2008.html#X08o</url>
</incollection>
```

Figura 2.3: Elemento tipo *InCollection*

Los elementos de tipo *InCollection* (Figura 2.3) corresponden principalmente a la publicación de capítulos de libros, cuya metadata contiene el título de la publicación, el título de la colección y el año de publicación; al igual que en los casos anteriores, tiene un conjunto de atributos opcionales.

Los elementos *InProceedings* (Figura 2.4) corresponden a artículos publicados en las actas de conferencias internacionales. Los metadatos de estos elementos incluyen una lista de autores, el título de la publicación, el título de las actas (*proceedings*) y el año de publicación,

además de algunos atributos opcionales. Los elementos de tipo *Proceedings* (Figura 2.5) corresponden a la publicación de actas de conferencias. Sus únicos campos requeridos son el título y el año de publicación, pero comúnmente contienen la lista de editores y el *publisher*. El resto de los campos son opcionales.

```
<inproceedings key="conf/aldt/ColorniT13" mdate="2017-05-23">
    <author>Alberto Colorni</author>
    <author>Alexis Tsoukiàs</author>
    <title>What Is a Decision Problem? Preliminary Statements.</title>
    <pages>139-153</pages>
    <year>2013</year>
    <booktitle>ADT</booktitle>
    <ee>https://doi.org/10.1007/978-3-642-41575-3_11</ee>
    <crossref>conf/aldt/2013</crossref>
    <url>db/conf/aldt/adt2013.html#ColorniT13</url>
</inproceedings>
```

Figura 2.4: Elemento tipo *InProceedings*

```
<proceedings key="conf/ivcnz/2012" mdate="2013-01-15">
    <editor>Brendan McCane</editor>
    <editor>Steven Mills</editor>
    <editor>Jeremiah D. Deng</editor>
    <title>Image and Vision Computing New Zealand, IVCNZ '12, Dunedin, New Zealand</title>
    <publisher>ACM</publisher>
    <booktitle>IVCNZ</booktitle>
    <year>2012</year>
    <isbn>978-1-4503-1473-2</isbn>
    <ee>http://dl.acm.org/citation.cfm?id=2425836</ee>
    <url>db/conf/ivcnz/ivcnz2012.html</url>
</proceedings>
```

Figura 2.5: Elemento tipo *Proceedings*

Finalmente, los elementos de tipo *Author* (Figura 2.6), codificados como elementos *www*, contienen los nombres de los autores identificados, y los aliases conocidos bajo los cuales cada uno publica. Este elemento también cuenta con atributos opcionales, como el ORCID del autor o sus afiliaciones.

```
<www key="homepages/28/5908" mdate="2016-06-29">
    <author>Leonard S. Haynes</author>
    <author>Leonard Stanley Haynes</author>
    <author>Leonard Haynes</author>
    <title>Home Page</title>
    <note type="affiliation">Intelligent Automation Inc., Rockville, MD, USA</note>
</www>
```

Figura 2.6: Elemento tipo *Author*

DBLP destaca por la gran cantidad de información disponible y las múltiples vías y formatos en las que dicha información se puede obtener. A esto se suma su especialización en el área de las Ciencias de la Computación, lo que evita la necesidad de identificar y filtrar por área los registros que se desean utilizar. Estos motivos hacen de DBLP un candidato ideal para su uso en la construcción del directorio. Sin embargo, cabe notar que las entradas de cada tipo de elemento tienen distintos niveles de completitud en sus atributos opcionales, por lo que es necesario complementar la información faltante con aquella disponible en otras fuentes; particularmente, la información relacionada con los autores es la más incompleta y difícil de obtener.

2.2. Mendeley

Mendeley [11] es un gestor de referencias bibliográficas/red social académica que recolecta gran cantidad de metadata relacionada con artículos científicos. Para acceder a esta información se puede hacer uso de una API [12], la cual cuenta con múltiples *endpoints*. Esta API es pública y está disponible para cualquier persona que cuente con una cuenta de usuario gratuita en dicha plataforma.

Dentro de los *endpoints* disponibles en Mendeley, los siguientes son de particular interés:

- ***/catalog***: Permite recuperar la metadata existente de un documento, dado un identificador. Los identificadores posibles son: ***arxiv***, ***doi***, ***isbn***, ***issn***, ***pmid***, ***scopus*** y ***ssrn***.

- ***/institution***: Permite recuperar información sobre una institución en base a una ubicación, un dominio de correo o una búsqueda por nombre.

- ***/search/catalog***: Permite realizar una búsqueda en el catálogo de documentos (artículos, libros, etc.). La búsqueda se basa en los siguientes parámetros: ***title***, ***author***, ***source***, ***abstract***, ***min_year***, ***max_year*** y ***open_access***.

La Figura 2.7 muestra un ejemplo de resultado obtenido al consultar por un documento en Mendeley. La estructura del resultado es la misma independientemente del endpoint utilizado, y la completitud de la información varía según el documento consultado. La Figura 2.8 presenta un ejemplo del resultado obtenido al consultar por una institución.


```
{
    "title": "Disseminating shared information in (...)",
    "authors": [
        {
            "first_name": "Rodrigo",
            "last_name": "Santos",
        },
        {
            "first_name": "Sergio F.",
            "last_name": "Ochoa",
            "scopus_author_id": "8605223400",
        }
    ],
    "year": 2011,
    "source": "Conference Proceedings - IEEE International (...)",
    "identifiers": {
        "doi": "10.1109/ICSMC.2011.6084202",
        "issn": "1062922X",
        "sgr": "83755173795",
        "isbn": "9781457706523",
        "pui": "363114349",
        "scopus": "2-s2.0-83755173795",
    },
    "keywords": [
        "communication computable model",
        "coordination activities",
        "disaster relief efforts",
        "opportunistic network"
    ],
    "abstract": "VHF radio systems commonly used to (...)"
}
```


Figura 2.7: Resultado de la búsqueda de un documento en Mendeley

```
{
    "id": "",
    "name": "University of Chile",
    "city": "Santiago de Chile",
    "state": "",
    "country": "CL",
    "urls": [
        "www.uchile.cl"
    ],
    "alt_names": [
        {
            "name": "SISIB"
        },
        {
            "name": "Universidad de Chile"
        }
    ]
}
```


Figura 2.8: Resultado de la búsqueda de una institución en Mendeley

2.3. Google Scholar

Google Scholar [9] ofrece información sobre autores en sus perfiles de usuario (*User Profiles*), particularmente datos como la afiliación de la persona, el dominio de correo, varios índices relativos al investigador (*h-Index*, *i10-Index*, *citations*) y los temas de investigación en los que trabaja (Figura 2.9). Si bien no se expone una API para realizar búsquedas, sí es posible procesar el HTML retornado para extraer la información mencionada.

Figura 2.9: Ejemplo de *User Profile* en Google Scholar

2.4. Elsevier

Elsevier [6] posee una gran cantidad de información, la cual está distribuida entre múltiples sitios y APIs [7]; por ejemplo Scopus, Embase, y ScienceDirect, entre otros. Sin embargo, sólo una pequeña fracción de los documentos (y de la información disponible de ellos) está disponible al público, mientras que la mayor parte están protegidos tras un *paywall*. Por este motivo, se descarta el uso de estas APIs como fuentes de información para la construcción del directorio de investigadores.

2.5. ISI Web of Knowledge

ISI Web of Knowledge [18] (ahora llamado *Web of Science*) es un servicio de suscripción, y como tal, protege todo su contenido tras un *paywall*, por lo que también se descarta su uso para la construcción del directorio.

2.6. Crossref

Crossref [2] es una agencia oficial de registro de DOIs[1]que contiene millones de registros de una gran variedad de tipos de publicaciones, incluidos *journals*, libros, actas (*proceedings*), papers y datasets. Crossref posee una API [3] con múltiples *endpoints* que permiten acceder de forma gratuita a esta información, de los cuales (en este caso) sólo interesa ***/works***. Cabe mencionar que Crossref sólo posee información de documentos registrados por la agencia, es decir, aquellos que posean un DOI emitido por Crossref. La Figura 2.10 muestra la estructura que poseen los resultados obtenidos desde las consultas a este endpoint (se muestran sólo los campos relevantes para la construcción del directorio).


```
{
    "title": ["Genotype-Related Effect of (...)"],
    "type": "journal-article",
    "container-title": ["BioMed Research International"],
    "volume": "2014",
    "page": "1-11",
    "abstract": "This study investigated the (...)",
    "author": [
        {
            "ORCID": "http://orcid.org/0000-0001-9757-3283",
            "authenticated-orcid": true,
            "given": "Peter",
            "family": "Slezak",
            "affiliation": [
                {"name": "Institute of Normal and Pathological Physiology"}
            ]
        },
        (...)
    ],
    "URL": "http://dx.doi.org/10.1155/2014/413629",
    "ISSN": ["2314-6133", "2314-6141"],
    "subject": ["Blood Pressure", "Vascular Function"],
    (...)
}
```


Figura 2.10: Resultado de la búsqueda de un documento en Crossref

[1]DOI: Digital Object Identifier.

Capítulo 3

Concepción del sistema

En este capítulo se describe la esencia de la solución y los principales componentes que forman parte de ella, partiendo por los principales requisitos que el sistema debe cumplir.

3.1. Requisitos del sistema

- **Recolección de información personal de los autores:** El sistema debe recopilar información personal sobre los autores, que sea relevante al problema planteado (por ej., afiliaciones, temas de investigación, índices de citaciones, etc.).
- **Identificación de coautores:** El sistema debe ser capaz de identificar y mostrar todos los coautores de un determinado autor, indicando el número de colaboraciones con cada uno de ellos.
- **Identificación de la conexión (distancia de coautoría) entre dos personas:** El sistema debe ser capaz de determinar si dos investigadores están conectados mediante vínculos de coautoría. En caso de estar conectados, el sistema permitirá visualizar el camino encontrado entre ellos.
- **Búsqueda de coautores por tema de investigación:** Para un autor dado, el sistema debe ser capaz de encontrar coautores que hayan publicado al menos un artículo del tema de investigación (usando palabras claves) indicado por el usuario.
- **Búsqueda de autor por nombre:** El sistema debe implementar una barra de búsqueda que permita encontrar autores a partir de un nombre completo o parcial.
- **Búsqueda de autores por temas de investigación:** El sistema debe implementar una búsqueda basada en un conjunto de temas de investigación, encontrando todos los autores que hayan publicado al menos un artículo de cada tema.
- **Búsqueda de revisores:** El sistema debe implementar una búsqueda que permita encontrar revisores para una lista de temas de investigación y un conjunto de autores dado.

- **Filtrado de información por fecha:** El sistema debe permitir filtrar la información del sistema (artículos, coautores, etc.) por fecha, indicando el rango de tiempo (en años) que se desea visualizar.
- **Entrega de información de los artículos:** El sistema debe mostrar adecuadamente la información bibliográfica de los artículos ingresados (por ej., título, autores, journal, issue, DOI, etc.).
- **Recopilación automática de información:** El sistema debe recopilar información automática y periódicamente desde las distintas fuentes utilizadas.
- **Idioma de las interfaces de usuario:** El texto de las vistas del sistema debe estar escrito en inglés.

3.2. Arquitectura del sistema

En esta sección se presenta la arquitectura física y lógica del sistema.

3.2.1. Arquitectura física

La arquitectura física del sistema (Figura 3.1) corresponde a una de tipo Cliente-Servidor, donde el usuario (a través de un navegador) realiza peticiones HTTP a un servidor, el cual las procesa y retorna al usuario el recurso asociado a la URL solicitada. En este caso, el encargado de procesar las peticiones es Flask [8], ejecutando las consultas necesarias a la base de datos para construir las vistas que serán retornadas al cliente. Flask además realiza las peticiones necesarias a las fuentes de información externas durante los procesos de construcción y actualización de la base de datos, la cual está implementada sobre Neo4j [13].

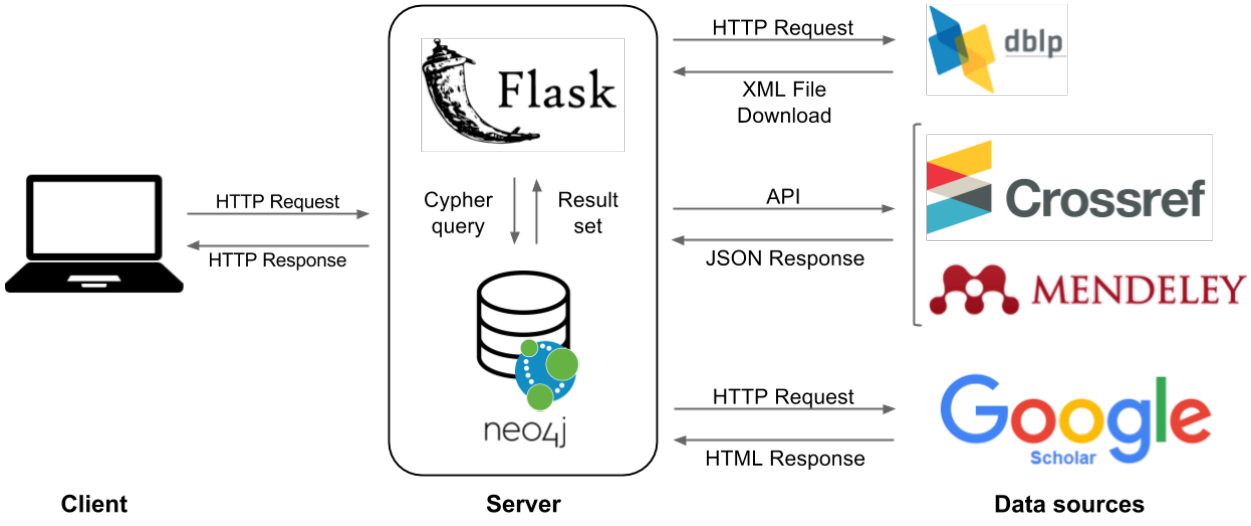

Figura 3.1: Arquitectura física del sistema

3.2.2. Arquitectura lógica

El sistema fue desarrollado utilizando Python 3.5, trabajando sobre el framework Flask para el desarrollo de la aplicación Web. Éste implementa una arquitectura MVC (Modelo-Vista-Controlador), la cual se detalla a continuación:

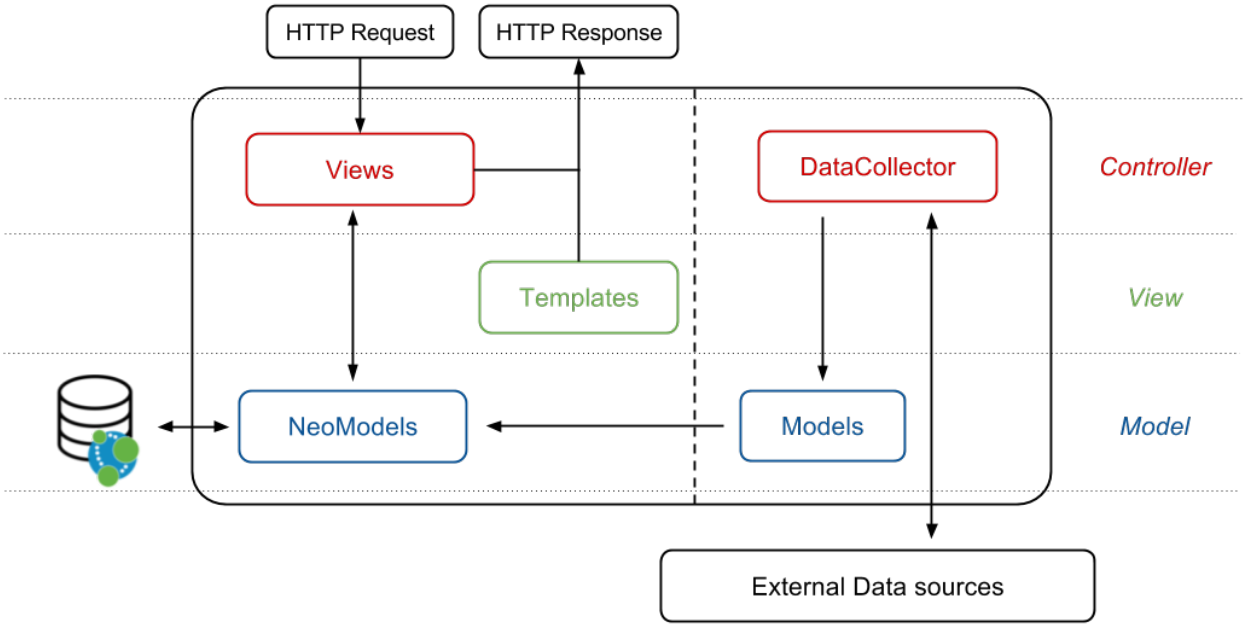

Figura 3.2: Arquitectura lógica del sistema

La Figura 3.2 muestra la interacción entre los componentes del sistema y la clasificación de éstos en Modelo, Vista y Controlador según corresponda. La figura muestra además la división de los componentes entre los subsistemas de **aplicación web** (*Views*, *Templates*, *NeoModels*) y de **recopilación de información** (*DataCollector*, *Models*).

En la **aplicación web**, el **modelo** está representado por el módulo *NeoModels*. En éste se definen los nodos y relaciones a utilizar dentro del grafo, estableciendo los campos de cada uno y las restricciones sobre éstos (tipo del campo, si es requerido o no, unicidad, etc.). Para facilitar este proceso de mapeo entre nodos y objetos, se hace uso de ***Neomodel*** [14], un OGM[1]para Neo4j que proporciona una serie de funcionalidades para la realización de consultas a la base de datos.

Las **vistas** están representadas por los *templates*, plantillas HTML con bloques de código en Python, que permiten generar vistas dinámicas a partir de un mismo archivo en función de los parámetros utilizados. El resultado es un archivo HTML, potencialmente complementado con CSS y Javascript, el cual puede ser enviado al cliente dentro de una respuesta HTTP.

En Flask, los **controladores** son las llamadas *views*, funciones que ejecutan la lógica de negocio necesaria para procesar las peticiones HTTP, y retornar una respuesta HTTP

[1]OGM: Object-Graph Mapper. El análogo a ORM para bases de datos de grafos.

apropiada. Los controladores *siempre* retornan una respuesta HTTP, pero el contenido de ésta puede variar según la petición (por ejemplo, podrían retornar JSON, XML, HTML generado a partir de un template, etc.). Cada *view* tiene una *ruta* asociada, un patrón de URL que determina qué peticiones procesa cada controlador. Estos patrones permiten, además, extraer variables desde la URL, lo que evita la necesidad de codificar información sobre los recursos a los que se accede como parámetros GET. Por ejemplo, al procesar una petición a la URL *'/users/123/documents/456'* con el patrón *'/users/<user_id>/documents/<doc_id>'*, se crean las variables *user_id* y *doc_id* con los valores 123 y 456, respectivamente.

En cuanto al subsistema de **recopilación de información**, el controlador corresponde al módulo *DataCollector*. Éste se encarga de recopilar la información desde las distintas fuentes externas durante los procesos de carga inicial y de actualización de datos. La información recopilada es integrada y consolidada en objetos, cuyas clases están definidas en el módulo *Models*, que constituyen la representación (en Python) de los elementos XML obtenidos desde DBLP. Estos elementos pueden ser complementados con la metadata adicional obtenida desde otras fuentes como Crossref y Mendeley. Finalmente, la información contenida en estos objetos es almacenada en la base de datos en forma de nodos y relaciones entre ellos (modelados en el módulo *NeoModels*).

3.3. Modelo de Datos

A continuación se presenta el modelo de datos (Figura 3.3) utilizado para representar la información recolectada, junto al detalle de los campos de cada entidad y relación entre entidades. Además de los campos mencionados, todos los nodos poseen un identificador único ***uid***. Las relaciones no mencionadas en el detalle no poseen atributos adicionales.

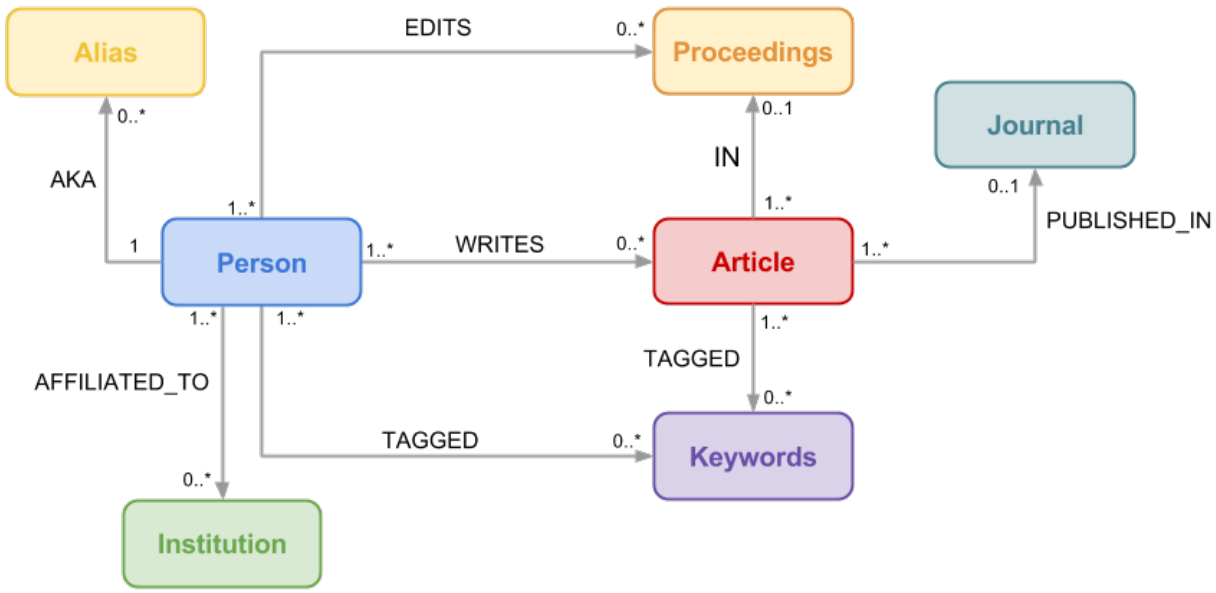

Figura 3.3: Modelo de Datos

- **Person**: Nodo que representa a una persona. Posee los campos name, ***first_name***, ***last_name***, ***dblp_id*** (esto es, el ID el elemento en DBLP), ***orcid***, ***email_domain***, ***scholar*** (cuya función se aclara más adelante), ***image***, ***citations***, ***i10_index*** y ***h_index***.

- **Alias**: Nodos que representan los múltiples nombres bajo los que publica una persona. Su único campo es ***name***.

- **Institution**: Nodo que representa a una institución. Posee los campos ***name*** y ***location***.

- **Article**: Nodo que representa una publicación (es decir, un artículo de *journal* o de una conferencia). Posee los campos ***title***, ***year***, ***dblp_id***, ***doi***, ***isbn*** e ***issn***.

- **Journal**: Nodo que representa un *journal*. Su único campo es ***name***.

- **Proceedings**: Nodo que representa a las actas de una conferencia. Posee los campos ***title***, ***year***, ***dblp_id***, ***doi***, ***isbn*** e ***issn***.

- **Keyword**: Nodo que representa un tema de investigación, con los cuales se etiqueta tanto a personas como a artículos. Su único campo es ***name***.

- **PUBLISHED_IN**: Establece la relación entre *Article* y *Journal*. Contiene información sobre la publicación en los campos ***volume***, ***issue*** y ***pages***.

- **IN**: Establece la relación entre *Article* y *Proceedings*. Representa a los artículos presentados en conferencias (*InProceedings*). Su único campo es ***pages***.

Cabe destacar que si bien las relaciones de coautoría son uno de los focos principales del sistema, éstas no son representadas directamente con relaciones entre entidades de tipo Persona, sino que son deducidas a partir de las relaciones de autoría entre un Artículo y sus autores, lo que permite, además, tener un mayor control a la hora de filtrar los resultados obtenidos.

Como se mencionó antes, dada la naturaleza de las relaciones entre las entidades y de las consultas que se desea realizar sobre los elementos, resulta conveniente almacenar la información en una base de datos de grafos, optando en este caso por utilizar Neo4j (y su lenguaje de consulta asociado, Cypher) como motor de bases de datos para el sistema.

Los nodos y relaciones que aparecen en el diagrama son modelados en forma de clases, mediante el uso del OGM neomodel mencionado anteriormente, y corresponden a la principal vía de interacción entre la aplicación web implementada y la base de datos.

Capítulo 4

Diseño detallado

A continuación se presentan los detalles de la implementación de los distintos componentes de la solución, desde la recolección de los datos desde las distintas fuentes, hasta la construcción de consultas a ser utilizadas por el usuario final.

4.1. Recolección de información

Como fue mencionado, el objetivo principal del sistema desarrollado es permitir la recopilación automática de información sobre investigadores en Ciencias de la Computación desde múltiples fuentes. Si bien todas las fuentes presentan una forma de recuperar metadata, la mayoría funciona en base a búsquedas de un documento en particular (ya sea por texto o con algún identificador), en lugar de listar todas las entradas disponibles. DBLP es la única fuente que posee una forma rápida y conveniente de obtener el conjunto de personas y documentos de interés (es decir, sólo aquellos relacionados con Ciencias de la Computación). A esto lo brinda a través de un dump de su base de datos, que se entrega en un archivo XML. Por este motivo, se decidió utilizar dicho dump como la base para instanciar el directorio, pasando a considerar el conjunto de registros contenidos en éste, como el universo de publicaciones y autores. A continuación se detallan los métodos de recopilación de información implementados para cada una de las fuentes de información utilizadas.

4.1.1. DBLP

El dump XML ofrecido por DBLP, en conjunto con el DTD[1] asociado, se encuentra disponible en formato comprimido en `http://dblp.uni-trier.de/xml/`.

Procesar el dump XML implica resolver tres problemas: (1) la incapacidad de cargar el archivo completo en memoria (dado su gran tamaño), (2) la detección de los elementos (es

[1]DTD: Document Type Definition. Corresponde a la descripción de la estructura y sintaxis del documento XML.

decir, reconocer los *tags* de apertura y cierre de los elementos, además de parsear sólo los que interesan) y (3) el procesamiento de los atributos (detectar la presencia de atributos requeridos y opcionales, normalizar tipos y formatos, etc.). Para resolver estos problemas se implementó la clase ***DblpParser***, cuyo funcionamiento se explica a continuación.

DblpParser está basado en *ElementTree*, un parser XML incluído en la librería estándar de Python, que provee una API con la cual se puede navegar y realizar operaciones básicas sobre el árbol XML contenido en un archivo. Entre las funcionalidades que ofrece esta API destaca ***iterparse***, un iterador[2] que parsea incrementalmente un archivo dado, manteniendo un subárbol en memoria y retornando cada vez que se abre o cierra un elemento. Esto permite tener un mayor control sobre la información mantenida en memoria, saltándose aquellos elementos que no son de interés, guardando en memoria la información con la que se está trabajando, y eliminando aquellos que ya fueron procesados. De esa manera se reduce el espacio de memoria requerido para procesar cada archivo, y se resuelve el primer problema mencionado.

El segundo problema involucra detectar cuáles de los *tags* de apertura y cierre son los que nos interesan, definiendo así cuándo empezar a guardar la información en memoria y hasta qué punto. El componente *iterparse* retorna cada vez que encuentra un *tag*, entregando el nombre del *tag* encontrado y el tipo de evento (*start* o *end*). El proceso de detección, entonces, funciona de la siguiente manera:

1. Mientras no se encuentre un *tag* de interés (es decir, *article*, *inproceedings*, etc.), se descartan todos los elementos retornados.

2. Al recibir un evento *start* para uno de los *tags* de interés, éste y todos los elementos siguientes son almacenados en memoria.

3. Si se recibe otro evento *start* de interés antes de cerrar el recibido anteriormente (por ejemplo, hay un *proceedings* dentro de un *article*), entonces se arroja una excepción, indicando que la estructura del XML parseado no es la esperada.

4. Finalmente, cuando se recibe el evento *end* correspondiente al elemento procesado actualmente, la información mantenida en memoria es procesada y retornada, liberando el espacio utilizado en memoria y reiniciando el proceso de detección en la próxima iteración.

Una vez obtenido el elemento, es necesario recuperar la metadata contenida en sus atributos y sub-elementos, estandarizando el tipo y formato de cada uno de ellos. Esta información es almacenada en objetos de tipo ***DblpElement*** y sus subclases ***DblpArticle***, ***DblpAuthor***, ***DblpBook***, ***DblpInCollection***, ***DblpInProceedings*** y ***DblpProceedings***[3], cada una con sus respectivos campos requeridos y opcionales, definiendo los valores por defecto necesarios en caso de no existir valor en el dump. El hecho de codificar los elementos del dump XML como objetos, permite tener una estructura estándar para éstos independientemente de su grado de completitud, simplificando el código de operaciones posteriores al evitar estructuras condicionales.

[2]Clase que implementa los métodos iter() y next(). Estos pueden ser utilizados como generador de elementos en un loop, como un for o un while.

[3]El prefijo "Dblp" es utilizado para diferenciar las clases de los elementos obtenidos desde el dump de las clases utilizadas para el OGM.

DblpParser es entonces un iterador construido sobre *ElementTree* que, dado un archivo XML y una lista de tipos de elementos, retorna en cada iteración un objeto de tipo ***DblpElement*** con su respectiva información ya parseada y procesada a un formato estándar.

4.1.2. Crossref

Al recuperar la metadata de un documento (es decir, un *article*, *book*, *proceedings* o *inproceedings*) desde Crossref, se distinguen dos escenarios:

- **El elemento posee DOI**: En este caso, se utiliza la API de Crossref para consultar directamente por el documento (es decir, utilizando el endpoint *'/works/<doi>'*). Si el DOI fue emitido por Crossref, la API retorna la metadata existente en formato JSON[4]. Si no fue emitido por Crossref, no entrega metadata adicional.

- **El elemento no posee DOI**: En este caso, se utiliza la API de búsqueda de Crossref para encontrar el elemento en base a la metadata conocida, particularmente el **título**, el **autor** y el **contenedor** (el *journal* o el *proceedings*, dependiendo del documento). La respuesta emitida por la API es un listado con todos los documentos encontrados y sus respectivos ***scores***, una medida de confianza en los resultados de la búsqueda que va entre 0 y 100. El score mínimo aceptado para insertar la metadata encontrada al directorio es de 85. Cualquier resultado con un *score* inferior es descartado.

La estructura de la metadata retornada por la API (Figura 2.10) es la misma independientemente del método de búsqueda utilizado, por lo que el procesamiento de ésta es el mismo en ambos casos. Antes de procesar la información recibida, se realizan verificaciones adicionales para tener mayor seguridad en los datos obtenidos, especialmente para los casos donde no se tiene un DOI. Estas verificaciones consisten en comparar el número de autores conocido (obtenido desde DBLP) con el obtenido desde Crossref, además de verificar que el apellido de cada autor obtenido desde Crossref (el campo **family**) esté contenido en el ***name*** obtenido desde DBLP. Esta última verificación permite asegurar que estamos añadiendo la metadata obtenida a las personas correspondientes, evitando así propagar errores a múltiples nodos.

4.1.3. Mendeley

El caso de Mendeley es esencialmente idéntico al de Crossref. Sin embargo, Mendeley no posee un *score* en el cual basarse para tomar la decisión de si insertar la información o no en caso de no existir un DOI. Pruebas realizadas durante el período de investigación revelaron un muy bajo porcentaje de acierto, por lo que se tomó la decisión de utilizar Mendeley sólo para aquellos elementos que posean DOI.

[4]JSON: JavaScript Object Notation

4.1.4. Google Scholar

Como ya se mencionó, la información de interés a recuperar desde Google Scholar está contenida en los *User Profiles*. Como no existe API para consultar esta información, es necesario recurrir a métodos de web scraping para recuperarla. Este proceso tiene dos partes:

- **Búsqueda del perfil**: A partir del nombre de un autor y del título de su publicación más reciente, se arma una URL de búsqueda avanzada de Google Scholar con el fin de encontrar un link al perfil del autor en el HTML generado. La Figura 4.1 muestra el HTML resultante al realizar la búsqueda de un autor. En caso de no ofrecer un link al perfil directamente como en la figura, dicho link se extrae desde el listado de autores (en verde) del primer resultado.

- **Extraer información del perfil**: Una vez encontrada la URL del perfil, es posible extraer la información del autor encontrando los *div* correspondientes mediante sus *ids*. De esta manera es posible extraer datos como la afiliación, temas de investigación, el dominio del correo electrónico, la imagen de perfil y los índices de citaciones (*i10-Index* y *h-Index*). Como es de esperar, el nivel de completitud de esta información varía de autor a autor.

Figura 4.1: Resultados de la búsqueda de un perfil de usuario en Google Scholar

Un problema que se presenta al utilizar Google Scholar como fuente de datos es la limitación de peticiones. Dado que no presenta una forma "oficial" de recuperar la información, no se sabe con seguridad la cantidad de peticiones que es posible realizar en un período de tiempo determinado antes de que se bloqueen las búsquedas. Para evitar caer en esa situación se optó por adoptar una estrategia *lazy*, realizando este proceso de recolección de información la primera vez que un usuario accede al perfil del autor en la aplicación web, utilizando el campo ***scholar*** de cada nodo Persona para indicar si ya se intentó recuperar esta información o no (evitando así consultar infinitamente en caso de que el autor no tenga *User Profile* en Google Scholar).

4.2. Integración de los datos

Una vez recopilada la información desde las múltiples fuentes, ésta es combinada en las clases definidas en el módulo *Models* ya mencionadas (*DblpElement* y sus subclases). En caso

de existir conflictos, éstos son resueltos estableciendo un orden de confianza para cada fuente:

- Al utilizarlo como base del directorio, asignamos de forma implícita el mayor grado de confianza a **DBLP**. Operar de esta manera permite evitar la propagación de cambios al encontrar conflictos que involucran a múltiples entidades.

- La segunda fuente más confiable es **Crossref**. Al tratarse de una agencia de registro de DOIs, la información que utiliza es obtenida desde la fuente. La excepción son aquellas publicaciones que no poseen DOI, o que poseen un DOI emitido por otra agencia, para las cuales no se cuenta con información alguna.

- La última prioridad la tiene **Mendeley**, el cual utiliza información ingresada por los usuarios de la plataforma para completar el catálogo, lo que disminuye drásticamente la confianza en los resultados, especialmente para aquellas publicaciones sin DOI.

- **Google Scholar** no presenta conflictos, ya que la información que entrega no es excluyente con la de otras fuentes (por ejemplo, temas de investigación, índices de citaciones, etc.).

4.3. Carga de datos

A continuación, se detalla el proceso de construcción y mantención de la base de datos del directorio, tanto la carga inicial de información al levantar el sistema por primera vez, como las posteriores actualizaciones a realizar periódicamente.

4.3.1. Carga inicial

El proceso inicia con la descarga del último dump disponible en DBLP, el cual, como ya fue explicado previamente, constituye el universo de autores y artículos a insertar en el directorio. Los elementos XML contenidos en el dump corresponden a aquellos mencionados anteriormente (*article*, *inproceedings*, etc.), los cuales son procesados uno a uno haciendo uso de *DblpParser* y luego complementados con la información disponible en otras fuentes, las cuales dependen del tipo de elemento:

- En el caso de los **documentos** (es decir, *article*, *proceedings*, etc.), la información de estos elementos se complementa con datos de Crossref y, en caso de poseer un DOI, con Mendeley, con el fin de obtener **identificadores**, **keywords** e información adicional sobre la publicación, como volumen, páginas, fecha, etc. Además, en algunos casos es posible recuperar información adicional sobre los autores, como identificadores y afiliaciones.

- En el caso de los **autores**, se complementa con Google Scholar, con el fin de obtener la **imagen de perfil**, las **afiliaciones**, los **temas de investigación**, índices de **citaciones** y el **dominio de correo electrónico** de la persona. Como ya se mencionó, esta

información se recupera de forma *lazy*, realizando las consultas la primera vez que se accede al perfil de un autor.

Una vez integrada la información de las distintas fuentes, el proceso continúa con la creación de los nodos respectivos y estableciendo las relaciones correspondientes entre ellos, en el siguiente orden:

1. **Personas**: Los elementos de tipo ***DblpAuthor*** son insertados en el grafo como nodos de tipo ***Person***. En caso de que el autor haya publicado bajo múltiples nombres, se crea un nodo tipo ***Alias*** por cada nombre adicional, conectándolos mediante relaciones de tipo *AKA* (*also known as*). Si existe alguna afiliación, se crea (o recupera, en caso de ya existir) el nodo de tipo ***Institution*** respectivo, conectándolo al nodo tipo ***Person*** con una relación de tipo *AFFILIATED_ TO*.

2. **Proceedings**: Los elementos de tipo ***DblpProceedings*** son insertados en el grafo como nodos de tipo ***Proceedings***. Los nodos correspondientes a los editores (insertados en el paso anterior) son recuperados y conectados al nuevo nodo mediante relaciones de tipo *EDITS*.

3. **Artículos**: Los elementos de tipo ***DblpArticle*** y ***DblpInProceedings*** son insertados en el grafo como nodos de tipo ***Article***. Los elementos de tipo *DblpArticle* son conectados al nodo de su respectivo ***Journal*** (creándolo en caso de no existir), mientras que los de tipo *DblpInProceedings* son conectados a los nodos de tipo ***Proceedings*** insertados en el paso anterior. Los nodos correspondientes a los autores (insertados en el primer paso) son recuperados y conectados al nuevo nodo mediante relaciones de tipo *WRITES*. Si existe alguna **keyword**, el nodo respectivo es conectado tanto a los nodos de los autores como al del artículo mediante relaciones de tipo *TAGGED*.

4.3.2. Actualización

El proceso de actualización es esencialmente idéntico al de carga inicial, con la excepción de que en este caso sólo se cargan o modifican datos que hayan sido modificados en DBLP desde la última actualización. Para esto, se hace uso del atributo ***mdate*** de cada elemento del dump XML, que corresponde a la fecha de la última modificación de dicho elemento. Si la fecha indicada en *mdate* es posterior a la fecha de la última actualización (la cual es almacenada al finalizar el proceso), el elemento es procesado. En caso contrario, el proceso continúa con el siguiente elemento del dump.

Además de actualizar la metadata de cada nodo, el campo ***scholar*** de cada nodo Person es cambiado a ***False***, con el fin de reiniciar el proceso de extracción *lazy* desde Google Scholar, y así mantener actualizados los índices de citaciones de cada autor.

El proceso de actualización se inicia de forma periódica mediante el uso de cron, ejecutando el script correspondiente según la programación utilizada. Se recomienda realizar la actualización de forma mensual, coincidiendo así con la generación mensual de los dumps de DBLP los primeros días de cada mes.

4.4. Construcción de consultas

Para evaluar la utilidad del sistema de recopilación de información se definieron una serie de consultas que el sistema debe ser capaz de resolver, estableciendo un set de funcionalidades mínimas a ofrecer a los usuarios finales del sistema. Además de las consultas triviales (es decir, recuperar los datos almacenados de cada entidad), se definieron 3 consultas a implementar:

- La **búsqueda de investigadores por tema de investigación**. Esta consulta permite determinar la comunidad de investigadores en torno a cada tema, e identificar a las personas más activas en investigación en el área el último tiempo.

- La **búsqueda de posibles revisores para un artículo o proyecto**, a partir de los autores del mismo y los temas de investigación asociados al artículo o proyecto a evaluar. Esta consulta busca identificar a investigadores con conocimientos en el área, que no tengan conflicto de interés con los autores de artículo o proponentes del proyecto especificado.

- La **búsqueda de conexión entre dos investigadores** a partir de las relaciones de coautoría de artículos científicos entre ellos. Esta consulta permite determinar la distancia entre dos personas, detallando el camino encontrado entre ellos. Esto permite evaluar si hay o no un posible conflicto de interés.

La complejidad de las consultas definidas escapa a las funcionalidades básicas ofrecidas por el OGM, por lo que es necesario implementar las consultas directamente en Cypher. Estas consultas hacen uso extensivo de las capacidades ofrecidas por un motor de bases de datos basado en grafos, como lo son la búsqueda por patrones y los algoritmos de búsqueda de caminos, incluídos en Neo4j. A modo de ejemplo, a continuación se detalla el proceso de construcción de la consulta de búsqueda de revisores para un artículo, ilustrada en la Figura 4.2.

La primera parte de la consulta busca los nodos de los autores seleccionados por el usuario; es decir, aquellos que escribieron el artículo a revisar. `MATCH (author:Person)` captura todos los nodos de tipo `Person` y les asigna el nombre `author`. Para seleccionar sólo aquellos de interés se agrega `WHERE author.uid IN {authors}`, donde `uid` es el ID único de cada nodo, y `{authors}` es la variable que contiene el arreglo de IDs de los autores seleccionados.

La segunda parte de la consulta encuentra todos los coautores a una distancia menor o igual a `min_distance` (indicada por el usuario). Para esto se hace uso de la búsqueda de nodos por patrones, que retorna todos los nodos que cumplan con el patrón dado. El patrón utilizado en este caso es `(author)-[:WRITES|:EDITS*1..{min_distance}]-(coAuthor:Person)`, que

```
// Seleccionar los autores de la consulta
MATCH (author:Person)
WHERE author.uid IN {authors}

// Encontrar los coautores a distancia <= min_distance
MATCH path=shortestPath(
    (author)-[:WRITES|:EDITS*1..{min_distance}]-(coAuthor:Person)
) WHERE author.uid <> coAuthor.uid
WITH DISTINCT coAuthor
WITH collect(coAuthor) as coauthors

// Buscar posibles revisores
MATCH (reviewer:Person)-[:TAGGED]->(tag:Keyword)
WHERE tag.uid IN {keywords} AND NOT reviewer IN coauthors
RETURN DISTINCT reviewer
```

Figura 4.2: Consulta Cypher para la búsqueda de revisores

representa un camino de largo variable (entre 1 y `{min_distance}`), compuesto por enlaces de tipo `:WRITES` o `:EDITS` (es decir, buscamos tanto coautores como coeditores) entre `author` (uno de los nodos encontrados en la primera parte) y `coAuthor` (los nodos de tipo Person que buscamos). El patrón está inserto dentro de un llamado a la función `shortestPath()`, uno de los algoritmos de búsqueda de caminos incorporados en Neo4j. Éste permite realizar la búsqueda de forma optimizada y eliminar ciclos y otros caminos redundantes entre personas. Finalmente, se eliminan los resultados duplicados y se recolectan todos los coautores en la colección `coauthors`, la cual contiene a todos los coautores **cercanos** y, por lo tanto, con potencial conflicto de interés.

La última parte de la consulta determina y retorna los posibles revisores a partir de la información ya recolectada. Los posibles revisores serán aquellas personas que tengan al menos una de las keywords seleccionadas, y que **no** sean cercanos a los autores seleccionados (`MATCH (reviewer:Person)-[:TAGGED]->(tag:Keyword) WHERE tag.uid IN {keywords} AND NOT reviewer IN coauthors`).

Capítulo 5

Implementación de la solución

5.1. Construcción de grafos

Dos de las principales funcionalidades del sistema son la búsqueda de coautores y la búsqueda de un camino o conexión entre dos personas. Si bien es posible expresar el resultado de estas consultas en listas o tablas, resulta mucho más natural el poder visualizarlas en forma de grafo.

Los grafos son construidos utilizando la librería **vis.js** [17], la cual fue escogida por la simplicidad de uso y personalización. Su simulador de física incorporado evita la necesidad de ubicar manualmente cada nodo y permite al usuario interactuar con el grafo arrastrando los nodos a su posición deseada. Además, facilita la navegación del sitio al incluir links en los nodos, llevando al usuario al perfil de cada autor al hacer click sobre estos últimos.

La Figura 5.1 muestra un grafo de coautores, construido a partir de las relaciones de coautoría entre un autor y sus cinco coautores. El nodo central, de forma cuadrada y mayor tamaño que los demás, representa al autor para el cual se realiza la consulta, mientras que los nodos de tamaño mediano (de colores morado, rojo, verde, rosado y amarillo), corresponden a sus cinco coautores. Los nodos más pequeños representan a los coautores de segundo orden (es decir, los coautores de los coautores), y su color corresponde al color del coautor que los conecta. El grosor de las líneas es proporcional a la cantidad de colaboraciones entre cada persona (típicamente, artículos en coautoría). Al hacer zoom, aparecen las etiquetas con el nombre de los autores y el número de colaboraciones, como muestra la Figura 5.2.

Por otro lado, la Figura 5.3 muestra el grafo construido a partir del camino encontrado entre dos autores. Los nodos de color verde y rojo son los extremos del camino, es decir, mientras que los nodos de color azul son los autores que conectan a dichos extremos mediante relaciones de coautoría y los nodos amarillos la representación del trabajo particular del que son coautores. Los nodos grises representan al resto de los autores que colaboraron en cada trabajo, pero que no son parte del camino encontrado. Cabe mencionar que el camino graficado corresponde al camino más corto encontrado entre los autores dados. En caso de haber múltiples caminos de un mismo largo, el sistema retorna el primero encontrado.

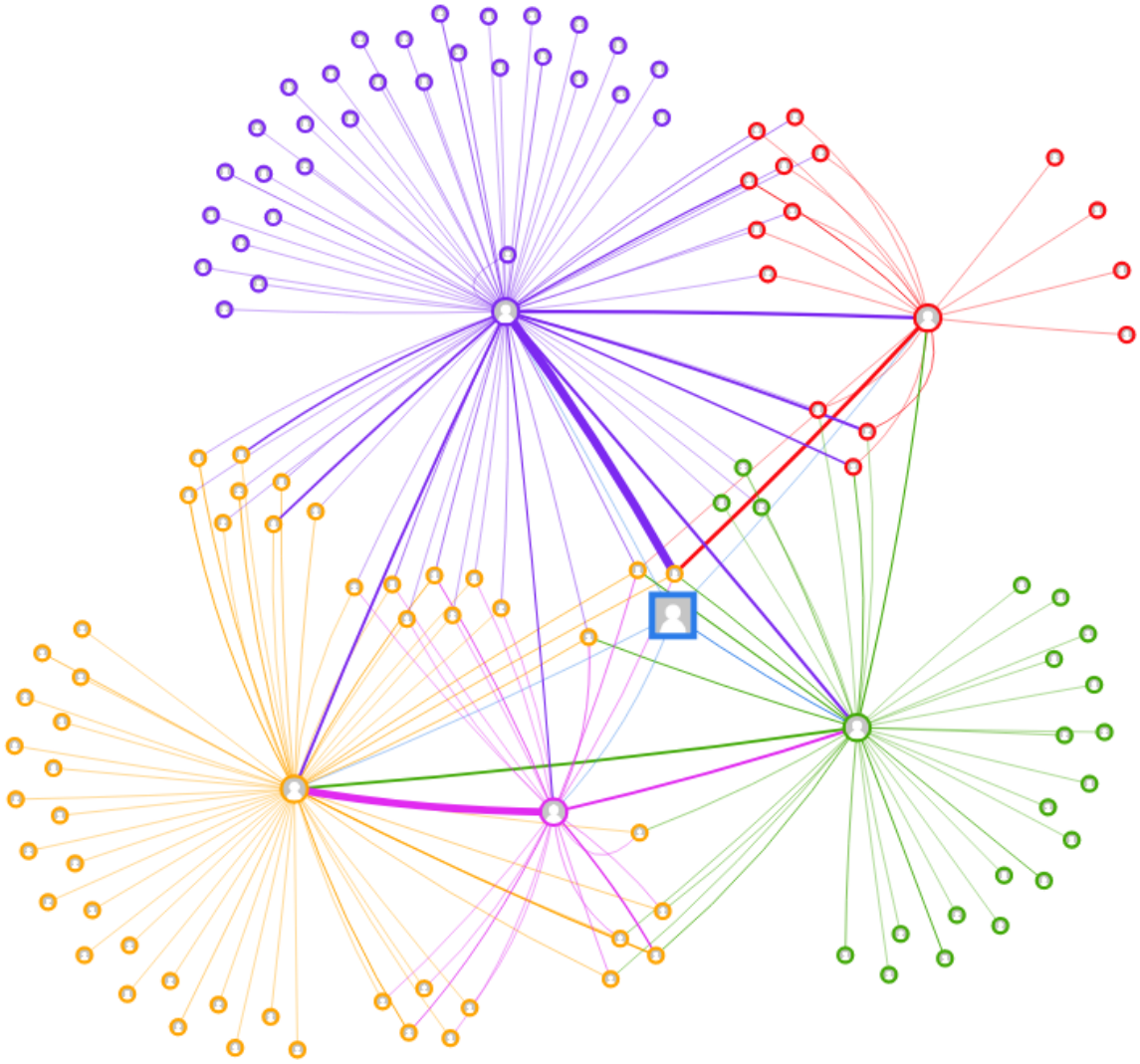

Figura 5.1: Ejemplo de grafo de coautores

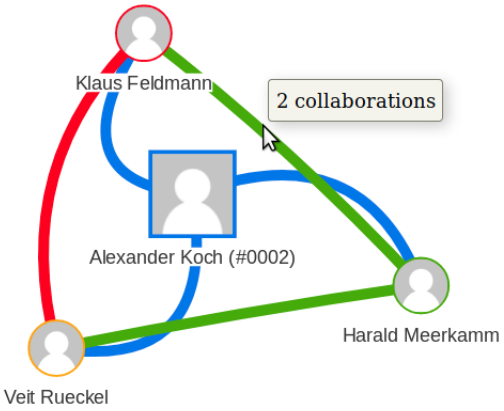

Figura 5.2: Ejemplo de grafo de coautores con etiquetas

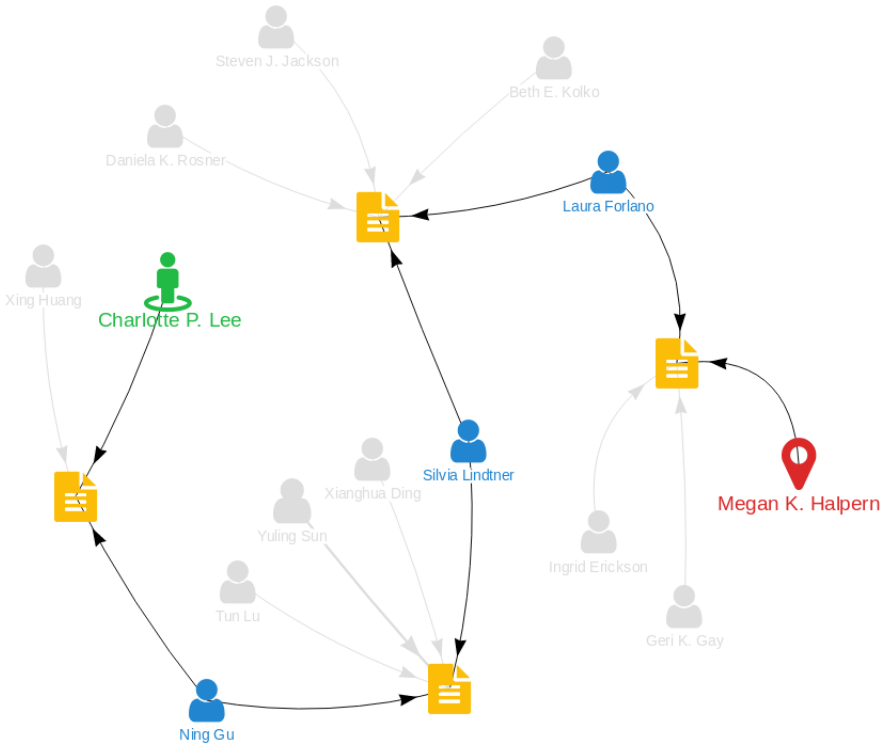

Figura 5.3: Ejemplo de grafo para un camino entre dos autores

5.2. Interfaces de usuario

A continuación se presentan las principales interfaces que provee la aplicación Web de consulta de datos.

5.2.1. Página principal

La página principal es la vista desde la cual se pueden realizar todas las consultas implementadas por el sistema. La Figura 5.4 muestra la barra de búsqueda de la página principal y los 4 tipos de consultas que se pueden realizar: la búsqueda de autores, la búsqueda por keywords, la búsqueda de revisores y la búsqueda de caminos entre autores.

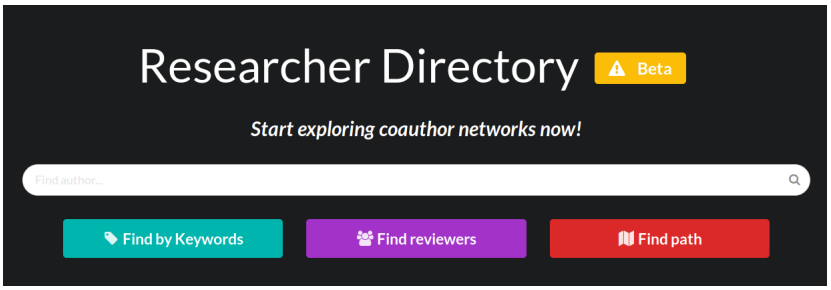

Figura 5.4: Barra de búsqueda de la página principal

5.2.2. Perfil de autor

El primer tipo de consulta disponible para los usuarios es la búsqueda de autores por nombre, mediante las barras de búsqueda de la página principal y del menú superior. Estas barras tienen una función de autocompletado que permite identificar al autor buscado desde la lista de resultados posibles.

Al seleccionar un autor, el usuario es redirigido a una vista de perfil de autor, la cual organiza la información recopilada en una barra lateral y 5 pestañas: **Network**, **Coauthors**, **Publications**, **Explore** y **Additional Info**, como se aprecia en la Figura 5.5.

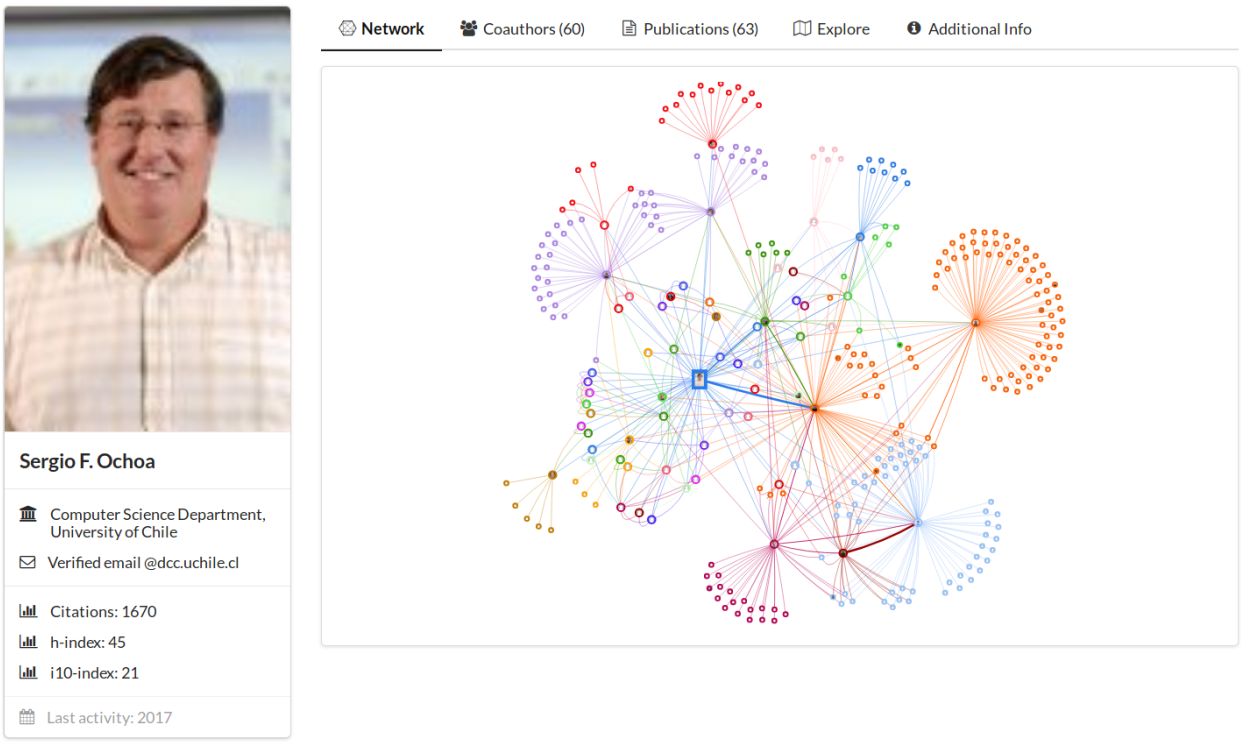

Figura 5.5: Ventana de perfil de un autor

La barra lateral muestra los datos personales del autor, acompañados de su foto de perfil. Estos datos incluyen el ORCID, la última afiliación conocida, el dominio del correo electrónico, los índices de citaciones y la última actividad, en un grado de completitud que varía de autor a autor según la información disponible en las fuentes.

La pestaña **Network** muestra la red de coautores de primer y segundo orden (es decir, coautores a distancia 1 y 2), mediante un grafo cuya construcción fue detallada en la sección anterior. La pestaña **Coauthors** muestra la información resumida de los coautores de primer orden (nombre, número de colaboraciones, índices de citaciones, última actividad), organizada en una tabla ordenable por columnas y con búsqueda por nombre. La pestaña **Publications** contiene la lista de publicaciones (es decir, artículos de *journal*, artículos de conferencia y *proceedings*) del autor, organizadas por año de publicación. La pestaña **Explore** contiene una serie de consultas que permiten explorar la red de coautores, como búsqueda por keywords o por distancia. Finalmente, la pestaña **Additional Info** contiene información adicional sobre

el autor, como son los alias conocidos (otros nombres bajo los que esta persona publica), las afiliaciones conocidas y las keywords de los temas en los que ha trabajado.

La información presentada en las distintas secciones puede además ser filtrada por un rango de fechas o por keyword (Figura 5.6), generando así nuevas visualizaciones a partir de las entidades que coincidan con el filtro aplicado. La Figura 5.6 muestra el resultado de filtrar el perfil de la Figura 5.5 por fecha y keyword con los valores que se muestran en el formulario.

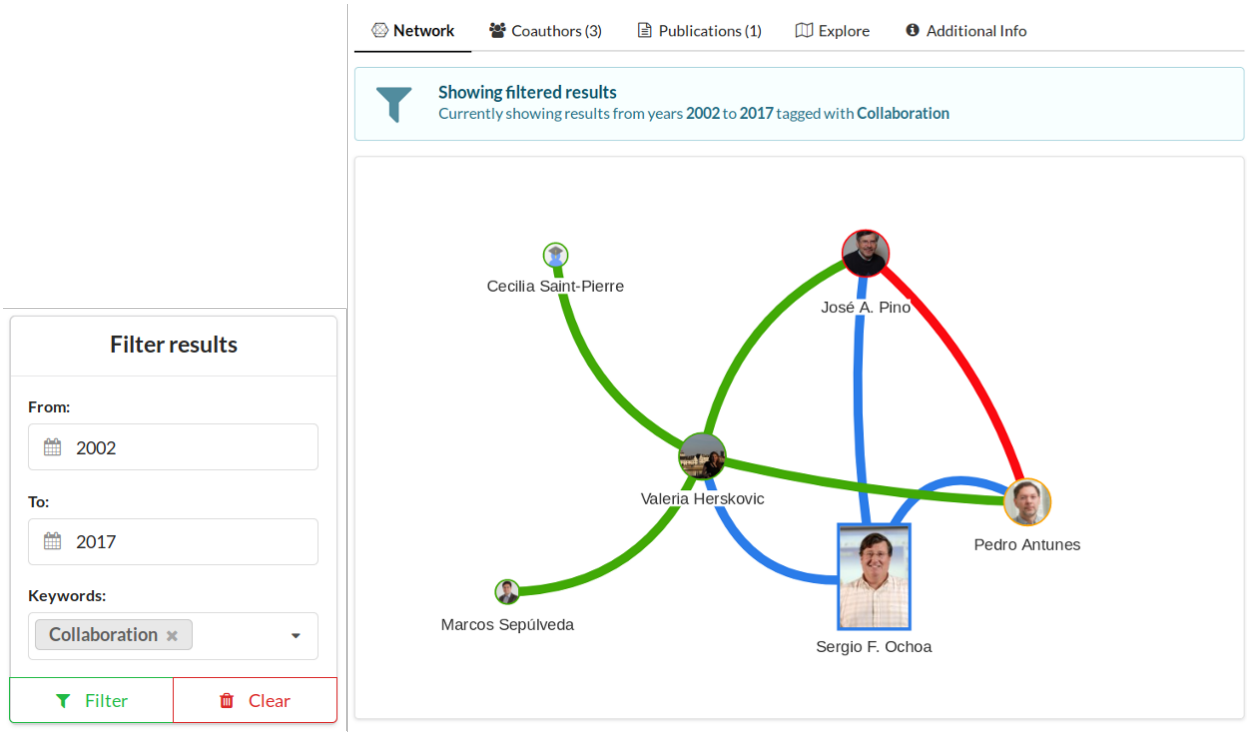

Figura 5.6: Ejemplo de filtros aplicados a un perfil de usuario

5.2.3. Búsqueda por keywords

El segundo tipo de búsqueda es la búsqueda por keywords. Esta consulta permite buscar autores por los temas de investigación en los que trabajan, obtenidos a través de las keywords de los artículos que han publicado. La Figura 5.7 muestra una búsqueda con los términos ***"Agile Software Development"*** y ***"Mobile Applications"***.

Los autores encontrados como resultado de la búsqueda son organizados en una tabla, mostrando parte de su información personal (índices de citaciones, última actividad) y la última actividad en los temas de investigación seleccionados. La Figura 5.8 muestra parte de los resultados encontrados para la consulta de la Figura 5.7

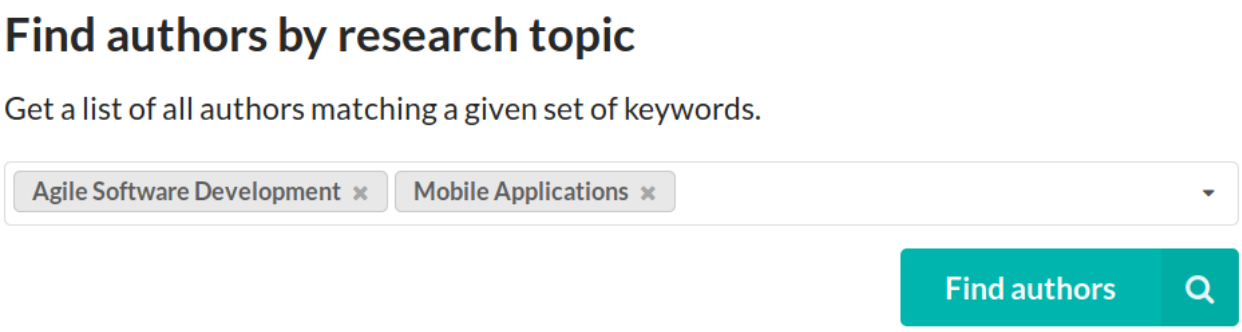

Figura 5.7: Ejemplo de búsqueda por keywords

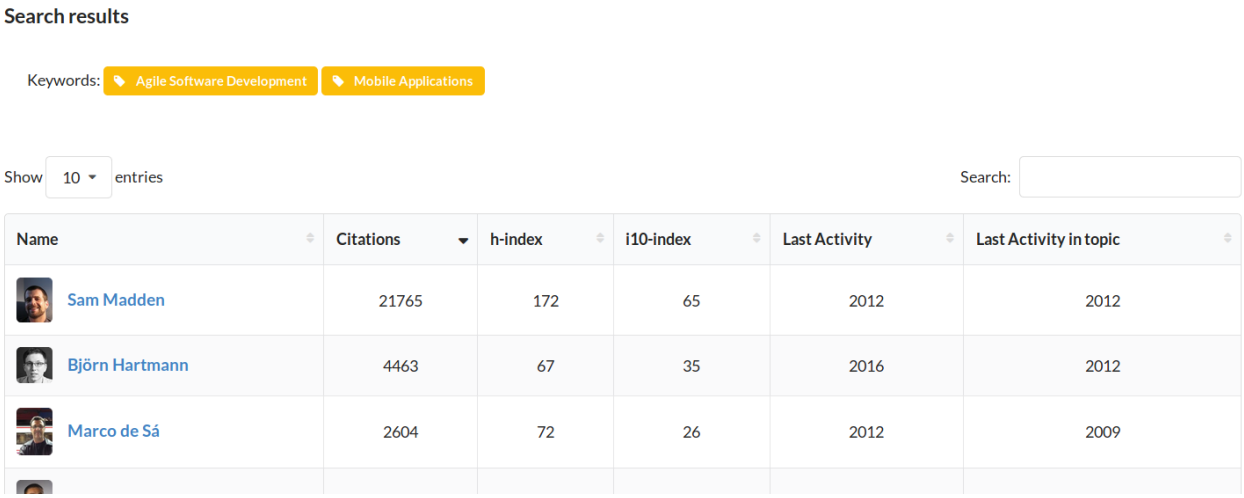

Figura 5.8: Resultado de búsqueda por keywords

5.2.4. Búsqueda de revisores

El tercer tipo de consulta implementado es la búsqueda de revisores para un artículo o proyecto, a partir de la lista de autores, el conjunto de keywords que lo describe y una distancia mínima. Los candidatos a revisores serán todos aquellos autores que tengan al menos una de las keywords dadas, y que estén a una distancia mayor a la mínima especificada por el usuario.

La Figura 5.9 muestra la consulta realizada para encontrar revisores para el artículo titulado ***"Bribecaster: documenting bribes through community participation"***, dadas sus respectivas keywords y autores y la distancia mínima a la que deben estar los revisores. La Figura 5.10 muestra parte de los resultados encontrados para dicha consulta.

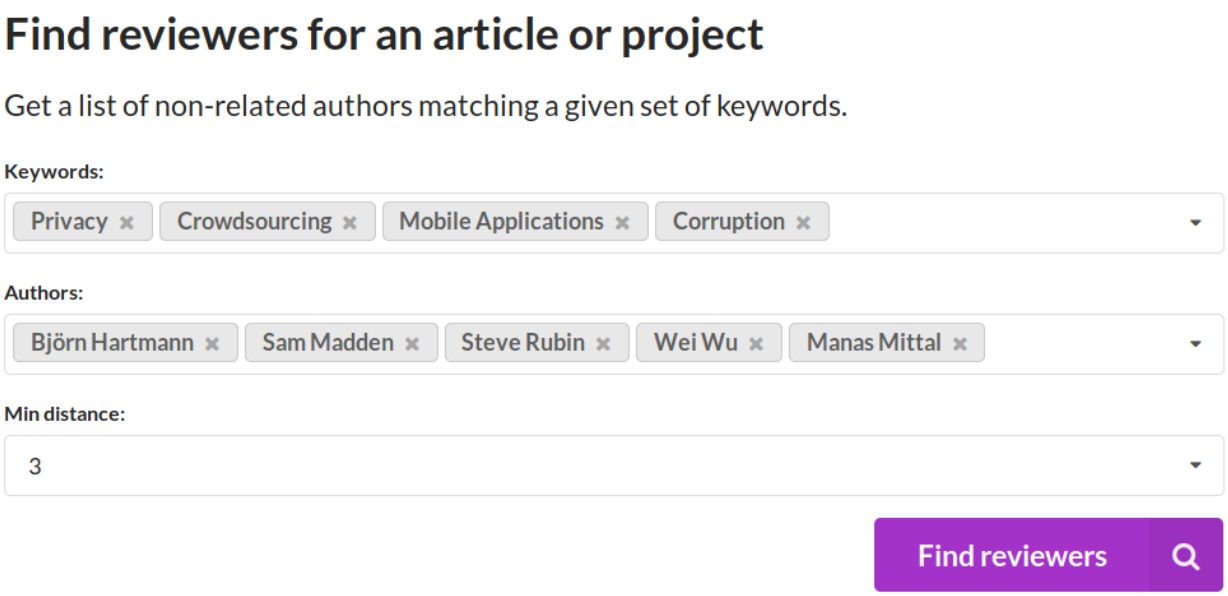

Figura 5.9: Ejemplo de búsqueda de revisores

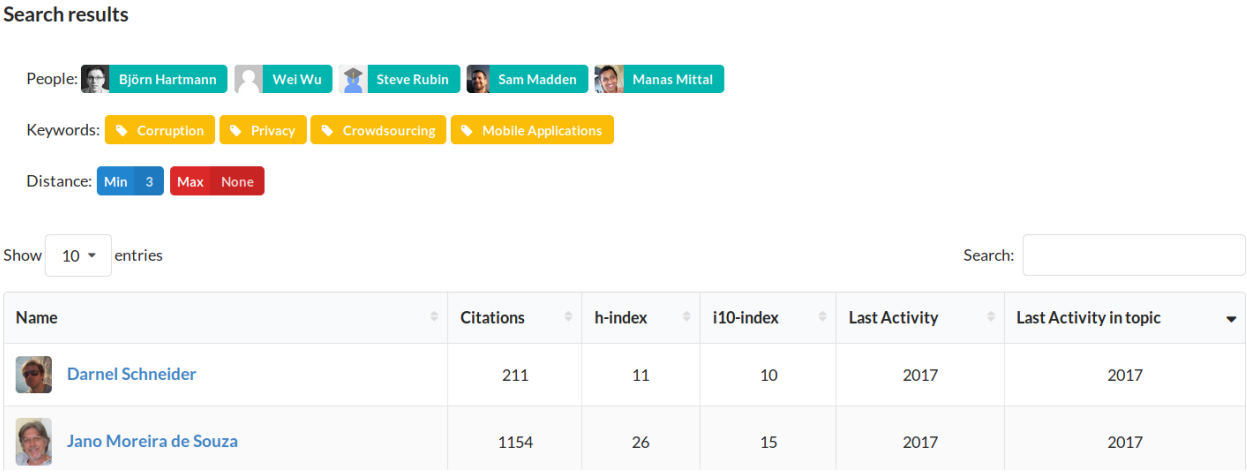

Figura 5.10: Resultado de búsqueda de revisores

5.2.5. Búsqueda de caminos

El último tipo de consulta corresponde a la búsqueda de caminos entre autores. Esta consulta encuentra el camino más corto (siguiendo relaciones de coautoría) entre dos autores dados, detallando los artículos que conectan a los autores en cada paso. La Figura 5.11 muestra el detalle del camino ilustrado en el grafo de la Figura 5.3, indicando los artículos que generan las relaciones de coautoría entre cada autor.

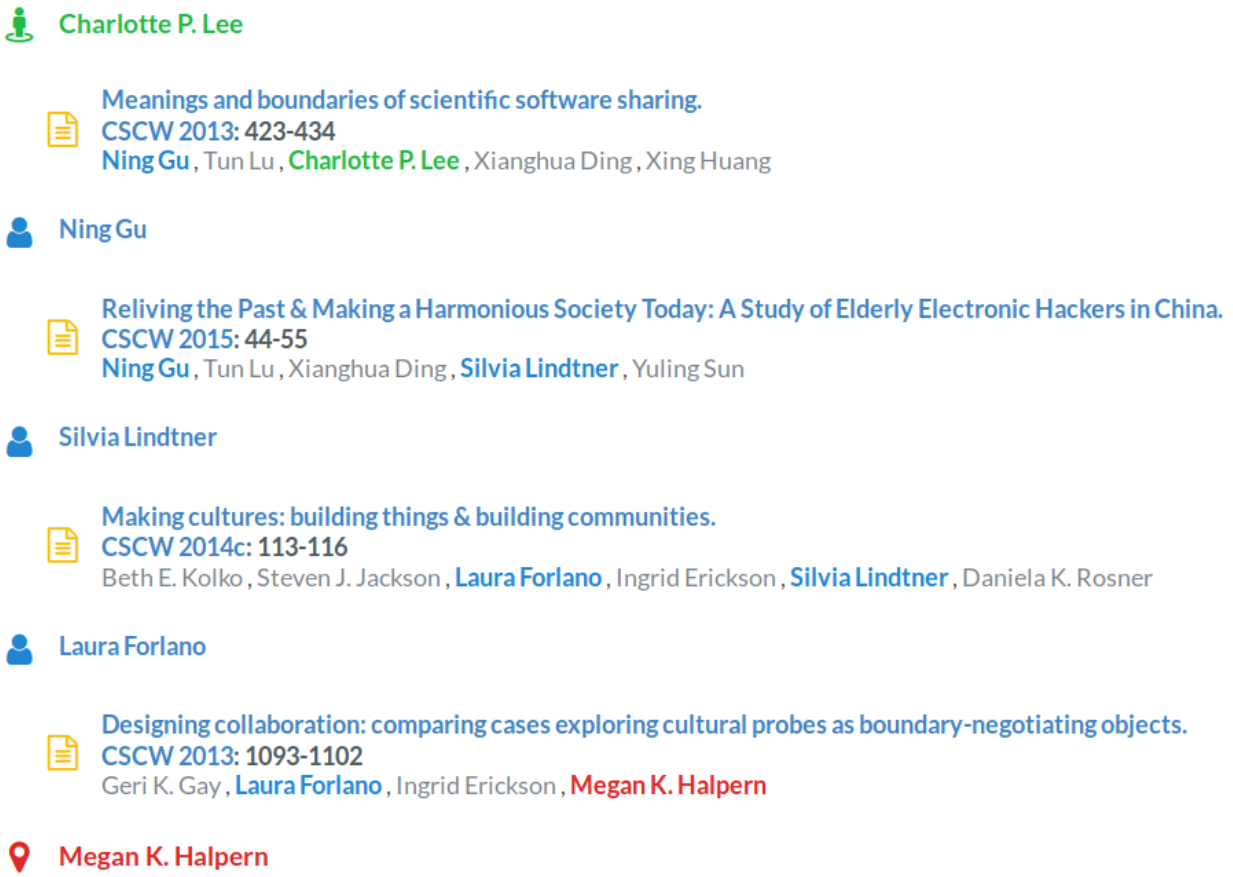

Figura 5.11: Resultado de búsqueda de revisores

Capítulo 6

Validación de la solución

Para validar la solución se realizó una sesión de demostración de la plataforma, a un grupo de posibles usuarios finales, todos académicos del DCC con experiencia en el uso de herramientas o fuentes de información similares. En la prueba se utilizó un *dataset* conformado por 94 proceedings de cinco conferencias del área de sistemas colaborativos (CRIWG, CSCWD, CSCW, ECSCW y Group), donde los evaluadores usualmente participan.

Con el fin de presentar la aplicación Web de consulta del repositorio, y recibir retroalimentación sobre temas como usabilidad y utilidad de sus servicios, se definió un formulario de evaluación (disponible en el Anexo A) que fue respondido por los asistentes al final de la sesión de demostración. El proceso de evaluación consideró, en primera instancia, la realización de 4 actividades que le fueron entregadas a los evaluadores, y ellos debían indicar cómo realizarlas utilizando la herramienta. La forma de llevar a cabo las cuatro actividades fue evidente para los participantes. Luego de eso tuvieron la oportunidad de idear requerimientos de consulta y de resolverlos usando la aplicación, lo cual también resultó en forma exitosa.

Finalmente, los participantes completaron el formulario de evaluación, en cual tiene una serie de afirmaciones que deben ser respaldados o refutadas utilizando una escala Likert [19] (indicando un valor entre **"Muy de acuerdo"** y **"Muy en desacuerdo"**). Los resultados obtenidos mostraron que el sistema tiene:

- **Baja dificultad de uso.** Los usuarios entienden la estructura del sistema, saben cómo navegar por las interfaces y se sienten capaces de utilizar la aplicación para realizar consultas por su cuenta.

- **Alta utilidad.** Los usuarios reconocen la utilidad del sistema. Las consultas existentes generan información útil a partir de los datos recopilados.

- **Alto potencial.** Los usuarios ven la plataforma como una herramienta que podrían utilizar, ofreciendo sugerencias de mejoras que les gustaría ver implementadas en el futuro.

Además de evaluar las afirmaciones mencionadas, se les solicitó a los asistentes expresar sus comentarios y sugerencias, entre las que destacan:

- La posibilidad de implementar un sistema de **ranking** para los posibles revisores, con el fin de orientar al usuario en la búsqueda del autor ideal entre todas las posibilidades.

- La necesidad de implementar *features* que atraigan a usuarios de otras plataformas y convencerlos de utilizar este servicio, en especial a aquellos ya acostumbrados a utilizar otras fuentes de datos.

- Evaluar los efectos que tendrá la carga total de los datos en la *performance* de la aplicación, además de un eventual cambio en la arquitectura de la solución a un enfoque más distribuido, según sea necesario.

Capítulo 7

Conclusión y trabajo a futuro

7.1. Conclusión

Actualmente, los sitios e instituciones dedicados a recopilar información sobre el estado de la comunidad de investigadores en ciencias de la computación están enfocados en los artículos que ésta produce más que en sus integrantes. Esto dificulta la generación de instancias de colaboración entre ellos, ya que además de la asistencia a eventos o conferencias comunes y recomendaciones entre conocidos, no existe forma fácil (o gratuita) de identificar comunidades activas en torno a un tema, contactar autores o identificar autores y los posibles conflictos de interés entre ellos. Además, la gran mayoría de los servicios interesantes para la comunidad, por ejemplo, la generación estadísticas o la búsqueda de revisores o potenciales colaboradores, están accesibles a través de un pago o una suscripción a las organizaciones que mantienen estos repositorios de información.

En este trabajo de memoria se diseñó e implementó un sistema de recopilación de información de autores en ciencias de la computación desde múltiples fuentes, junto a una aplicación web que permite explorar los datos recolectados y generar nueva información a partir de éstos. Este trabajo representa una prueba de concepto, que muestra la factibilidad técnica, operativa y eventualmente económica de desarrollar una solución definitiva para atacar los problemas mencionados anteriormente.

En primer lugar se realizó una investigación sobre las fuentes de datos existentes y la información disponible en cada una de ellas, además de sus respectivos métodos de extracción. Esta investigación estuvo enfocada en fuentes con libre acceso a la información, descartando aquellas con *paywalls* y otras restricciones en el acceso a los datos.

Una vez identificadas las fuentes de información a utilizar, se implementaron los respectivos sistemas de recopilación automática, junto a los procesos de resolución de conflictos necesarios para integrar los datos provenientes de cada fuente en una misma entidad. Se generó, además, un modelo de datos que expande el esquema utilizado por DBLP (el cual está basado en entradas de BibTeX) al agregar entidades para los autores, instituciones y temas de investigación, reorganizando la información almacenada pero manteniendo la semántica

de las relaciones existentes entre entidades.

En tercer lugar se definieron consultas a responder con la información recopilada, con el fin de implementar herramientas que ayuden a aliviar los problemas ya mencionados anteriormente. Estas consultas se definieron a modo de prueba de concepto, para evaluar la utilidad que el sistema podría entregar a los usuarios finales.

Finalmente, se implementó una aplicación web mediante la cual los usuarios pueden realizar consultas sobre los datos recopilados y utilizar las visualizaciones construidas para explorar los resultados encontrados.

Para comprobar la utilidad y usabilidad del sistema, se realizó una validación con un grupo de potenciales usuarios finales, quienes, tras una demostración, realizaron actividades representativas de los casos de uso y corroboraron la utilidad del sistema y el potencial que una herramienta de este tipo puede tener.

Si bien el sistema es capaz de recopilar la información desde las distintas fuentes y responder las consultas correctamente, una de las principales preocupaciones a la hora de implementar el sistema en su totalidad es la escalabilidad. Durante el desarrollo del trabajo de memoria se trabajó con un subconjunto pequeño de los datos (10.000 artículos, 15.000 autores), con el cual el sistema se comportó de buena manera, sin presentar casos de lentitud considerable al evaluar las consultas, pero se desconoce el comportamiento que tendrá al cargar la totalidad de los datos y el volumen de peticiones que será capaz de procesar una vez puesto a disposición de la comunidad. Es necesario, entonces, evaluar un eventual cambio en el motor de base de datos para manejar grandes volúmenes de datos (por ejemplo, pasar de la versión Comunitaria a la Empresarial) y en la arquitectura utilizada para manejar la cantidad de peticiones realizadas al servidor (por ejemplo, implementar el sistema de forma distribuida).

Cabe recordar que la aplicación web fue construída como prueba de concepto de la posible utilidad de la información recolectada y no como principal foco de esta memoria. Además, dada la estructura utilizada al implementar la solución, el sistema de recopilación de información es totalmente independiente de la aplicación web y del sistema de base de datos utilizado, por lo que puede ser adaptado con facilidad a una nueva arquitectura en caso de ser necesario.

Otro problema de escalabilidad que surge es el costo (en tiempo) del procesamiento de los datos. Pruebas de *profiling* mostraron que entre un 80 y un 95 % del tiempo (dependiendo de la máquina utilizada y la calidad de la conexión a internet) que toma procesar cada entidad, es decir, parsear sus atributos y complementar con otras fuentes, corresponde a las consultas a las APIs de Crossref y Mendeley. Si bien el tiempo puede ser manejable para cantidades reducidas de entidades (como el subconjunto utilizado durante el desarrollo), una estimación aproximada extrapolada desde el tiempo que tomó cargar los datos utilizados durante las pruebas muestra que el proceso de cargar la totalidad de los datos de DBLP y complementarlos con la metadata obtenida desde Crossref y Mendeley tomaría alrededor de 400 horas en completarse. Este costo, sin embargo, corresponde sólo a la carga inicial; las actualizaciones posteriores tardarán considerablemente menos tiempo, ya que sólo se procesan las entidades que han sido modificadas en el último tiempo. Aún así, es importante

evaluar la necesidad real de utilizar Crossref y Mendeley, por lo que se propone evaluar una medida de conformidad con la cantidad de información recopilada que permita realizar las consultas a los sistemas adicionales sólo cuando se estime que la información obtenida para una entidad dada no sea suficiente, reduciendo así el número de consultas y el costo en tiempo del procesamiento de cada entidad.

Para concluir, se puede decir que el sistema implementado permite corroborar la factibilidad y utilidad de un sistema enfocado en los autores y sus relaciones con el medio, dando un paso inicial hacia la construcción de una alternativa real a las opciones utilizadas por los usuarios finales actualmente y hacia la disminución del tiempo y trabajo manual requerido por las actividades que se busca apoyar, pero que aún presenta desafíos a resolver en futuras iteraciones antes de ser puesto a disposición de la comunidad.

7.2. Trabajo a futuro

A continuación se plantean posibles mejoras al sistema, las cuales fueron consideradas en este trabajo pero no incluidas en el desarrollo de la memoria:

- **Uso de ORCID como identificador de autores.** Actualmente, el sistema incorpora la notación utilizada por DBLP para reconocer los múltiples nombres bajo los que publica una persona. Sin embargo, durante el desarrollo de esta memoria, el uso de ORCID como identificador de investigadores ha estado ganando terreno, al punto de que muchos *publishers* han empezado a requerir este ID a la hora de recibir un artículo. En la medida que ORCID siga siendo adoptado por más servicios, el problema de recuperar la metadata de un autor desde distintas fuentes se hace más fácil, simplificando el proceso de recopilación de datos.

- **Implementación de un ranking para los revisores.** Se plantea la posibilidad de implementar una métrica que ayude a identificar los mejores candidatos a revisores, basándose en parámetros como los temas de investigación seleccionados, la última actividad en el área, el número de publicaciones en el área o la distancia a los autores dados.

- **Etiquetado de artículos en base al título o abstract.** Se plantea la posibilidad de implementar un sistema de extracción de keywords desde el texto de una publicación (título o abstract) para los casos donde no se tenga clasificación alguna; en particular, para aquellos casos donde los autores no declararon ninguna keyword al momento de publicar un artículo.

- **Ubicación geográfica de los autores.** Una de las principales motivaciones detrás de la elaboración del directorio fue la posibilidad de tener una lista con todos los investigadores y sus temas de investigación por cada país. Actualmente, esta información no se encuentra disponible en ninguna de las fuentes utilizadas, siendo las afiliaciones los únicos indicios del lugar de residencia del autor.

- **Elaboración de un diccionario de keywords.** Actualmente, el sistema inserta directamente las keywords recibidas desde Crossref, Mendeley y Google Scholar con un mínimo pre-procesamiento (remover puntuación y pasar a minúsculas). Esto lleva a la existencia de múltiples duplicados para cada término, dadas las diferencias en detalle, sinónimos, pluralidad, etc. Se plantea la posibilidad de implementar un sistema de mapeo que transforme las keywords recibidas desde los distintos servicios, a keywords pertenecientes a un conjunto pre-definido, con el fin de mejorar el sistema de búsqueda de usuarios y filtros por temática abordada.

Bibliografía

[1] ACM.org: ACM Digital Library. http://dl.acm.org/dl.cfm. Última consulta: Enero 2018.

[2] Crossref.org. https://www.crossref.org/. Última consulta: Enero 2018.

[3] Crossref.org: Crossref REST API. http://api.crossref.org/. Última consulta: Enero 2018.

[4] DBLP.org: Computer science bibliography. http://dblp.org/. Última consulta: Enero 2018.

[5] DBLP.org: Statistics. http://dblp.dagstuhl.de/statistics/. Última consulta: Enero 2018.

[6] Elsevier.com: An information analytics company. https://www.elsevier.com/. Última consulta: Enero 2018.

[7] Elsevier.com: Developer Portal. https://dev.elsevier.com/. Última consulta: Enero 2018.

[8] Flask: A python microframework. http://flask.pocoo.org/. Última consulta: Enero 2018.

[9] Google.com: Google Scholar. https://scholar.google.com. Última consulta: Enero 2018.

[10] IEEE.org: IEEE. http://www.ieee.org/index.html. Última consulta: Enero 2018.

[11] Mendeley.com. https://www.mendeley.com/. Última consulta: Enero 2018.

[12] Mendeley.com: Developer Portal. http://dev.mendeley.com/. Última consulta: Enero 2018.

[13] Neo4j.com: The neo4j graph platform. https://neo4j.com/. Última consulta: Enero 2018.

[14] Neomodel documentation. https://neomodel.readthedocs.io/. Última consulta: Enero 2018.

[15] OpenDataCommons.org: ODC Attributions Summary. https://opendatacommons.org/licenses/by/summary/. Última consulta: Enero 2018.

[16] Springer.com: Springer Link. https://link.springer.com/. Última consulta: Enero 2018.

[17] VisJS.org: A dynamic, browser based visualization library. http://visjs.org/. Última consulta: Enero 2018.

[18] WebOfKnowledge.com: ISI Web of Science. https://apps.webofknowledge.com/. Última consulta: Enero 2018.

[19] Wikipedia.org: Escala de likert. https://es.wikipedia.org/wiki/Escala_Likert. Última consulta: Enero 2018.

Anexo A

Formulario de evaluación

ACTIVIDADES A REALIZAR

1. Dado un autor, recuperar sus coautores y sus publicaciones.
2. Dadas 2 keywords, determinar la comunidad y los investigadores activos en el área en los últimos 2 años.
3. Determinar posibles revisores para el siguiente artículo (descrito a través de sus autores y keywords):

   Nelson Baloian, Jonathan Frez, José A. Pino, Gustavo Zurita. Supporting collaborative decision making in geo-collaboration scenarios. In *Proceedings of CRIWG 2015*: 63-71, 2015

   Keywords: Collaborative decision making, Geo-collaboration
4. Dado un autor y un posible revisor, determinar el potencial conflicto de interés entre ellos basándose en la distancia de coautorías.

EVALUACIÓN: Marque con una X la opción que considere más adecuada.

Ítem	Muy en desacuerdo	En desacuerdo	Indeciso	De acuerdo	Muy de acuerdo
La estructura de la interfaz de usuario es adecuada					
No está claro el significado de los íconos que se muestran en pantalla.					
La aplicación es lenta.					
Me siento capaz de utilizar la aplicación.					
La aplicación es de poca utilidad.					
La aplicación tiene mucho potencial.					

Ítem	Muy en desacuerdo	En desacuerdo	Indeciso	De acuerdo	Muy de acuerdo
Fue difícil realizar las tareas solicitadas con la herramienta.					
Está claro cómo navegar la aplicación y acceder a sus componentes.					
Cualquier estudiante o investigador podría ser capaz de utilizar esta aplicación.					

Indique comentarios, sugerencias de mejora o errores que deberíamos corregir: