Universidad Politécnica de Madrid

Escuela Técnica Superior de Ingenieros Informáticos

Grado en Ingeniería Informática

Trabajo Fin de Grado

Análisis del decaimiento de información en internet: El caso de los recursos científicos.

Autor: SARA SUSANO RUIZ
Tutor(a): OSCAR DIESTE TUBIO

Madrid, Junio 2024

Este Trabajo Fin de Grado se ha depositado en la ETSI Informáticos de la Universidad Politécnica de Madrid para su defensa.

Trabajo Fin de Grado
Grado en Ingeniería Informática

Título: Análisis del decaimiento de información en internet: El caso de los recursos científicos.

Junio 2024

Autor: SARA SUSANO RUIZ
Tutor: OSCAR DIESTE TUBIO
Departamento de Lenguajes y Sistemas Informáticos e Ingeniería de Software
Escuela Técnica Superior de Ingenieros Informáticos
Universidad Politécnica de Madrid

Resumen

Este Trabajo de Fin de Grado trata de explorar la desaparición que se produce de la información en internet. Tiene el objetivo de poder realizar un análisis de cómo va desapareciendo dicha información e intentar descubrir el por qué. Para ello, se utilizó una aplicación, desarrollada por ciclos.

Dicha aplicación, se encarga de extraer de un fichero con terminación XML, de la biblioteca digital *DBLP*, los títulos de los artículos contenidos en él. Una vez extraídos se encarga de buscar, gracias a SerpApi de Google Scholar, dichos artículos en formato PDF para posteriormente descargarlos. Una vez descargados, se extráen los enlaces contenidos en el apartado de referencias de dichos PDFs y se verifica su existencia, guardando los datos obtenidos en una base de datos. Esto último, se realiza para poder llevar a cabo las estadísticas sobre cómo desaparece la información que se encuentra recogida en Internet.

Los resultados de los análisis realizados mostraron que cuanto mayor es el tiempo de publicación del artículo, más propensa es la información a desaparecer. Además, hay relación entre los servicios online donde se almacena la información y su desaparición, como repositorios o servicios cloud que son personales.

Estos resultados sugieren que se deben de tomar medidas para poder mantener la información disponible en la red, como almacenarla en sitios web oficiales. Además, todavía es necesario mejorar las técnicas de extracción y manipulación de las URLs para que el análisis pueda ser lo más verídico posible.

En este documento se abordan el diseño de la aplicación, junto con sus ciclos de desarrollo, los antecedentes utilizados, los objetivos a cumplir, los resultados que se han obtenido de los análisis realizados y el trabajo futuro que se debe de realizar.

Abstract

This document explores the phenomenon of information disappearance on the internet. Its objective is to conduct an analysis of how this information disappears and attempt to uncover the reasons behind it. To achieve this, an application was used, developed through iterative cycles.

This application is responsible for extracting titles of articles from a file with XML extension, obtained from the digital library *DBLP*. Once they are extracted, it searches for these articles in PDF format using SerpApi from Google Scholar, and later it downloads them. After downloading, it extracts the links that are in the references section of these PDFs and verifies their existence, storing the obtained data in a database. This is done in order to conduct statistics on how the information collected from the internet disappears over time.

The results of the conducted analyses showed that the longer the publication time of the article, the more likely the information is to disappear. Furthermore, there is a relationship between online services where information is stored and its disappearance, such as personal repositories or cloud services.

These results suggest that measures should be taken to keep information available on the web, such as storing it on official websites. Additionally, there is still a need to improve techniques for extracting and manipulating URLs so that the analysis can be as accurate as possible.

This document addresses the design of the application, along with its development cycles, the background used, the objectives to be achieved, the results obtained from the analyses performed, and the future work to be done.

Tabla de contenidos

Índice de figuras

Capítulo 1

Introducción

La investigación científica por lo general, se apoya bastante en Internet. Los distintos recursos, como los artículos o los análisis, son publicados en la red en distintos servicios, como repositorios o servicios cloud entre otros, no siempre de forma voluntaria.

Estos recursos se identifican mediante *URIs* (Uniform Resource Identifiers), más concretamente se suelen utilizar *URNs* (Uniform Resource Names) y los *URLs* (Uniform Resource Locators).

Un URN es un *"persistent, location-independent resource identifier"*. Un ejemplo bien conocido de URN es el *Digital Object Identifier* (DOI).

Los URIs más frecuentemente utilizados son los URLs, los cuales definen el protocolo de acceso, servidor y puerto donde los recursos están accesibles, aunque, nada garantiza que el recurso esté disponible en el momento de acceso.

Los URNs no indican la localización del recurso al que identifican, por lo que es necesario, realizar una traslación URN $\rightarrow$ URL. En consecuencia, los URNs también pueden estar no disponibles.

Se conoce como *linkrot* o *link decay* a *"the disassociation between web addresses and their content"*. Los motivos de esta disociación pueden ser muy variados. Por ejemplo, el propietario de un recurso podría desear eliminarlo de Internet. Otras razones por los que una URL puede no ser accesible son: reorganización de los recursos en los servidores son proporcionar una redirección, gestión inadecuada de los recursos, servidores o dominios que desaparecen, etc. [11].

La situación se exacerba cuando los recursos son publicados en servicios cuyo objetivo no es la preservación, sino el almacenamiento y la compartición de archivos, como por ejemplo, el uso de Dropbox o Google Drive. El acceso a un archivo compartido se pierde cuando es movido de posición, borrado, o el usuario cancela su cuenta en dichos servicios Cloud. Por este motivo, se recomienda evitar este tipo de servicios para la preservación de recursos científicos [12].

El *linkrot* es un problema importante, ya que, se ha observado que, al año siguiente de la publicación, un 10 % de los links que aparecen en artículos científi-

ficos ya no están disponibles [11, 13], y dicho porcentaje aumenta con el tiempo [14].

Este trabajo tiene como objetivo el desarrollo de una aplicación. Dicha aplicación debe ser capaz de poder acceder a los artículos en formato PDF de un fichero extraído de la biblioteca digital DBLP. En los PDF, debe poder extraer las URL que se encuentran en el apartado de referencias. Por último, debe comprobar si las URLs extraídas existen o no para, poder realizar un análisis de la desaparición de la información en Internet.

Para poder lograr el objetivo se utiliza como método el desarrollo incremental iterativo. Esto significa que en cada ciclo de desarrollo, se van introduciendo nuevas funcionalidades y a la vez se van añadiendo mejoras sobre aquellas que ya estaban implementadas previamente.

La aplicación es capaz de buscar un enlace para poder obtener un artículo en formato PDF. Además, lo descarga, únicamente si no ha sido descargado previamente, y lee el contenido del mismo. Tras leer el contenido, es capaz de identificar y extraer las URLs que aparecen en las referencias de dicho PDF, además de verificar su existencia. Por último, una vez se ha realizado el análisis, cuenta con la capacidad de exportar los datos obtenidos para poder realizar estadísticas sobre los mismos y así, efectuar un análisis de la pérdida de información en internet.

En el siguiente repositorio se encuentra el código de desarrollo de la aplicación: Repositorio.

Capítulo 2

Antecedentes

En este capítulo se tratan las herramientas y los aspectos claves que han jugado un papel fundamental en el desarrollo del proyecto.

2.1. Maven [1] [2]

Apache Maven se trata de una herramienta que sirve para la gestión y la comprensión de proyectos de software. Su enfoque se basa en el Modelo de Objeto de Proyecto (POM), permitiendo a Maven gestionar la construcción, informes y documentación de un proyecto a través de una fuente centralizada de información.

Maven desempeña funciones cruciales como la compilación, el empaquetamiento y la ejecución de los test. Se basa en un fichero denominado pom.xml, donde se definen todos los elementos esenciales para el proyecto, tales como la gestión de dependencias, las cuales se descargan e incorporan al classpath al definirlas. Esta herramienta cuenta además con unos arquetipos, los cuales son los encargados de crear la estructura del proyecto , los datos contenidos en pom.xml, la estructura utilizada por las carpetas y los ficheros que se incluyen por defecto. Estos arquetipos facilitan la creación inicial del proyecto al definir automáticamente aspectos fundamentales.

En el marco de este proyecto, se ha utilizado para organizar el código de la aplicación de manera efectiva. Resulta beneficioso tener las clases debidamente estructuradas y las bibliotecas utilizadas gestionadas a través del archivo pom.xml, evitando así la necesidad de descargar el archivo .jar e incorporarlas individualmente.

Figura 2.1: Logo Maven.

2.2. DBLP [3]

DBLP computer science bibliography, o simplemente dblp, es una base de datos bibliográfica destacada por su acceso abierto, eliminando la necesidad de suscripción o registro. Principalmente indexa congresos y revistas académicas aunque tiene una variada tipología documental.

La misión de dblp es respaldar a los investigadores del campo informático, ofreciendo acceso de forma gratuita a los metadatos bibliográficos que contienen alta calidad y enlaces que se refieren a ediciones electrónicas de publicaciones.

En el contexto de este trabajo, se ha empleado para obtener los enlaces a los artículos, permitiendo así comprobar si los enlaces en sus referencias siguen existiendo o si su información ha desaparecido.

Figura 2.2: Logo DBLP.

2.3. Google Scholar [4] [5]

Google Scholar es una herramienta de buscador que permite la localización de documentos académicos como por ejemplo artículos, tesis, libros y resúmenes de diferentes fuentes como editoriales pertenecientes a universidades, asociaciones profesionales u otras organizaciones que tienen carácter académico.

Esta herramienta es un buen recurso para la búsqueda en muchas fuentes a la vez, sin embargo, no todos los autores permiten a Google Scholar a acceder a sus artículos por lo que hay parte de la producción científica y académica a la que no se puede acceder desde el buscador.

Google Scholar proporciona información sobre la cantidad de citas que un artículo ha recibido y quiénes son los citante, además de contar con la función de guardar tanto citas como artículos para leerlos posteriormente. Esto lo convierte en la herramienta ideal para completar búsquedas hechas en bases de datos científicas.

En el contexto de este proyecto, Google Scholar es utilizada obtener un acceso gratuitos a los artículos que se van a analizar cuando, al acceder a través de liberías digitales nos encontramos que son de pago.

Figura 2.3: Logo Google Scholar.

2.4. SerpApi [6]

SerpApi es una API gratuita, siempre y cuando no se excedan las 100 búsquedas al mes, que permite obtener información de buscadores como Google de manera fácil y eficiente.

Para poder utilizarla es necesario tener una clave personal API-KEY. Para obtenerla es necesario indicar un correo electrónico y un número telefónico.

Esta herramienta resuelve además los problemas relacionados con el alquiler de proxies, resolver captchas y analizar archivos JSON.

En el contexto de este trabajo, se utiliza para poder manejar los problemas mencionados anteriormente.

2.5. Librería PDFBox [7]

La libería Apache PDFBox es una herramienta de código abierto para el lenguaje de programación JAVA, que se utiliza para trabajar con documentos PDF. Permite la creación de nuevos PDFs, la manipulación de aquellos existentes y tiene la habilidad de extraer el contenido de los mismos. Además, contiene varias utilidades de línea de comandos.

En el contexto del proyecto, se usa para poder extraer el contenido de los PDFs descargados.

2.6. Librería JSOUP [8]

La librería JSOUP es una herramienta para JAVA que simplifica la manipulación de HTML y XML. Ofrece una API fácil de usar para la obtención de URLs, análisis de datos, extracción y manipulación usando métodos DOM API, CSS, y selectores xpath.

JSOUP implementa la especificación denominada *HTML5* de WHATWG y analiza HTML en el mismo DOM en el que lo realizan los navegadores modernos.

Con esta librería se puede:

- Scrapear y parsear HTML de una URL, fichero o cadena de texto.
- Encontrar y extraer datos mediante recorridos del DOM o selectores CSS.
- Manipular elementos HTML, atributos y texto.
- Limpiar el contenido enviado por otros usuarios para prevenir ataques XSS.
- Generar HTML ordenados.

En el contexto del trabajo, se utiliza para poder obtener los títulos de los artículos del archivo xml obtenido de DBLP.

2.7. JSON [9] [10]

JSON es un formato para se utiliza para el intercambio de datos, ya que, destaca por su legibilidad y facilidad.

Está basado en un subconjunto del lenguaje de programación denominado *JavaScript*, aunque, JSON es independiente del lenguaje de programación utlizado. Admite valores de tipo cadena, numérico y booleano, pero no permite valores octales ni hexadecimales, por lo que, es recomendable que los tipos coincidan entre los datos JSON y los objetos de datos correspondientes.

En el contexto del trabajo se utiliza para poder obtener el JSON proporcionado por SerpApi donde se encuentra la url para poder descargar el artículo a analizar en formato PDF.

Capítulo 3

Objetivos y metodología

3.1. Objetivos

A continuación, se exponen los objetivos a conseguir con la aplicación.

Objetivos:

- Poder acceder a los pdfs de los artículos que se encuentran en DBLP.
- En los pdfs ser capaz de poder localizar los enlaces a las referencias utilizadas.
- Acceder a esos enlaces para comprobar si existen, y si no existen estimar el motivo.

3.1.1. Acceso a los pdfs de los artículos

Este objetivo se centra en proveer a los usuarios la información contenida en los artículos, muchos de los cuales son de pago, de forma totalmente gratuita. Esta parte resulta crucial para la investigación científica.

3.1.2. Localizar los enlaces utilizados como referencias

En este caso, el objetivo se centra en poder localizar todas los enlaces que aparecen en el apartado de referencias de los artículos. Esto resulta de mucha utilidad, ya que de esta forma, podemos comprobar si dichos enlaces siguen vigentes en la actualidad o han sido eliminados.

3.1.3. Acceso a enlaces y estimación

Este objetivo se centra en, acceder a los enlaces extraídos del apartado de referencias de los artículos, para poder verificar su existencia. Además, si dichos enlaces ya no existen, el poder estimar el motivo de su desaparición ayuda a poder realizar estadísticas sobre ello.

3.2. Metodología

El desarrollo de la aplicación se basa en ciclos. En el primer ciclo, se realizó una versión básica de la aplicación que cumplía con los objetivos descritos anteriormente.

Esto se consigue recopilando todas las URLs que aparecen en las referencias de los artículos. Una vez las tenemos, se establece una conexión con ellas y se comprueba que el código devuelto por el servidor es un código de éxito (número comprendido entre 200 – 299). Esto se realiza ya que, con un código devuelto satisfactorio se verifica la existencia de la URL.

Para poder llevar a cabo este análisis, la aplicación sigue una serie de pasos. Primero lee la información contenida en un archivo .xml procedente de la librería digital DBLP, para después extraer los títulos de los artículos presentes. A continuación, busca, utilizando SerpApi de Google Scholar [6], una URL que permite la descarga de los artículos en formato PDF, gracias a los títulos extraídos. Una vez se obtienen los enlaces de descarga, se procede a descargar los PDFs solo una vez. Después, se lee el contenido del PDF para poder extraer las URLs que aparecen en las referencias del artículo, procediendo a verificar su existencia.

Se espera que esta aplicación pueda detectar si las URLs, que se toman como referencias en los artículos científicos, existen o no para poder realizar un análisis de cómo va desapareciendo la información que está contenida en Internet.

En el segundo ciclo, se incluyen mejoras para la eficiencia y el rendimiento de la aplicación, como la adicción de una clase que permite ejecutar la aplición seguidamente, y no ejecutar las distintas clases de forma individual.

Además, se especifican una serie de requisitos tanto funcionales como no funcionales que se pueden encontrar en el Anexo A.

Capítulo 4

Plan de trabajo

4.1. Plan de trabajo preliminar

Antes de iniciar con el desarrollo del TFG, se elaboró un Plan de trabajo, el cuál, se comentará a continuación.

4.1.1. Descripción general del trabajo

Consiste en la realización de una aplicación que nos permita ir comprobando si los enlaces utilizados en los artículos, para apoyar o justificar sus conclusiones, siguen existiendo.

Está relacionado con el llamado "Fraude científico".

El objetivo principal es estudiar la cantidad de información científica que se pierde en internet, identificando los mecanismos y el tiempo en el que se producen, de forma automatizada.

Objetivos:

- Poder acceder a los pdfs de los artículos que se encuentran en DBLP.
- En los pdfs ser capaz de poder localizar los enlaces a las referencias utilizadas.
- Acceder a esos enlaces para comprobar si existen, y si no existen estimar su fecha de desaparición.

4.1.2. Lista de tareas

A continuación se muestra una lista con las tareas que debe realizar la aplicación:

- Leer el archivo .xml de DBLP para poder acceder a los artículos.
- Localizar los enlaces a los pdf de los artículos.
- Descargar el pdf de los artículos.

- Ser capaz de leer el contenido del pdf descargado.
- Ser capaz de encontrar los enlaces a las referencias.
- Acceder a los enlaces ver si existen.
- Si no existen los enlaces, estimar cuándo han podido desaparecer.
- Manejar los captchas.
- Manejar la situación de cambio de red pública a red privada y viceversa.
- Tener una caché para que en el caso en el que tengamos que pagar por acceso a un pdf no se vuelva a comprar si ya se ha realizado previamente.

4.1.3. Diagrama de Gantt

Figura 4.1: Diagrama de Gantt - Plan de trabajo preliminar

4.2. Plan de trabajo intermedio

Al alcanzar la mitad del trabajo, el plan de trabajo fue revisado para poder realizar las modificaciones necesarias.

4.2.1. Revisión de la lista de objetivos del trabajo

En cuanto a los objetivos del trabajo se ha añadido el exportar en formato CSV los datos para poder realizar un análisis de lo obtenido a través de la aplicación:

- Poder acceder a los pdfs de los artículos que se encuentran en DBLP.
- En los pdfs ser capaz de poder localizar los enlaces a las referencias utilizadas.
- Acceder a esos enlaces para comprobar si existen, y si no existen estimar su fecha de desaparición.
- Exportar en formato CSV los datos obtenidos.

4.2.2. Revisión de la lista de tareas

- Leer el archivo .xml de DBLP para poder acceder a los artículos.
- Localizar los enlaces a los pdf de los artículos.
- Descargar el pdf de los artículos.
- Ser capaz de leer el contenido del pdf descargado.
- Ser capaz de encontrar los enlaces a las referencias.
- Acceder a los enlaces ver si existen.
- Si no existen los enlaces, estimar cuándo han podido desaparecer.
- Manejar los captchas.
- Manejar la situación de cambio de red pública a red privada y viceversa.
- Tener una caché para que en el caso en el que tengamos que pagar por acceso a un pdf no se vuelva a comprar si ya se ha hecho previamente.
- Obtener un archivo CSV con los datos de la base de datos.
- Analizar los datos contenidos en el CSV y obtener una conclusión.

Podemos ver que se han añadido al final de la lista dos puntos relacionados a la exportación y análisis de los datos obtenidos. Esto se ha producido, ya que, tras obtener los datos y almacenarlos en la base de datos a través de la aplicación, también es necesario realizar un aálisis de esos datos. Esto nos ayuda a buscar la causa del por qué desaparece la información en internet además de poder realizar una conclusión factible de lo mismo.

4.2.3. Revisión del Diagrama de Gantt

En cuanto al diagrama de Gantt, cabe recalcar que solo hay una ligera variación.

Esta variación afecta a la tarea de *Elaborar las estadísticas*. Según el diagrama de Gantt se debería haber empezado con esta tarea el 22/03/2024, sin embargo, comencé con el desarrollo del script en Python para poder exportar los datos el 18/04/2024. Por consiguiente, esto también afectó a la tarea de *Realizar el estudio del decaimiento de la información científica en internet*, ya que se está extendiendo por la mejoras que se quieren introducir hasta calculo 06/05/2024.

Además, la tarea de *Redactar TFG* la empecé el 10/04/2024, cuando estaba indicado que la empezaría el 15/04/2024.

En cuanto al resto de tareas se mantienen los tiempos.

Figura 4.2: Diagrama de Gantt - Plan de trabajo intermedio

4.3. Plan de trabajo final

A medida que el proyecto va avanzando, se han ido introduciendo nuevas mejoras en las tareas listadas anteriormente, que han implicado modificaciones en el diagrama de Gantt.

4.3.1. Revisión del Diagrama de Gantt

En cuanto al diagrama de Gantt, hay una ligera modificación en los tiempos de las tareas *Realizar el estudio del decaimiento de la información científica en internet*, *Elaborar las estadísticas*, *Redactar TFG*, *Realizar ficha resumen del TFG* y *Realizar PowerPoint para la presentación*, debido a la inclusión de mejoras en la aplicación. Además, se añadió una nueva tarea denominada *Inclusión de mejoras para la aplicación*.

Las nuevas fechas para dichas tareas son:

- Realizar el estudio del decaimineto de la información científica en internet: del 06/03/2024 al 30/05/2024
- Elaborar las estadísticas: del 30/05/2024 al 01/06/2024
- Redactar TFG: del 10/04/2024 al 02/06/2024
- Realizar ficha resumen del TFG: del 05/06/2024 al 07/06/2024
- Realizar PowerPoint para la presentación: del 05/06/2024 al 14/06/2024

Figura 4.3: Diagrama de Gantt - Plan de trabajo final

Capítulo 5

Desarrollo

El desarrollo de la aplicación se ha dividido en tres ciclos.

En el primer ciclo se ha desarrollado la aplicación con la funcionalidad básica que permite la extracción y la verificación de la existencia de las URLs obtenidas de la descarga, en formato PDF de un artículo online contenido en la librería digital DBLP.

En el segundo ciclo se han añadido las siguientes funcionalidades:

- Exportación de la base de datos a un fichero con terminación .csv
- Adición de la clase **RunEntireFlow.java** que al ejecutarla se encarga de invocar a las distintas clases que conforman la aplicación.

En cuanto al tercer ciclo, la funcionalidad de la aplicación no ha variado, pero se han incluido una serie de mejoras en cada una de las clases que se especificarán con detalle.

5.1. Ciclo 1

5.1.1. Clases y relaciones entre clases

Al inicio, la aplicación carecía de una base de datos para poder almacenar el título de los PDFs descargados, ni sus URLs tanto accesibles como no:

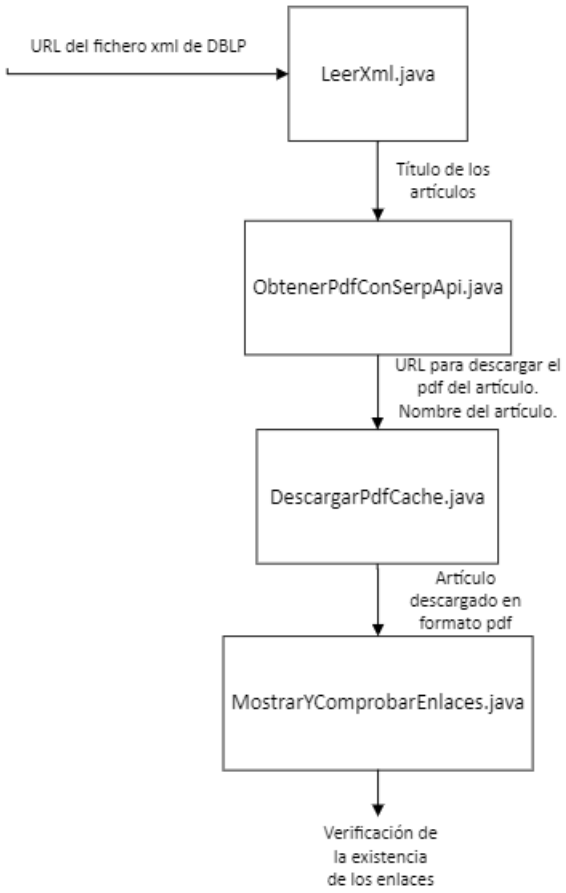


Figura 5.1: Conexión de clases sin base de datos.

Sin embargo, para poder mantener un orden, incluí la base de datos, por lo que agregué una clase más denominada **ConexionBaseDeDatos.java**. Esta clase, permite a la clase **DescargarPdfCache.java** introducir el título de un artículo en formato PDF al descargarlo, y controlar que no se descargue más de una vez. Además, posibilita a la clase **MostrarYComprobarEnlaces.java** introducir las URLs de cada PDF, indicando si son accesibles o no y haciendo referencia a el título del artículo que corresponde para poder identificar a cuál de ellos pertenece.

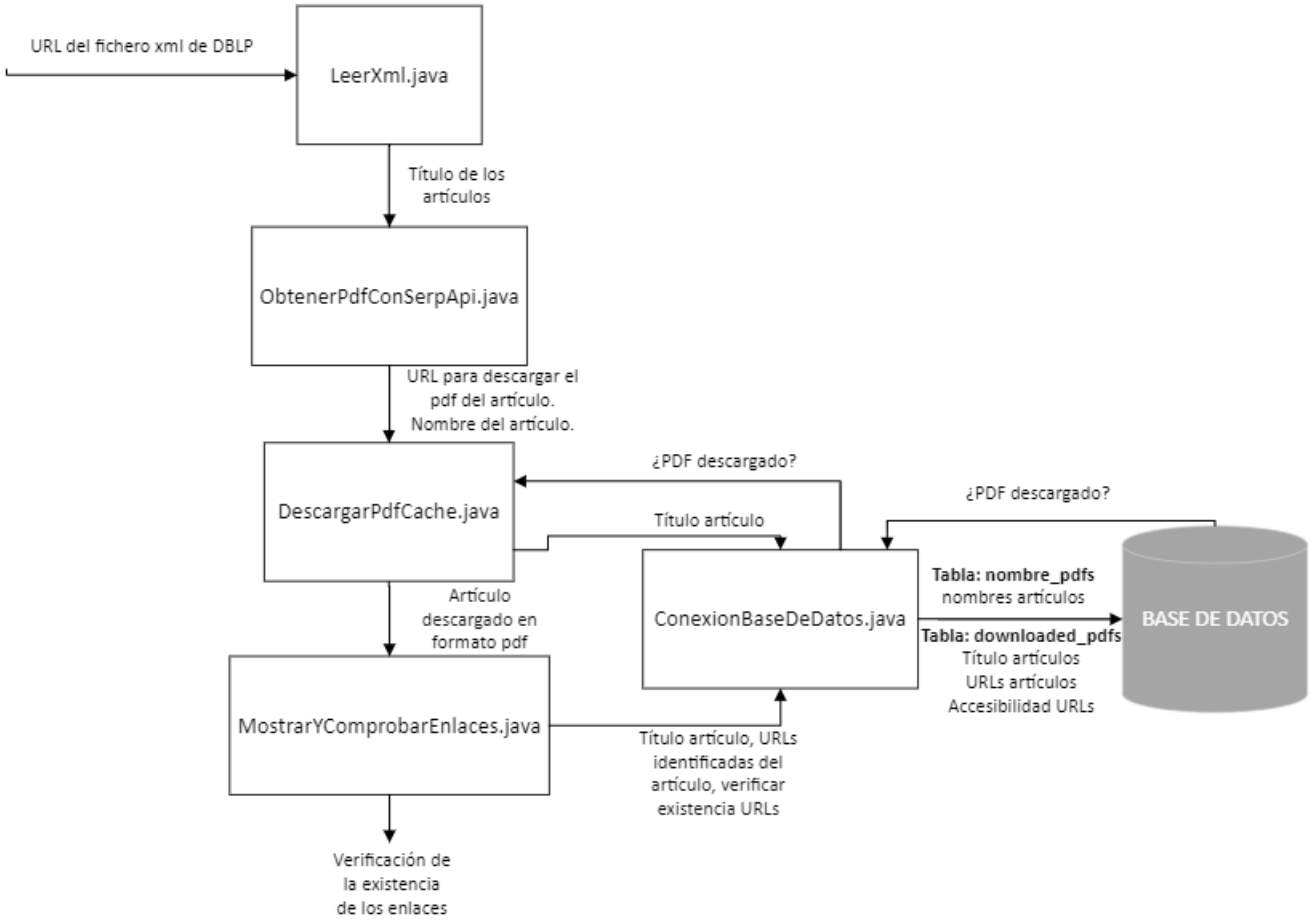


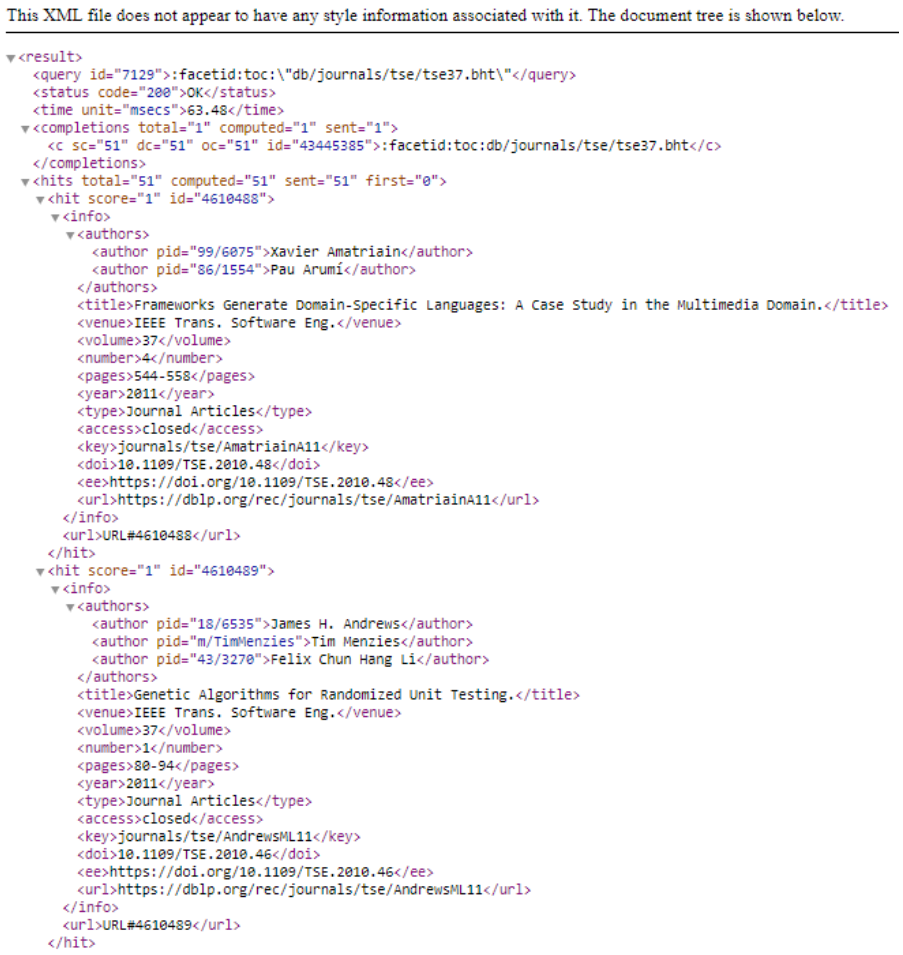
Figura 5.2: Conexión de clases con base de datos.

5.1.2. Ejecución

A continuación, describiré el funcionamiento de la aplicación en relación a ambas conexiones de clases indicadas anteriormente.

Clases sin base de datos

En primer lugar, se ejecuta la clase **LeerXml.java** con la URL del fichero xml obtenido desde la base de datos digital DBLP. Esta clase se encarga de extraer los títulos de los artículos contenidos en el fichero xml. Un ejemplo de fichero xml:



This XML file does not appear to have any style information associated with it. The document tree is shown below.


```
<result>
  <query id="7129">:facetid:toc:\"db/journals/tse/tse37.bht\"</query>
  <status code="200">OK</status>
  <time unit="msecs">63.48</time>
  <completions total="1" computed="1" sent="1">
    <c sc="51" dc="51" oc="51" id="43445385">:facetid:toc:db/journals/tse/tse37.bht</c>
  </completions>
  <hits total="51" computed="51" sent="51" first="0">
    <hit score="1" id="4610488">
      <info>
        <authors>
          <author pid="99/6075">Xavier Amatriain</author>
          <author pid="86/1554">Pau Arumí</author>
        </authors>
        <title>Frameworks Generate Domain-Specific Languages: A Case Study in the Multimedia Domain.</title>
        <venue>IEEE Trans. Software Eng.</venue>
        <volume>37</volume>
        <number>4</number>
        <pages>544-558</pages>
        <year>2011</year>
        <type>Journal Articles</type>
        <access>closed</access>
        <key>journals/tse/AmatriainA11</key>
        <doi>10.1109/TSE.2010.48</doi>
        <ee>https://doi.org/10.1109/TSE.2010.48</ee>
        <url>https://dblp.org/rec/journals/tse/AmatriainA11</url>
      </info>
      <url>URL#4610488</url>
    </hit>
    <hit score="1" id="4610489">
      <info>
        <authors>
          <author pid="18/6535">James H. Andrews</author>
          <author pid="m/TimMenzies">Tim Menzies</author>
          <author pid="43/3270">Felix Chun Hang Li</author>
        </authors>
        <title>Genetic Algorithms for Randomized Unit Testing.</title>
        <venue>IEEE Trans. Software Eng.</venue>
        <volume>37</volume>
        <number>1</number>
        <pages>80-94</pages>
        <year>2011</year>
        <type>Journal Articles</type>
        <access>closed</access>
        <key>journals/tse/AndrewsML11</key>
        <doi>10.1109/TSE.2010.46</doi>
        <ee>https://doi.org/10.1109/TSE.2010.46</ee>
        <url>https://dblp.org/rec/journals/tse/AndrewsML11</url>
      </info>
      <url>URL#4610489</url>
    </hit>
```


Figura 5.3: Ejemplo de fichero xml.

Ejemplo de títulos obtenidos:

```
Titulo: Frameworks Generate Domain-Specific Languages: A Case Study in the Multimedia Domain.
Titulo: Genetic Algorithms for Randomized Unit Testing.
Titulo: Self-Supervising BPEL Processes.
Titulo: Loupe: Verifying Publish-Subscribe Architectures with a Magnifying Lens.
Titulo: FlowTalk: Language Support for Long-Latency Operations in Embedded Devices.
Titulo: Toward a Formalism for Conservative Claims about the Dependability of Software-Based Systems.
Titulo: Developing a Single Model and Test Prioritization Strategies for Event-Driven Software.
Titulo: Zebu: A Language-Based Approach for Network Protocol Message Processing.
```

Figura 5.4: Títulos obtenidos.

A continuación, se ejecuta la clase **ObtenerPdfConSerpApi.java** con los títulos obtenidos a partir de las etiquetas <title>. Esta clase devuelve, el nombre del artículo junto con la primera URL de descarga del propio artículo en formato PDF, la cuál, se obtiene a partir del JSON del artículo cuyo nombre es buscado a través de SerpApi de Google Scholar. La URL de descarga en formato PDF se encuentra en la sección de resources"del JSON:


```
      }
    ]
  },
  "resources": [
    {
      "title": "arxiv.org",
      "file_format": "PDF",
      "link": "https://arxiv.org/pdf/2107.11789"
    }
  ],
  "inline links": {
```


Figura 5.5: Sección resources de un JSON.

Luego, se ejecuta la clase **DescargarPdfCache.java** con la URL de descarga del artículo en formato PDF, que descarga el PDF en local solo si no se ha descargado previamente. Además, proporciona la ruta local donde se descarga.

Si no se ha descargado previamente:

```
PDF descargado correctamente en:
C:\Users\Admin\Desktop\PDF descargado\Self-supervising BPEL Processes.pdf
```

Figura 5.6: Mensaje si no se ha descargado previamente.

Si se ha descargado el PDF previamente se muestra el mensaje siguiente:

```
El PDF ya ha sido descargado previamente.
```

Figura 5.7: Mensaje si se ha descargado previamente.

Finalmente, se ejecuta la clase **MostrarYComprobarEnlaces.java** con la ruta de descarga local del PDF. Esta clase se encarga de leer el contenido del PDF para extraer las URLs de referencia, además de verificar si estas existen o no:

```
Enlaces encontrados y verificados:
http://www.clam-project.org
El enlace existe.
```

Figura 5.8: Ejemplo de enlaces.

Clases con base de datos

El funcionamiento es muy parecido al anterior, solo modificándose por la introducción de la clase **ConexionBaseDeDatos.java** para poder manipular la base de datos.

En primer lugar, se ejecuta la clase **LeerXml.java** junto con la URL del fichero .xml obtenido de DBLP. Esta ejecución nos proporciona los títulos de los artículos contenidos en el fichero. Ejemplo de fichero .xml :


```
<hit score="1" id="4646506">
  <info>
    <authors>
      <author pid="15/2194">Radu Calinescu</author>
      <author pid="55/4092">Lars Grunske</author>
      <author pid="k/MartaZKwiatkowska">Marta Z. Kwiatkowska</author>
      <author pid="80/191">Raffaela Mirandola</author>
      <author pid="02/2374">Giordano Tamburrelli</author>
    </authors>
    <title>Dynamic QoS Management and Optimization in Service-Based Systems.</title>
    <venue>IEEE Trans. Software Eng.</venue>
    <volume>37</volume>
    <number>3</number>
    <pages>387-409</pages>
    <year>2011</year>
    <type>Journal Articles</type>
    <access>closed</access>
    <key>journals/tse/CalinescuGKMT11</key>
    <doi>10.1109/TSE.2010.92</doi>
    <ee>https://doi.org/10.1109/TSE.2010.92</ee>
    <url>https://dblp.org/rec/journals/tse/CalinescuGKMT11</url>
  </info>
  <url>URL#4646506</url>
</hit>
<hit score="1" id="4646507">
  <info>
    <authors>
      <author pid="77/5805">Laura Carnevali</author>
      <author pid="02/7786">Lorenzo Ridi</author>
      <author pid="50/3792">Enrico Vicario</author>
    </authors>
    <title>Putting Preemptive Time Petri Nets to Work in a V-Model SW Life Cycle.</title>
    <venue>IEEE Trans. Software Eng.</venue>
    <volume>37</volume>
    <number>6</number>
    <pages>826-844</pages>
    <year>2011</year>
    <type>Journal Articles</type>
    <access>closed</access>
    <key>journals/tse/CarnevaliRV11</key>
    <doi>10.1109/TSE.2011.4</doi>
    <ee>https://doi.org/10.1109/TSE.2011.4</ee>
    <url>https://dblp.org/rec/journals/tse/CarnevaliRV11</url>
  </info>
  <url>URL#4646507</url>
</hit>
```


Figura 5.9: Ejemplo de fichero xml 2.

Ejemplo de títulos obtenidos:

```
Titulo: Dynamic Software Updating Using a Relaxed Consistency Model.
Titulo: On the Distribution of Bugs in the Eclipse System.
Titulo: A Controlled Experiment for Program Comprehension through Trace Visualization.
Titulo: A Classification Framework for Software Component Models.
Titulo: Measuring the Discriminative Power of Object-Oriented Class Cohesion Metrics.
Titulo: Bristlecone: Language Support for Robust Software Applications.
Titulo: Systematic Review and Aggregation of Empirical Studies on Elicitation Techniques.
Titulo: From UML to Petri Nets: The PCM-Based Methodology.
Titulo: Automatically Detecting and Tracking Inconsistencies in Software Design Models.
Titulo: A Comparison of Tabular Expression-Based Testing Strategies.
Titulo: Swarm Verification Techniques.
Titulo: An Analysis and Survey of the Development of Mutation Testing.
```

Figura 5.10: Títulos obtenidos 2.

A continuación, se ejecuta la clase **ObtenerPdfConSerpApi.java** con los títulos conseguidos, obteniéndose el nombre del artículo y la primera URL de descarga del artículo en formato PDF. Esta URL se obtiene de la sección de resources"del JSON del artículo obtenido por SerpApi de Google Scholar a partir de los títulos indicados.

```
Título del artículo: Swarm Verification Techniques.
El primer enlace encontrado es: https://ir.library.oregonstate.edu/downloads/2801ph654
```

Figura 5.11: Titulo y ruta de PDF.

Luego, se ejecuta la clase **DescargarPdfCache.java** junto con la URL de descarga en formato PDF. Esta clase, descarga el PDF en local solo si no se ha descargado previamente, además de proporcionar la ruta local donde se almacena.

Para saber si el PDF no se ha descargado de forma previa, esta clase se comunica con **ConexionBaseDeDatos.java** que es la clase utilizada para poder manipular la base de datos. Antes de descargar el PDF, **DescargarPdfCache.java** mira en la tabla *nombre_pdfs* de la base de datos si se encuentra el título del artículo que se quiere descargar:

- Si se encuentra no descarga el PDF e indica que ya está descargado.
- Si no se encuentra presente introduce el DOI del artículo a través de la clase **ConexionBaseDeDatos.java** en la tabla y descarga el artículo en formato PDF.

Una vez está descargado el PDF se procede a la ejecución de la clase **MostrarY-ComprobarEnlaces.java** indicando la ruta local de descarga del artículo.

Esta clase lee el contenido del PDF descargado y se encarga de extraer las URLs que se encuentran en el mismo, además de verificar si existen o no.

Las URLs extraídas las introduce en la tabla *downloaded_pdfs* gracias a la clase **ConexionBaseDeDatos.java** junto con el título del artículo del que se han extraído. Además, también indica el código de respuesta del servidor, que si se encuentra entre 200 y 300 significa que la URL es accesible, es decir, que existe; si no se encuentra en ese intervalo implica que no está accesible y al contar con

el código de respuesta podemos saber la razón por la cuál no se pudo acceder a ella.

Para poder visualizar las tablas creadas en la base de datos se puede utilizar la aplicación "DB Browser (SQLite)", ya que se trata de una base de datos SQLite. Con esa aplicación, una vez se ha abierto la base de datos, se puede visualizar el contenido de la misma, las tablas y los datos que contienen, además de la estructura:

Estructura de la base de datos:

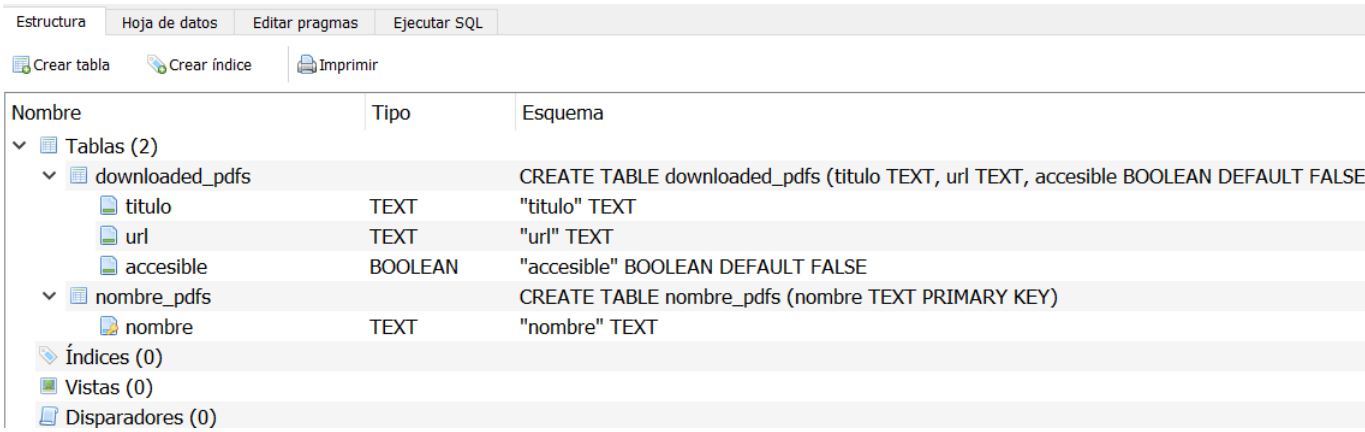


Figura 5.12: Estructura Base de Datos.

Ejemplo de contenido de la tabla *nombre-pdfs*:

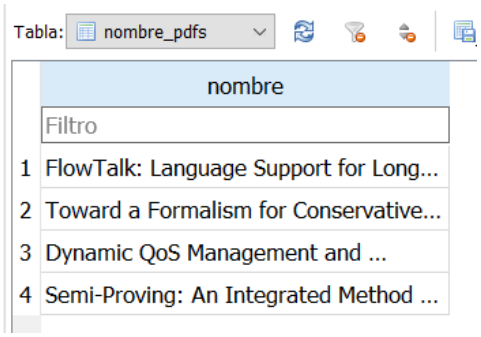


Figura 5.13: Tabla nombre_pdfs.

Ejemplo de contenido de la tabla *downloaded-pdfs*:

Estructura | Hoja de datos | Editar pragmas | Ejecutar SQL

Tabla: downloaded_pdfs — Filtrar en cualquier columna

	doi	url	accesible
	Filtro	Filtro	Filtro
1	FlowTalk_ Language Support for Long...	http://www.xbow.com/Products/...	-1
2	FlowTalk_ Language Support for Long...	http://bergel.eu/flowtalk.html	200
3	FlowTalk_ Language Support for Long...	http://www.sunspotworld.com/docs/javadoc	-1
4	FlowTalk_ Language Support for Long...	http://nescc.sourceforge.net	200
5	FlowTalk_ Language Support for Long...	http://www.tinyos.net	200
6	Toward a Formalism for Conservative...	https://openaccess.city.ac.uk/id/eprint/1070/	200
7	Toward a Formalism for Conservative...	https://doi.org/10.1109/tse.2010.67	200
8	Toward a Formalism for Conservative...	http://openaccess.city.ac.uk/	200
9	Dynamic QoS Management and ...	https://inria.hal.science/hal-00663216	200
10	Semi-Proving: An Integrated Method ...	http://pleuma.cc.gatech.edu/aristotle/Tools/...	-1
11	Semi-Proving: An Integrated Method ...	http://babelfish.arc.nasa.gov/trac/	-1

Figura 5.14: Tabla downloaded_pdfs.

5.2. Ciclo 2

5.2.1. Clases y relaciones entre clases

En este ciclo, se añadió una clase denominada **RunEntireFlow.java**, la cuál, al ejecutarse se encarga de ir llamando a todas las clases de forma automática, pasándoles los argumentos que precisan cada una, para evitar que el usuario tenga que ir ejecutando cada una de forma individual.

En este caso la conexión de clases se produce según el siguiente diagrama:

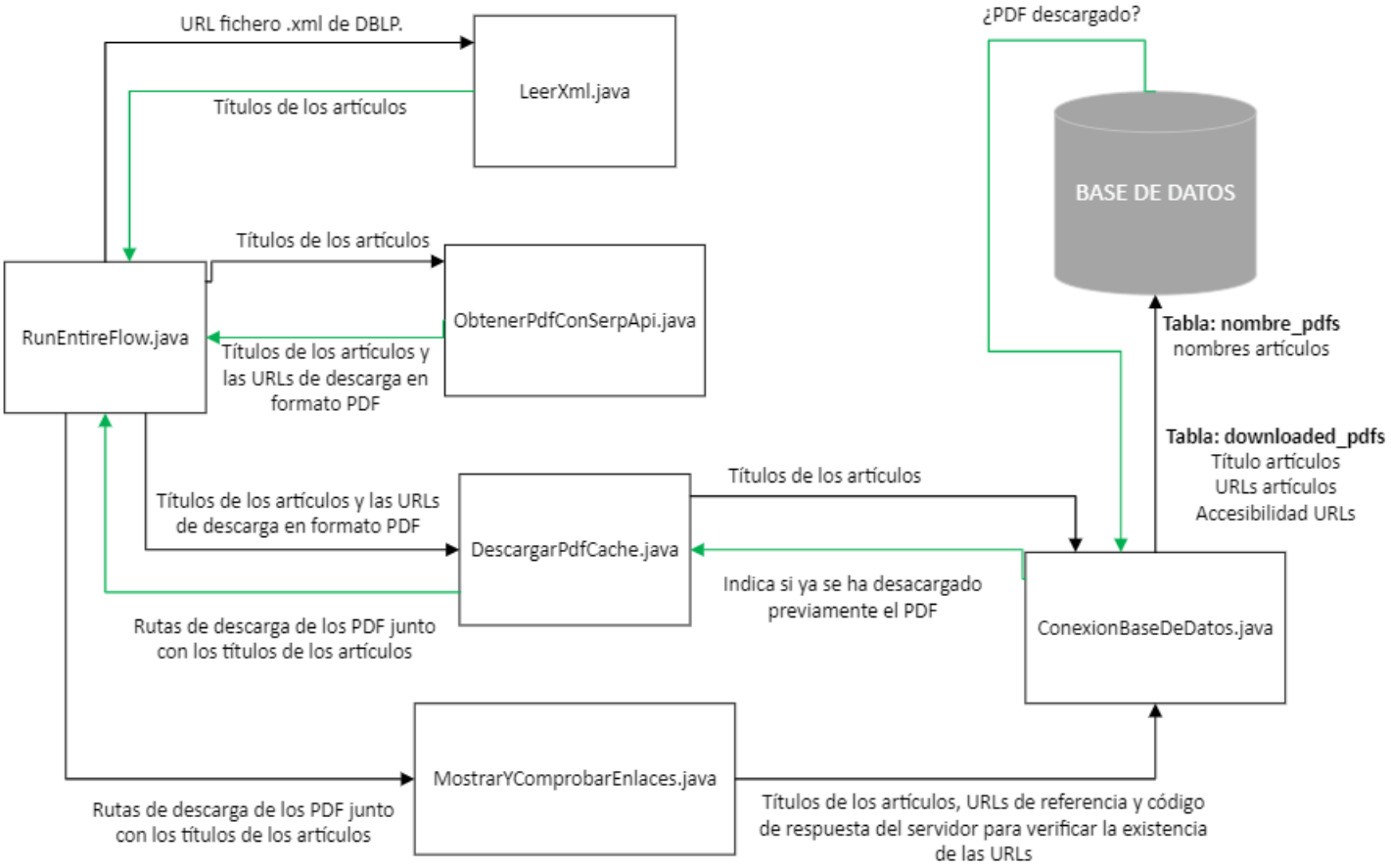


Figura 5.15: Diagrama conexión de clases ciclo 2.

5.2.2. Ejecución

A continuación, se expone el funcionamiento de la aplicación gracias a la adición de la clase **RunEntireFlow.java**.

Se ejecuta dicha clase añadida con la URL del fichero .xml obtenido de DBLP.

Ejemplo de fichero .xml :


```
▼<result>
  <query id="612670">conf* icse* 2024* workshop*</query>
  <status code="200">OK</status>
  <time unit="msecs">140.15</time>
 ▼<completions total="1" computed="1" sent="1">
    <c sc="68" dc="68" oc="68" id="55576254">workshop</c>
  </completions>
 ▼<hits total="68" computed="68" sent="3" first="0">
  ▼<hit score="5" id="83838">
   ▼<info>
    ▼<authors>
       <author pid="09/6502">Junqiang Li</author>
       <author pid="312/1887">Senyi Li</author>
       <author pid="257/1399">Keyao Li</author>
       <author pid="367/5897">Falin Luo</author>
       <author pid="94/60">Hongfang Yu</author>
       <author pid="66/5479-1">Shanshan Li 0001</author>
       <author pid="40/1491-78">Xiang Li 0078</author>
     </authors>
     <title>ECFuzz: Effective Configuration Fuzzing for Large-Scale Systems.</title>
     <venue>ICSE</venue>
     <pages>48:1-48:12</pages>
     <year>2024</year>
     <type>Conference and Workshop Papers</type>
     <access>closed</access>
     <key>conf/icse/LiLLLYLL24</key>
     <doi>10.1145/3597503.3623315</doi>
     <ee>https://doi.org/10.1145/3597503.3623315</ee>
     <url>https://dblp.org/rec/conf/icse/LiLLLYLL24</url>
    </info>
    <url>URL#83838</url>
   </hit>
  ▼<hit score="4" id="83808">
```


Figura 5.16: Ejemplo de fichero xml 3.

Tras la ejecución, **RunEntireFlow.java** invoca a **LeerXml.java** que proporciona el nombre de los artículos que contiene el fichero .xml. Dichos títulos se guardan en una variable denominada *titulosArticulos*, que utiliza **RunEntireFlow.java** para llamar a **ObtenerPdfConSerpApi.java**.

Esta clase se encarga, si no encuentra una URL de descarga del artículo en formato PDF, de mostrar el JSON del artículo. Si por el contrario encuentra una URL de descarga, se encarga de guardar en una variable de tipo String [][], llamada *titulosYEnlaces*, el título de cada artículo con su URL de descarga correspondiente.

Con esa variable se invoca a **DescargarPdfCache.java**, la cual, descarga, si no se ha descargado previamente, el artículo en formato PDF en una carpeta denominada *cache*. Además, introduce el nombre de dicho PDF en la base de datos, en la tabla *nombre_pdfs*. Además, guarda en una variable de tipo String [][],

denominada *rutasYTitulos*, la ruta de descarga del PDF junto con el título del artículo.

Utilizando dicha variable, se llama a la clase **MostrarYComprobarEnlaces.java** que se encarga de buscar las URLs que aparecen en las referencias de los artículos y verifica su existencia. Además, añade el nombre del artículo, la URL identificada y el código devuelto por el servidor (verificación de la exitencia de la URL) a la base de datos, a la tabla *downloaded_pdfs*.

5.3. Ciclo 3

Tras haber desarrollado el ciclo 2 de la aplicación, se presentó el problema de no poder utilizar las clases de forma individual.

Cuando se ejecuta la aplicación, se requiere la funcionalidad que permite introducir por línea de comandos un argumento, realizado por el propio usuario. Para poder conseguir esto, es necesario que cada clase pueda ser ejecutada de forma individual, es decir, que pueda utilizar argumentos pasados por la línea de comandos y, no solo aquellos que les proporciona en muchos casos la clase anterior. Por ejemplo, en el momento en el que la clase **ObtenerPdfConSerpApi.java** empieza a buscar una URL de descarga del artículo en formato PDF, en el JSON del propio artículo, cabe la posibilidad de que no lo encuentre. En ese caso, muestra por pantalla el JSON completo para poder dar la oportunidad al usuario de buscar la URL. Por ello, si el usuario encuentra dicha URL, es necesario que pueda ejecutar la siguiente clase, **DescargarPdfCache.java** directamente desde la linea de comandos.

Para poder resolver este tipo conflicto, se han modificado las clases que conforman la aplicación de la manera que se explica a acontinuación.

5.3.1. Modificaciones de las clases

LeerXml.java

La funcionalidad de esta clase no se ha modificado, sin embargo, se ha añadido una mejora.

Esta consta de agregar en el código, una variable que funciona como un contador de títulos, para poder mostrar por pantalla la cantidad de ellos que se encuentran en el fichero de DBLP.

ObtenerPdfConSerpApi.java

En esta clase, se añadió una variable contador también, que indica el número de enlaces leídos.

Además, se agregó una vaiable de tipo boolean que se encarga de no buscar el enlace de descarga de un artículo en formato PDF si ya se ha realizado con anterioridad. He decidido incluir esta nueva funcionalidad porque de esta forma,

se puede minimizar el número de búsquedas que realiza SerpApi, ya que son limitadas por mes.

DescargarPdfCache.java

En esta clase decidí cambiar el *main*. Anteriormente, se encontraba de forma *public static String[][] main(String[][] titulosYEnlaces)* y, actualmente se encuentra como *public static void main(String[] args) throws ClassNotFoundException*. Este cambio es debido a que por línea de comandos no se puede ejecutar la clase con un *String[][]*.

Además, agregué una funcionalidad, que permite forzar la descarga de los archivos PDFs en la caché. Para ello, tuve también que realizar un método que se encarga de chequear si existe el directorio cache, y si no lo crea, e incluir una variable de tipo boolean para indicar si es necesario forzar esa descarga.

Por último, también modifiqué ligeramente la lógica que se encarga de comprobar si el PDF ya se había descargado previamente. Anteriormente se realizaba a través de un *for each* en el *main*, utilizando la variable *tituloYEnlace* que se obtenía de la clase **ObtenerPdfConSerpApi.java**. Actualmente, esto se realiza en un método, el cuál, sigue guardando los títulos de los artículos y sus rutas de descarga, pero tiene en cuenta que la variable booleana que, indica si hay que forzar la descarga, se encuentra en estado *false*.

MostrarYComprobarEnlaces.java

En esta clase, se ha mejorado la interacción con la línea de comandos, pudiendo:

- Volver a transformar el PDF a texto, buscando de nuevo los enlaces y comprobando su accesibilidad. Para ello, se debe indicar al inicio *–regenerate-text*.
- Analizar de nuevo los enlaces de los ficheros, usando el texto existente y comprobar su accesibilidad. Para ello, se debe indicar al inicio *–force-link-search*.
- Comprobar la accesibilidad de los enlaces que están en la base de datos sin la necesidad de utilizar el texto del artículo. Para ello, se debe indicar al inicio *–use-db-only*.

Además, se ha mejorado la funcionalidad de la extracción y procesamiento de los enlaces. Esto se consiguió mediante la introducción de múltiples expresiones regulares para mejorar la precisión. También, se introdujo un marcador que permite identificar de forma más eficiente y precisa las URLs, además de que, permite manipular el texto del PDF para poder optimizar la extracción de las mismas.

Por último, se ha incluído una nueva funcionalidad que proporciona la capacidad de eliminar enlaces antiguos antes de insertar las nuevas versiones de los mismos, además de permitir, obtener y actualizar las URLs directamente desde la base de datos.

ConexionBaseDeDatos.java

En este caso, he incluído una serie de columnas a la tabla *downloaded_pdfs* de la base de datos. Estas nuevas columnas son *finalURL*, *contexto* y *type*, las cuales, proprocionan una visión más compleja y completa de la representación de los datos almacenados.

Además, he agregado nuevos métodos que se especifican a continuación:

- EliminarEnlacesDeUnArticulo, para poder limpiar la base de datos de enlaces que se encuentran obsoletos.
- getAllURLs, extrae las URLs que han sido identificadas y las devuelve como una lista.
- saveURLType, se encarga de clasificar o categorizar las URLs que se encuentran en la tabla *downloaded_pdfs*.
- BDExist, se encarga de verificar si la base de datos existe.
- updateResponseCode, actualiza el código de respuesta del servidor al intentar acceder a una URL.

Por último, se han modificado métodos existentes y el manejo de las excepciones para poder garantizar la compatibilidad con la nueva versión de la base de datos.

RunEntireFlow.java

En esta clase, he añadido mejoras en el manejo de los errores. Además, se integró una nueva clase denominada **ClasificarURL.java**.

5.3.2. Clases y relaciones entre clases

Como he indicado anteriormente, he agregado una nueva clase denominada **ClasificarURL.java**, que se encarga de clasificar las URLs presentes en la tabla *downloaded_pdfs* de la base de datos, gracias a un fichero llamado *sites.properties*.

En este ciclo, la conexión de clases se produce según se indica en el diagrama a continuación:

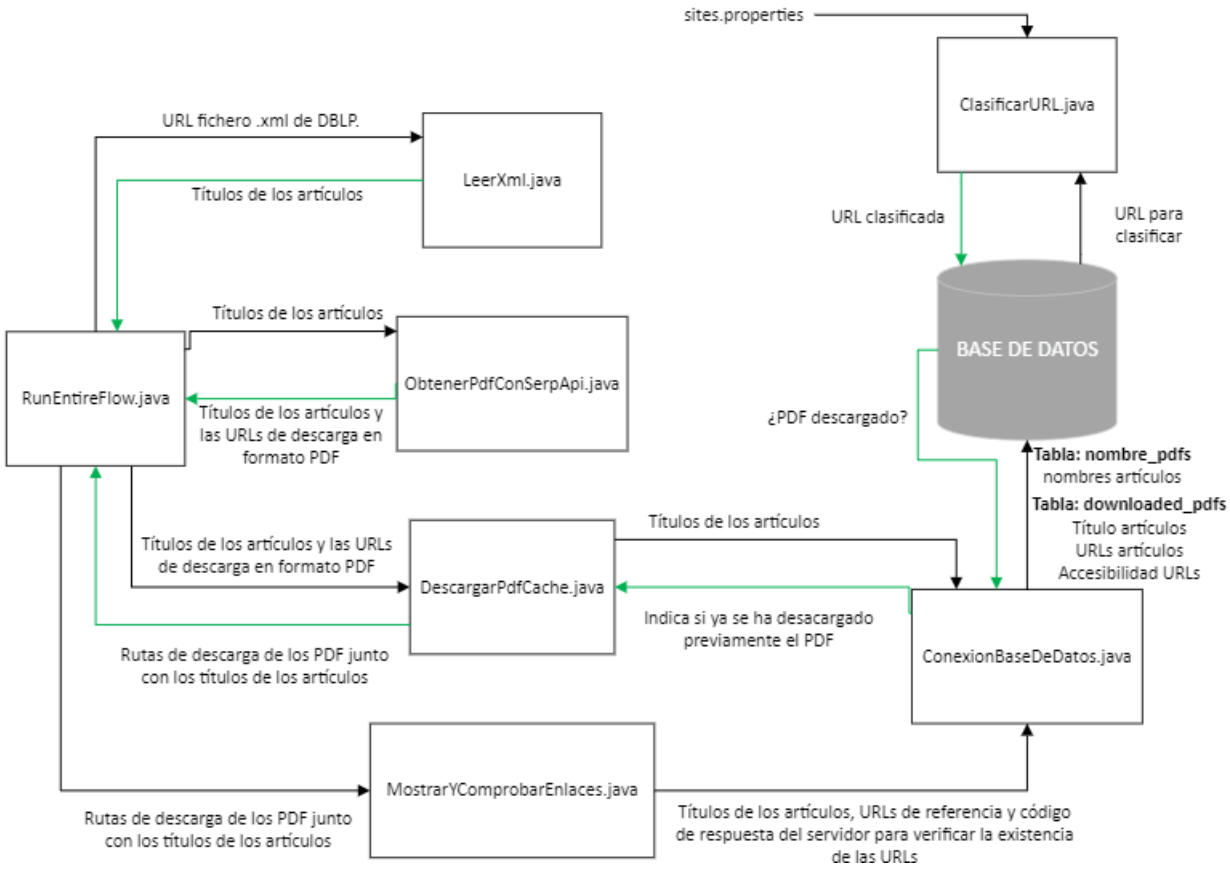


Figura 5.17: Diagrama conexión de clases ciclo 3.

5.3.3. Ejecución

En este apartado, explicaré el funcionamiento de la aplicación con las mejoras introducidas y la adición de la nueva clase.

Hay dos opciones para poder ejecutar la aplicación, las cuáles se detallan a continuación.

Opción 1: Ejecución individual de las clases

Se indican por línea de comandos, aquellos que están comprendidos en el fichero **runEntireFlow.sh**:

```
java -cp checkurl-0.0.1-SNAPSHOT-jar-with-dependencies.jar es.upm.grise.checkurl.LeerXml
$1 > titleList.txt 2> errorTitleList.txt

cat titleList.txt | xargs java -cp checkurl-0.0.1-SNAPSHOT-jar-with-dependencies.jar es.upm.grise.checkurl.ObtenerPdfConSerpApi
> urlList.txt 2> errorUrlList.txt

cat urlList.txt | xargs java -cp checkurl-0.0.1-SNAPSHOT-jar-with-dependencies.jar es.upm.grise.checkurl.DescargarPdfCache
> fileList.txt 2> errorFileList.txt

cat fileList.txt | xargs java -cp checkurl-0.0.1-SNAPSHOT-jar-with-dependencies.jar es.upm.grise.checkurl.MostrarYComprobarEnlaces
> links.txt 2> errorLinks.txt

java -cp checkurl-0.0.1-SNAPSHOT-jar-with-dependencies.jar es.upm.grise.checkurl.ClasificarURL
> classification.txt 2> errorClassification.txt
```

Figura 5.18: Comandos ejecución ciclo 3.

Tras la ejecución del primer comando, siendo el argumento la URL del fichero .xml de DBLP, crea dos ficheros denominados *titleList.txt* y *errorTitleList.txt*, si no ha sido creados previamente. A continuación, guarda en el fichero *titleList.txt* todos los nombres de los artículos que extrae **LeerXml.java**, y en *errorTitleList.txt* los errores si se producen.

Ejemplo de *titleList.txt*:

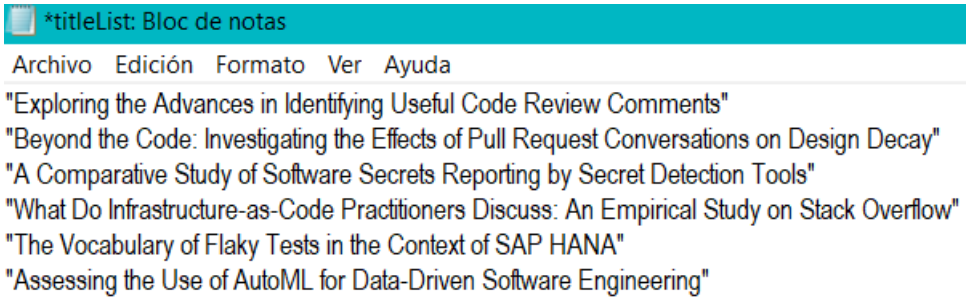


Figura 5.19: Ejemplo fichero *titleList.txt* ciclo 3.

Ahora, se indica el segundo comando, que señala que la clase **ObtenerPdfConSerpAPi.java** utiliza los títulos que se encuentran en el fichero generado *titleList.txt*, para buscar sus enlaces de descarga en formato PDF. Además, cuando encuentra los enlaces, los almacena en un fichero, que si no existe crea, denominado *urlList.txt*. Además, guarda los posibles errores en otro fichero denominado *errorUrlList.txt*.

Ejemplo de *urlList.txt*:

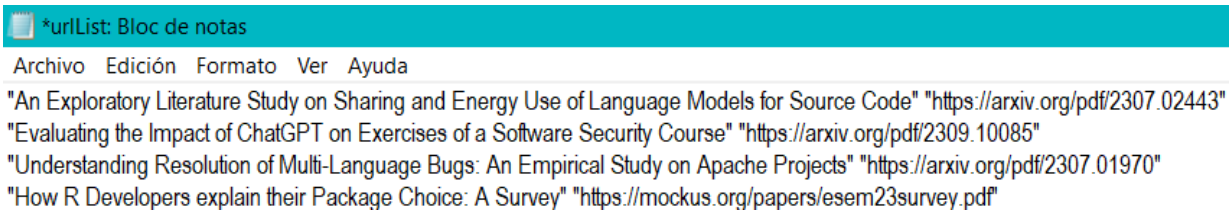


Figura 5.20: Ejemplo fichero *urlList.txt* ciclo 3.

A continución,se indica el tercer comando, que señala que la clase **DescargarPdfCache.java** utiliza los títulos y sus enlaces de descarga en formato PDF para poder descargar los artículos. Además genera si no existen dos ficheros de texto denominados *fileList.txt* y *errorFileList.txt*. En el primero se guardan la ruta de descarga del artículo junto con su título, y en el segundo los posibles errores.

Ejemplo de *fileList.txt*:

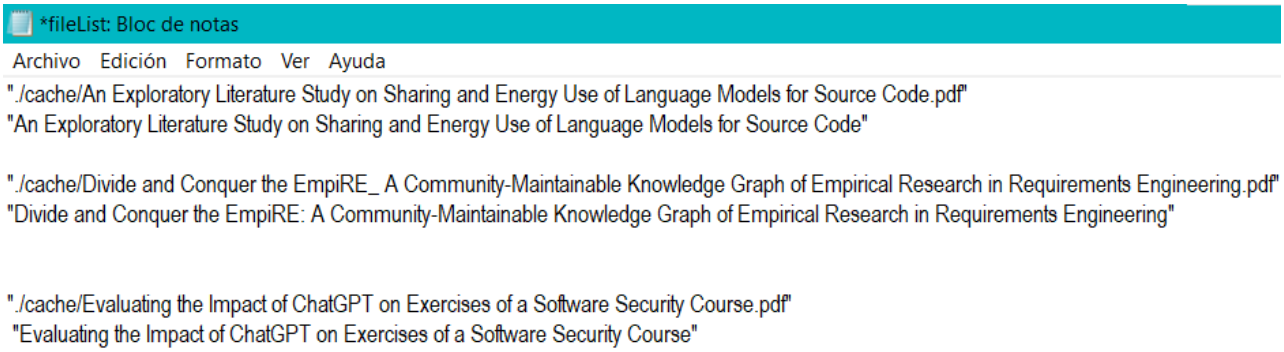


Figura 5.21: Ejemplo fichero *fileList.txt* ciclo 3.

Luego, se indica el cuarto comando, que señala que la clase **MostrarYComprobarEnlaces.java** utiliza las rutas de descarga y el nombre de los artículos. Además, almacena en un fichero, que si no existe crea, denominado *links.txt* los enlaces encontrados y en *errorLinks.txt* los posibles errores producidos.

Por último, se ejecuta el quinto comando, que señala que la clase **ClasificarURL.java** además de clasificar los enlaces en la base de datos, también indica dicha clasificación en un fichero denominado *classification.txt*, que crea si no existe previamente, y en *errorClassification.txt* los posibles errores generados.

Ejemplo de *classification.txt*:

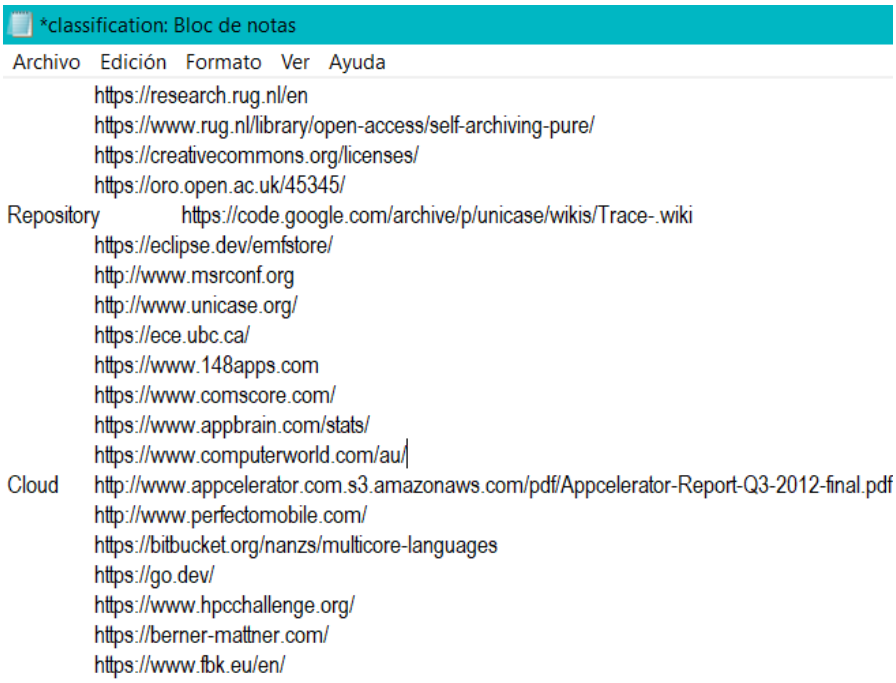


Figura 5.22: Ejemplo fichero *classification.txt* ciclo 3.

Ejemplo de base de datos:

Tabla: downloaded_pdfs — Filtrar en cualquier columna

	titulo	URL	finalURL	accesible	contexto	type
	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro
111	Personalized Guidelines for Design, ...	https://drive.google.com/file/d/...	https://drive.google.com/file/d/...	200	https://www.core.edu.au/.[38] "Cor...	Cloud
112	Personalized Guidelines for Design, ...	https://figshare.com/s/...	https://figshare.com/s/...	404	ew," Comput-ers & Security, vol. 105,...	
113	ToxiSpanSE: An Explainable Toxicity ...	https://github.com/WSU-SEAL/...	https://github.com/WSU-SEAL/...	200	dataset available for further...	Repository
114	ToxiSpanSE: An Explainable Toxicity ...	https://github.com/WSU-SEAL/...	https://github.com/WSU-SEAL/...	200	with 88% F11 score. We have made...	Repository
115	ToxiSpanSE: An Explainable Toxicity ...	https://www.perspectiveapi.com/	https://www.perspectiveapi.com/	200	016....	
116	ToxiSpanSE: An Explainable Toxicity ...	https://boxofcables.dev/toxicity-in-...	https://boxofcables.dev/toxicity-in-...	200	5, no. CSCW2, pp. 1–35, 2021....	
117	ToxiSpanSE: An Explainable Toxicity ...	https://medium.com/linuxforeveryone/	https://medium.com/linuxforeveryone	200	, "Windows is sh*t:' linux users...	
118	ToxiSpanSE: An Explainable Toxicity ...	https://www.linux4everyone.com/	https://www.linux4everyone.com/	200	860/...	
119	ToxiSpanSE: An Explainable Toxicity ...	https://2014.jsconf.eu/speakers/	https://2014.jsconf.eu/speakers/	200	. Reinhard, "This is bigger than us: ...	
120	ToxiSpanSE: An Explainable Toxicity ...	https://www.zdnet.com/article/	https://www.zdnet.com/?...	200	html...	
121	ToxiSpanSE: An Explainable Toxicity ...	https://arstechnica.com/gadgets/...	https://arstechnica.com/gadgets/...	200	"The perl foundation is fragment-...	
122	ToxiSpanSE: An Explainable Toxicity ...	https://doi.org/...	https://dl.acm.org/doi/...	200	g, ser. ASE...	Article
123	ToxiSpanSE: An Explainable Toxicity ...	https://github.com/heartexlabs/label...	https://github.com/HumanSignal/lab...	200	N. Liubimov, "Label...	Repository
124	ToxiSpanSE: An Explainable Toxicity ...	https://github.com/heartexlabs/label...	https://github.com/HumanSignal/lab...	200	ce software...	Repository
125	ToxiSpanSE: An Explainable Toxicity ...	https://github.com/tamuhey/textspan	https://github.com/tamuhey/textspan	200	preprint arXiv:1912.06872, 2019....	Repository
126	ToxiSpanSE: An Explainable Toxicity ...	https://github.com/fchollet/keras	https://github.com/keras-team/keras	200	lagiarism detection," in Coling 2010: ...	Repository
127	ToxiSpanSE: An Explainable Toxicity ...	https://github.com/WSU-SEAL/...	https://github.com/WSU-SEAL/...	200	015....	Repository
128	ToxiSpanSE: An Explainable Toxicity ...	https://competitions.codalab.org/...	https://competitions.codalab.org/...	200	16–27.[42] J. Pavlopoulos, "Semeval...	
129	ToxiSpanSE: An Explainable Toxicity ...	https://github.com/grrrr/krippendorff...	https://github.com/grrrr/krippendorff...	200	ll and S. Castro, "Python ...	Repository
130	ToxiSpanSE: An Explainable Toxicity ...	https://aclanthology.org/2020.emnlp-...	https://aclanthology.org/2020.emnlp-...	200	tions. Online: Association for ...	Book

Figura 5.23: Ejemplo tabla *downloaded_pdfs* ciclo 3.

Opción 2: Ejecución clase RunEntireFlow.java

Con esta opción, los resultados son los mismos, en relación a la base de datos, que en la Opción 1. Sin embargo, todos los documentos de texto que se generan con esta opción no se producen.

En cuanto al flujo de ejecución, es exactamente igual que el indicado en el ciclo 2, es decir, se ejecuta la clase **RunEntireFlow.java** junto con la URL del fichero .xml de DBLP. Luego, dicha clase se encarga de ir ejecutando y proporcionando los argumentos necesarios a cada clase individual.

Capítulo 6

Resultados

En este capítulo se mostrarán los resultados que se han obtenido a partir de la realización de un caso de estudio.

En dicho caso de estudio, se escogieron dos revistas con determinados artículos de los años 2013 y 2023 para comparar cómo desaparecía la información en internet en dos décadas distintas.

6.1. Revista 2013

Primero realicé el caso de estudio con la revista de 2013. Cuento con un fichero *Shell Script* denominado **runEntireFlow.sh** en el que se muestran los comandos para poder realizar la ejecución de la aplicación con el fichero .xml de DBLP seleccionado.

runEntireFlow.sh

```
java -cp checkurl-0.0.1-SNAPSHOT-jar-with-dependencies.jar es.upm.grise.checkurl.LeerXml
$1 > titleList.txt 2> errorTitleList.txt

cat titleList.txt | xargs java -cp checkurl-0.0.1-SNAPSHOT-jar-with-dependencies.jar es.upm.grise.checkurl.ObtenerPdfConSerpApi
> urlList.txt 2> errorUrlList.tex

cat urlList.txt | xargs java -cp checkurl-0.0.1-SNAPSHOT-jar-with-dependencies.jar es.upm.grise.checkurl.DescargarPdfCache
> fileList.txt 2> errorFileList.txt

cat fileList.txt | xargs java -cp checkurl-0.0.1-SNAPSHOT-jar-with-dependencies.jar es.upm.grise.checkurl.MostrarYComprobarEnlaces
> links.txt 2> errorLinks.txt

java -cp checkurl-0.0.1-SNAPSHOT-jar-with-dependencies.jar es.upm.grise.checkurl.ClasificarURL
> classification.txt 2> errorClassification.txt
```

Figura 6.1: Comandos de runEntireFlow.sh

Ejecuté la aplicación siguiendo los comandos indicados. Tras esta ejecución, obtuve la base de datos con las URLs y su verificación de existencia.

	título	URL	finalURL	accesible	contexto	type
	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro
1	An Embedded Multiple-Case Study on...	http://www.rug.nl/research/portal	https://research.rug.nl/en	200	investigate your claim....	
2	An Embedded Multiple-Case Study on...	https://www.rug.nl/library/open-...	https://www.rug.nl/library/open-...	200	ated by the "Taverne" license.More ...	
3	IR in Software Traceability: From a ...	https://creativecommons.org/licenses/	https://creativecommons.org/licenses/	200	the URL identifying the publication in ...	
4	Towards a Metric Suite Proposal to ...	https://oro.open.ac.uk/45345/	https://oro.open.ac.uk/45345/	200	...	
5	Tracing Requirements and Source Co...	http://code.google.com/p/unicase/...	https://code.google.com/archive/p/...	200	ware...	Repository
6	Tracing Requirements and Source Co...	http://eclipse.org/emfstore/	https://eclipse.dev/emfstore/	200	on...	
7	Tracing Requirements and Source Co...	http://www.msrconf.org	http://www.msrconf.org	200	hop on Cooperative and...	
8	Tracing Requirements and Source Co...	http://www.unicase.org/	http://www.unicase.org/	403	race Client, http://code.google.com/...	
9	Real Challenges in Mobile App ...	http://www.ece.ubc.ca/	https://ece.ubc.ca/	200	between 1-3 years, to 16% less tha...	
10	Real Challenges in Mobile App ...	http://148apps.biz/app-store-metrics/	https://www.148apps.com	200	ve Systems...	
11	Real Challenges in Mobile App ...	http://www.comscore.com/	https://www.comscore.com/	200	ore Metrics," >>>>>>>http://...	
12	Real Challenges in Mobile App ...	http://www.appbrain.com/stats/	https://www.appbrain.com/stats/	200	/comScore Reports January 2013 U.S...	
13	Real Challenges in Mobile App ...	http://www.computerworld.com.au/...	https://www.computerworld.com/au/	200	ybrid mobile-app development,"...	
14	Real Challenges in Mobile App ...	http://code.google.com/p/robotium/	https://github.com/robotiumtech/...	200	ed Theory: Issues and Discussions. ...	Repository
15	Real Challenges in Mobile App ...	http://developer.android.com/tools/...	https://developer.android.com/tools	200	California, 1998....	
16	Real Challenges in Mobile App ...	http://www.gorillalogic.com/	https://www.gorillalogic.com:443/	200	/developer.android.com/tools/help/...	
17	Real Challenges in Mobile App ...	https://github.com/square/	https://github.com/square/	200	ttp://www.gorillalogic.com/...	Repository
18	Real Challenges in Mobile App ...	http://cocoadev.com/wiki/...	http://cocoadev.com/wiki/...	200	egration Testing Framework," ...	
19	Real Challenges in Mobile App ...	https://developer.apple.com/library/...	https://developer.apple.com/library/...	200	"SenTestingKit Framework," ...	
20	Real Challenges in Mobile App ...	http://...	https://...	200	ightapp.com/....	
21	Real Challenges in Mobile App ...	http://www.berginsight.com/...	http://www.berginsight.com/...	404	/drippler.com/rim/blackberry ...	
22	Real Challenges in Mobile App ...	http://...	http://...	403	"Voice of the Next-Generation Mobile ...	Cloud
23	Real Challenges in Mobile App ...	http://www.perfectomobile.com/	http://www.perfectomobile.com/	403	tion/Introduction.html.[30] ...	

Figura 6.2: Base de datos - Caso de estudio 2013

Como se puede observar en la imágen, obtenemos la tabla de *downloaded_pdfs* con:

- Título del artículo
- URL que se verifica
- URL final, la cual es el resultado de la modificación de una URL que se encontraba rota, debido a por ejemplo, saltos de líneas o caracteres especiales.
- La columna de accedible, que nos indica el código de retorno del servidor para verificar la existencia de la URL.
- La columna de contexto, nos proporciona inforación sobre la URL.
- La columna type indica la clasificación de la URL.

Además de proporcionar la tabla *downloaded_pdfs*, también brinda cuatro documentos de texto:

titleList.txt 2013

Este documento recoge los títulos de todos los artículos presentes en el fichero .xml de DBLP:

"IR in Software Traceability: From a Bird's Eye View"
"Impact of Peer Code Review on Peer Impression Formation: A Survey"
"The Case for Knowledge Translation"
"Lessons from Conducting a Distributed Quasi-experiment"
"Towards a Metric Suite Proposal to Quantify Confirmation Biases of Developers"
"Identifying Barriers to the Systematic Literature Review Process"
"Tracing Requirements and Source Code during Software Development: An Empirical Study"
"Cost Effectiveness of Unit Testing: A Case Study in a Financial Institution"
"Application of Statistical Process Control to Software Defect Metrics: An Industry Experience Report"
"A Replicated Experiment on the Effectiveness of Test-First Development"
"Learning from Open-Source Projects: An Empirical Study on Defect Prediction"
"ScrumBut, But Does it Matter? A Mixed-Method Study of the Planning Process of a Multi-team Scrum Organization"
"Real Challenges in Mobile App Development"
"Experimental Comparison of Two Safety Analysis Methods and Its Replication"
"Towards Understanding Replication of Software Engineering Experiments"
"Message from the RESER 2013 Workshop Chairs"
"Message from the MTD 2013 Workshop Chairs"

Figura 6.3: Fragmento fichero *titleList.txt* 2013

urlList.txt 2013

En este documento se recogen las URLs que se ha verificado que existen actualmente.

"IR in Software Traceability: From a Bird's Eye View" "https://portal.research.lu.se/files/6380240/4091779.pdf"
"Impact of Peer Code Review on Peer Impression Formation: A Survey" "https://amiangshu.com/papers/Bosu-ESEM-2013.pdf"
"Towards a Metric Suite Proposal to Quantify Confirmation Biases of Developers" "https://oro.open.ac.uk/45345/1/ESEM2013.pdf"
"Tracing Requirements and Source Code during Software Development: An Empirical Study" "http://se.ifi.uni-heidelberg.de/fileadmin/pdf/publications/2013_ESEM_Delater.pdf"
"Real Challenges in Mobile App Development" "https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=b3076e3108068f6a7579dca49ba21551693fe869"

Figura 6.4: Fragmento fichero *urlList.txt* 2013

fileList.txt 2013

En este documento se guardan los títulos de los artículos que se encuentran en la carpeta *caché*, es decir, que han sido descargados.

"./cache/IR in Software Traceability_ From a Bird's Eye View.pdf" "IR in Software Traceability: From a Bird's Eye View"
"./cache/Impact of Peer Code Review on Peer Impression Formation_ A Survey.pdf" "Impact of Peer Code Review on Peer Impression Formation: A Survey"
"./cache/Towards a Metric Suite Proposal to Quantify Confirmation Biases of Developers.pdf" "Towards a Metric Suite Proposal to Quantify Confirmation Biases of Developers"

Figura 6.5: Fragmento fichero *fileList.txt* 2013

classification.txt 2013

Es este fichero, se clasifican según su tipo a las URLs.

Como se puede observar en este caso, hay un mayor número de URLs que se han identificado como *Cloud*.

Repository	https://github.com/square/
	http://cocoadev.com/wiki/SenTestingKit
	https://developer.apple.com/library/archive/
	https://www.keynotedeviceanywhere.com/
	http://www.berginsight.com/ReportPDF/ProductSheet/bi-app1-ps.pdf
Cloud	http://www.appcelerator.com.s3.amazonaws.com/pdf/Appcelerator-Report-Q3-2012-final.pdf
	http://www.perfectomobile.com/
	https://bitbucket.org/nanzs/multicore-languages
	https://go.dev/
	https://www.hpcchallenge.org/
	https://berner-mattner.com/
	https://www.fbk.eu/en/
	http://www.berner-mattner.com/en/berner-mattner-home/products/cte
	https://researchrepository.ul.ie/articles/conference_contribution/Can_automated_text_classification_improve_content_analysis_of_software_profect_data_/19850593
	https://www.oemr.org/wiki/
	http://snowball.tartarus.org/texts/introduction.html
	https://ece.ubc.ca/
	https://www.bugzilla.org/
	https://www.jslint.com/
	https://developers.google.com/closure/compiler/?csw=1

Figura 6.6: Fragmento fichero *classification.txt* 2013

6.2. Revista 2023

Una vez realicé el análisis de la desaparición de la información en internet, utilizando los artículos de la revista del año 2013, realicé el mismo análisis con los artículos de la revista del año 2023.

Llevé a cabo la ejecución de la aplicación siguiendo los comandos indicados en el fichero **runEntireFlow.sh**. Tras esta ejecución, obtuve la base de datos con las URLs y su verificación.

	titulo	URL	finalURL	accesible	contexto	type
	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro
1	A Comparative Study of Software ...	https://blog.gitguardian.com/uber-...	https://blog.gitguardian.com/uber-...	200	Online; accessed April 25, 2023]....	
2	A Comparative Study of Software ...	https://learn.g2.com/alert-fatigue	https://learn.g2.com/alert-fatigue	200	. 76–79, 2004....	
3	A Comparative Study of Software ...	https://cloud.google.com/storage	https://cloud.google.com/storage	200	ataset of Software Secrets," arXiv e-...	
4	A Comparative Study of Software ...	https://cloud.google.com/bigquery	https://cloud.google.com/bigquery	200	>>>>>>>https://cloud.google.com/...	
5	A Comparative Study of Software ...	https://dl.acm.org	https://dl.acm.org	200	>>>https://www.sciencedirect.com/...	Article
6	A Comparative Study of Software ...	https://link.springer.com	https://link.springer.com/	200	igital Library," >>>>>>>https://...	
7	A Comparative Study of Software ...	https://ieeexplore.ieee.org/Xplore/...	https://ieeexplore.ieee.org/Xplore/...	200	ngerLink," >>>>>>>https://...	Article
8	A Comparative Study of Software ...	https://dblp.org	https://dblp.org	200	>>>>>>>https://ieeexplore.ieee.or...	
9	A Comparative Study of Software ...	https://secdevtools.azurewebsites.n...	https://secdevtools.azurewebsites.n...	200	n open-source platforms with machin...	
10	A Comparative Study of Software ...	https://cycode.com	https://cycode.com	200	credscan.html,...	
11	A Comparative Study of Software ...	https://github.com/anshumanbh/git-...	https://github.com/anshumanbh/git-...	200	https://github.com/Yelp/detect- ...	Repository
12	A Comparative Study of Software ...	https://github.com/tillson/git-hound	https://github.com/tillson/git-hound	200	>https://github.com/anshumanbh/gi...	Repository
13	A Comparative Study of Software ...	https://github.com/michenriksen/...	https://github.com/michenriksen/...	200	und," >>>>>>>https://github.com/...	Repository
14	A Comparative Study of Software ...	https://github.com/kootenpv/gittyleaks	https://github.com/kootenpv/gittyleaks	200	>>>>>>>https://github.com/...	Repository
15	A Comparative Study of Software ...	https://github.com/awslabs/git-secrets	https://github.com/awslabs/git-secrets	200	erence on COMmunication Systems &...	Repository
16	A Comparative Study of Software ...	https://github.com/awslabs	https://github.com/awslabs	200	thub.com/awslabs/git-secrets, [Onlin...	Repository
17	A Comparative Study of Software ...	https://github.com/gitleaks/gitleaks	https://github.com/gitleaks/gitleaks	200	ces - Labs," >>>>>>>https://...	Repository
18	A Comparative Study of Software ...	https://github.com/auth0/repo-...	https://github.com/auth0/repo-...	200	>>>https://github.com/gitleaks/...	Repository
19	A Comparative Study of Software ...	https://github.com/trufflesecurity/...	https://github.com/trufflesecurity/...	200	>>>>https://github.com/auth0/repo...	Repository
20	A Comparative Study of Software ...	https://trufflesecurity.com	https://trufflesecurity.com	200	/github.com/trufflesecurity/trufflehog...	
21	A Comparative Study of Software ...	https://github.com/Skyscanner/...	https://github.com/Skyscanner/...	200	Security," >>>>>>>https://...	Repository
22	A Comparative Study of Software ...	https://github.com/GitGuardian/...	https://github.com/GitGuardian/...	200	>>>>>>>https://github.com/...	Repository
23	A Comparative Study of Software ...	https://api.gitguardian.com/docs	https://api.gitguardian.com/docs	200	n," >>>>>>>https://ww...	API

Figura 6.7: Base de datos - Caso de estudio 2023

Además de proporcionar la tabla denominada *downloaded_pdfs*, también muestra los tres primeros documentos de texto mencionados con anterioridad y adicionalmente uno nuevo.

titleList.txt 2023

Este documento muestra los títulos de los artículos que se encuntran en el fichero de DBLP del año 2023.

"Beyond the Code: Investigating the Effects of Pull Request Conversations on Design Decay"
"A Comparative Study of Software Secrets Reporting by Secret Detection Tools"
"What Do Infrastructure-as-Code Practitioners Discuss: An Empirical Study on Stack Overflow"
"On the Impact and Lessons Learned from Mindfulness Practice in a Real-World Software Company"
"The Vocabulary of Flaky Tests in the Context of SAP HANA"

Figura 6.8: Fragmento fichero *titleList.txt* 2023

urlList.txt 2023

En este caso, el documento presenta las URLs verificadas encontradas en los artículos de 2023.

"An Exploratory Literature Study on Sharing and Energy Use of Language Models for Source Code" "https://arxiv.org/pdf/2307.02443"
"Divide and Conquer the EmpiRE: A Community-Maintainable Knowledge Graph of Empirical Research in Requirements Engineering" "https://arxiv.org/pdf/2306.16791"
"Evaluating the Impact of ChatGPT on Exercises of a Software Security Course" "https://arxiv.org/pdf/2309.10085"
"Understanding Resolution of Multi-Language Bugs: An Empirical Study on Apache Projects" "https://arxiv.org/pdf/2307.01970"

Figura 6.9: Fragmento fichero *urlList.txt* 2023

fileList.txt 2023

Incluye el título de los artículos descargados correspondientes al año 2023.

"./cache/ToxiSpanSE_ An Explainable Toxicity Detection in Code Review Comments.pdf" "ToxiSpanSE: An Explainable Toxicity Detection in Code Review Comments"
"./cache/Replicability Study_ Corpora For Understanding Simulink Models & Projects.pdf" "Replicability Study: Corpora For Understanding Simulink Models & Projects"
"./cache/Comparing Mobile Testing Tools Using Documentary Analysis.pdf" "Comparing Mobile Testing Tools Using Documentary Analysis"
"./cache/Manual Tests Do Smell! Cataloging and Identifying Natural Language Test Smells.pdf" "Manual Tests Do Smell! Cataloging and Identifying Natural Language Test Smells"

Figura 6.10: Fragmento fichero *fileList.txt* 2023

error.txt

En este caso, decidí que la aplicación facilitara un documento donde se exponen las URLs que se han podido verificar por cada artículo, además de indicar el número de aquellas cuya verificación no ha resultado satisfactoria.

Articulo: Divide and Conquer the EmpiRE: A Community-Maintainable Knowledge Graph of Empirical Research in Requirements Engineering
---

Link identificado: https://www.w3.org/2000/01/rdf-schema (200)
Link identificado: https://orkg.org/comparison/R255464 (200)
Link identificado: https://hi-knowledge.org/ (200)
Link identificado: https://orkg.org/comparison/R112387 (200)
Link identificado: https://orkg.org/comparison/R114155 (200)
Link identificado: https://kgbook.org/ (200)
Link identificado: https://covid-aqs.fz-juelich.de (200)
Link identificado: https://zenodo.org/records/4456034 (200)
Link identificado: https://zenodo.org/records/4455951 (200)
Link identificado: https://zenodo.org/records/8083529 (200)
Link identificado: https://www.w3.org/TR/2017/REC-shacl-20170720/ (200)
Link identificado: https://paperswithcode.com/about (200)
Link identificado: https://www.w3.org/TR/2013/REC-sparql11-query-20130321/ (200)

Faltan por recuperar 3 enlaces en el fichero: ./cache/Divide and Conquer the EmpiRE_ A Community-Maintainable Knowledge Graph of Empirical Research in Requirements Engineering.tx

Figura 6.11: Fragmento fichero *error.txt* 2023

6.3. Comparación de los resultados obtenidos

A continuación, se van a comparar los resultados obtenidos en el análisis de la desaparición de la información en internet, realizado con las revistas del año 2013 y 2023.

En cuanto al número de títulos que hay en cada fichero .xml de cada revista tenemos que en el año 2013 contamos con 57 títulos y en 2023 con 38 títulos.

Podemos comprobar que en el año 2013 en la base de datos, concretamente en la tabla *downloaded_pdfs* se indica, que se han encontrado 49 URLs en total:

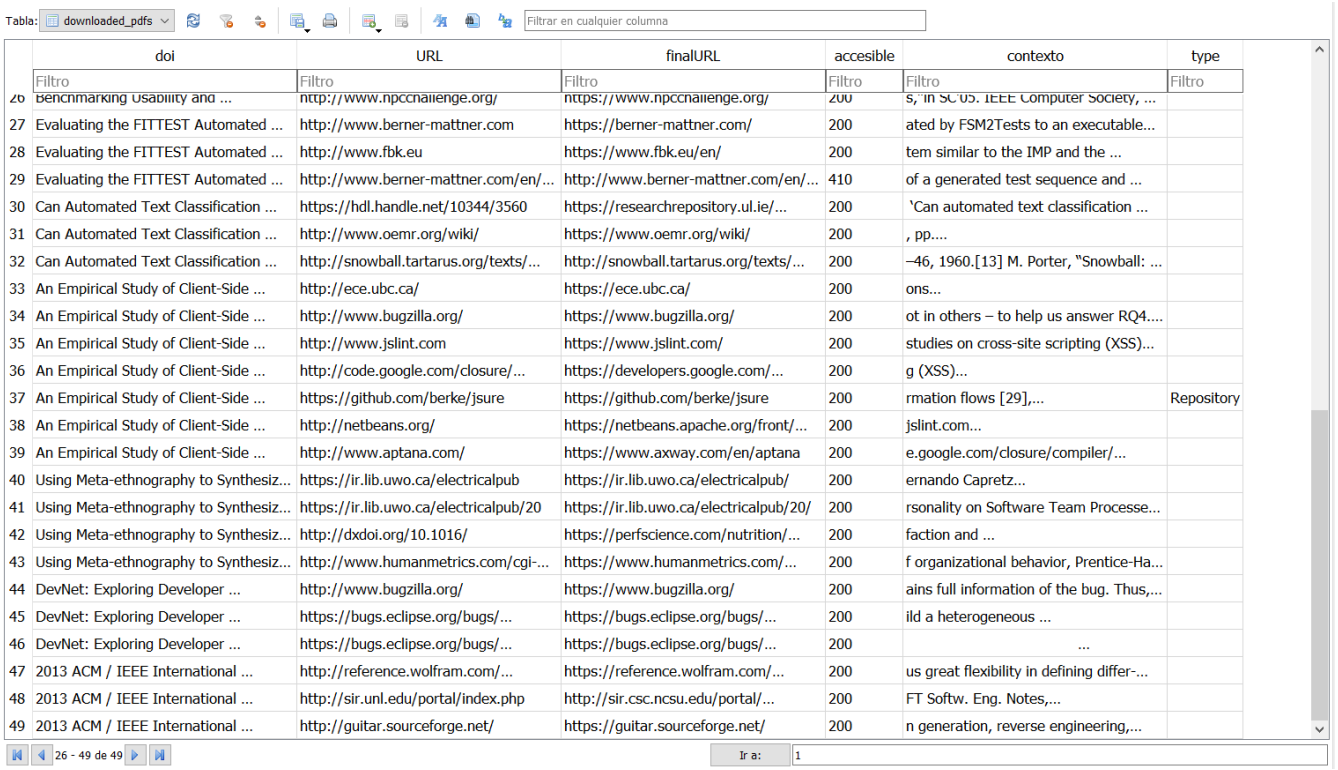


Tabla: downloaded_pdfs — Filtrar en cualquier columna

	doi	URL	finalURL	accesible	contexto	type
	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro
26	Benchmarking Usability and ...	http://www.npcchallenge.org/	https://www.npcchallenge.org/	200	s, in SC 05. IEEE Computer Society, ...	
27	Evaluating the FITTEST Automated ...	http://www.berner-mattner.com	https://berner-mattner.com/	200	ated by FSM2Tests to an executable...	
28	Evaluating the FITTEST Automated ...	http://www.fbk.eu	https://www.fbk.eu/en/	200	tem similar to the IMP and the ...	
29	Evaluating the FITTEST Automated ...	http://www.berner-mattner.com/en/...	http://www.berner-mattner.com/en/...	410	of a generated test sequence and ...	
30	Can Automated Text Classification ...	https://hdl.handle.net/10344/3560	https://researchrepository.ul.ie/...	200	'Can automated text classification ...	
31	Can Automated Text Classification ...	http://www.oemr.org/wiki/	https://www.oemr.org/wiki/	200	, pp....	
32	Can Automated Text Classification ...	http://snowball.tartarus.org/texts/...	http://snowball.tartarus.org/texts/...	200	–46, 1960.[13] M. Porter, "Snowball: ...	
33	An Empirical Study of Client-Side ...	http://ece.ubc.ca/	https://ece.ubc.ca/	200	ons...	
34	An Empirical Study of Client-Side ...	http://www.bugzilla.org/	https://www.bugzilla.org/	200	ot in others – to help us answer RQ4....	
35	An Empirical Study of Client-Side ...	http://www.jslint.com	https://www.jslint.com/	200	studies on cross-site scripting (XSS)...	
36	An Empirical Study of Client-Side ...	http://code.google.com/closure/...	https://developers.google.com/...	200	g (XSS)...	
37	An Empirical Study of Client-Side ...	https://github.com/berke/jsure	https://github.com/berke/jsure	200	rmation flows [29],...	Repository
38	An Empirical Study of Client-Side ...	http://netbeans.org/	https://netbeans.apache.org/front/...	200	jslint.com...	
39	An Empirical Study of Client-Side ...	http://www.aptana.com/	https://www.axway.com/en/aptana	200	e.google.com/closure/compiler/...	
40	Using Meta-ethnography to Synthesiz...	https://ir.lib.uwo.ca/electricalpub	https://ir.lib.uwo.ca/electricalpub/	200	ernando Capretz...	
41	Using Meta-ethnography to Synthesiz...	https://ir.lib.uwo.ca/electricalpub/20	https://ir.lib.uwo.ca/electricalpub/20/	200	rsonality on Software Team Processe...	
42	Using Meta-ethnography to Synthesiz...	http://dxdoi.org/10.1016/	https://perfscience.com/nutrition/...	200	faction and ...	
43	Using Meta-ethnography to Synthesiz...	http://www.humanmetrics.com/cgi-...	https://www.humanmetrics.com/...	200	f organizational behavior, Prentice-Ha...	
44	DevNet: Exploring Developer ...	http://www.bugzilla.org/	https://www.bugzilla.org/	200	ains full information of the bug. Thus,...	
45	DevNet: Exploring Developer ...	https://bugs.eclipse.org/bugs/...	https://bugs.eclipse.org/bugs/...	200	ild a heterogeneous ...	
46	DevNet: Exploring Developer ...	https://bugs.eclipse.org/bugs/...	https://bugs.eclipse.org/bugs/...	200	...	
47	2013 ACM / IEEE International ...	http://reference.wolfram.com/...	https://reference.wolfram.com/...	200	us great flexibility in defining differ-...	
48	2013 ACM / IEEE International ...	http://sir.unl.edu/portal/index.php	http://sir.csc.ncsu.edu/portal/...	200	FT Softw. Eng. Notes,...	
49	2013 ACM / IEEE International ...	http://guitar.sourceforge.net/	https://guitar.sourceforge.net/	200	n generation, reverse engineering,...	

26 - 49 de 49 — Ir a: 1

Figura 6.12: Número de URLs - Caso de estudio 2013

Sin embargo, al filtrar las URLs por código devuelto por el servidor: 200, me encontré con que, en el caso de la revista de 2013, el número de URLs existentes es 44:

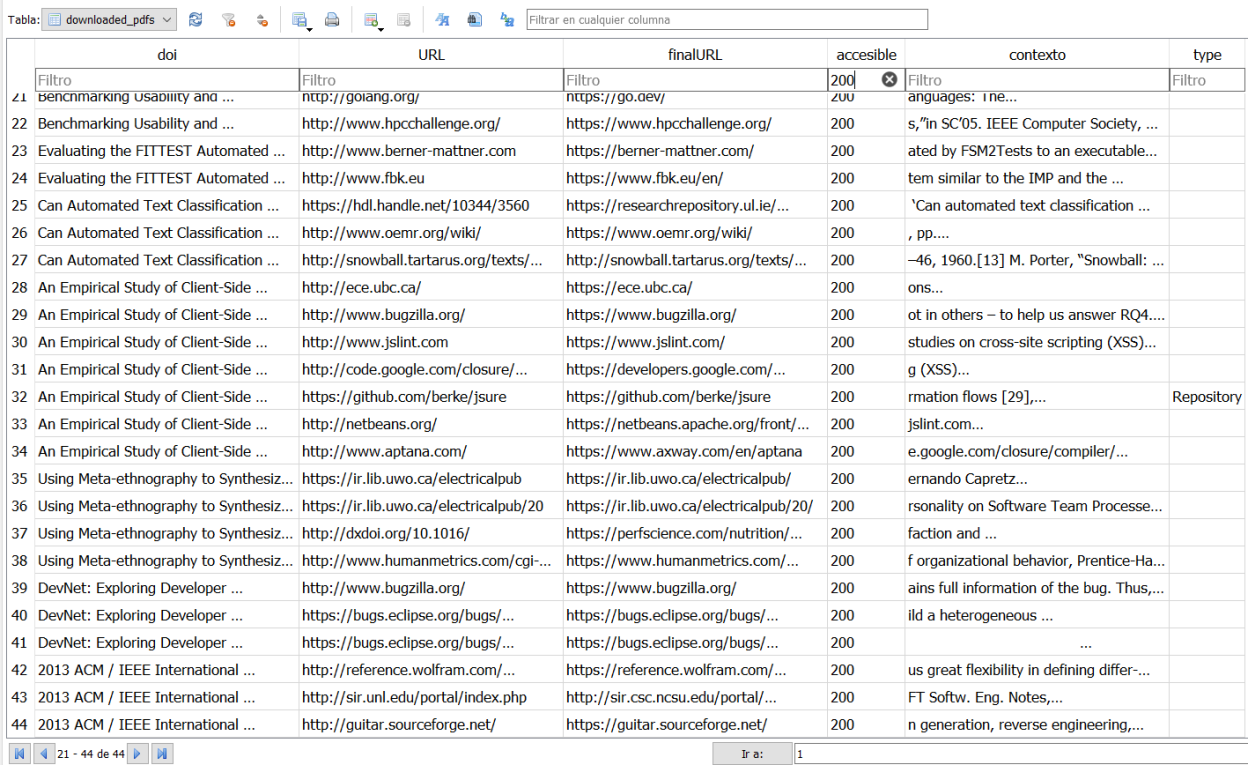


Tabla: downloaded_pdfs — Filtrar en cualquier columna

	doi	URL	finalURL	accesible	contexto	type
	Filtro	Filtro	Filtro	200	Filtro	Filtro
21	Benchmarking Usability and ...	http://golang.org/	https://go.dev/	200	anguages: The...	
22	Benchmarking Usability and ...	http://www.hpcchallenge.org/	https://www.hpcchallenge.org/	200	s,"in SC'05. IEEE Computer Society, ...	
23	Evaluating the FITTEST Automated ...	http://www.berner-mattner.com	https://berner-mattner.com/	200	ated by FSM2Tests to an executable...	
24	Evaluating the FITTEST Automated ...	http://www.fbk.eu	https://www.fbk.eu/en/	200	tem similar to the IMP and the ...	
25	Can Automated Text Classification ...	https://hdl.handle.net/10344/3560	https://researchrepository.ul.ie/...	200	'Can automated text classification ...	
26	Can Automated Text Classification ...	http://www.oemr.org/wiki/	https://www.oemr.org/wiki/	200	, pp....	
27	Can Automated Text Classification ...	http://snowball.tartarus.org/texts/...	http://snowball.tartarus.org/texts/...	200	–46, 1960.[13] M. Porter, "Snowball: ...	
28	An Empirical Study of Client-Side ...	http://ece.ubc.ca/	https://ece.ubc.ca/	200	ons...	
29	An Empirical Study of Client-Side ...	http://www.bugzilla.org/	https://www.bugzilla.org/	200	ot in others – to help us answer RQ4....	
30	An Empirical Study of Client-Side ...	http://www.jslint.com	https://www.jslint.com/	200	studies on cross-site scripting (XSS)...	
31	An Empirical Study of Client-Side ...	http://code.google.com/closure/...	https://developers.google.com/...	200	g (XSS)...	
32	An Empirical Study of Client-Side ...	https://github.com/berke/jsure	https://github.com/berke/jsure	200	rmation flows [29],...	Repository
33	An Empirical Study of Client-Side ...	http://netbeans.org/	https://netbeans.apache.org/front/...	200	jslint.com...	
34	An Empirical Study of Client-Side ...	http://www.aptana.com/	https://www.axway.com/en/aptana	200	e.google.com/closure/compiler/...	
35	Using Meta-ethnography to Synthesiz...	https://ir.lib.uwo.ca/electricalpub	https://ir.lib.uwo.ca/electricalpub/	200	ernando Capretz...	
36	Using Meta-ethnography to Synthesiz...	https://ir.lib.uwo.ca/electricalpub/20	https://ir.lib.uwo.ca/electricalpub/20/	200	rsonality on Software Team Processe...	
37	Using Meta-ethnography to Synthesiz...	http://dxdoi.org/10.1016/	https://perfscience.com/nutrition/...	200	faction and ...	
38	Using Meta-ethnography to Synthesiz...	http://www.humanmetrics.com/cgi-...	https://www.humanmetrics.com/...	200	f organizational behavior, Prentice-Ha...	
39	DevNet: Exploring Developer ...	http://www.bugzilla.org/	https://www.bugzilla.org/	200	ains full information of the bug. Thus,...	
40	DevNet: Exploring Developer ...	https://bugs.eclipse.org/bugs/...	https://bugs.eclipse.org/bugs/...	200	ild a heterogeneous ...	
41	DevNet: Exploring Developer ...	https://bugs.eclipse.org/bugs/...	https://bugs.eclipse.org/bugs/...	200	...	
42	2013 ACM / IEEE International ...	http://reference.wolfram.com/...	https://reference.wolfram.com/...	200	us great flexibility in defining differ-...	
43	2013 ACM / IEEE International ...	http://sir.unl.edu/portal/index.php	http://sir.csc.ncsu.edu/portal/...	200	FT Softw. Eng. Notes,...	
44	2013 ACM / IEEE International ...	http://guitar.sourceforge.net/	https://guitar.sourceforge.net/	200	n generation, reverse engineering,...	

21 - 44 de 44 — Ir a: 1

Figura 6.13: Número de URLs existentes - Caso de estudio 2013

Por lo tanto, comparando los números obtenidos, se llega a la conclusión de que aproximadamente el 10.20 % de las URLs que se han extraído de los artículos de la revista de 2013 no existen.

Por otro lado, en el año 2023, se puede comprobar, gracias a la base de datos, que se han identificado 220 URLs:

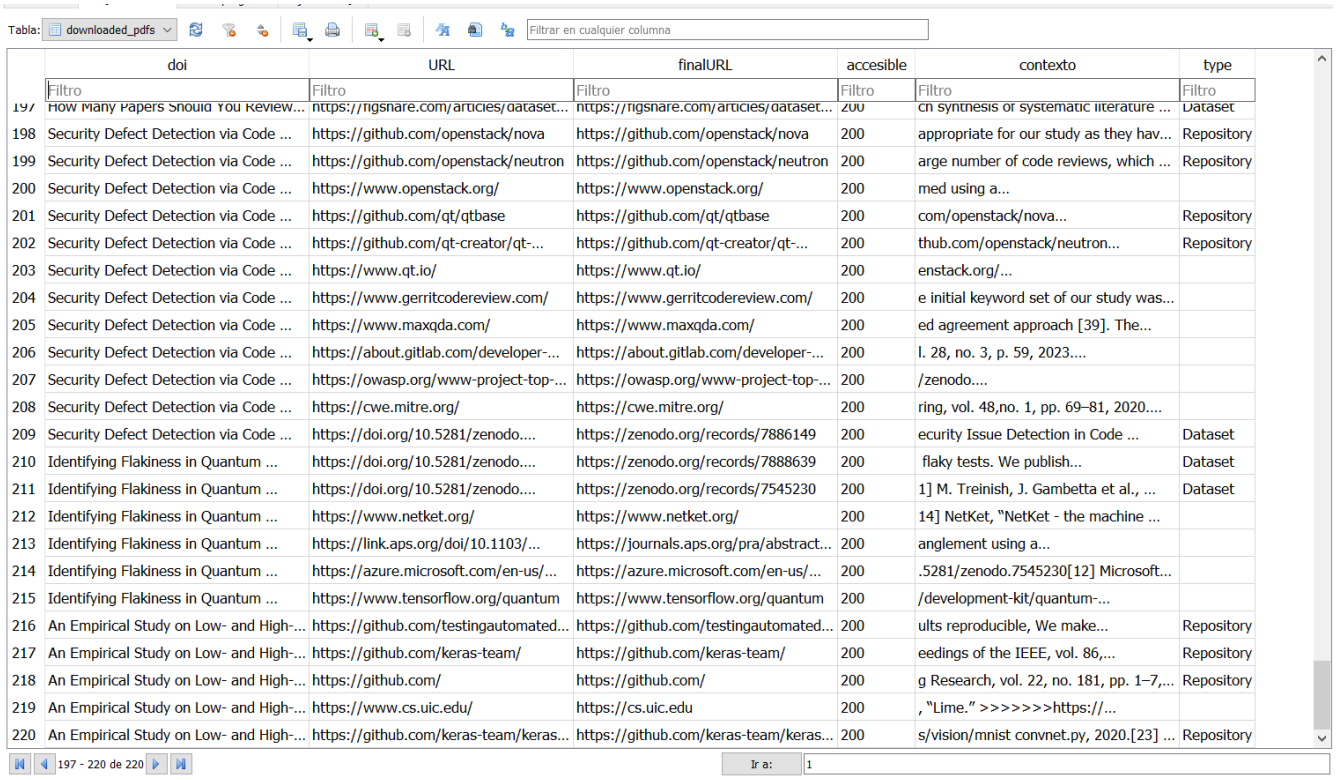


Tabla: downloaded_pdfs — Filtrar en cualquier columna

	doi	URL	finalURL	accesible	contexto	type
	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro
197	How Many Papers Should You Review...	https://figshare.com/articles/dataset...	https://figshare.com/articles/dataset...	200	ch synthesis of systematic literature ...	Dataset
198	Security Defect Detection via Code ...	https://github.com/openstack/nova	https://github.com/openstack/nova	200	appropriate for our study as they hav...	Repository
199	Security Defect Detection via Code ...	https://github.com/openstack/neutron	https://github.com/openstack/neutron	200	arge number of code reviews, which ...	Repository
200	Security Defect Detection via Code ...	https://www.openstack.org/	https://www.openstack.org/	200	med using a...	
201	Security Defect Detection via Code ...	https://github.com/qt/qtbase	https://github.com/qt/qtbase	200	com/openstack/nova...	Repository
202	Security Defect Detection via Code ...	https://github.com/qt-creator/qt-...	https://github.com/qt-creator/qt-...	200	thub.com/openstack/neutron...	Repository
203	Security Defect Detection via Code ...	https://www.qt.io/	https://www.qt.io/	200	enstack.org/...	
204	Security Defect Detection via Code ...	https://www.gerritcodereview.com/	https://www.gerritcodereview.com/	200	e initial keyword set of our study was...	
205	Security Defect Detection via Code ...	https://www.maxqda.com/	https://www.maxqda.com/	200	ed agreement approach [39]. The...	
206	Security Defect Detection via Code ...	https://about.gitlab.com/developer-...	https://about.gitlab.com/developer-...	200	l. 28, no. 3, p. 59, 2023....	
207	Security Defect Detection via Code ...	https://owasp.org/www-project-top-...	https://owasp.org/www-project-top-...	200	/zenodo....	
208	Security Defect Detection via Code ...	https://cwe.mitre.org/	https://cwe.mitre.org/	200	ring, vol. 48,no. 1, pp. 69–81, 2020....	
209	Security Defect Detection via Code ...	https://doi.org/10.5281/zenodo....	https://zenodo.org/records/7886149	200	ecurity Issue Detection in Code ...	Dataset
210	Identifying Flakiness in Quantum ...	https://doi.org/10.5281/zenodo....	https://zenodo.org/records/7888639	200	flaky tests. We publish...	Dataset
211	Identifying Flakiness in Quantum ...	https://doi.org/10.5281/zenodo....	https://zenodo.org/records/7545230	200	1] M. Treinish, J. Gambetta et al., ...	Dataset
212	Identifying Flakiness in Quantum ...	https://www.netket.org/	https://www.netket.org/	200	14] NetKet, "NetKet - the machine ...	
213	Identifying Flakiness in Quantum ...	https://link.aps.org/doi/10.1103/...	https://journals.aps.org/pra/abstract...	200	anglement using a...	
214	Identifying Flakiness in Quantum ...	https://azure.microsoft.com/en-us/...	https://azure.microsoft.com/en-us/...	200	.5281/zenodo.7545230[12] Microsoft...	
215	Identifying Flakiness in Quantum ...	https://www.tensorflow.org/quantum	https://www.tensorflow.org/quantum	200	/development-kit/quantum-...	
216	An Empirical Study on Low- and High-...	https://github.com/testingautomated...	https://github.com/testingautomated...	200	ults reproducible, We make...	Repository
217	An Empirical Study on Low- and High-...	https://github.com/keras-team/	https://github.com/keras-team/	200	eedings of the IEEE, vol. 86,...	Repository
218	An Empirical Study on Low- and High-...	https://github.com/	https://github.com/	200	g Research, vol. 22, no. 181, pp. 1–7,...	Repository
219	An Empirical Study on Low- and High-...	https://www.cs.uic.edu/	https://cs.uic.edu	200	, "Lime." >>>>>>>https://...	
220	An Empirical Study on Low- and High-...	https://github.com/keras-team/keras...	https://github.com/keras-team/keras...	200	s/vision/mnist convnet.py, 2020.[23] ...	Repository

197 - 220 de 220 — Ir a: 1

Figura 6.14: Número de URLs - Caso de estudio 2023

Sin embargo, al filtrar con el código devuelto por el servidor, con el valor 200, se nos indica que se han verificado correctamente 208 URLs.

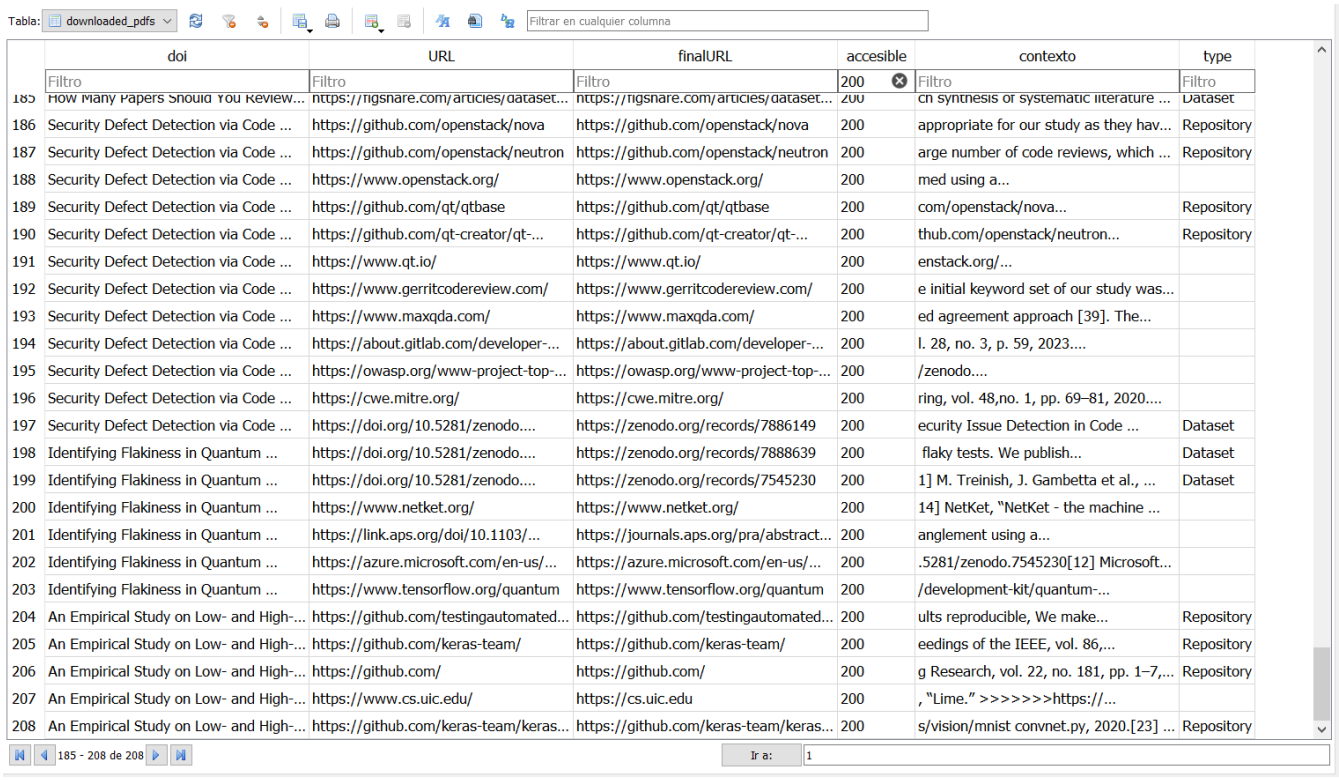


Tabla: downloaded_pdfs — Filtrar en cualquier columna

	doi	URL	finalURL	accesible	contexto	type
	Filtro	Filtro	Filtro	200	Filtro	Filtro
185	How Many Papers Should You Review...	https://figshare.com/articles/dataset...	https://figshare.com/articles/dataset...	200	ch synthesis of systematic literature ...	Dataset
186	Security Defect Detection via Code ...	https://github.com/openstack/nova	https://github.com/openstack/nova	200	appropriate for our study as they hav...	Repository
187	Security Defect Detection via Code ...	https://github.com/openstack/neutron	https://github.com/openstack/neutron	200	arge number of code reviews, which ...	Repository
188	Security Defect Detection via Code ...	https://www.openstack.org/	https://www.openstack.org/	200	med using a...	
189	Security Defect Detection via Code ...	https://github.com/qt/qtbase	https://github.com/qt/qtbase	200	com/openstack/nova...	Repository
190	Security Defect Detection via Code ...	https://github.com/qt-creator/qt-...	https://github.com/qt-creator/qt-...	200	thub.com/openstack/neutron...	Repository
191	Security Defect Detection via Code ...	https://www.qt.io/	https://www.qt.io/	200	enstack.org/...	
192	Security Defect Detection via Code ...	https://www.gerritcodereview.com/	https://www.gerritcodereview.com/	200	e initial keyword set of our study was...	
193	Security Defect Detection via Code ...	https://www.maxqda.com/	https://www.maxqda.com/	200	ed agreement approach [39]. The...	
194	Security Defect Detection via Code ...	https://about.gitlab.com/developer-...	https://about.gitlab.com/developer-...	200	l. 28, no. 3, p. 59, 2023....	
195	Security Defect Detection via Code ...	https://owasp.org/www-project-top-...	https://owasp.org/www-project-top-...	200	/zenodo....	
196	Security Defect Detection via Code ...	https://cwe.mitre.org/	https://cwe.mitre.org/	200	ring, vol. 48,no. 1, pp. 69–81, 2020....	
197	Security Defect Detection via Code ...	https://doi.org/10.5281/zenodo....	https://zenodo.org/records/7886149	200	ecurity Issue Detection in Code ...	Dataset
198	Identifying Flakiness in Quantum ...	https://doi.org/10.5281/zenodo....	https://zenodo.org/records/7888639	200	flaky tests. We publish...	Dataset
199	Identifying Flakiness in Quantum ...	https://doi.org/10.5281/zenodo....	https://zenodo.org/records/7545230	200	1] M. Treinish, J. Gambetta et al., ...	Dataset
200	Identifying Flakiness in Quantum ...	https://www.netket.org/	https://www.netket.org/	200	14] NetKet, "NetKet - the machine ...	
201	Identifying Flakiness in Quantum ...	https://link.aps.org/doi/10.1103/...	https://journals.aps.org/pra/abstract...	200	anglement using a...	
202	Identifying Flakiness in Quantum ...	https://azure.microsoft.com/en-us/...	https://azure.microsoft.com/en-us/...	200	.5281/zenodo.7545230[12] Microsoft...	
203	Identifying Flakiness in Quantum ...	https://www.tensorflow.org/quantum	https://www.tensorflow.org/quantum	200	/development-kit/quantum-...	
204	An Empirical Study on Low- and High-...	https://github.com/testingautomated...	https://github.com/testingautomated...	200	ults reproducible, We make...	Repository
205	An Empirical Study on Low- and High-...	https://github.com/keras-team/	https://github.com/keras-team/	200	eedings of the IEEE, vol. 86,...	Repository
206	An Empirical Study on Low- and High-...	https://github.com/	https://github.com/	200	g Research, vol. 22, no. 181, pp. 1–7,...	Repository
207	An Empirical Study on Low- and High-...	https://www.cs.uic.edu/	https://cs.uic.edu	200	, "Lime." >>>>>>>https://...	
208	An Empirical Study on Low- and High-...	https://github.com/keras-team/keras...	https://github.com/keras-team/keras...	200	s/vision/mnist convnet.py, 2020.[23] ...	Repository

185 - 208 de 208 — Ir a: 1

Figura 6.15: Número de URLs existentes - Caso de estudio 2023

Por lo tanto, comparando los números obtenidos, se llega a la conclusión de que aproximadamente el 5.45 % de las URLs que se han extraído de los artículos de la revista de 2023 no existen.

Una vez que se cuenta con los resultados de las revistas de ambos años, se puede concluir que en 2023 el porcentaje de URLs que ya no están disponibles es bastante inferior al de 2013. Sin embargo, hay que tener en cuenta que 2013 corresponde a la década anterior y todavía no ha pasado ni un año desde 2023. Por ello, considero que, en este caso de estudio, el porcentaje de desaparición de la información en internet es superior en 2023 que en 2013.

Capítulo 7

Análisis de impacto

En este capítulo se trata el impacto de los resultados obtenidos en relación con Objetivos de Desarrollo Sostenible (ODS).

7.1. Educación de calidad (ODS 4)

La aplicación desarrollada ayuda a proporcionar una educación de calidad. Esto se debe a que al poder realizar un análisis de cómo va desapareciendo la información en internet, se pueden tomar medidas para reducir al mínimo dicha desaparición. De esta forma, las bibliografías estarán más completas y se podrá relaizar reproducciones de los trabajo de forma más verídica y sencilla.

7.2. Industria, innovación e infraestructura (ODS 9)

La aplicación promueve una innovación en relación con el almacenamiento de la información, ya que como se han podido ver en los resultados obtenidos, uno de los motivos de desaparición que se han identificado es los sitios web donde se guarda dicha información. De esta forma, se busca elaborar sitios oficiales y verificados para poder almacenar la documentación oficial.

7.3. Paz, justicia e instituciones sólidas (ODS 16)

La aplicación desarrollada contribuye a la transparencia de la información y al poder acceder a ella. Esto ayuda sobre todo a las instituciones, ya que al tener acceso a la información fortalece la capacidad de desarrollo, investigación y toma de decisiones de ellas.

Capítulo 8

Conclusiones y trabajo futuro

En este trabajo de fin de curso se ha investigado cómo desaparece la información que se encuentra almacenada en internet. Para ello, se han analizado artículos de revistas obtenidas de la biblioteca digital DBLP de distintos años. El resultado de este ánalisis abre el horizonte de que cuanto más tiempo transcurre desde la publicación de un artículo, la información citada en él es más propensa a desaparecer.

Gracias a los resultados obtenidos podemos comentar que, se debe de tener en cuenta que, la información que deja de estar disponible se relaciona con encontrarse almacenada en sitios específicos en la red, como repositorios o servicios de cloud. Además, esto nos indica que uno de los motivos de la desaparición, es el hecho de que se almacena la información en servicios personales, en vez de estar disponibles en servicios públicos para que pueda ayudar a mantenerse a través del tiempo con mayor facilidad. También, otro motivo es la eliminación o actualización de dominios, además de redirecciones, que pueden afectar a la contrucción de las URLs.

Sin embargo, pese a los resultados que se han obtenido la aplicación cuenta con una serie de limitaciones que pueden variar el análisis realizado. Una de esas limitaciones es el problema de los denominados *linkrot* que implican la disociación entre las direcciones web y su contenido, que son difíciles de identificar. Otra de las limitaciones importantes es que las direcciones web identifiquen que es una aplicación la que está intentando acceder a ella, y por lo tanto la clasifica como un *bot*, restringindo el acceso. También, existe el problema de identificar de forma correcta el motivo de la desaparición de dicha información.

Para trabajo futuro se debería mejorar el proceso de camuflar la aplicación para que no sea detectada y se pueda reducir al minímo el problema de resultado falso de verificaciones de URLs por acceso denegado. Además, sería muy beneficioso afinar aún más los distintos dominios utilizados para la clasificación de los enlaces en distintas categorías, para realizarlo con más precisión y veracidad.

La realización de este trabajo de fin de curso me ha permitido adquirir conocimientos sobre la manipulación de archivos en formato PDF ya que estos cuentan

con estructuras específicas. Además, he podido enfrentarme a el manejo de enlaces, como por ejemplo las técnicas para extraerlos sin producir una rotura de ellos. También, he podido darme cuenta del problema que supone que la información desaparezca, además de el propio hecho de que va desapareciendo sin que nos demos cuenta.

En conclusión, este proyecto ha ayudado a realizar una investigación y un análisis de cómo desaparece la información en internet y el periodo aproximado de esa desaparicón. Debido a esto espero que se pueda realizar una concienciación de la importancia de mantener la información disponible, y que se tomen medidas para ello.

Bibliografía

[1] Apache Software Foundation. (2024) Apache Maven. [Online]. Available: https://maven.apache.org/

[2] Genbeta. (2011) Introducción a Maven. [Online]. Available: https://www.genbeta.com/desarrollo/introduccion-a-maven

[3] Universidad de La Coruña. (2023) dblp: una base de datos fantástica, gratuita y especializada exclusivamente en computación. [Online]. Available: https://www.udc.es/es/biblioteca.fic/recursos_informacion/bases_de_datos-00001/dblp/

[4] Universidad de Las Palmas de Gran Canaria. (2020) Google Académico: acceso al texto completo ULPGC. [Online]. Available: https://biblioteca.ulpgc.es/google_academico#:~:text=Google%20Acad%C3%A9mico%20es%20un%20buscador,universidades%20y%20otras%20organizaciones%20acad%C3%A9micas.

[5] Universidad Autónoma de Madrid. (2023) Google Académico, paso a paso. [Online]. Available: https://biblioteca.ulpgc.es/google_academico#:~:text=Google%20Acad%C3%A9mico%20es%20un%20buscador,universidades%20y%20otras%20organizaciones%20acad%C3%A9micas.

[6] Daniel Albarracín Morales. (2023) Introducción SerpApi. [Online]. Available: https://www.kaggle.com/code/danielalbarracinm/introducci-n-serpapi

[7] Apache Software Foundation. (2024) Apache PDFBox. [Online]. Available: https://pdfbox.apache.org./

[8] Jsoup. (2023) Jsoup Java HTML Parser. [Online]. Available: https://jsoup.org/

[9] IBM. (2022) Formato JSON (JavaScript Object Notation). [Online]. Available: https://www.ibm.com/docs/es/baw/20.x?topic=formats-javascript-object-notation-json-format

[10] IBM. (2022) Propiedades y conversiones de tipos de datos del formato JSON. [Online]. Available: https://www.ibm.com/docs/es/baw/20.x?topic=format-json-properties-data-type-conversions

[11] J. Zittrain, K. Albert, and L. Lessig, "Perma: Scoping and addressing the problem of link and reference rot in legal citations," *Legal Information Management*, vol. 14, no. 2, pp. 88–99, 2014.

[12] D. G. Gomes, P. Pottier, R. Crystal-Ornelas, E. J. Hudgins, V. Foroughirad, L. L. Sánchez-Reyes, R. Turba, P. A. Martinez, D. Moreau, M. G. Bertram *et al.*, "Why don't we share data and code? perceived barriers and benefits to public archiving practices," *Proceedings of the Royal Society B*, vol. 289, no. 1987, p. 20221113, 2022.

[13] D. E. Ott, "Reference hygiene and death on the internet–decay, rot, half-life, deterioration, and corruption," *JSLS: Journal of the Society of Laparoscopic & Robotic Surgeons*, vol. 26, no. 1, 2022.

[14] J. Hennessey and S. X. Ge, "A cross disciplinary study of link decay and the effectiveness of mitigation techniques," in *BMC bioinformatics*, vol. 14. Springer, 2013, pp. 1–11.

Anexos

Apéndice A

Especificación de Requisitos

A.1. Introducción

Este documento detalla la Especificación de Requisitos Software (ERS) de una aplicación que extrae y analiza las URLS de pdfs descargados con el fin de conocer si existen o no.

A.1.1. Propósito del proyecto

La realización de esta aplicación tiene el propósito de comprobar si los enlaces utilizados en los artículos, para apoyar o justificar sus conclusiones, siguen existiendo.

Está relacionado con el llamado "Fraude científico". El objetivo principal es estudiar la cantidad de información científica que se pierde en internet, identificando los mecanismos y el tiempo en el que se producen, de forma automatizada.

Objetivos:

- Poder acceder a los PDFs de los artículos que se encuentran en DBLP.
- En los PDFs ser capaz de poder localizar los enlaces a las referencias utilizadas.
- Acceder a esos enlaces para comprobar si existen, y si no existen estimar su motivo de desaparición.

A.1.2. Definiciones, Acrónimos y Abreviaturas

API: Se trata de: Interfaz de Programación de Aplicaciones.
DOI: Digital Object Identifier. Se trata de un identificador único y permanente que contienen las publicaciones que se realizan en internet.

A.2. Requisitos Específicos Ciclo 1

En esta sección se enumeran los requisitos que el sistema tendrá que cumplir, tanto funcionales como no funcionales.

A.2.1. Requisitos Funcionales

Línea de comandos

- Req1. La aplicación se deberá lanzar desde la línea de comandos.
- Req2. La aplicación deberá aceptar como entrada un fichero xml.
- Req3. La aplicación deberá un código de éxito(0) si se procesa de forma correcta el fichero xml o un código de fallo(1) en caso contrario.

Obtener los títulos de los artículos desde un fichero xml de DBLP

- Req4. La aplicación deberá extraer los títulos de los artículos presentes en los ficheros xml de entrada de la base de datos DBLP.

Obtener URL de descarga del PDF con SerpApi de Google Scholar

- Req5. La aplicación deberá utilizar SerpApi de Google Scholar para buscar los títulos de los artículos y obtener sus JSON.
- Req6. La aplicación deberá obtener la URL de descarga de los PDF del JSON obtenido.

Descargar PDF desde URL

- Req7. La aplicación deberá descargar el PDF desde la URL obtenida a través de SerpApi.

Analizar el contenido del PDF y comprobar la existencia de las URLs de referencias

- Req8. La aplicación deberá leer el contenido del PDF descargado y extraer las URLs de referencia.
- Req9. La aplicación deberá verificar la existenci de las URLs extraídas sin considerar su contenido.
- Req10. La aplicación deberá asociar los resultados de la verificación con el PDF al que corresponden las URLs.

Caché local para evitar descargas duplicadas de PDFs

- Req11. La aplicación deberá mantener una caché local para evitar las descargas duplicadas de los PDFs.
- Req12. La aplicación deberá identificar a los PDFs por su título.

Almacenar los datos extraídos en una base de datos local

- Req13. La aplicación deberá guardar los datos extraídos en una base de datos local.
- Req14. La aplicación deberá clasificar esos datos por el DOI del pdf, la URL verificada y el resultado de la verificación de la URL.

A.2.2. Requisitos no funcionales

Bloqueo de servidores debido a bots

- Req15. La aplicación deberá implementar una estrategia para evitar que los servidores la consideren un bot.

Fiabilidad en la detección y manejo de errores

- Req16. La aplicación deberá demostrar alta fiabilidad a la hora de detectar y gestionar errores.

Idempotencia del proceso

- Req17. El proceso deberá ser idempotente, lo que quiere decir, que la información asociada a un PDF se sobreescriba si hay alguna modificación.

Manejo de captchas de SerpApi y Google Scholar

- Req18. La aplicación deberá ser capaz de manejar los captchas de forma eficaz generados por SerpApi y Google Scholar. garantizando que no se produzcan bloqueos en el flujo de trabajo automatizado.

Arquitectura JAVA

- Req19. La aplicación contará con una arquitectura JAVA, lo que implica que el diseño y el desarrollo deben estar dentro de los estándares de esta tecnología.

A.3. Requisitos Específicos Ciclo 2

En esta sección se especifican los requisitos que se añaden a la aplicación.

A.3.1. Requisitos Funcionales

Se ha añadido el siguiente requisito:

Exportación de datos

- Req20. La aplicación deberá poder exportar en formato CSV los datos que se encuentran en la base de datos.

A.3.2. Requisitos no funcionales

En cuanto a los requisitos no funcionales se ha añadido el siguiente requisito:

Análisis de datos

- Req21. La aplicación deberá contar con un análisis de los datos exportados en formato CSV.

A.4. Requisitos Específicos Ciclo 3

En esta sección se comentan los requisitos que se han añadido a la aplicación.

A.4.1. Requisitos Funcionales

Descarga del PDF

- Req22. La aplicación deberá mostrar el número de títulos y el número de enlaces encontrados.
- Req23. La aplicación deberá permitir forzar la descarga de un artículo aunque ya se haya descargado previamente.
- Req24. La aplicación deberá bloquear la búsqueda del enlace de descarga de un artículo en formato PDF si ya se ha realizado esa búsqueda con anterioridad.

Contenido del PDF y análisis de las URLs extraídas

- Req25. La aplicación deberá permitir transformar un PDF a texto.
- Req26. La aplicación deberá permitir analizar nuevamente los enlaces encontrados.
- Req27. La aplicación deberá permitir comprobar la accesibilidad de los enlaces directamente en la base de datos.
- Req28. La aplicación deberá permitir eliminar enlaces obsoletos de la base de datos.
- Req29. La aplicación deberá permitir clasificar o categorizar las URLs extraídas.

A.4.2. Requisitos no funcionales

Ejecución

- Req30. La aplicación deberá permitir su ejecución de dos formas. Una de ellas es, a través de la línea de comandos, ejecutando cada clase de forma individual. La otra es, ejecutando una clase que se encarga de activar el flujo de ejecución.

Apéndice B

Diseño del programa

B.1. Ciclo 1

Este ciclo cuenta con dos versiones del programa: la versión con y sin base de datos.

B.1.1. Versión sin base de datos

En esta versión el programa cuenta con las clases:

- LeerXml.java
- ObtenerPdfConSerpApi.java
- DescargarPdfCache.java
- MostrarYComprobarEnlaces.java

A continuación se explicará el propósito de cada clase así como los métodos que incluyen.

LeerXml.java

Esta clase se encarga de extraer el nombre de los títulos de los artículos que se encuentran en el fichero .xml obtenido de DBLP, el cuál, se debe indicar al ejecutarla proprocionando su URL.

Métodos:

- obtenerTitulos: establece una conexión con la URL del fichero .xml de DBLP a través de JSOUP con Jsoup.connect(url). Después, guarda el contenido de la URL en un documento con .get(), sobre el cuál se itera para poder obtener los títulos de los artículos utilizando un forEach. Esto se logra extrayendo el contenido de las etiquetas <title>, suprimiendo el punto final de los títulos y guardándolos en un ArrayList llamado "titulos".

ObtenerPdfConSerpApi.java

Esta clase se encarga de obtener, a partir de los títulos obtenidos en **LeerXml.java**, las URL de descarga de los artículos en PDF, con un formato de salida:
titulo artículo y *URL descarga*.

Métodos:

- obtenerEnlacePDF: se establece una conexión de tipo *OkHttpClient*. Se codifican los títulos en un formato adecuado para poder ser utilizado en la URL de búsqueda de SerpApi de Google Scholar. Si se encuentra una URL para el título del artículo proporcionado, se imprime solo esa URl, y si no, se imprime todo el JSON obtenido gracias a SerpApi, para poder buscar la URL de forma manual.
- obtenerPrimerLink: se encarga de obtener el primer enlace de descarga del artículo, en formato PDF, que aparece en la sección *resources* del JSON de dicho artículo obtenido gracias a SerApi. Este método se utiliza en *obtenerEnlacePDF*.

DescargarPdfCache.java

Esta clase se encarga de descargar, en formato PDF, el artículo gracias al título y a la URL de descarga obtenida de **ObtenerPdfConSerpApi.java**, sólo si no ha sido descargado previamente.

Métodos:

- descargarPDF: lee el contenido presente en la URL utilizando *BufferedInputStream* y después lo escribe en un fichero de destino gracias a *FileOutputStream*. Con esto se logra descargar el PDF desde una URL.
- archivoExiste: este método se encarga de mirar si un archivo ya existe en el destino para evitar descargar el PDF más de una vez. Con *Path* tenemos el destino y comprobamos si ya existe con *Files.exists(path)*.
- limpiarNombreArticulo: se encarga de cambiar los símbolos que pueden causar problemas en los títulos de los artículos por una barra baja.

MostrarYComprobarEnlaces.java

Esta clase se encarga de buscar en el artículo en formato PDF las URL que aparecen como referencias. Una vez que las encuentra intenta entrar en ellas, indicando el código de respuesta del servidor, por lo que nos permite saber si las URLs existen o no.

Métodos:

- en el main se utiliza *PDFTextStripper* para poder buscar en el PDF las URLs que aparecen como referencias.
- verificarExistencia: con JSOUP se establece una conexión con las URLs encontradas, y con *.statusCode()* se obtiene el mensaje devuelto por el servidor.

B.1.2. Versión con base de datos

En esta versión, el programa cuenta con una base de datos para almacenar los resultados obtenidos.

Las clases **LeerXml.java** y **ObtenerPdfConSerpApi.java** se mantienen exactamente igual que en la versión sin base de datos.

En cuanto a la clase **DescargarPdfCache.java** en el *main* se inicializa la base de datos, si no ha sido creada previamente. Se comprueba que no se encuentre el PDF a descargar en la base de datos gracias a *.isNombrePDFExistente(tituloArticulo)*, y si no se encuentra se procede a descargar el PDF y se añade el nombre a la tabla *nombre_pdfs*.

En relación a la clase **MostrarYComprobarEnlaces.java** se añade un método:

- guardarVerificacionEnBaseDeDatos: como su nombre indica, se encarga de guardar las verificaciones de las URLs encontradas en las referencias de los artículos, en la base de datos. Utiliza un objeto *Pattern* que representa el patrón de símbolos de las URLs. Además, cuenta con un objeto *Matcher* que se encarga de encontrar las coincidencias del patrón en el texto. Después, llama al método *verificarExistencia* para obtener el código de respuesta del servidor, y con *.insertPDF(doi, enlace, codigoRespuesta)*, el cuál se verá a continuación, se introducen los datos en la base de datos.

Además, se ha añadido una clase que se encarga de manejar la base de datos llamada **ConexionBaseDeDatos.java**.

ConexionBaseDeDatos.java

Métodos:

- Al inicio se establece que en la tabla *nombre_pdfs* solo se puede introducir una vez el nombre de un artículo. Sin embargo, en la tabla *downloaded_pdfs* se pueden introducir más de una vez, ya que, un PDF puede tener más de una URL como referencia.
- initializeDatabase: se encarga de inicializar las dos tablas de la base de datos : *nombre_pdfs* y *downloaded_pdfs*.
- insertPDF: se encarga de establecer el formato de la tabla *downloaded_pdfs*, utilizando *PreparedStatement* y *Connection*.

- insertNombrePDF: se encarga de establecer el formato de la tabla *nombre_pdfs*, utilizando *PreparedStatement* y *Connection*.
- isNombrePDFExistente: se encarga de consultar a la base de datos si el nombre del PDF ya existe previamente en la tabla *nombre_pdfs*.
- isPDFDownloaded: se encarga de consultar a la base de datos si el nombre del PDF ya existe previamente en la tabla *downloaded_pdfs*.

B.2. Ciclo 2

En este ciclo se han añadido una clase **RunEntireFlow.java** y un script de python, para introducir mejoras en el programa.

En cuanto a la clase **RunEntireFlow.java** se encarga de llamar a las distintas clases que conforman al programa con el objetivo de que se ejecuten de seguido, sin necesidad de ejecutar cada clase de forma individual. Para ello, fue necesario la modificación de las clases para poder enlazar la entrada de una clase con la salida de la anterior.

Flujo de datos entre clases:

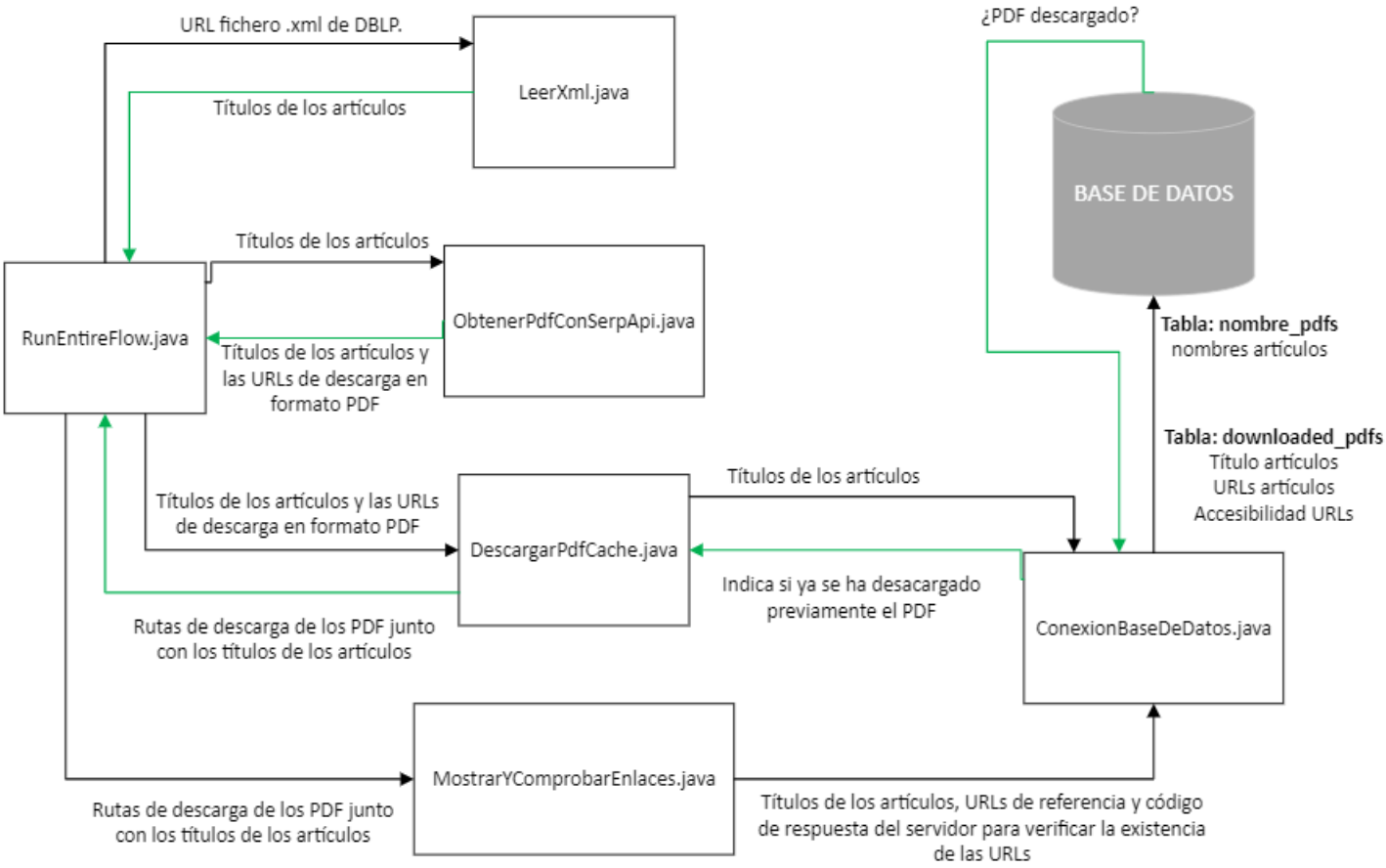


Figura B.1: Diagrama conexión de clases ciclo 2.

Para hacer posible el intercambio de datos entre las clases tal y como se observa en la imagen anterior, ha sido necesario modificar las entradas y salidas de las clases de la manera que se muestra a continuación.

LeerXml.java

La entrada continúa siendo la URL que contiene el fichero .xml de DBLP obtenida desde la línea de comandos. Sin embargo, la salida, en vez de mostrar los títulos de los artículos por pantalla directamente, los va guardando en un array llamado *títulos*.

ObtenerPdfConSerpApi.java

La entrada de esta clase se ha modificado para que coja los nombres de los títulos del array devuelto por **LeerXml.java** en vez de obtenerlos de la línea de comandos. En cuanto a la salida, también se ha modificado, devolviéndose ahora una matriz bidimensinal denominada *titulosYEnlaces* cuyo formato es: *String[título][enlace de descarga PDF del artículo].*

DescargarPdfCache.java

La entrada de esta clase se ha modificado para que utilice los títulos de los artículos, junto con su URL de descarga en formato PDF, obtenidos de **ObtenerPdfConSerpApi.java**. Su salida también ha sido modificada, ya que ahora devuelve una matriz bidimensional denominada *rutasYTitulos* cuyo formato es: *String[rutas de destino para la descarga de los artículos][títulos de los artículos].*

MostrarYComprobarEnlaces.java

La entrada de esta clase se ha modificado para que utilice las rutas de destino y los títulos de los artículos obtenidos de **DescargarPdfCache.java**. Su salida no ha sido modificada, sigue introduciendo en la tabla *downloaded_pdfs*:

- Título del artículo
- URL de referencia
- Código de respuesta del servidor que nos ayuda a identificar si la URL existe o no.

Además, en este ciclo se cuenta con un script de Python, denominado *ExportarCSV*. Este script exporta los datos recogidos en la base de datos para poder realizar un análisis y unas estadísticas de los datos obtenidos.

B.3. Ciclo 3

Representa el último ciclo de desarrollo de la aplicación. En él se modificaron las clases existentes, para poder ejecutar la aplicación tanto con la clase **RunEntireFlow.java** como ejecutar de forma individual las distintas clases que la conforman. Además, se añadió una clase adicional llamada **ClasificarURL.java**.

A continuación, se explican las modificaciones realizadas a las clases existentes, así como se detallará la clase nueva.

LeerXml.java

Se añadió como mejora la posibilidad de mostrar por pantalla el número de títulos que se encuentran en el argumento con el que se ejecuta. Para ello, se utiliza una variable que funciona como contador.

ObtenerPdfConSerpApi.java

Se añadió como mejora una variable contador, para poder mostrar por pantalla el número de enlaces de descarga de un artículo en formato PDF, que se han obtenido gracias a SerpApi de Google Scholar.

Además, se añadió también otra variable de tipo boolean que se encarga de no buscar el enlace de descarga del artículo si ya se ha buscado con anterioridad. Esta mejora es debida a que las búsquedas que se pueden realizar con SerpApi son limitadas por mes, por lo que una vez que se han buscado no es necesario utilizar otra búsqueda para el mismo artículo.

DescargarPdfCache.java

En este caso procedo a detallar los métodos de la clase, ya que han habido mejoras significativas.

- El método *main* en este caso vuelve a ser de la forma *public static void main(String[] args) throws ClassNotFoundException* para poder ejecutar la clase de forma individual.
- Se introduce la casuística de *forzar descarga* la cual, permite volver a descargar los artículos en la caché aunque ya se hubieran descargado previamente.
- Método *chequearDirectorioCache* se encarga de cmprobar si existe el directorio caché, y si no existe lo crea.
- Método *comprobarCacheyDescargarPDF* se encarga de comprobar en la base de datos y en la caché el artículo que se quiere descargar. Además, llama al método *descargarPDF* si no se ha descargado previamente el artículo, o si se indica que se fuerce la descarga.

MostrarYComprobarEnlaces.java

Debido a las mejoras introducidas se expone la estructura de la clase a continuación.

- El método *main* en este caso vuelve también a ser de la forma *public static void main(String[] args) throws SQLException* para poder ejecutar la clase de forma individual.

- Se introducen las casuísticas de *regenerate-text*, *force-link-search* y *use-db-only*.
  - *regenerate-text*: Se vuelve transformar el PDF a texto, se buscan de nuevo los enlaces y se comprueba su accesibilidad.
  - *force-link-search*: Se analizan de nuevo los enlaces de los ficheros, usando el texto existente. Se comprueba su accesibilidad.
  - *use-db-only*: Se comprueba la accesibilidad de los enlaces que están en la base de datos sin utilizar el texto del artículo.
- Método *extraerTodosLosEnlaces* se encarga de extraer todas las URLs que se utilizan como referencias en los artículos.
- Método *extraeEnlacesDeArticulo* se encarga de manipular los enlaces obtenidos. Primero se comprueba si la version .txt del documento ya existe en la cache. A continuación, se eliminan los enlaces existentes en la base de datos. Si no hay un fichero de texto asociado, el PDF corresponde con un nuevo fichero, entonces se obtiene el documento y se transforma a texto. Luego, se marcan todos los "http", ya que, esto permite realizar manipulaciones del texto, como añadir o eliminar, conociendo qué URLs se han procesado y cuáles están pendientes. También, se aplican varias estrategias, en forma de expresiones regulares, para poder maximizar los enlaces correctos. De esta forma, se mejoran los problemas con los matches ya que estos proporcionan muchos falsos negativos por OCR incorrecto o por el formato de los artículos.
- Método *getURLsFromPaperText* se obtienen y se procesan los enlaces del artículo. Para ello, se eliminan los espacios de las URLs para poder procesarlas de forma correcta.
- Método *unmarkURLS* elimina los markers de una cadena de texto solo en posiciones especificadas.
- Método *extractURLContext* extrae el contexto de las URLs obtenidas del artículo.
- Método *generateNewPaperTextWithMarkedHTTP* utiliza un marker para marcar todas apariciones de *http* para poder identificar de forma más eficiente las URLs.
- Para verificar la existencia se ha añadido un agente.
- Método *obtenerEnlaceFinal* se encarga de devolver la URL después de todas las redirecciones que puede tener.

ConexionBaseDeDatos.java

En este caso se han añadido las columnas, a la tabla *downloaded_pdfs*, *finalURL*, *contexto* y *type*. Además, se han añadido los siguientes métodos:

- Método *EliminarEnlacesDeUnArticulo* se encarga de eliminar los enlaces de un artículo que se encuentran almacenados en la tabla *downloaded_pdfs* de la base de datos.
- Método *getAllURLs* extrae las URLs de los artículos que se han identificado y las muestra en una lista.
- Método *saveURLType* guarda en la base de datos, en la tabla *downloaded_pdfs*, la clasificación de las URLs.
- Método *BDExist* indica si la base de datos existe.
- Método *updateResponseCode* indica el código de respuesta que devuelve el servidor al intentar conectar con cada URL para verificar su existencia.

RunEntireFlow.java

Se añadió al flujo de ejecución de la aplicación la nueva clase **ClasificarURL.java**.

ClasificarURL.java

Gracias a un fichero denominado *sites.properties* se clasificam las URLs identificadas.

Contiene un método denominado *getURLType* que categoriza las URLs.

Ejemplo del fichero *sites.properties*:

```
Article = doi\.org/10\.48550,arxiv\.org,academia\.edu,researchgate\.net,sciencedirect\.com,ieeexplore\.ieee\.org,dl\.acm\
Book = taylorfrancis\.com/chapters,aspbooks\.org/publications,books\.google,library\.oapen\.org,academic\
Document = eric\.ed\.gov/\?id=,nvlpubs\.nist\.gov\/nistpubs\/ir,
Regulation = eur-lex\.europa\.eu/legal-content,normlex\.ilo\.org\/dyn\/normlex,uscode\.house\.gov\/view\
Standard = iso\.org/standard,etsi\.org\/deliver\/etsi_ts,w3\.org\/TR\/,astm\.org\/[a-zA-Z]][0-9]{4},webstore\
Dataset = zenodo\.org/records,dataverse\.harvard\.edu/dataset\.xhtml,yareta\.unige\.ch/archives,datadryad\
Cloud = cloud\.box\.com,dropbox\.com/s/,drive\.google\.com/file/d,drive\.google\.com/drive/folders,dropbox\
Repository = \/\/gitlab\.com,\/\/github\.com,sourceforge\.net/projects,code\.google\.com,bitbucket\.com,git\
Social = facebook\.com,instagram\.com,linkedin\.com,pinterest\.com,snapchat\.com,twitter\.com,youtube\.com,reddit\.com,tiktok\
Messaging = whatsapp\.com,
API = \/\/api\.,
```

Figura B.2: Ejemplo fichero *sites.properties*.

Apéndice C

Pruebas

C.1. Ciclo 1

Para este primer ciclo se realizaron una serie de pruebas en base a la versión con base de datos.

C.1.1. Versión con base de datos

Lo primero que hay que tener en cuenta es que es necesario tener instalado en el ordenador donde se va a ejecutar el programa *Maven*. Esto es necesario para que se pueda leer el archivo *pom.xml* y funcionen las dependencias. Además, se miró previamente las URLs presentes en los artículos para verifiar su existencia y así, poder comprobar si el programa realizaba satisfactoriamente su función.

A continuación, se ejecutó la clase **LeerXml.java** con el argumento: `https://dblp.org/search/publ/api?q=toc%3Adb/journals/tse/tse37.bht%3A&h=1000&format=xml`. Como resultado se obtuvieron los títulos de los artículos que se encuentran en la url proporcionada como argumento. Se decidió hacer las pruebas con estos artículos:

1. Frameworks Generate Domain-Specific Languages: A Case Study in the Multimedia Domain.
2. FlowTalk: Language Support for Long-Latency Operations in Embedded Devices.
3. Dynamic QoS Management and Optimization in Service-Based Systems.

Prueba 1

Se ejecutó la clase **ObtenerPdfConSerpApi.java** con el título de un artículo como argumento: *Frameworks Generate Domain-Specific Languages: A Case Study in the Multimedia Domain.*. Como resultado se mostró el título del artículo y una URL de descarga de este en formato PDF:

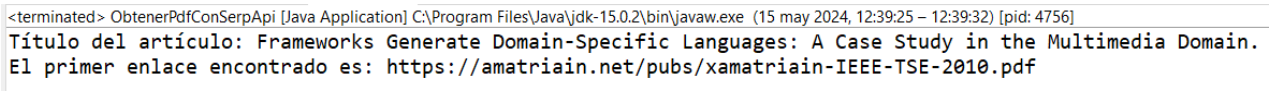


Figura C.1: Resultado obtenido por ObtenerPdfConSerpApi.java - Artículo 1

Posteriormente, se ejecutó la clase **DescargarPdfCache.java** con el título del artículo y su URL de descarga. Como resultado, se descargó el artículo en formato PDF de forma local. Asímismo, se añadió el nombre del artículo descargado a la base de datos, en la tabla *nombre_pdfs*:

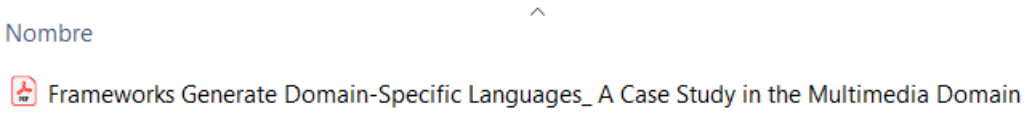


Figura C.2: PDF descargado - Artículo 1

Para poder visualizar la base de datos, se utilizó la aplicación *DB Browser for SQLite*:

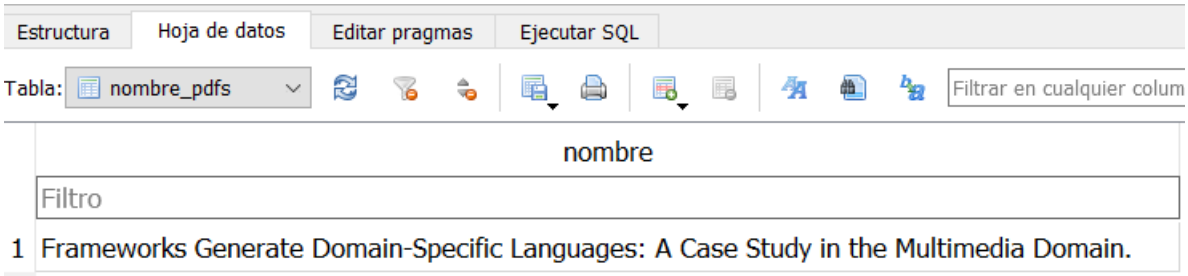


Figura C.3: Tabla nombre_pdfs - Artículo 1

A continuación, se ejecutó la clase **MostrarYComprobarEnlaces.java**, utilizando la ruta de descarga del artículo y su título. Como resultado se introdujo en la base de datos, en la tabla *downloaded_pdfs* el título del artículo, las URLs de referencia y el código devuelto por el servidor al intentar acceder a la URL:

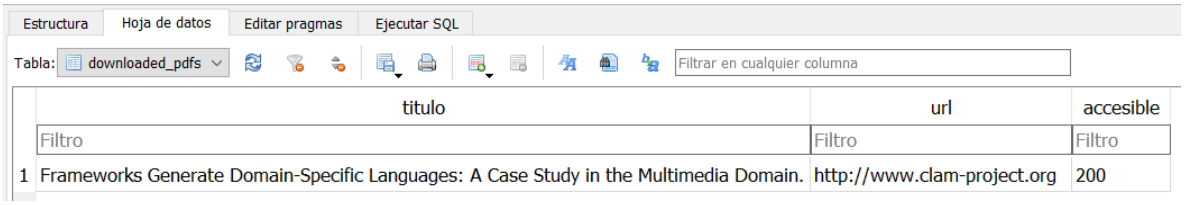


Figura C.4: Tabla downloaded_pdfs - Artículo 1

Una vez realizado el proceso completo, se intentó volver a ejecutar la clase **DescargarPdfCache.java** con el mismo artículo para comprobar que no se descarga, ya que, ya ha sido descargado previamente:

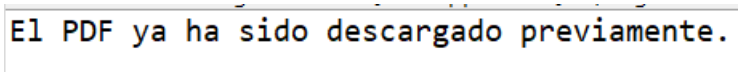


Figura C.5: Mensaje tras intentar volver a descargar un PDF - Artículo 1

Una vez se terminó de realizar la prueba 1 con el primer artículo, se procedió a realizar la prueba 2.

Prueba 2

Se ejecutó la clase **ObtenerPdfConSerpApi.java** con el título de un artículo como argumento: *FlowTalk: Language Support for Long-Latency Operations in Embedded Devices*. Como resultado se mostró el título del artículo y una URL de descarga de este en formato PDF:

Figura C.6: Resultado obtenido por ObtenerPdfConSerpApi.java - Arículo 2

Luego, se ejecutó la clase **DescargarPdfCache.java** con el título del artículo y su URL de descarga. Como resultado, se descargó el artículo en formato PDF de forma local. Asímismo, se añadió el nombre del artículo descargado a la base de datos, en la tabla *nombre_pdfs*:

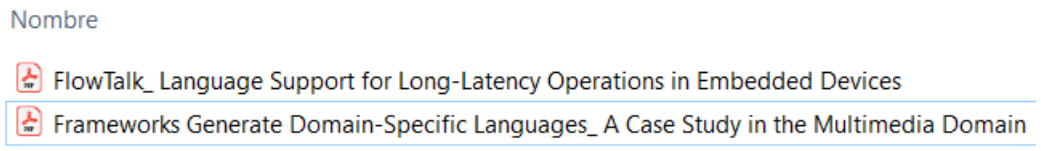


Figura C.7: PDF descargado - Artículo 2

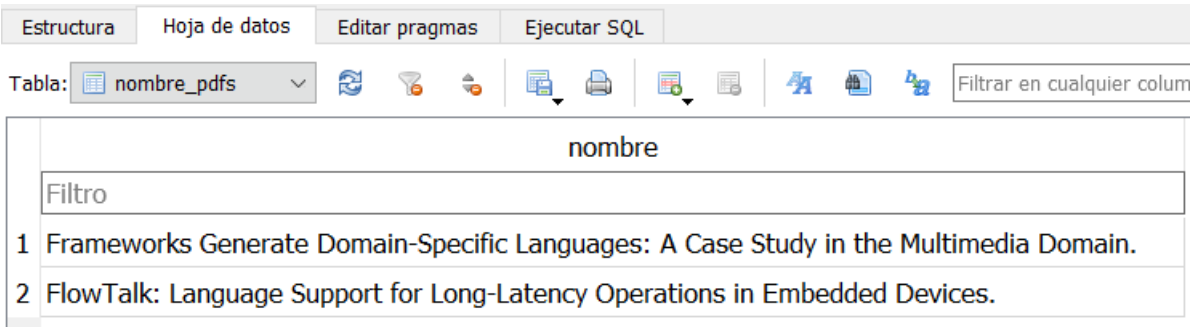
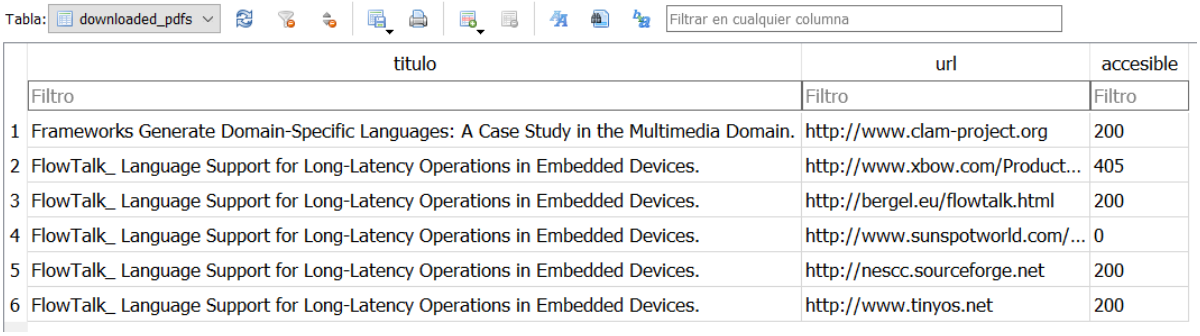


Figura C.8: Tabla nombre_pdfs - Artículo 2

Posteriormente, se ejecutó la clase **MostrarYComprobarEnlaces.java**, utilizando la ruta de descarga del artículo y su título. Como resultado se introdujo en la base de datos, en la tabla *downloaded_pdfs* el título del artículo, las URLs de referencia y el código devuelto por el servidor al intentar acceder a la URL:



	titulo	url	accesible
	Filtro	Filtro	Filtro
1	Frameworks Generate Domain-Specific Languages: A Case Study in the Multimedia Domain.	http://www.clam-project.org	200
2	FlowTalk_ Language Support for Long-Latency Operations in Embedded Devices.	http://www.xbow.com/Product...	405
3	FlowTalk_ Language Support for Long-Latency Operations in Embedded Devices.	http://bergel.eu/flowtalk.html	200
4	FlowTalk_ Language Support for Long-Latency Operations in Embedded Devices.	http://www.sunspotworld.com/...	0
5	FlowTalk_ Language Support for Long-Latency Operations in Embedded Devices.	http://nescc.sourceforge.net	200
6	FlowTalk_ Language Support for Long-Latency Operations in Embedded Devices.	http://www.tinyos.net	200

Figura C.9: Tabla downloaded_pdfs - Artículo 2

Una vez realizado el proceso completo, se intentó volver a ejecutar la clase **DescargarPdfCache.java** con el mismo artículo para comprobar que no se descarga, ya que, ya ha sido descargado previamente:

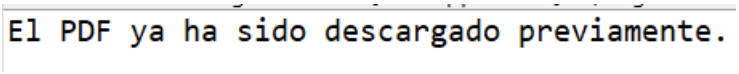


Figura C.10: Mensaje tras intentar volver a descargar un PDF - Artículo 2

Una vez se terminó de realizar la prueba 2 con el segundo artículo, se procedió a realizar la prueba 3.

Prueba 3

Se ejecutó la clase **ObtenerPdfConSerpApi.java** con el título de un artículo como argumento: *Dynamic QoS Management and Optimization in Service-Based Systems*.. Como resultado se mostró el título del artículo y una URL de descarga de este en formato PDF:

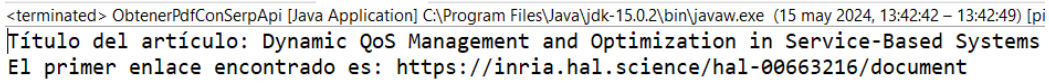


Figura C.11: Resultado obtenido por ObtenerPdfConSerpApi.java - Arículo 3

Luego, se ejecutó la clase **DescargarPdfCache.java** con el título del artículo y su URL de descarga. Como resultado, se descargó el artículo en formato PDF de forma local. Asímismo, se añadió el nombre del artículo descargado a la base de datos, en la tabla *nombre_pdfs*:

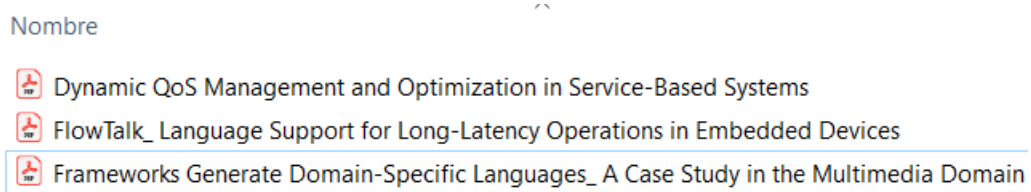
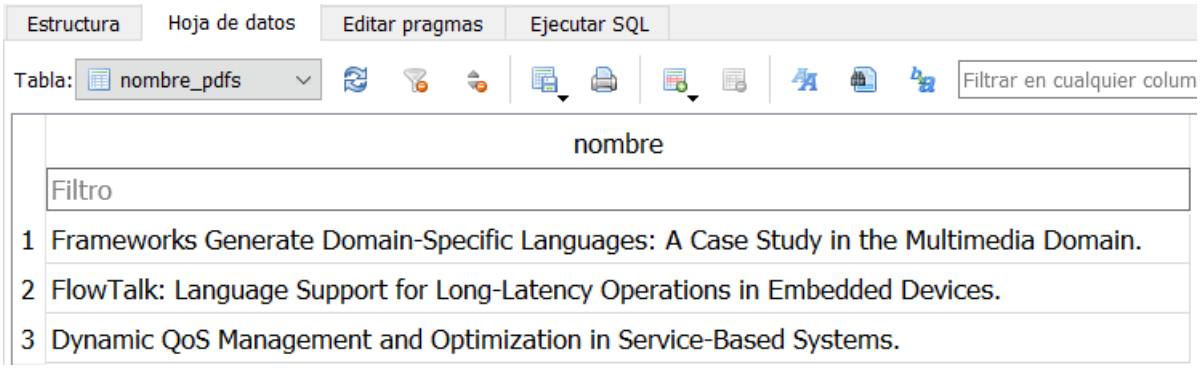


Figura C.12: PDF descargado - Artículo 3



	nombre
	Filtro
1	Frameworks Generate Domain-Specific Languages: A Case Study in the Multimedia Domain.
2	FlowTalk: Language Support for Long-Latency Operations in Embedded Devices.
3	Dynamic QoS Management and Optimization in Service-Based Systems.

Figura C.13: Tabla nombre_pdfs - Artículo 3

Posteriormente, se ejecutó la clase **MostrarYComprobarEnlaces.java**, utilizando la ruta de descarga del artículo y su título. Como resultado se introdujo en la base de datos, en la tabla *downloaded_pdfs* el título del artículo, las URLs de referencia y el código devuelto por el servidor al intentar acceder a la URL:

	titulo	url	accesible
	Filtro	Filtro	Filtro
1	Frameworks Generate Domain-Specific Languages: A Case Study in the Multimedia Domain.	http://www.clam-project.org	200
2	FlowTalk_ Language Support for Long-Latency Operations in Embedded Devices.	http://www.xbow.com/Product...	405
3	FlowTalk_ Language Support for Long-Latency Operations in Embedded Devices.	http://bergel.eu/flowtalk.html	200
4	FlowTalk_ Language Support for Long-Latency Operations in Embedded Devices.	http://www.sunspotworld.com/...	0
5	FlowTalk_ Language Support for Long-Latency Operations in Embedded Devices.	http://nescc.sourceforge.net	200
6	FlowTalk_ Language Support for Long-Latency Operations in Embedded Devices.	http://www.tinyos.net	200
7	Dynamic QoS Management and Optimization in Service-Based Systems.	https://inria.hal.science/...	200

Figura C.14: Tabla downloaded_pdfs - Artículo 3

Una vez realizado el proceso completo, se intentó volver a ejecutar la clase **DescargarPdfCache.java** con el mismo artículo para comprobar que no se descarga, ya que, ya ha sido descargado previamente:

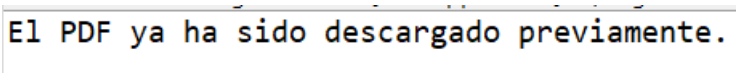


Figura C.15: Mensaje tras intentar volver a descargar un PDF - Artículo 3

C.2. Ciclo 2

En este ciclo, se añadió una clase denominada **RunEntireFlow.java** la cual, al lanzarla se encarga de ejecutar las distintas clases que conforman el programa.

Prueba 1

Ejecutar la clase **RunEntireFlow.java** con el argumento: `https://dblp.org/search/publ/api?q=conf/icse/2024/workshop&h=3&format=xml`, el cual, contiene un fichero .xml de DBLP. Al ejecutar la clase con la URL especificada, esta llama a la clase **LeerXml.java** que devuelve el nombre de los artículos que contiene el fichero .xml:

```
<<<<<<<<<<LeerXml>>>>>>>>>>
"ECFuzz: Effective Configuration Fuzzing for Large-Scale Systems"
"Enabling Runtime Verification of Causal Discovery Algorithms with Automated Conditional Independence Reasoning"
"Inferring Data Preconditions from Deep Learning Models for Trustworthy Prediction in Deployment"
```

Figura C.16: Salida LeerXml - Ciclo 2

A continuación, llama a **ObtenerPdfConSerpApi.java** la cual, solo muestra por pantalla el JSON del artículo si no encontró ningún enlace para descargar dicho artículo en formato PDF, ya que así, se puede explorar de forma manual. En el caso que encuentre la URL de descarga, llama directamente a **DescargarPdfCache.java** la cual, crea una carpeta denominada *Cache* en la que guarda los artículos descargados en formato PDF.

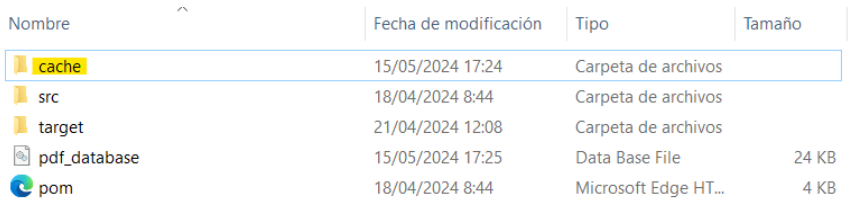


Figura C.17: Carpeta caché creada

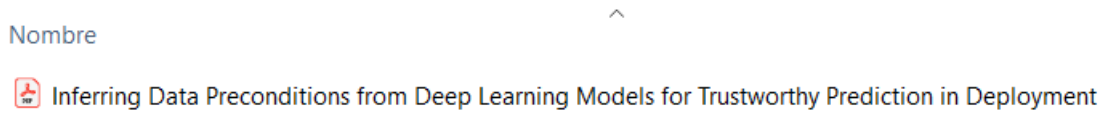


Figura C.18: PDF descargado - Ciclo 2

Además, se introduce en la base de datos, en la tabla *nombre_pdfs* el nombre del PDF descargado: A continuación, llama a **MostrarYComprobarEnlaces.java**, la cual, introduce en la base de datos, en la tabla *downloaded_pdfs* el título del artículo, las URLs que aparecen como referencias y el código del servidor para verificar la existencia de ellas:

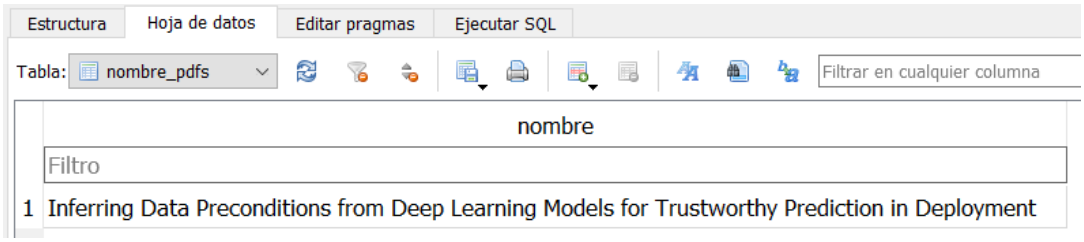
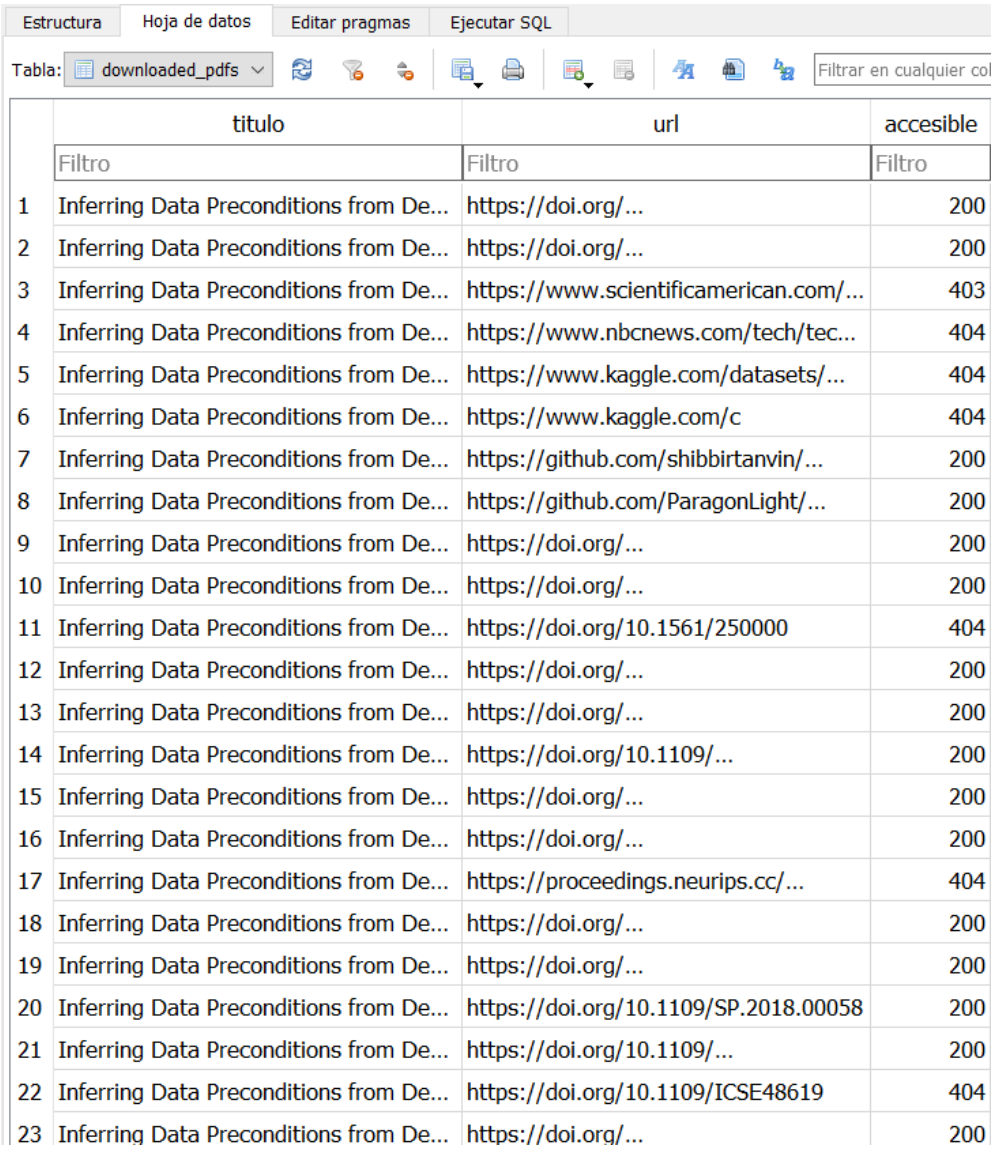


Figura C.19: Tabla nombre_pdfs - Ciclo 2



	titulo	url	accesible
	Filtro	Filtro	Filtro
1	Inferring Data Preconditions from De...	https://doi.org/...	200
2	Inferring Data Preconditions from De...	https://doi.org/...	200
3	Inferring Data Preconditions from De...	https://www.scientificamerican.com/...	403
4	Inferring Data Preconditions from De...	https://www.nbcnews.com/tech/tec...	404
5	Inferring Data Preconditions from De...	https://www.kaggle.com/datasets/...	404
6	Inferring Data Preconditions from De...	https://www.kaggle.com/c	404
7	Inferring Data Preconditions from De...	https://github.com/shibbirtanvin/...	200
8	Inferring Data Preconditions from De...	https://github.com/ParagonLight/...	200
9	Inferring Data Preconditions from De...	https://doi.org/...	200
10	Inferring Data Preconditions from De...	https://doi.org/...	200
11	Inferring Data Preconditions from De...	https://doi.org/10.1561/250000	404
12	Inferring Data Preconditions from De...	https://doi.org/...	200
13	Inferring Data Preconditions from De...	https://doi.org/...	200
14	Inferring Data Preconditions from De...	https://doi.org/10.1109/...	200
15	Inferring Data Preconditions from De...	https://doi.org/...	200
16	Inferring Data Preconditions from De...	https://doi.org/...	200
17	Inferring Data Preconditions from De...	https://proceedings.neurips.cc/...	404
18	Inferring Data Preconditions from De...	https://doi.org/...	200
19	Inferring Data Preconditions from De...	https://doi.org/...	200
20	Inferring Data Preconditions from De...	https://doi.org/10.1109/SP.2018.00058	200
21	Inferring Data Preconditions from De...	https://doi.org/10.1109/...	200
22	Inferring Data Preconditions from De...	https://doi.org/10.1109/ICSE48619	404
23	Inferring Data Preconditions from De...	https://doi.org/...	200

Figura C.20: Tabla downloaded_pdfs - Ciclo 2

24	Inferring Data Preconditions from De...	https://doi.org/...	200
25	Inferring Data Preconditions from De...	https://doi.org/10.1145/3546947	200
26	Inferring Data Preconditions from De...	https://doi.org/...	200
27	Inferring Data Preconditions from De...	https://doi.org/...	200
28	Inferring Data Preconditions from De...	https://proceedings.mlr.press/v80/...	200
29	Inferring Data Preconditions from De...	https://doi.org/10.1109/...	200
30	Inferring Data Preconditions from De...	https://doi.org/10.1145/3290354	200
31	Inferring Data Preconditions from De...	https://doi.org/...	200
32	Inferring Data Preconditions from De...	https://doi.org/...	200
33	Inferring Data Preconditions from De...	https://doi.org/...	200
34	Inferring Data Preconditions from De...	https://doi.org/10.1109/TDSC....	200
35	Inferring Data Preconditions from De...	https://doi.org/10.1	404
36	Inferring Data Preconditions from De...	https://doi.org/...	200

Figura C.21: Tabla downloaded_pdfs - Ciclo 2

Como se puede observar, se utilizó un artículo que contaba con un amplio número de URLs, en concreto 36, para poder comprobar si aún con un número elevado de ellas, el programa lograba verificar su existencia. El resultado fue satisfactorio.

tfg_etsiinf_Sara.pdf

by SARA SUSANO RUIZ

Submission date: 03-Jun-2024 12:11PM (UTC+0200)
Submission ID: 2394498249
File name: 14778_SARA_SUSANO_RUIZ_tfg_etsiinf_Sara_530678_1824891039.pdf (2.53M)
Word count: 14331
Character count: 74614

Universidad Politécnica de Madrid

Escuela Técnica Superior de Ingenieros Informáticos

Grado en Ingeniería Informática

Trabajo Fin de Grado

Análisis del decaimiento de información en internet: El caso de los recursos científicos.

Autor: SARA SUSANO RUIZ
Tutor(a): OSCAR DIESTE TUBIO

Madrid, Junio 2024

Este Trabajo Fin de Grado se ha depositado en la ETSI Informáticos de la Universidad Politécnica de Madrid para su defensa.

Trabajo Fin de Grado
Grado en Ingeniería Informática

Título: Análisis del decaimiento de información en internet: El caso de los recursos científicos.

Junio 2024

Autor: SARA SUSANO RUIZ
Tutor: OSCAR DIESTE TUBIO
Departamento de Lenguajes y Sistemas Informáticos e Ingeniería de Software
Escuela Técnica Superior de Ingenieros Informáticos
Universidad Politécnica de Madrid

Resumen

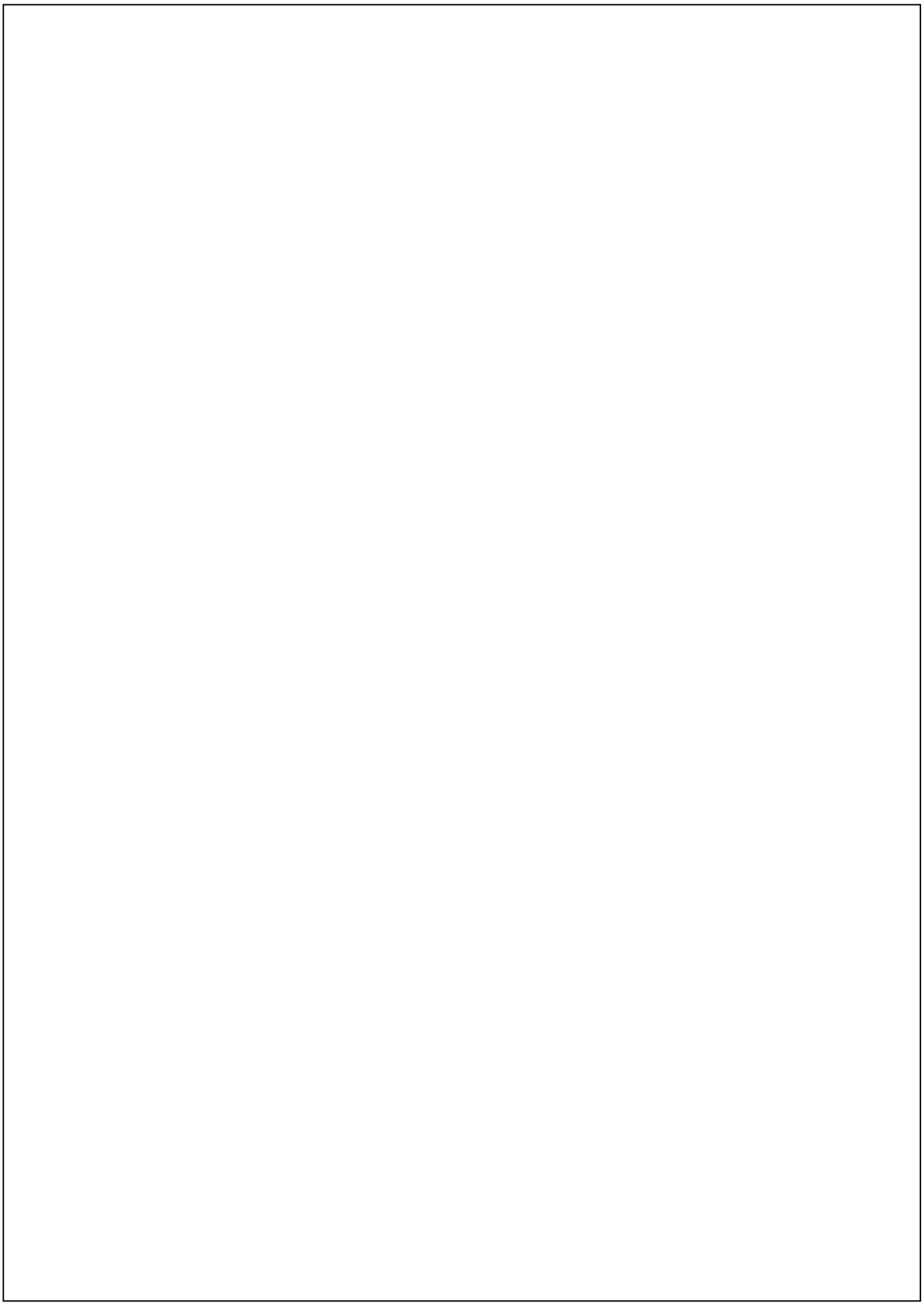
Este Trabajo de Fin de Grado trata de explorar la desaparición que se produce de la información en internet. Tiene el objetivo de poder realizar un análisis de cómo va desapareciendo dicha información e intentar descubrir el por qué. Para ello, se utilizó una aplicación, desarrollada por ciclos.

Dicha aplicación, se encarga de extraer de un fichero con terminación XML, de la biblioteca digital *DBLP*, los títulos de los artículos contenidos en él. Una vez extraídos se encarga de buscar, gracias a SerpApi de Google Scholar, dichos artículos en formato PDF para posteriormente descargarlos. Una vez descargados, se extráen los enlaces contenidos en el apartado de referencias de dichos PDFs y se verifica su existencia, guardando los datos obtenidos en una base de datos. Esto último, se realiza para poder llevar a cabo las estadísticas sobre cómo desaparece la información que se encuentra recogida en Internet.

Los resultados de los análisis realizados mostraron que cuanto mayor es el tiempo de publicación del artículo, más propensa es la información a desaparecer. Además, hay relación entre los servicios online donde se almacena la información y su desaparición, como repositorios o servicios cloud que son personales.

Estos resultados sugieren que se deben de tomar medidas para poder mantener la información disponible en la red, como almacenarla en sitios web oficiales. Además, todavía es necesario mejorar las técnicas de extracción y manipulación de las URLs para que el análisis pueda ser lo más verídico posible.

En este documento se abordan el diseño de la aplicación, junto con sus ciclos de desarrollo, los antecedentes utilizados, los objetivos a cumplir, los resultados que se han obtenido de los análisis realizados y el trabajo futuro que se debe de realizar.



Abstract

This document explores the phenomenon of information disappearance on the internet. Its objective is to conduct an analysis of how this information disappears and attempt to uncover the reasons behind it. To achieve this, an application was used, developed through iterative cycles.

This application is responsible for extracting titles of articles from a file with XML extension, obtained from the digital library *DBLP*. Once they are extracted, it searches for these articles in PDF format using SerpApi from Google Scholar, and later it downloads them. After downloading, it extracts the links that are in the references section of these PDFs and verifies their existence, storing the obtained data in a database. This is done in order to conduct statistics on how the information collected from the internet disappears over time.

The results of the conducted analyses showed that the longer the publication time of the article, the more likely the information is to disappear. Furthermore, there is a relationship between online services where information is stored and its disappearance, such as personal repositories or cloud services.

These results suggest that measures should be taken to keep information available on the web, such as storing it on official websites. Additionally, there is still a need to improve techniques for extracting and manipulating URLs so that the analysis can be as accurate as possible.

This document addresses the design of the application, along with its development cycles, the background used, the objectives to be achieved, the results obtained from the analyses performed, and the future work to be done.

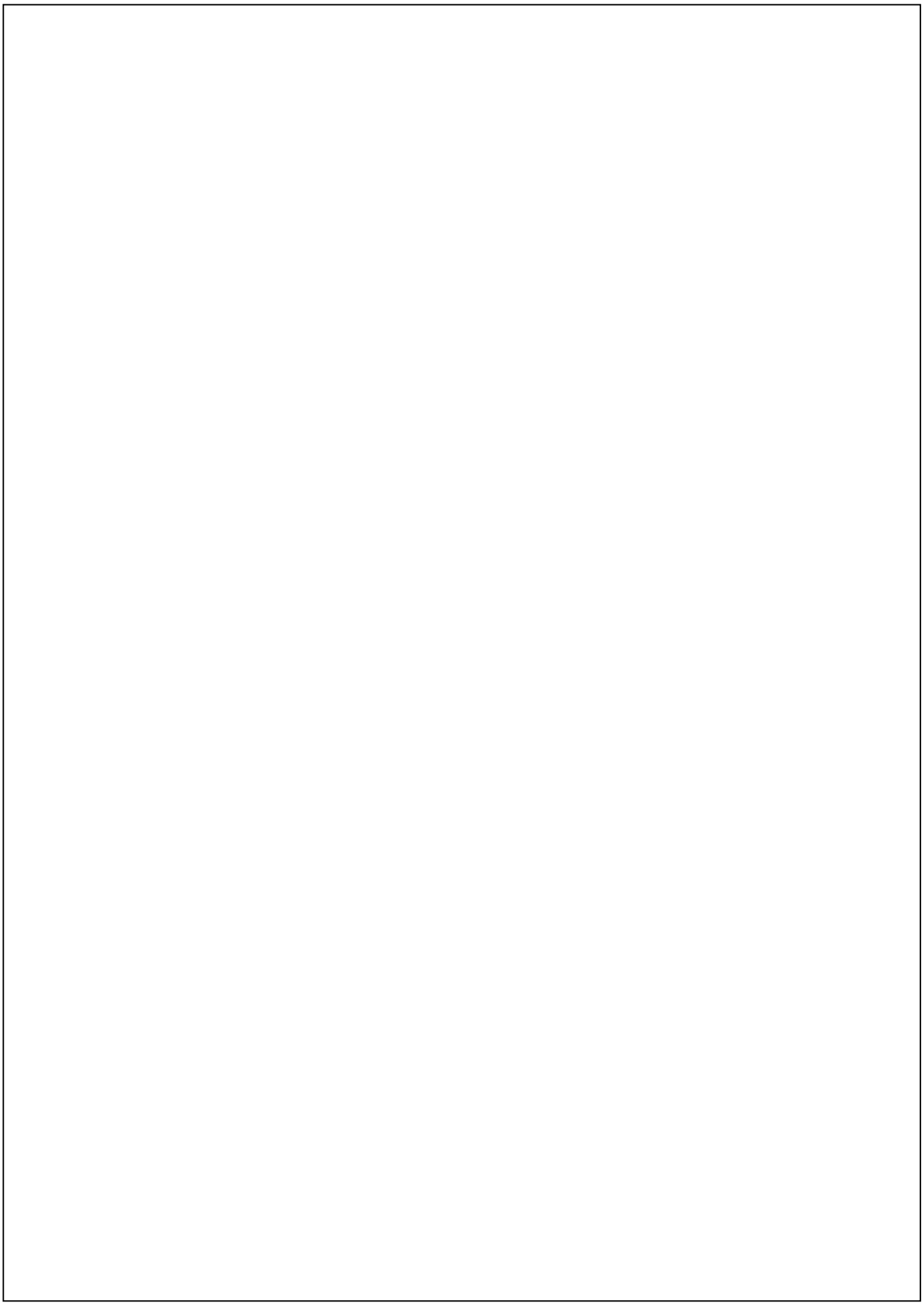


Tabla de contenidos

Índice de figuras

Capítulo 1

Introducción

La investigación científica por lo general, se apoya bastante en Internet. Los distintos recursos, como los artículos o los análisis, son publicados en la red en distintos servicios, como repositorios o servicios cloud entre otros, no siempre de forma voluntaria.

Estos recursos se identifican mediante *URIs* (Uniform Resource Identifiers), más concretamente se suelen utilizar *URNs* (Uniform Resource Names) y los *URLs* (Uniform Resource Locators).

Un URN es un *"persistent, location-independent resource identifier"*. Un ejemplo bien conocido de URN es el *Digital Object Identifier* (DOI).

Los URIs más frecuentemente utilizados son los URLs, los cuales definen el protocolo de acceso, servidor y puerto donde los recursos están accesibles, aunque, nada garantiza que el recurso esté disponible en el momento de acceso.

Los URNs no indican la localización del recurso al que identifican, por lo que es necesario, realizar una traslación URN $\rightarrow$ URL. En consecuencia, los URNs también pueden estar no disponibles.

Se conoce como *linkrot* o *link decay* a *"the disassociation between web addresses and their content"*. Los motivos de esta disociación pueden ser muy variados. Por ejemplo, el propietario de un recurso podría desear eliminarlo de Internet. Otras razones por los que una URL puede no ser accesible son: reorganización de los recursos en los servidores son proporcionar una redirección, gestión inadecuada de los recursos, servidores o dominios que desaparecen, etc. [11].

La situación se exacerba cuando los recursos son publicados en servicios cuyo objetivo no es la preservación, sino el almacenamiento y la compartición de archivos, como por ejemplo, el uso de Dropbox o Google Drive. El acceso a un archivo compartido se pierde cuando es movido de posición, borrado, o el usuario cancela su cuenta en dichos servicios Cloud. Por este motivo, se recomienda evitar este tipo de servicios para la preservación de recursos científicos [12].

El *linkrot* es un problema importante, ya que, se ha observado que, al año siguiente de la publicación, un 10 % de los links que aparecen en artículos científ-

ficos ya no están disponibles [11, 13], y dicho porcentaje aumenta con el tiempo [14].

Este trabajo tiene como objetivo el desarrollo de una aplicación. Dicha aplicación debe ser capaz de poder acceder a los artículos en formato PDF de un fichero extraído de la biblioteca digital DBLP. En los PDF, debe poder extraer las URL que se encuentran en el apartado de referencias. Por último, debe comprobar si las URLs extraídas existen o no para, poder realizar un análisis de la desaparición de la información en Internet.

Para poder lograr el objetivo se utiliza como método el desarrollo incremental iterativo. Esto significa que en cada ciclo de desarrollo, se van introduciendo nuevas funcionalidades y a la vez se van añadiendo mejoras sobre aquellas que ya estaban implementadas previamente.

La aplicación es capaz de buscar un enlace para poder obtener un artículo en formato PDF. Además, lo descarga, únicamente si no ha sido descargado previamente, y lee el contenido del mismo. Tras leer el contenido, es capaz de identificar y extraer las URLs que aparecen en las referencias de dicho PDF, además de verificar su existencia. Por último, una vez se ha realizado el análisis, cuenta con la capacidad de exportar los datos obtenidos para poder realizar estadísticas sobre los mismos y así, efectuar un análisis de la pérdida de información en internet.

En el siguiente repositorio se encuentra el código de desarrollo de la aplicación: Repositorio.

Capítulo 2

Antecedentes

En este capítulo se tratan las herramientas y los aspectos claves que han jugado un papel fundamental en el desarrollo del proyecto.

2.1. Maven [1] [2]

Apache Maven se trata de una herramienta que sirve para la gestión y la comprensión de proyectos de software. Su enfoque se basa en el Modelo de Objeto de Proyecto (POM), permitiendo a Maven gestionar la construcción, informes y documentación de un proyecto a través de una fuente centralizada de información.

Maven desempeña funciones cruciales como la compilación, el empaquetamiento y la ejecución de los test. Se basa en un fichero denominado pom.xml, donde se definen todos los elementos esenciales para el proyecto, tales como la gestión de dependencias, las cuales se descargan e incorporan al classpath al definirlas. Esta herramienta cuenta además con unos arquetipos, los cuales son los encargados de crear la estructura del proyecto , los datos contenidos en pom.xml, la estructura utilizada por las carpetas y los ficheros que se incluyen por defecto. Estos arquetipos facilitan la creación inicial del proyecto al definir automáticamente aspectos fundamentales.

En el marco de este proyecto, se ha utilizado para organizar el código de la aplicación de manera efectiva. Resulta beneficioso tener las clases debidamente estructuradas y las bibliotecas utilizadas gestionadas a través del archivo pom.xml, evitando así la necesidad de descargar el archivo .jar e incorporarlas individualmente.

Figura 2.1: Logo Maven.

2.2. DBLP [3]

DBLP computer science bibliography, o simplemente dblp, es una base de datos bibliográfica destacada por su acceso abierto, eliminando la necesidad de suscripción o registro. Principalmente indexa congresos y revistas académicas aunque tiene una variada tipología documental.

La misión de dblp es respaldar a los investigadores del campo informático, ofreciendo acceso de forma gratuita a los metadatos bibliográficos que contienen alta calidad y enlaces que se refieren a ediciones electrónicas de publicaciones.

En el contexto de este trabajo, se ha empleado para obtener los enlaces a los artículos, permitiendo así comprobar si los enlaces en sus referencias siguen existiendo o si su información ha desaparecido.

Figura 2.2: Logo DBLP.

2.3. Google Scholar [4] [5]

Google Scholar es una herramienta de buscador que permite la localización de documentos académicos como por ejemplo artículos, tesis, libros y resúmenes de diferentes fuentes como editoriales pertenecientes a universidades, asociaciones profesionales u otras organizaciones que tienen carácter académico.

Esta herramienta es un buen recurso para la búsqueda en muchas fuentes a la vez, sin embargo, no todos los autores permiten a Google Scholar a acceder a sus artículos por lo que hay parte de la producción científica y académica a la que no se puede acceder desde el buscador.

Google Scholar proporciona información sobre la cantidad de citas que un artículo ha recibido y quiénes son los citante, además de contar con la función de guardar tanto citas como artículos para leerlos posteriormente. Esto lo convierte en la herramienta ideal para completar búsquedas hechas en bases de datos científicas.

En el contexto de este proyecto, Google Scholar es utilizada obtener un acceso gratuitos a los artículos que se van a analizar cuando, al acceder a través de liberías digitales nos encontramos que son de pago.

Figura 2.3: Logo Google Scholar.

2.4. SerpApi [6]

SerpApi es una API gratuita, siempre y cuando no se excedan las 100 búsquedas al mes, que permite obtener información de buscadores como Google de manera fácil y eficiente.

Para poder utilizarla es necesario tener una clave personal API-KEY. Para obtenerla es necesario indicar un correo electrónico y un número telefónico.

Esta herramienta resuelve además los problemas relacionados con el alquiler de proxies, resolver captchas y analizar archivos JSON.

En el contexto de este trabajo, se utiliza para poder manejar los problemas mencionados anteriormente.

2.5. Librería PDFBox [7]

La libería Apache PDFBox es una herramienta de código abierto para el lenguaje de programación JAVA, que se utiliza para trabajar con documentos PDF. Permite la creación de nuevos PDFs, la manipulación de aquellos existentes y tiene la habilidad de extraer el contenido de los mismos. Además, contiene varias utilidades de línea de comandos.

En el contexto del proyecto, se usa para poder extraer el contenido de los PDFs descargados.

2.6. Librería JSOUP [8]

La librería JSOUP es una herramienta para JAVA que simplifica la manipulación de HTML y XML. Ofrece una API fácil de usar para la obtención de URLs, análisis de datos, extracción y manipulación usando métodos DOM API, CSS, y selectores xpath.

JSOUP implementa la especificación denominada *HTML5* de WHATWG y analiza HTML en el mismo DOM en el que lo realizan los navegadores modernos.

Con esta librería se puede:

- Scrapear y parsear HTML de una URL, fichero o cadena de texto.
- Encontrar y extraer datos mediante recorridos del DOM o selectores CSS.
- Manipular elementos HTML, atributos y texto.
- Limpiar el contenido enviado por otros usuarios para prevenir ataques XSS.
- Generar HTML ordenados.

En el contexto del trabajo, se utiliza para poder obtener los títulos de los artículos del archivo xml obtenido de DBLP.

2.7. JSON [9] [10]

JSON es un formato para se utiliza para el intercambio de datos, ya que, destaca por su legibilidad y facilidad.

Está basado en un subconjunto del lenguaje de programación denominado *JavaScript*, aunque, JSON es independiente del lenguaje de programación utlizado. Admite valores de tipo cadena, numérico y booleano, pero no permite valores octales ni hexadecimales, por lo que, es recomendable que los tipos coincidan entre los datos JSON y los objetos de datos correspondientes.

En el contexto del trabajo se utiliza para poder obtener el JSON proporcionado por SerpApi donde se encuentra la url para poder descargar el artículo a analizar en formato PDF.

Capítulo 3

Objetivos y metodología

3.1. Objetivos

A continuación, se exponen los objetivos a conseguir con la aplicación.

Objetivos:

- Poder acceder a los pdfs de los artículos que se encuentran en DBLP.
- En los pdfs ser capaz de poder localizar los enlaces a las referencias utilizadas.
- Acceder a esos enlaces para comprobar si existen, y si no existen estimar el motivo.

3.1.1. Acceso a los pdfs de los artículos

Este objetivo se centra en proveer a los usuarios la información contenida en los artículos, muchos de los cuales son de pago, de forma totalmente gratuita. Esta parte resulta crucial para la investigación científica.

3.1.2. Localizar los enlaces utilizados como referencias

En este caso, el objetivo se centra en poder localizar todas los enlaces que aparecen en el apartado de referencias de los artículos. Esto resulta de mucha utilidad, ya que de esta forma, podemos comprobar si dichos enlaces siguen vigentes en la actualidad o han sido eliminados.

3.1.3. Acceso a enlaces y estimación

Este objetivo se centra en, acceder a los enlaces extraídos del apartado de referencias de los artículos, para poder verificar su existencia. Además, si dichos enlaces ya no existen, el poder estimar el motivo de su desaparición ayuda a poder realizar estadísticas sobre ello.

3.2. Metodología

El desarrollo de la aplicación se basa en ciclos. En el primer ciclo, se realizó una versión básica de la aplicación que cumplía con los objetivos descritos anteriormente.

Esto se consigue recopilando todas las URLs que aparecen en las referencias de los artículos. Una vez las tenemos, se establece una conexión con ellas y se comprueba que el código devuelto por el servidor es un código de éxito (número comprendido entre 200 – 299). Esto se realiza ya que, con un código devuelto satisfactorio se verifica la existencia de la URL.

Para poder llevar a cabo este análisis, la aplicación sigue una serie de pasos. Primero lee la información contenida en un archivo .xml procedente de la librería digital DBLP, para después extraer los títulos de los artículos presentes. A continuación, busca, utilizando SerpApi de Google Scholar [6], una URL que permite la descarga de los artículos en formato PDF, gracias a los títulos extraídos. Una vez se obtienen los enlaces de descarga, se procede a descargar los PDFs solo una vez. Después, se lee el contenido del PDF para poder extraer las URLs que aparecen en las referencias del artículo, procediendo a verificar su existencia.

Se espera que esta aplicación pueda detectar si las URLs, que se toman como referencias en los artículos científicos, existen o no para poder realizar un análisis de cómo va desapareciendo la información que está contenida en Internet.

En el segundo ciclo, se incluyen mejoras para la eficiencia y el rendimiento de la aplicación, como la adicción de una clase que permite ejecutar la aplición seguidamente, y no ejecutar las distintas clases de forma individual.

Además, se especifican una serie de requisitos tanto funcionales como no funcionales que se pueden encontrar en el Anexo A.

Capítulo 4

Plan de trabajo

4.1. Plan de trabajo preliminar

Antes de iniciar con el desarrollo del TFG, se elaboró un Plan de trabajo, el cuál, se comentará a continuación.

4.1.1. Descripción general del trabajo

36
Consiste en la realización de una aplicación que nos permita ir comprobando si los enlaces utilizados en los artículos, para apoyar o justificar sus conclusiones, siguen existiendo.

Está relacionado con el llamado "Fraude científico".

El objetivo principal es estudiar la cantidad de información científica que se pierde en internet, identificando los mecanismos y el tiempo en el que se producen, de forma automatizada.

Objetivos:

- Poder acceder a los pdfs de los artículos que se encuentran en DBLP.
- En los pdfs ser capaz de poder localizar los enlaces a las referencias utilizadas.
- Acceder a esos enlaces para comprobar si existen, y si no existen estimar su fecha de desaparición.

4.1.2. Lista de tareas

46
A continuación se muestra una lista con las tareas que debe realizar la aplicación:

- Leer el archivo .xml de DBLP para poder acceder a los artículos.
- Localizar los enlaces a los pdf de los artículos.
- Descargar el pdf de los artículos.

- Ser capaz de leer el contenido del pdf descargado.
- Ser capaz de encontrar los enlaces a las referencias.
- Acceder a los enlaces ver si existen.
- Si no existen los enlaces, estimar cuándo han podido desaparecer.
- Manejar los captchas.
- Manejar la situación de cambio de red pública a red privada y viceversa.
- Tener una caché para que en el caso en el que tengamos que pagar por acceso a un pdf no se vuelva a comprar si ya se ha realizado previamente.

4.1.3. Diagrama de Gantt

Figura 4.1: Diagrama de Gantt - Plan de trabajo preliminar

4.2. Plan de trabajo intermedio

Al alcanzar la mitad del trabajo, el plan de trabajo fue revisado para poder realizar las modificaciones necesarias.

4.2.1. Revisión de la lista de objetivos del trabajo

En cuanto a los objetivos del trabajo se ha añadido el exportar en formato CSV los datos para poder realizar un análisis de lo obtenido a través de la aplicación:

- Poder acceder a los pdfs de los artículos que se encuentran en DBLP.
- En los pdfs ser capaz de poder localizar los enlaces a las referencias utilizadas.
- Acceder a esos enlaces para comprobar si existen, y si no existen estimar su fecha de desaparición.
- Exportar en formato CSV los datos obtenidos.

4.2.2. Revisión de la lista de tareas

- Leer el archivo .xml de DBLP para poder acceder a los artículos.
- Localizar los enlaces a los pdf de los artículos.
- Descargar el pdf de los artículos.
- Ser capaz de leer el contenido del pdf descargado.
- Ser capaz de encontrar los enlaces a las referencias.
- Acceder a los enlaces ver si existen.
- Si no existen los enlaces, estimar cuándo han podido desaparecer.
- Manejar los captchas.
- Manejar la situación de cambio de red pública a red privada y viceversa.
- Tener una caché para que en el caso en el que tengamos que pagar por acceso a un pdf no se vuelva a comprar si ya se ha hecho previamente.
- Obtener un archivo CSV con los datos de la base de datos.
- Analizar los datos contenidos en el CSV y obtener una conclusión.

Podemos ver que se han añadido al final de la lista dos puntos relacionados a la exportación y análisis de los datos obtenidos. Esto se ha producido, ya que, tras obtener los datos y almacenarlos en la base de datos a través de la aplicación, también es necesario realizar un aálisis de esos datos. Esto nos ayuda a buscar la causa del por qué desaparece la información en internet además de poder realizar una conclusión factible de lo mismo.

4.2.3. Revisión del Diagrama de Gantt

En cuanto al diagrama de Gantt, cabe recalcar que solo hay una ligera variación.

Esta variación afecta a la tarea de *Elaborar las estadísticas*. Según el diagrama de Gantt se debería haber empezado con esta tarea el 22/03/2024, sin embargo, comencé con el desarrollo del script en Python para poder exportar los datos el 18/04/2024. Por consiguiente, esto también afectó a la tarea de *Realizar el estudio del decaimiento de la información científica en internet*, ya que se está extendiendo por la mejoras que se quieren introducir hasta calculo 06/05/2024.

Además, la tarea de *Redactar TFG* la empecé el 10/04/2024, cuando estaba indicado que la empezaría el 15/04/2024.

En cuanto al resto de tareas se mantienen los tiempos.

Figura 4.2: Diagrama de Gantt - Plan de trabajo intermedio

4.3. Plan de trabajo final

A medida que el proyecto va avanzando, se han ido introduciendo nuevas mejoras en las tareas listadas anteriormente, que han implicado modificaciones en el diagrama de Gantt.

4.3.1. Revisión del Diagrama de Gantt

En cuanto al diagrama de Gantt, hay una ligera modificación en los tiempos de las tareas *Realizar el estudio del decaimiento de la información científica en internet*, *Elaborar las estadísticas*, *Redactar TFG*, *Realizar ficha resumen del TFG* y *Realizar PowerPoint para la presentación*, debido a la inclusión de mejoras en la aplicación. Además, se añadió una nueva tarea denominada *Inclusión de mejoras para la aplicación*.

Las nuevas fechas para dichas tareas son:

- Realizar el estudio del decaimineto de la información científica en internet: del 06/03/2024 al 30/05/2024
- Elaborar las estadísticas: del 30/05/2024 al 01/06/2024
- Redactar TFG: del 10/04/2024 al 02/06/2024
- Realizar ficha resumen del TFG: del 05/06/2024 al 07/06/2024
- Realizar PowerPoint para la presentación: del 05/06/2024 al 14/06/2024

Figura 4.3: Diagrama de Gantt - Plan de trabajo final

Capítulo 5

Desarrollo

El desarrollo de la aplicación se ha dividido en tres ciclos.

En el primer ciclo se ha desarrollado la aplicación con la funcionalidad básica que permite la extracción y la verificación de la existencia de las URLs obtenidas de la descarga, en formato PDF de un artículo online contenido en la librería digital DBLP.

En el segundo ciclo se han añadido las siguientes funcionalidades:

- Exportación de la base de datos a un fichero con terminación .csv
- Adición de la clase **RunEntireFlow.java** que al ejecutarla se encarga de invocar a las distintas clases que conforman la aplicación.

En cuanto al tercer ciclo, la funcionalidad de la aplicación no ha variado, pero se han incluido una serie de mejoras en cada una de las clases que se especificarán con detalle.

5.1. Ciclo 1

5.1.1. Clases y relaciones entre clases

Al inicio, la aplicación carecía de una base de datos para poder almacenar el título de los PDFs descargados, ni sus URLs tanto accesibles como no:

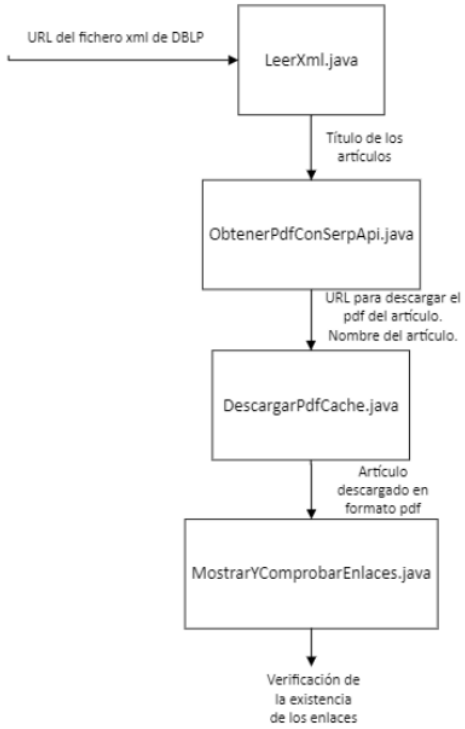


Figura 5.1: Conexión de clases sin base de datos.

Sin embargo, para poder mantener un orden, incluí la base de datos, por lo que agregué una clase más denominada **ConexionBaseDeDatos.java**. Esta clase, permite a la clase **DescargarPdfCache.java** introducir el título de un artículo en formato PDF al descargarlo, y controlar que no se descargue más de una vez. Además, posibilita a la clase **MostrarYComprobarEnlaces.java** introducir las URLs de cada PDF, indicando si son accesibles o no y haciendo referencia a el título del artículo que corresponde para poder identificar a cuál de ellos pertenece.

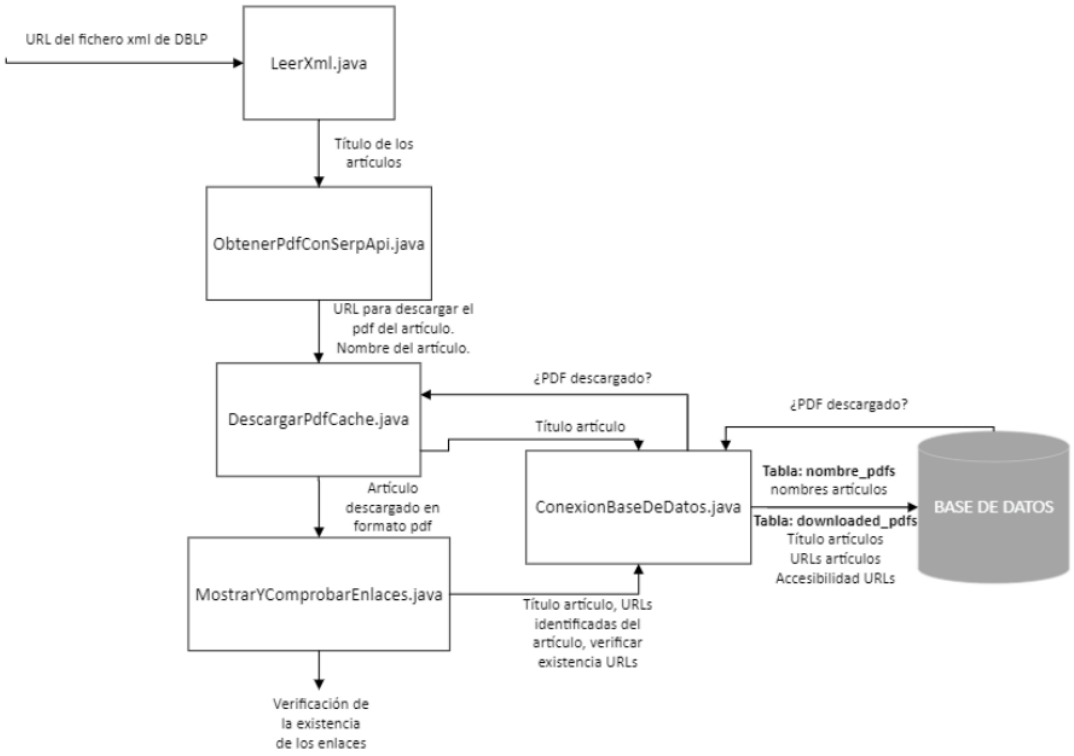


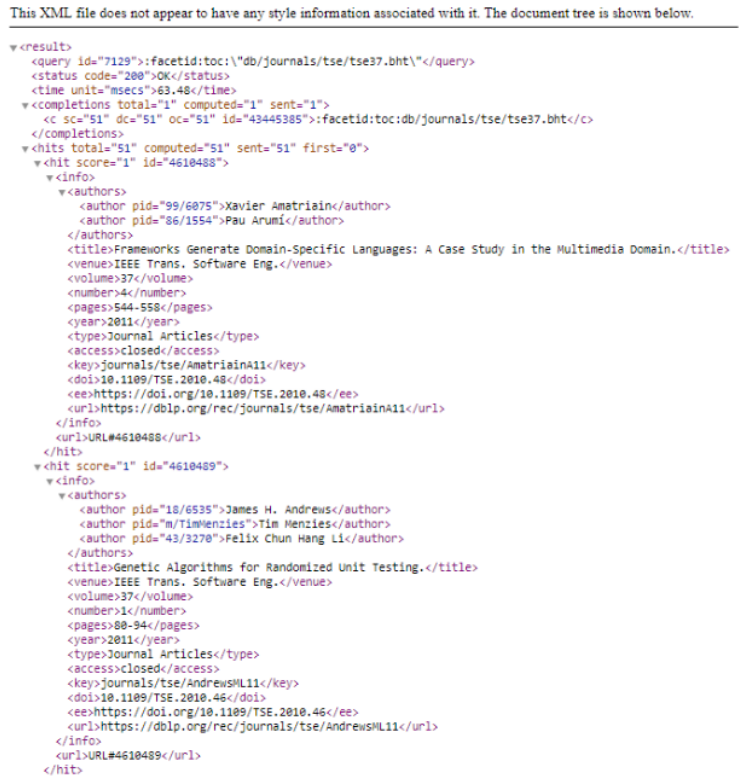
Figura 5.2: Conexión de clases con base de datos.

5.1.2. Ejecución

A continuación, describiré el funcionamiento de la aplicación en relación a ambas conexiones de clases indicadas anteriormente.

Clases sin base de datos

En primer lugar, se ejecuta la clase **LeerXml.java** con la URL del fichero xml obtenido desde la base de datos digital DBLP. Esta clase se encarga de extraer los títulos de los artículos contenidos en el fichero xml. Un ejemplo de fichero xml:



This XML file does not appear to have any style information associated with it. The document tree is shown below.


```
▼<result>
   <query id="7129">:facetid:toc:\"db/journals/tse/tse37.bht\"</query>
   <status code="200">OK</status>
   <time unit="msecs">63.48</time>
  ▼<completions total="1" computed="1" sent="1">
     <c sc="51" dc="51" oc="51" id="43445385">:facetid:toc:db/journals/tse/tse37.bht</c>
   </completions>
  ▼<hits total="51" computed="51" sent="51" first="0">
    ▼<hit score="1" id="4610488">
      ▼<info>
        ▼<authors>
           <author pid="99/6075">Xavier Amatriain</author>
           <author pid="86/1554">Pau Arumí</author>
         </authors>
         <title>Frameworks Generate Domain-Specific Languages: A Case Study in the Multimedia Domain.</title>
         <venue>IEEE Trans. Software Eng.</venue>
         <volume>37</volume>
         <number>4</number>
         <pages>544-558</pages>
         <year>2011</year>
         <type>Journal Articles</type>
         <access>closed</access>
         <key>journals/tse/AmatriainA11</key>
         <doi>10.1109/TSE.2010.48</doi>
         <ee>https://doi.org/10.1109/TSE.2010.48</ee>
         <url>https://dblp.org/rec/journals/tse/AmatriainA11</url>
       </info>
       <url>URL#4610488</url>
     </hit>
    ▼<hit score="1" id="4610489">
      ▼<info>
        ▼<authors>
           <author pid="18/6535">James H. Andrews</author>
           <author pid="m/TimMenzies">Tim Menzies</author>
           <author pid="43/3270">Felix Chun Hang Li</author>
         </authors>
         <title>Genetic Algorithms for Randomized Unit Testing.</title>
         <venue>IEEE Trans. Software Eng.</venue>
         <volume>37</volume>
         <number>1</number>
         <pages>80-94</pages>
         <year>2011</year>
         <type>Journal Articles</type>
         <access>closed</access>
         <key>journals/tse/AndrewsML11</key>
         <doi>10.1109/TSE.2010.46</doi>
         <ee>https://doi.org/10.1109/TSE.2010.46</ee>
         <url>https://dblp.org/rec/journals/tse/AndrewsML11</url>
       </info>
       <url>URL#4610489</url>
     </hit>
```


Figura 5.3: Ejemplo de fichero xml.

Ejemplo de títulos obtenidos:

```
Titulo: Frameworks Generate Domain-Specific Languages: A Case Study in the Multimedia Domain.
Titulo: Genetic Algorithms for Randomized Unit Testing.
Titulo: Self-Supervising BPEL Processes.
Titulo: Loupe: Verifying Publish-Subscribe Architectures with a Magnifying Lens.
Titulo: FlowTalk: Language Support for Long-Latency Operations in Embedded Devices.
Titulo: Toward a Formalism for Conservative Claims about the Dependability of Software-Based Systems.
Titulo: Developing a Single Model and Test Prioritization Strategies for Event-Driven Software.
Titulo: Zebu: A Language-Based Approach for Network Protocol Message Processing.
```

Figura 5.4: Títulos obtenidos.

A continuación, se ejecuta la clase **ObtenerPdfConSerpApi.java** con los títulos obtenidos a partir de las etiquetas <title>. Esta clase devuelve, el nombre del artículo junto con la primera URL de descarga del propio artículo en formato PDF, la cuál, se obtiene a partir del JSON del artículo cuyo nombre es buscado a través de SerpApi de Google Scholar. La URL de descarga en formato PDF se encuentra en la sección de resources"del JSON:


```
      }
    ]
  },
  "resources": [
    {
      "title": "arxiv.org",
      "file_format": "PDF",
      "link": "https://arxiv.org/pdf/2107.11789"
    }
  ],
  "inline links": {
```


Figura 5.5: Sección resources de un JSON.

Luego, se ejecuta la clase **DescargarPdfCache.java** con la URL de descarga del artículo en formato PDF, que descarga el PDF en local solo si no se ha descargado previamente. Además, proporciona la ruta local donde se descarga.

Si no se ha descargado previamente:

```
PDF descargado correctamente en:
C:\Users\Admin\Desktop\PDF descargado\Self-supervising BPEL Processes.pdf
```

Figura 5.6: Mensaje si no se ha descargado previamente.

Si se ha descargado el PDF previamente se muestra el mensaje siguiente:

```
El PDF ya ha sido descargado previamente.
```

Figura 5.7: Mensaje si se ha descargado previamente.

Finalmente, se ejecuta la clase **MostrarYComprobarEnlaces.java** con la ruta de descarga local del PDF. Esta clase se encarga de leer el contenido del PDF para extraer las URLs de referencia, además de verificar si estas existen o no:

```
Enlaces encontrados y verificados:
http://www.clam-project.org
El enlace existe.
```

Figura 5.8: Ejemplo de enlaces.

Clases con base de datos

El funcionamiento es muy parecido al anterior, solo modificándose por la introducción de la clase **ConexionBaseDeDatos.java** para poder manipular la base de datos.

En primer lugar, se ejecuta la clase **LeerXml.java** junto con la URL del fichero .xml obtenido de DBLP. Esta ejecución nos proporciona los títulos de los artículos contenidos en el fichero. Ejemplo de fichero .xml :


```
<hit score="1" id="4646506">
  <info>
    <authors>
      <author pid="15/2194">Radu Calinescu</author>
      <author pid="55/4092">Lars Grunske</author>
      <author pid="k/MartaZKwiatkowska">Marta Z. Kwiatkowska</author>
      <author pid="80/191">Raffaela Mirandola</author>
      <author pid="02/2374">Giordano Tamburrelli</author>
    </authors>
    <title>Dynamic QoS Management and Optimization in Service-Based Systems.</title>
    <venue>IEEE Trans. Software Eng.</venue>
    <volume>37</volume>
    <number>3</number>
    <pages>387-409</pages>
    <year>2011</year>
    <type>Journal Articles</type>
    <access>closed</access>
    <key>journals/tse/CalinescuGKMT11</key>
    <doi>10.1109/TSE.2010.92</doi>
    <ee>https://doi.org/10.1109/TSE.2010.92</ee>
    <url>https://dblp.org/rec/journals/tse/CalinescuGKMT11</url>
  </info>
  <url>URL#4646506</url>
</hit>
<hit score="1" id="4646507">
  <info>
    <authors>
      <author pid="77/5805">Laura Carnevali</author>
      <author pid="02/7786">Lorenzo Ridi</author>
      <author pid="50/3792">Enrico Vicario</author>
    </authors>
    <title>Putting Preemptive Time Petri Nets to Work in a V-Model SW Life Cycle.</title>
    <venue>IEEE Trans. Software Eng.</venue>
    <volume>37</volume>
    <number>6</number>
    <pages>826-844</pages>
    <year>2011</year>
    <type>Journal Articles</type>
    <access>closed</access>
    <key>journals/tse/CarnevaliRV11</key>
    <doi>10.1109/TSE.2011.4</doi>
    <ee>https://doi.org/10.1109/TSE.2011.4</ee>
    <url>https://dblp.org/rec/journals/tse/CarnevaliRV11</url>
  </info>
  <url>URL#4646507</url>
</hit>
```


Figura 5.9: Ejemplo de fichero xml 2.

Ejemplo de títulos obtenidos:

```
Titulo: Dynamic Software Updating Using a Relaxed Consistency Model.
Titulo: On the Distribution of Bugs in the Eclipse System.
Titulo: A Controlled Experiment for Program Comprehension through Trace Visualization.
Titulo: A Classification Framework for Software Component Models.
Titulo: Measuring the Discriminative Power of Object-Oriented Class Cohesion Metrics.
Titulo: Bristlecone: Language Support for Robust Software Applications.
Titulo: Systematic Review and Aggregation of Empirical Studies on Elicitation Techniques.
Titulo: From UML to Petri Nets: The PCM-Based Methodology.
Titulo: Automatically Detecting and Tracking Inconsistencies in Software Design Models.
Titulo: A Comparison of Tabular Expression-Based Testing Strategies.
Titulo: Swarm Verification Techniques.
Titulo: An Analysis and Survey of the Development of Mutation Testing.
```

Figura 5.10: Títulos obtenidos 2.

A continuación, se ejecuta la clase **ObtenerPdfConSerpApi.java** con los títulos conseguidos, obteniéndose el nombre del artículo y la primera URL de descarga del artículo en formato PDF. Esta URL se obtiene de la sección de resources"del JSON del artículo obtenido por SerpApi de Google Scholar a partir de los títulos indicados.

```
Título del artículo: Swarm Verification Techniques.
El primer enlace encontrado es: https://ir.library.oregonstate.edu/downloads/2801ph654
```

Figura 5.11: Titulo y ruta de PDF.

Luego, se ejecuta la clase **DescargarPdfCache.java** junto con la URL de descarga en formato PDF. Esta clase, descarga el PDF en local solo si no se ha descargado previamente, además de proporcionar la ruta local donde se almacena.

Para saber si el PDF no se ha descargado de forma previa, esta clase se comunica con **ConexionBaseDeDatos.java** que es la clase utilizada para poder manipular la base de datos. Antes de descargar el PDF, **DescargarPdfCache.java** mira en la tabla *nombre_pdfs* de la base de datos si se encuentra el título del artículo que se quiere descargar:

- Si se encuentra no descarga el PDF e indica que ya está descargado.
- Si no se encuentra presente introduce el DOI del artículo a través de la clase **ConexionBaseDeDatos.java** en la tabla y descarga el artículo en formato PDF.

Una vez está descargado el PDF se procede a la ejecución de la clase **MostrarY-ComprobarEnlaces.java** indicando la ruta local de descarga del artículo.

Esta clase lee el contenido del PDF descargado y se encarga de extraer las URLs que se encuentran en el mismo, además de verificar si existen o no.

Las URLs extraídas las introduce en la tabla *downloaded_pdfs* gracias a la clase **ConexionBaseDeDatos.java** junto con el título del artículo del que se han extraído. Además, también indica el código de respuesta del servidor, que si se encuentra entre 200 y 300 significa que la URL es accesible, es decir, que existe; si no se encuentra en ese intervalo implica que no está accesible y al contar con

el código de respuesta podemos saber la razón por la cuál no se pudo acceder a ella.

Para poder visualizar las tablas creadas en la base de datos se puede utilizar la aplicación "DB Browser (SQLite)", ya que se trata de una base de datos SQLite. Con esa aplicación, una vez se ha abierto la base de datos, se puede visualizar el contenido de la misma, las tablas y los datos que contienen, además de la estructura:

Estructura de la base de datos:

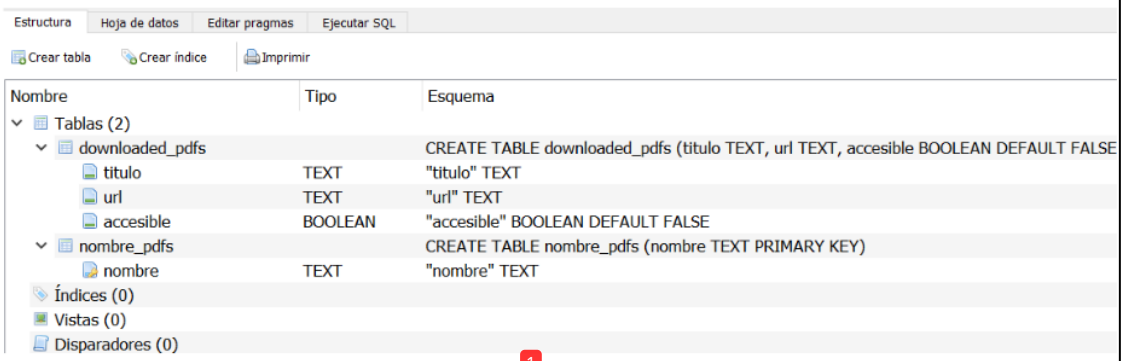


Figura 5.12: Estructura Base de Datos.

Ejemplo de contenido de la tabla *nombre-pdfs*:

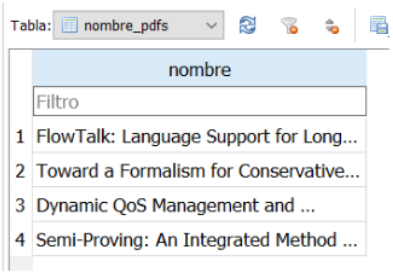
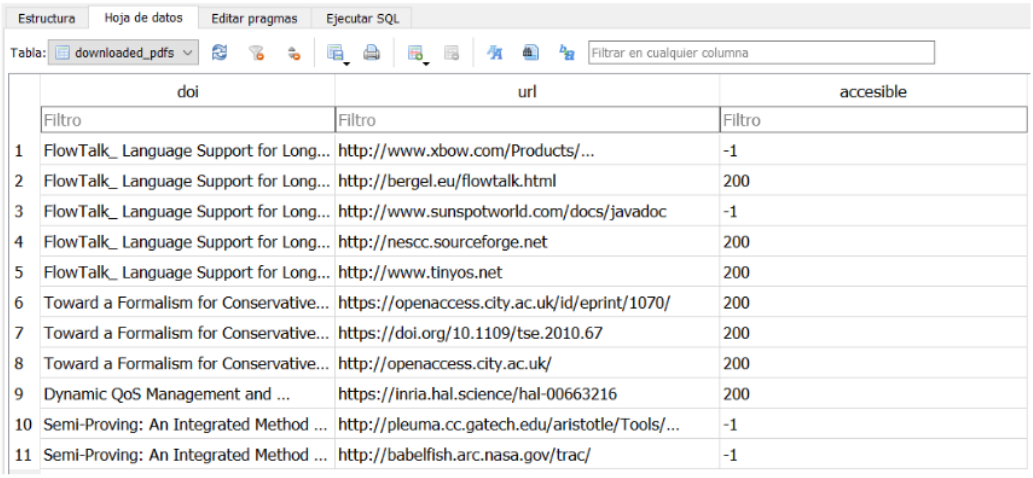


Figura 5.13: Tabla nombre_pdfs.

Ejemplo de contenido de la tabla *downloaded-pdfs*:



Estructura | Hoja de datos | Editar pragmas | Ejecutar SQL

Tabla: downloaded_pdfs | Filtrar en cualquier columna

	doi	url	accesible
	Filtro	Filtro	Filtro
1	FlowTalk_ Language Support for Long...	http://www.xbow.com/Products/...	-1
2	FlowTalk_ Language Support for Long...	http://bergel.eu/flowtalk.html	200
3	FlowTalk_ Language Support for Long...	http://www.sunspotworld.com/docs/javadoc	-1
4	FlowTalk_ Language Support for Long...	http://nescc.sourceforge.net	200
5	FlowTalk_ Language Support for Long...	http://www.tinyos.net	200
6	Toward a Formalism for Conservative...	https://openaccess.city.ac.uk/id/eprint/1070/	200
7	Toward a Formalism for Conservative...	https://doi.org/10.1109/tse.2010.67	200
8	Toward a Formalism for Conservative...	http://openaccess.city.ac.uk/	200
9	Dynamic QoS Management and ...	https://inria.hal.science/hal-00663216	200
10	Semi-Proving: An Integrated Method ...	http://pleuma.cc.gatech.edu/aristotle/Tools/...	-1
11	Semi-Proving: An Integrated Method ...	http://babelfish.arc.nasa.gov/trac/	-1

Figura 5.14: Tabla downloaded_pdfs.

5.2. Ciclo 2

5.2.1. Clases y relaciones entre clases

En este ciclo, se añadió una clase denominada **RunEntireFlow.java**, la cuál, al ejecutarse se encarga de ir llamando a todas las clases de forma automática, pasándoles los argumentos que precisan cada una, para evitar que el usuario tenga que ir ejecutando cada una de forma individual.

En este caso la conexión de clases se produce según el siguiente diagrama:

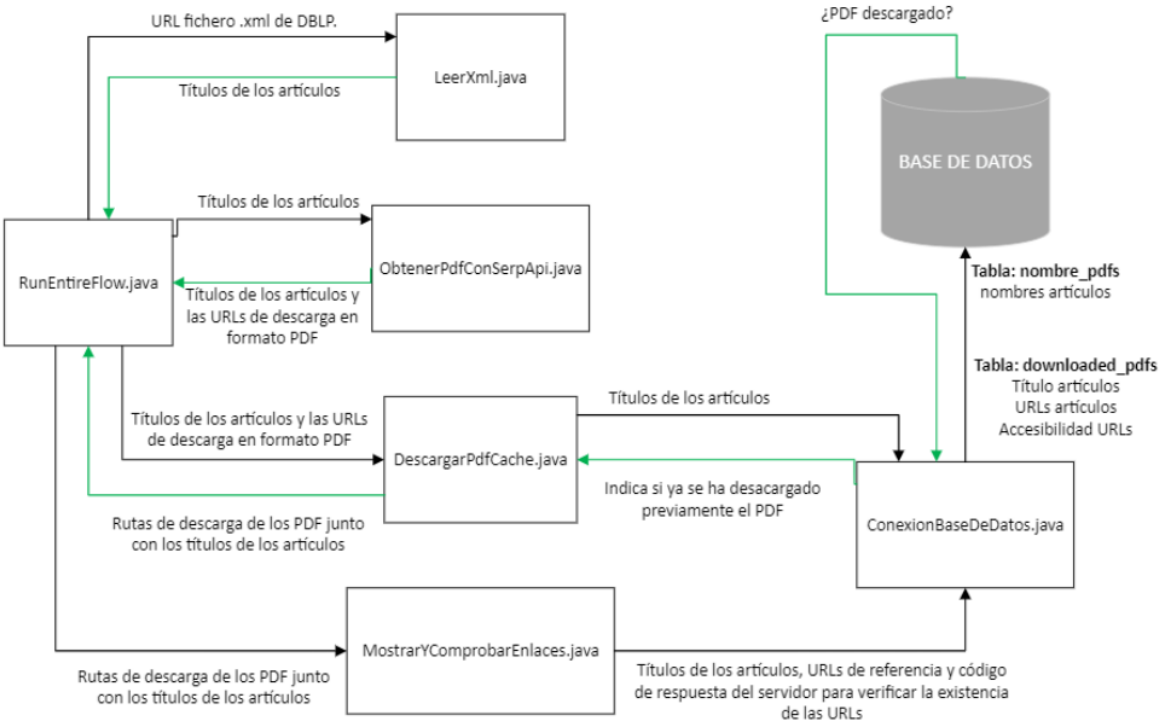


Figura 5.15: Diagrama conexión de clases ciclo 2.

5.2.2. Ejecución

A continuación, se expone el funcionamiento de la aplicación gracias a la adición de la clase **RunEntireFlow.java**.

Se ejecuta dicha clase añadida con la URL del fichero .xml obtenido de DBLP.

Ejemplo de fichero .xml :


```
▼<result>
  <query id="612670">conf* icse* 2024* workshop*</query>
  <status code="200">OK</status>
  <time unit="msecs">140.15</time>
 ▼<completions total="1" computed="1" sent="1">
    <c sc="68" dc="68" oc="68" id="55576254">workshop</c>
  </completions>
 ▼<hits total="68" computed="68" sent="3" first="0">
  ▼<hit score="5" id="83838">
   ▼<info>
    ▼<authors>
       <author pid="09/6502">Junqiang Li</author>
       <author pid="312/1887">Senyi Li</author>
       <author pid="257/1399">Keyao Li</author>
       <author pid="367/5897">Falin Luo</author>
       <author pid="94/60">Hongfang Yu</author>
       <author pid="66/5479-1">Shanshan Li 0001</author>
       <author pid="40/1491-78">Xiang Li 0078</author>
     </authors>
     <title>ECFuzz: Effective Configuration Fuzzing for Large-Scale Systems.</title>
     <venue>ICSE</venue>
     <pages>48:1-48:12</pages>
     <year>2024</year>
     <type>Conference and Workshop Papers</type>
     <access>closed</access>
     <key>conf/icse/LiLLLYLL24</key>
     <doi>10.1145/3597503.3623315</doi>
     <ee>https://doi.org/10.1145/3597503.3623315</ee>
     <url>https://dblp.org/rec/conf/icse/LiLLLYLL24</url>
   </info>
   <url>URL#83838</url>
  </hit>
 ▼<hit score="4" id="83808">
```


Figura 5.16: Ejemplo de fichero xml 3.

Tras la ejecución, **RunEntireFlow.java** invoca a **LeerXml.java** que proporciona el nombre de los artículos que contiene el fichero .xml. Dichos títulos se guardan en una variable denominada *titulosArticulos*, que utiliza **RunEntireFlow.java** para llamar a **ObtenerPdfConSerpApi.java**.

Esta clase se encarga, si no encuentra una URL de descarga del artículo en formato PDF, de mostrar el JSON del artículo. Si por el contrario encuentra una URL de descarga, se encarga de guardar en una variable de tipo String [][], llamada *titulosYEnlaces*, el título de cada artículo con su URL de descarga correspondiente.

Con esa variable se invoca a **DescargarPdfCache.java**, la cual, descarga, si no se ha descargado previamente, el artículo en formato PDF en una carpeta denominada *cache*. Además, introduce el nombre de dicho PDF en la base de datos, en la tabla *nombre_pdfs*. Además, guarda en una variable de tipo String [][],

denominada *rutasYTitulos*, la ruta de descarga del PDF junto con el título del artículo.

Utilizando dicha variable, se llama a la clase **MostrarYComprobarEnlaces.java** que se encarga de buscar las URLs que aparecen en las referencias de los artículos y verifica su existencia. Además, añade el nombre del artículo, la URL identificada y el código devuelto por el servidor (verificación de la exitencia de la URL) a la base de datos, a la tabla *downloaded_pdfs*.

5.3. Ciclo 3

Tras haber desarrollado el ciclo 2 de la aplicación, se presentó el problema de no poder utilizar las clases de forma individual.

Cuando se ejecuta la aplicación, se requiere la funcionalidad que permite introducir por línea de comandos un argumento, realizado por el propio usuario. Para poder conseguir esto, es necesario que cada clase pueda ser ejecutada de forma individual, es decir, que pueda utilizar argumentos pasados por la línea de comandos y, no solo aquellos que les proporciona en muchos casos la clase anterior. Por ejemplo, en el momento en el que la clase **ObtenerPdfConSerpApi.java** empieza a buscar una URL de descarga del artículo en formato PDF, en el JSON del propio artículo, cabe la posibilidad de que no lo encuentre. En ese caso, muestra por pantalla el JSON completo para poder dar la oportunidad al usuario de buscar la URL. Por ello, si el usuario encuentra dicha URL, es necesario que pueda ejecutar la siguiente clase, **DescargarPdfCache.java** directamente desde la linea de comandos.

Para poder resolver este tipo conflicto, se han modificado las clases que conforman la aplicación de la manera que se explica a acontinuación.

5.3.1. Modificaciones de las clases

LeerXml.java

La funcionalidad de esta clase no se ha modificado, sin embargo, se ha añadido una mejora.

Esta consta de agregar en el código, una variable que funciona como un contador de títulos, para poder mostrar por pantalla la cantidad de ellos que se encuentran en el fichero de DBLP.

ObtenerPdfConSerpApi.java

En esta clase, se añadió una variable contador también, que indica el número de enlaces leídos.

Además, se agregó una vaiable de tipo boolean que se encarga de no buscar el enlace de descarga de un artículo en formato PDF si ya se ha realizado con anterioridad. He decidido incluir esta nueva funcionalidad porque de esta forma,

se puede minimizar el número de búsquedas que realiza SerpApi, ya que son limitadas por mes.

DescargarPdfCache.java

En esta clase decidí cambiar el *main*. Anteriormente, se encontraba de forma *public static String[][] main(String[][] titulosYEnlaces)* y, actualmente se encuentra como *public static void main(String[] args) throws ClassNotFoundException*. Este cambio es debido a que por línea de comandos no se puede ejecutar la clase con un *String[][]*.

Además, agregué una funcionalidad, que permite forzar la descarga de los archivos PDFs en la caché. Para ello, tuve también que realizar un método que se encarga de chequear si existe el directorio cache, y si no lo crea, e incluir una variable de tipo boolean para indicar si es necesario forzar esa descarga.

Por último, también modifiqué ligeramente la lógica que se encarga de comprobar si el PDF ya se había descargado previamente. Anteriormente se realizaba a través de un *for each* en el *main*, utilizando la variable *tituloYEnlace* que se obtenía de la clase **ObtenerPdfConSerpApi.java**. Actualmente, esto se realiza en un método, el cuál, sigue guardando los títulos de los artículos y sus rutas de descarga, pero tiene en cuenta que la variable booleana que, indica si hay que forzar la descarga, se encuentra en estado *false*.

MostrarYComprobarEnlaces.java

En esta clase, se ha mejorado la interacción con la línea de comandos, pudiendo:

- Volver a transformar el PDF a texto, buscando de nuevo los enlaces y comprobando su accesibilidad. Para ello, se debe indicar al inicio *–regenerate-text*.
- Analizar de nuevo los enlaces de los ficheros, usando el texto existente y comprobar su accesibilidad. Para ello, se debe indicar al inicio *–force-link-search*.
- Comprobar la accesibilidad de los enlaces que están en la base de datos sin la necesidad de utilizar el texto del artículo. Para ello, se debe indicar al inicio *–use-db-only*.

Además, se ha mejorado la funcionalidad de la extracción y procesamiento de los enlaces. Esto se consiguió mediante la introducción de múltiples expresiones regulares para mejorar la precisión. También, se introdujo un marcador que permite identificar de forma más eficiente y precisa las URLs, además de que, permite manipular el texto del PDF para poder optimizar la extracción de las mismas.

Por último, se ha incluído una nueva funcionalidad que proporciona la capacidad de eliminar enlaces antiguos antes de insertar las nuevas versiones de los mismos, además de permitir, obtener y actualizar las URLs directamente desde la base de datos.

ConexionBaseDeDatos.java

En este caso, he incluído una serie de columnas a la tabla *downloaded_pdfs* de la base de datos. Estas nuevas columnas son *finalURL*, *contexto* y *type*, las cuales, proprocionan una visión más compleja y completa de la representación de los datos almacenados.

Además, he agregado nuevos métodos que se especifican a continuación:

- EliminarEnlacesDeUnArticulo, para poder limpiar la base de datos de enlaces que se encuentran obsoletos.
- getAllURLs, extrae las URLs que han sido identificadas y las devuelve como una lista.
- saveURLType, se encarga de clasificar o categorizar las URLs que se encuentran en la tabla *downloaded_pdfs*.
- BDExist, se encarga de verificar si la base de datos existe.
- updateResponseCode, actualiza el código de respuesta del servidor al intentar acceder a una URL.

Por último, se han modificado métodos existentes y el manejo de las excepciones para poder garantizar la compatibilidad con la nueva versión de la base de datos.

RunEntireFlow.java

En esta clase, he añadido mejoras en el manejo de los errores. Además, se integró una nueva clase denominada **ClasificarURL.java**.

5.3.2. Clases y relaciones entre clases

Como he indicado anteriormente, he agregado una nueva clase denominada **ClasificarURL.java**, que se encarga de clasificar las URLs presentes en la tabla *downloaded_pdfs* de la base de datos, gracias a un fichero llamado *sites.properties*.

En este ciclo, la conexión de clases se produce según se indica en el diagrama a continuación:

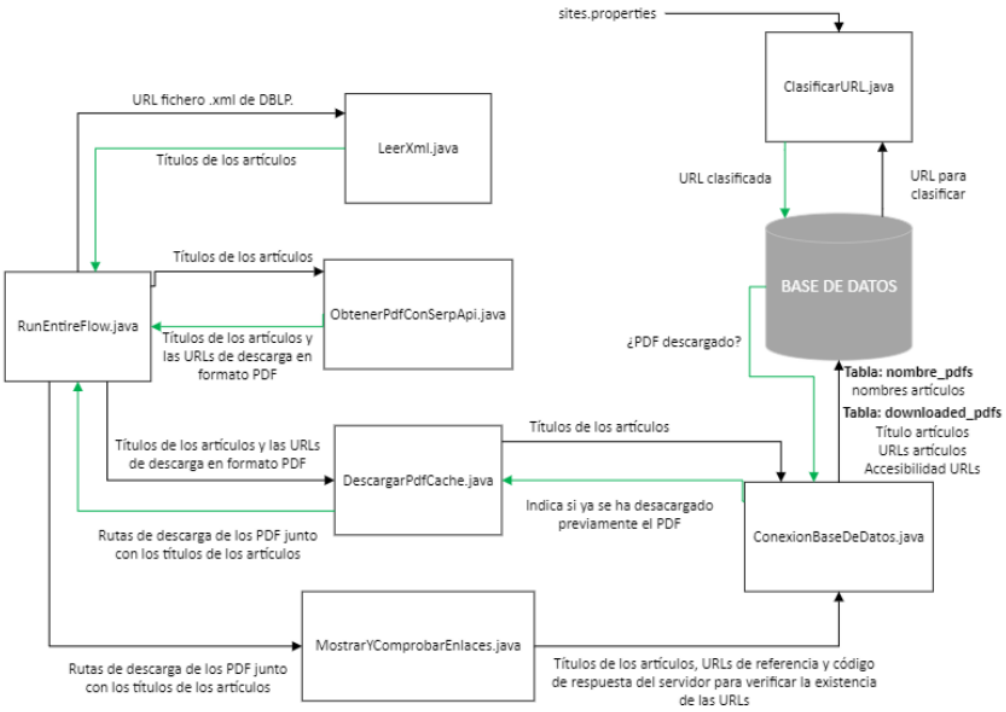


Figura 5.17: Diagrama conexión de clases ciclo 3.

5.3.3. Ejecución

En este apartado, explicaré el funcionamiento de la aplicación con las mejoras introducidas y la adición de la nueva clase.

Hay dos opciones para poder ejecutar la aplicación, las cuáles se detallan a continuación.

Opción 1: Ejecución individual de las clases

Se indican por línea de comandos, aquellos que están comprendidos en el fichero **runEntireFlow.sh**:

```
java -cp checkurl-0.0.1-SNAPSHOT-jar-with-dependencies.jar es.upm.grise.checkurl.LeerXml
$1 > titleList.txt 2> errorTitleList.txt

cat titleList.txt | xargs java -cp checkurl-0.0.1-SNAPSHOT-jar-with-dependencies.jar es.upm.grise.checkurl.ObtenerPdfConSerpApi
> urlList.txt 2> errorUrlList.txt

cat urlList.txt | xargs java -cp checkurl-0.0.1-SNAPSHOT-jar-with-dependencies.jar es.upm.grise.checkurl.DescargarPdfCache
> fileList.txt 2> errorFileList.txt

cat fileList.txt | xargs java -cp checkurl-0.0.1-SNAPSHOT-jar-with-dependencies.jar es.upm.grise.checkurl.MostrarYComprobarEnlaces
> links.txt 2> errorLinks.txt

java -cp checkurl-0.0.1-SNAPSHOT-jar-with-dependencies.jar es.upm.grise.checkurl.ClasificarURL
> classification.txt 2> errorClassification.txt
```

Figura 5.18: Comandos ejecución ciclo 3.

Tras la ejecución del primer comando, siendo el argumento la URL del fichero .xml de DBLP, crea dos ficheros denominados *titleList.txt* y *errorTitleList.txt*, si no ha sido creados previamente. A continuación, guarda en el fichero *titleList.txt* todos los nombres de los artículos que extrae **LeerXml.java**, y en *errorTitleList.txt* los errores si se producen.

Ejemplo de *titleList.txt*:

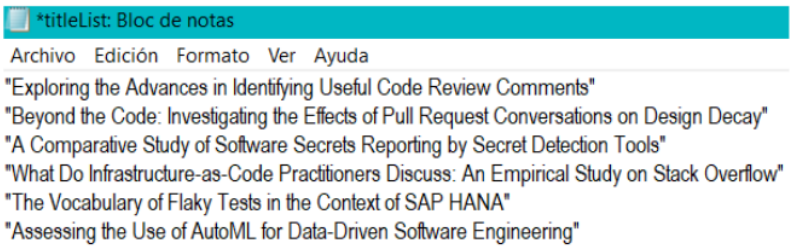


Figura 5.19: Ejemplo fichero *titleList.txt* ciclo 3.

Ahora, se indica el segundo comando, que señala que la clase **ObtenerPdfConSerpAPi.java** utiliza los títulos que se encuentran en el fichero generado *titleList.txt*, para buscar sus enlaces de descarga en formato PDF. Además, cuando encuentra los enlaces, los almacena en un fichero, que si no existe crea, denominado *urlList.txt*. Además, guarda los posibles errores en otro fichero denominado *errorUrlList.txt*.

Ejemplo de *urlList.txt*:

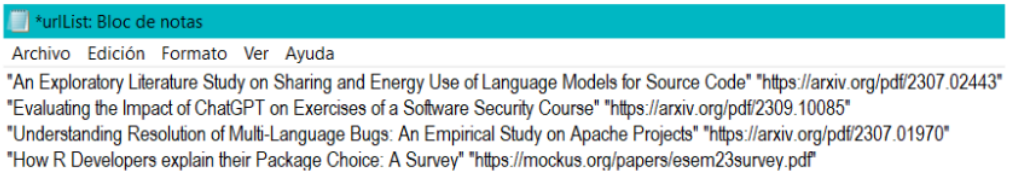


Figura 5.20: Ejemplo fichero *urlList.txt* ciclo 3.

A continución,se indica el tercer comando, que señala que la clase **DescargarPdfCache.java** utiliza los títulos y sus enlaces de descarga en formato PDF para poder descargar los artículos. Además genera si no existen dos ficheros de texto denominados *fileList.txt* y *errorFileList.txt*. En el primero se guardan la ruta de descarga del artículo junto con su título, y en el segundo los posibles errores.

Ejemplo de *fileList.txt*:

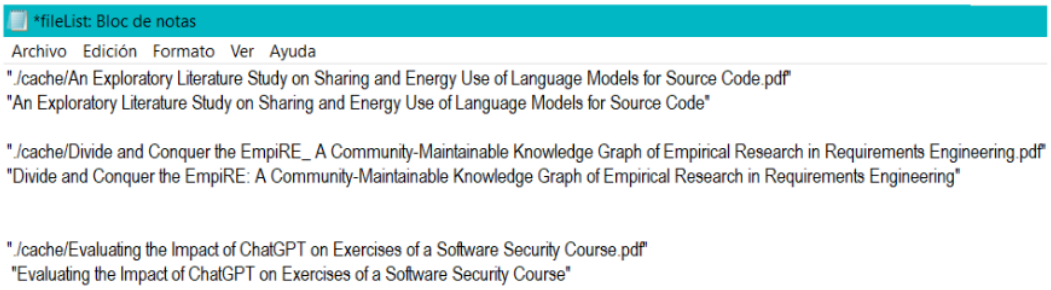


Figura 5.21: Ejemplo fichero *fileList.txt* ciclo 3.

Luego, se indica el cuarto comando, que señala que la clase **MostrarYComprobarEnlaces.java** utiliza las rutas de descarga y el nombre de los artículos. Además, almacena en un fichero, que si no existe crea, denominado *links.txt* los enlaces encontrados y en *errorLinks.txt* los posibles errores producidos.

Por último, se ejecuta el quinto comando, que señala que la clase **ClasificarURL.java** además de clasificar los enlaces en la base de datos, también indica dicha clasificación en un fichero denominado *classification.txt*, que crea si no existe previamente, y en *errorClassification.txt* los posibles errores generados.

Ejemplo de *classification.txt*:

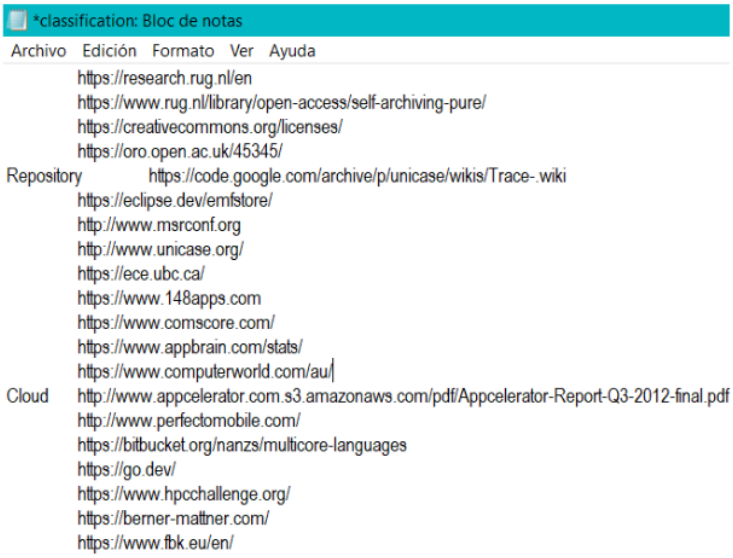


Figura 5.22: Ejemplo fichero *classification.txt* ciclo 3.

Ejemplo de base de datos:

Tabla: downloaded_pdfs — Filtrar en cualquier columna

	titulo	URL	finalURL	accesible	contexto	type
	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro
111	Personalized Guidelines for Design, ...	https://drive.google.com/file/d/...	https://drive.google.com/file/d/...	200	https://www.core.edu.au/.[38] "Cor...	Cloud
112	Personalized Guidelines for Design, ...	https://figshare.com/s/...	https://figshare.com/s/...	404	ew," Comput-ers & Security, vol. 105,...	
113	ToxiSpanSE: An Explainable Toxicity ...	https://github.com/WSU-SEAL/...	https://github.com/WSU-SEAL/...	200	dataset available for further...	Repository
114	ToxiSpanSE: An Explainable Toxicity ...	https://github.com/WSU-SEAL/...	https://github.com/WSU-SEAL/...	200	with 88% F11 score. We have made...	Repository
115	ToxiSpanSE: An Explainable Toxicity ...	https://www.perspectiveapi.com/	https://www.perspectiveapi.com/	200	016....	
116	ToxiSpanSE: An Explainable Toxicity ...	https://boxofcables.dev/toxicity-in-...	https://boxofcables.dev/toxicity-in-...	200	5, no. CSCW2, pp. 1–35, 2021....	
117	ToxiSpanSE: An Explainable Toxicity ...	https://medium.com/linuxforeveryone/	https://medium.com/linuxforeveryone	200	, "Windows is sh*t:' linux users...	
118	ToxiSpanSE: An Explainable Toxicity ...	https://www.linux4everyone.com/	https://www.linux4everyone.com/	200	860/...	
119	ToxiSpanSE: An Explainable Toxicity ...	https://2014.jsconf.eu/speakers/	https://2014.jsconf.eu/speakers/	200	. Reinhard, "This is bigger than us: ...	
120	ToxiSpanSE: An Explainable Toxicity ...	https://www.zdnet.com/article/	https://www.zdnet.com/?...	200	html...	
121	ToxiSpanSE: An Explainable Toxicity ...	https://arstechnica.com/gadgets/...	https://arstechnica.com/gadgets/...	200	"The perl foundation is fragment-...	
122	ToxiSpanSE: An Explainable Toxicity ...	https://doi.org/...	https://dl.acm.org/doi/...	200	g, ser. ASE...	Article
123	ToxiSpanSE: An Explainable Toxicity ...	https://github.com/heartexlabs/label...	https://github.com/HumanSignal/lab...	200	N. Liubimov, "Label...	Repository
124	ToxiSpanSE: An Explainable Toxicity ...	https://github.com/heartexlabs/label...	https://github.com/HumanSignal/lab...	200	ce software...	Repository
125	ToxiSpanSE: An Explainable Toxicity ...	https://github.com/tamuhey/textspan	https://github.com/tamuhey/textspan	200	preprint arXiv:1912.06872, 2019....	Repository
126	ToxiSpanSE: An Explainable Toxicity ...	https://github.com/fchollet/keras	https://github.com/keras-team/keras	200	lagiarism detection," in Coling 2010: ...	Repository
127	ToxiSpanSE: An Explainable Toxicity ...	https://github.com/WSU-SEAL/...	https://github.com/WSU-SEAL/...	200	015....	Repository
128	ToxiSpanSE: An Explainable Toxicity ...	https://competitions.codalab.org/...	https://competitions.codalab.org/...	200	16–27.[42] J. Pavlopoulos, "Semeval...	
129	ToxiSpanSE: An Explainable Toxicity ...	https://github.com/grrrr/krippendorff...	https://github.com/grrrr/krippendorff...	200	ll and S. Castro, "Python ...	Repository
130	ToxiSpanSE: An Explainable Toxicity ...	https://aclanthology.org/2020.emnlp-...	https://aclanthology.org/2020.emnlp-...	200	tions. Online: Association for ...	Book

Figura 5.23: Ejemplo tabla *downloaded_pdfs* ciclo 3.

Opción 2: Ejecución clase RunEntireFlow.java

Con esta opción, los resultados son los mismos, en relación a la base de datos, que en la Opción 1. Sin embargo, todos los documentos de texto que se generan con esta opción no se producen.

En cuanto al flujo de ejecución, es exactamente igual que el indicado en el ciclo 2, es decir, se ejecuta la clase **RunEntireFlow.java** junto con la URL del fichero .xml de DBLP. Luego, dicha clase se encarga de ir ejecutando y proporcionando los argumentos necesarios a cada clase individual.

Capítulo 6

Resultados

En este capítulo se mostrarán los resultados que se han obtenido a partir de la realización de un caso de estudio.

En dicho caso de estudio, se escogieron dos revistas con determinados artículos de los años 2013 y 2023 para comparar cómo desaparecía la información en internet en dos décadas distintas.

6.1. Revista 2013

Primero realicé el caso de estudio con la revista de 2013. Cuento con un fichero *Shell Script* denominado **runEntireFlow.sh** en el que se muestran los comandos para poder realizar la ejecución de la aplicación con el fichero .xml de DBLP seleccionado.

runEntireFlow.sh

```
java -cp checkurl-0.0.1-SNAPSHOT-jar-with-dependencies.jar es.upm.grise.checkurl.LeerXml
$1 > titleList.txt 2> errorTitleList.txt

cat titleList.txt | xargs java -cp checkurl-0.0.1-SNAPSHOT-jar-with-dependencies.jar es.upm.grise.checkurl.ObtenerPdfConSerpApi
> urlList.txt 2> errorUrlList.tex

cat urlList.txt | xargs java -cp checkurl-0.0.1-SNAPSHOT-jar-with-dependencies.jar es.upm.grise.checkurl.DescargarPdfCache
> fileList.txt 2> errorFileList.txt

cat fileList.txt | xargs java -cp checkurl-0.0.1-SNAPSHOT-jar-with-dependencies.jar es.upm.grise.checkurl.MostrarYComprobarEnlaces
> links.txt 2> errorLinks.txt

java -cp checkurl-0.0.1-SNAPSHOT-jar-with-dependencies.jar es.upm.grise.checkurl.ClasificarURL
> classification.txt 2> errorClassification.txt
```

Figura 6.1: Comandos de runEntireFlow.sh

Ejecuté la aplicación siguiendo los comandos indicados. Tras esta ejecución, obtuve la base de datos con las URLs y su verificación de existencia.

	título	URL	finalURL	accesible	contexto	type
	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro
1	An Embedded Multiple-Case Study on...	http://www.rug.nl/research/portal	https://research.rug.nl/en	200	investigate your claim....	
2	An Embedded Multiple-Case Study on...	https://www.rug.nl/library/open-...	https://www.rug.nl/library/open-...	200	ated by the "Taverne" license.More ...	
3	IR in Software Traceability: From a ...	https://creativecommons.org/licenses/	https://creativecommons.org/licenses/	200	the URL identifying the publication in ...	
4	Towards a Metric Suite Proposal to ...	https://oro.open.ac.uk/45345/	https://oro.open.ac.uk/45345/	200	...	
5	Tracing Requirements and Source Co...	http://code.google.com/p/unicase/...	https://code.google.com/archive/p/...	200	ware...	Repository
6	Tracing Requirements and Source Co...	http://eclipse.org/emfstore/	https://eclipse.dev/emfstore/	200	on...	
7	Tracing Requirements and Source Co...	http://www.msrconf.org	http://www.msrconf.org	200	hop on Cooperative and...	
8	Tracing Requirements and Source Co...	http://www.unicase.org/	http://www.unicase.org/	403	race Client, http://code.google.com/...	
9	Real Challenges in Mobile App ...	http://www.ece.ubc.ca/	https://ece.ubc.ca/	200	between 1-3 years, to 16% less tha...	
10	Real Challenges in Mobile App ...	http://148apps.biz/app-store-metrics/	https://www.148apps.com	200	ve Systems...	
11	Real Challenges in Mobile App ...	http://www.comscore.com/	https://www.comscore.com/	200	ore Metrics," >>>>>>>http://...	
12	Real Challenges in Mobile App ...	http://www.appbrain.com/stats/	https://www.appbrain.com/stats/	200	/comScore Reports January 2013 U.S...	
13	Real Challenges in Mobile App ...	http://www.computerworld.com.au/...	https://www.computerworld.com/au/	200	ybrid mobile-app development,"...	
14	Real Challenges in Mobile App ...	http://code.google.com/p/robotium/	https://github.com/robotiumtech/...	200	ed Theory: Issues and Discussions. ...	Repository
15	Real Challenges in Mobile App ...	http://developer.android.com/tools/...	https://developer.android.com/tools	200	California, 1998....	
16	Real Challenges in Mobile App ...	http://www.gorillalogic.com/	https://www.gorillalogic.com:443/	200	/developer.android.com/tools/help/...	
17	Real Challenges in Mobile App ...	https://github.com/square/	https://github.com/square/	200	ttp://www.gorillalogic.com/...	Repository
18	Real Challenges in Mobile App ...	http://cocoadev.com/wiki/...	http://cocoadev.com/wiki/...	200	egration Testing Framework," ...	
19	Real Challenges in Mobile App ...	https://developer.apple.com/library/...	https://developer.apple.com/library/...	200	"SenTestingKit Framework," ...	
20	Real Challenges in Mobile App ...	http://...	https://...	200	ightapp.com/....	
21	Real Challenges in Mobile App ...	http://www.berginsight.com/...	http://www.berginsight.com/...	404	/drippler.com/rim/blackberry ...	
22	Real Challenges in Mobile App ...	http://...	http://...	403	"Voice of the Next-Generation Mobile ...	Cloud
23	Real Challenges in Mobile App ...	http://www.perfectomobile.com/	http://www.perfectomobile.com/	403	tion/Introduction.html.[30] ...	

Figura 6.2: Base de datos - Caso de estudio 2013

Como se puede observar en la imágen, obtenemos la tabla de *downloaded_pdfs* con:

- Título del artículo
- URL que se verifica
- URL final, la cual es el resultado de la modificación de una URL que se encontraba rota, debido a por ejemplo, saltos de líneas o caracteres especiales.
- La columna de accedible, que nos indica el código de retorno del servidor para verificar la existencia de la URL.
- La columna de contexto, nos proporciona inforación sobre la URL.
- La columna type indica la clasificación de la URL.

Además de proporcionar la tabla *downloaded_pdfs*, también brinda cuatro documentos de texto:

titleList.txt 2013

Este documento recoge los títulos de todos los artículos presentes en el fichero .xml de DBLP:

"IR in Software Traceability: From a Bird's Eye View"
"Impact of Peer Code Review on Peer Impression Formation: A Survey"
"The Case for Knowledge Translation"
"Lessons from Conducting a Distributed Quasi-experiment"
"Towards a Metric Suite Proposal to Quantify Confirmation Biases of Developers"
"Identifying Barriers to the Systematic Literature Review Process"
"Tracing Requirements and Source Code during Software Development: An Empirical Study"
"Cost Effectiveness of Unit Testing: A Case Study in a Financial Institution"
"Application of Statistical Process Control to Software Defect Metrics: An Industry Experience Report"
"A Replicated Experiment on the Effectiveness of Test-First Development"
"Learning from Open-Source Projects: An Empirical Study on Defect Prediction"
"ScrumBut, But Does it Matter? A Mixed-Method Study of the Planning Process of a Multi-team Scrum Organization"
"Real Challenges in Mobile App Development"
"Experimental Comparison of Two Safety Analysis Methods and Its Replication"
"Towards Understanding Replication of Software Engineering Experiments"
"Message from the RESER 2013 Workshop Chairs"
"Message from the MTD 2013 Workshop Chairs"

Figura 6.3: Fragmento fichero *titleList.txt* 2013

urlList.txt 2013

En este documento se recogen las URLs que se ha verificado que existen actualmente.

"IR in Software Traceability: From a Bird's Eye View" "https://portal.research.lu.se/files/6380240/4091779.pdf"
"Impact of Peer Code Review on Peer Impression Formation: A Survey" "https://amiangshu.com/papers/Bosu-ESEM-2013.pdf"
"Towards a Metric Suite Proposal to Quantify Confirmation Biases of Developers" "https://oro.open.ac.uk/45345/1/ESEM2013.pdf"
"Tracing Requirements and Source Code during Software Development: An Empirical Study" "http://se.ifi.uni-heidelberg.de/fileadmin/pdf/publications/2013_ESEM_Delater.pdf"
"Real Challenges in Mobile App Development" "https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=b3076e3108068f6a7579dca49ba21551693fe869"

Figura 6.4: Fragmento fichero *urlList.txt* 2013

fileList.txt 2013

En este documento se guardan los títulos de los artículos que se encuentran en la carpeta *caché*, es decir, que han sido descargados.

"./cache/IR in Software Traceability_ From a Bird's Eye View.pdf" "IR in Software Traceability: From a Bird's Eye View"
"./cache/Impact of Peer Code Review on Peer Impression Formation_ A Survey.pdf" "Impact of Peer Code Review on Peer Impression Formation: A Survey"
"./cache/Towards a Metric Suite Proposal to Quantify Confirmation Biases of Developers.pdf" "Towards a Metric Suite Proposal to Quantify Confirmation Biases of Developers"

Figura 6.5: Fragmento fichero *fileList.txt* 2013

classification.txt 2013

Es este fichero, se clasifican según su tipo a las URLs.

Como se puede observar en este caso, hay un mayor número de URLs que se han identificado como *Cloud*.

Repository https://github.com/square/
http://cocoadev.com/wiki/SenTestingKit
https://developer.apple.com/library/archive/
https://www.keynotedeviceanywhere.com/
http://www.berginsight.com/ReportPDF/ProductSheet/bi-app1-ps.pdf
Cloud http://www.appcelerator.com.s3.amazonaws.com/pdf/Appcelerator-Report-Q3-2012-final.pdf
http://www.perfectomobile.com/
https://bitbucket.org/nanzs/multicore-languages
https://go.dev/
https://www.hpcchallenge.org/
https://berner-mattner.com/
https://www.fbk.eu/en/
http://www.berner-mattner.com/en/berner-mattner-home/products/cte
https://researchrepository.ul.ie/articles/conference_contribution/Can_automated_text_classification_improve_content_analysis_of_software_profect_data_/19850593
https://www.oemr.org/wiki/
http://snowball.tartarus.org/texts/introduction.html
https://ece.ubc.ca/
https://www.bugzilla.org/
https://www.jslint.com/
https://developers.google.com/closure/compiler/?csw=1

Figura 6.6: Fragmento fichero *classification.txt* 2013

6.2. Revista 2023

Una vez realicé el análisis de la desaparición de la información en internet, utilizando los artículos de la revista del año 2013, realicé el mismo análisis con los artículos de la revista del año 2023.

Llevé a cabo la ejecución de la aplicación siguiendo los comandos indicados en el fichero **runEntireFlow.sh**. Tras esta ejecución, obtuve la base de datos con las URLs y su verificación.

	titulo	URL	finalURL	accesible	contexto	type
	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro
1	A Comparative Study of Software ...	https://blog.gitguardian.com/uber-...	https://blog.gitguardian.com/uber-...	200	Online; accessed April 25, 2023]....	
2	A Comparative Study of Software ...	https://learn.g2.com/alert-fatigue	https://learn.g2.com/alert-fatigue	200	. 76–79, 2004....	
3	A Comparative Study of Software ...	https://cloud.google.com/storage	https://cloud.google.com/storage	200	ataset of Software Secrets," arXiv e-...	
4	A Comparative Study of Software ...	https://cloud.google.com/bigquery	https://cloud.google.com/bigquery	200	>>>>>>>https://cloud.google.com/...	
5	A Comparative Study of Software ...	https://dl.acm.org	https://dl.acm.org	200	>>>https://www.sciencedirect.com/...	Article
6	A Comparative Study of Software ...	https://link.springer.com	https://link.springer.com/	200	igital Library," >>>>>>>https://...	
7	A Comparative Study of Software ...	https://ieeexplore.ieee.org/Xplore/...	https://ieeexplore.ieee.org/Xplore/...	200	ngerLink," >>>>>>>https://...	Article
8	A Comparative Study of Software ...	https://dblp.org	https://dblp.org	200	>>>>>>>https://ieeexplore.ieee.or...	
9	A Comparative Study of Software ...	https://secdevtools.azurewebsites.n...	https://secdevtools.azurewebsites.n...	200	n open-source platforms with machin...	
10	A Comparative Study of Software ...	https://cycode.com	https://cycode.com	200	credscan.html,...	
11	A Comparative Study of Software ...	https://github.com/anshumanbh/git-...	https://github.com/anshumanbh/git-...	200	https://github.com/Yelp/detect- ...	Repository
12	A Comparative Study of Software ...	https://github.com/tillson/git-hound	https://github.com/tillson/git-hound	200	>https://github.com/anshumanbh/gi...	Repository
13	A Comparative Study of Software ...	https://github.com/michenriksen/...	https://github.com/michenriksen/...	200	und," >>>>>>>https://github.com/...	Repository
14	A Comparative Study of Software ...	https://github.com/kootenpv/gittyleaks	https://github.com/kootenpv/gittyleaks	200	>>>>>>>https://github.com/...	Repository
15	A Comparative Study of Software ...	https://github.com/awslabs/git-secrets	https://github.com/awslabs/git-secrets	200	erence on COMmunication Systems &...	Repository
16	A Comparative Study of Software ...	https://github.com/awslabs	https://github.com/awslabs	200	thub.com/awslabs/git-secrets, [Onlin...	Repository
17	A Comparative Study of Software ...	https://github.com/gitleaks/gitleaks	https://github.com/gitleaks/gitleaks	200	ces - Labs," >>>>>>>https://...	Repository
18	A Comparative Study of Software ...	https://github.com/auth0/repo-...	https://github.com/auth0/repo-...	200	>>>https://github.com/gitleaks/...	Repository
19	A Comparative Study of Software ...	https://github.com/trufflesecurity/...	https://github.com/trufflesecurity/...	200	>>>>https://github.com/auth0/repo...	Repository
20	A Comparative Study of Software ...	https://trufflesecurity.com	https://trufflesecurity.com	200	/github.com/trufflesecurity/trufflehog...	
21	A Comparative Study of Software ...	https://github.com/Skyscanner/...	https://github.com/Skyscanner/...	200	Security," >>>>>>>https://...	Repository
22	A Comparative Study of Software ...	https://github.com/GitGuardian/...	https://github.com/GitGuardian/...	200	>>>>>>>https://github.com/...	Repository
23	A Comparative Study of Software ...	https://api.gitguardian.com/docs	https://api.gitguardian.com/docs	200	n," >>>>>>>https://ww...	API

Figura 6.7: Base de datos - Caso de estudio 2023

Además de proporcionar la tabla denominada *downloaded_pdfs*, también muestra los tres primeros documentos de texto mencionados con anterioridad y adicionalmente uno nuevo.

titleList.txt 2023

Este documento muestra los títulos de los artículos que se encuntran en el fichero de DBLP del año 2023.

"Beyond the Code: Investigating the Effects of Pull Request Conversations on Design Decay"
"A Comparative Study of Software Secrets Reporting by Secret Detection Tools"
"What Do Infrastructure-as-Code Practitioners Discuss: An Empirical Study on Stack Overflow"
"On the Impact and Lessons Learned from Mindfulness Practice in a Real-World Software Company"
"The Vocabulary of Flaky Tests in the Context of SAP HANA"

Figura 6.8: Fragmento fichero *titleList.txt* 2023

urlList.txt 2023

En este caso, el documento presenta las URLs verificadas encontradas en los artículos de 2023.

"An Exploratory Literature Study on Sharing and Energy Use of Language Models for Source Code" "https://arxiv.org/pdf/2307.02443"
"Divide and Conquer the EmpiRE: A Community-Maintainable Knowledge Graph of Empirical Research in Requirements Engineering" "https://arxiv.org/pdf/2306.16791"
"Evaluating the Impact of ChatGPT on Exercises of a Software Security Course" "https://arxiv.org/pdf/2309.10085"
"Understanding Resolution of Multi-Language Bugs: An Empirical Study on Apache Projects" "https://arxiv.org/pdf/2307.01970"

Figura 6.9: Fragmento fichero *urlList.txt* 2023

fileList.txt 2023

Incluye el título de los artículos descargados correspondientes al año 2023.

"./cache/ToxiSpanSE_ An Explainable Toxicity Detection in Code Review Comments.pdf" "ToxiSpanSE: An Explainable Toxicity Detection in Code Review Comments"
"./cache/Replicability Study_ Corpora For Understanding Simulink Models & Projects.pdf" "Replicability Study: Corpora For Understanding Simulink Models & Projects"
"./cache/Comparing Mobile Testing Tools Using Documentary Analysis.pdf" "Comparing Mobile Testing Tools Using Documentary Analysis"
"./cache/Manual Tests Do Smell! Cataloging and Identifying Natural Language Test Smells.pdf" "Manual Tests Do Smell! Cataloging and Identifying Natural Language Test Smells"

Figura 6.10: Fragmento fichero *fileList.txt* 2023

error.txt

En este caso, decidí que la aplicación facilitara un documento donde se exponen las URLs que se han podido verificar por cada artículo, además de indicar el número de aquellas cuya verificación no ha resultado satisfactoria.

Articulo: Divide and Conquer the EmpiRE: A Community-Maintainable Knowledge Graph of Empirical Research in Requirements Engineering
---

Link identificado: https://www.w3.org/2000/01/rdf-schema (200)
Link identificado: https://orkg.org/comparison/R255464 (200)
Link identificado: https://hi-knowledge.org/ (200)
Link identificado: https://orkg.org/comparison/R112387 (200)
Link identificado: https://orkg.org/comparison/R114155 (200)
Link identificado: https://kgbook.org/ (200)
Link identificado: https://covid-aqs.fz-juelich.de (200)
Link identificado: https://zenodo.org/records/4456034 (200)
Link identificado: https://zenodo.org/records/4455951 (200)
Link identificado: https://zenodo.org/records/8083529 (200)
Link identificado: https://www.w3.org/TR/2017/REC-shacl-20170720/ (200)
Link identificado: https://paperswithcode.com/about (200)
Link identificado: https://www.w3.org/TR/2013/REC-sparql11-query-20130321/ (200)

Faltan por recuperar 3 enlaces en el fichero: ./cache/Divide and Conquer the EmpiRE_ A Community-Maintainable Knowledge Graph of Empirical Research in Requirements Engineering.txt

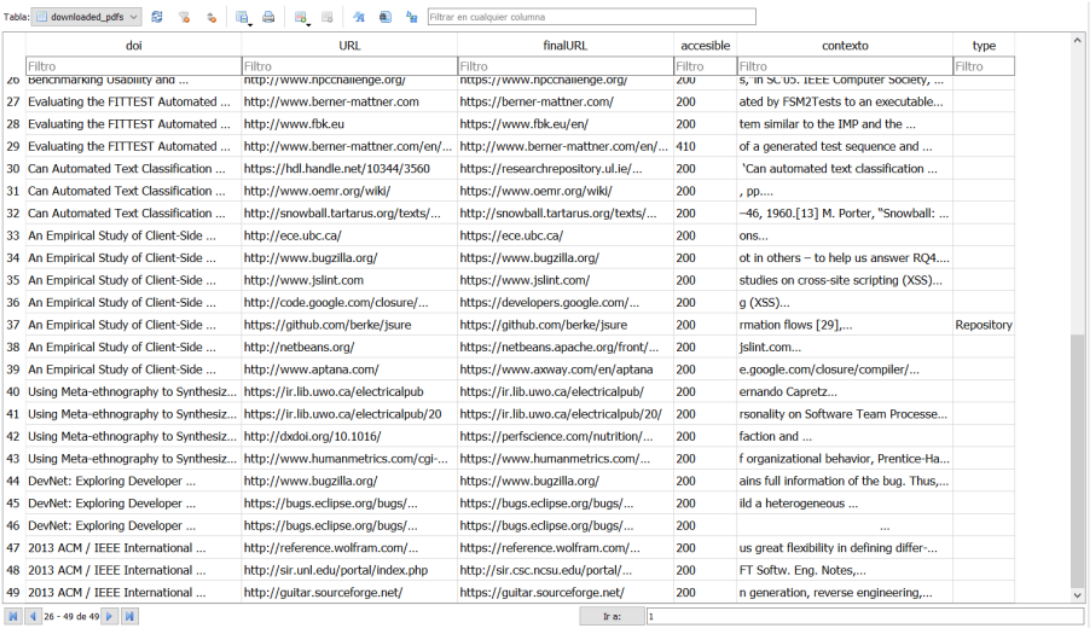
Figura 6.11: Fragmento fichero *error.txt* 2023

6.3. Comparación de los resultados obtenidos

A continuación, se van a comparar los resultados obtenidos en el análisis de la desaparición de la información en internet, realizado con las revistas del año 2013 y 2023.

En cuanto al número de títulos que hay en cada fichero .xml de cada revista tenemos que en el año 2013 contamos con 57 títulos y en 2023 con 38 títulos.

Podemos comprobar que en el año 2013 en la base de datos, concretamente en la tabla *downloaded_pdfs* se indica, que se han encontrado 49 URLs en total:



	doi	URL	finalURL	accesible	contexto	type
	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro
26	benchmarking usability and ...	http://www.npcchallenge.org/	https://www.npcchallenge.org/	200	s, in SC 05. IEEE Computer Society, ...	
27	Evaluating the FITTEST Automated ...	http://www.berner-mattner.com	https://berner-mattner.com/	200	ated by FSM2Tests to an executable...	
28	Evaluating the FITTEST Automated ...	http://www.fbk.eu	https://www.fbk.eu/en/	200	tem similar to the IMP and the ...	
29	Evaluating the FITTEST Automated ...	http://www.berner-mattner.com/en/...	http://www.berner-mattner.com/en/...	410	of a generated test sequence and ...	
30	Can Automated Text Classification ...	https://hdl.handle.net/10344/3560	https://researchrepository.ul.ie/...	200	'Can automated text classification ...	
31	Can Automated Text Classification ...	http://www.oemr.org/wiki/	https://www.oemr.org/wiki/	200	, pp....	
32	Can Automated Text Classification ...	http://snowball.tartarus.org/texts/...	http://snowball.tartarus.org/texts/...	200	–46, 1960.[13] M. Porter, "Snowball: ...	
33	An Empirical Study of Client-Side ...	http://ece.ubc.ca/	https://ece.ubc.ca/	200	ons...	
34	An Empirical Study of Client-Side ...	http://www.bugzilla.org/	https://www.bugzilla.org/	200	ot in others – to help us answer RQ4....	
35	An Empirical Study of Client-Side ...	http://www.jslint.com	https://www.jslint.com/	200	studies on cross-site scripting (XSS)...	
36	An Empirical Study of Client-Side ...	http://code.google.com/closure/...	https://developers.google.com/...	200	g (XSS)...	
37	An Empirical Study of Client-Side ...	https://github.com/berke/jsure	https://github.com/berke/jsure	200	rmation flows [29],...	Repository
38	An Empirical Study of Client-Side ...	http://netbeans.org/	https://netbeans.apache.org/front/...	200	jslint.com...	
39	An Empirical Study of Client-Side ...	http://www.aptana.com/	https://www.axway.com/en/aptana	200	e.google.com/closure/compiler/...	
40	Using Meta-ethnography to Synthesiz...	https://ir.lib.uwo.ca/electricalpub	https://ir.lib.uwo.ca/electricalpub/	200	ernando Capretz...	
41	Using Meta-ethnography to Synthesiz...	https://ir.lib.uwo.ca/electricalpub/20	https://ir.lib.uwo.ca/electricalpub/20/	200	rsonality on Software Team Processe...	
42	Using Meta-ethnography to Synthesiz...	http://dxdoi.org/10.1016/	https://perfscience.com/nutrition/...	200	faction and ...	
43	Using Meta-ethnography to Synthesiz...	http://www.humanmetrics.com/cgi-...	https://www.humanmetrics.com/...	200	f organizational behavior, Prentice-Ha...	
44	DevNet: Exploring Developer ...	http://www.bugzilla.org/	https://www.bugzilla.org/	200	ains full information of the bug. Thus,...	
45	DevNet: Exploring Developer ...	https://bugs.eclipse.org/bugs/...	https://bugs.eclipse.org/bugs/...	200	ild a heterogeneous ...	
46	DevNet: Exploring Developer ...	https://bugs.eclipse.org/bugs/...	https://bugs.eclipse.org/bugs/...	200	...	
47	2013 ACM / IEEE International ...	http://reference.wolfram.com/...	https://reference.wolfram.com/...	200	us great flexibility in defining differ-...	
48	2013 ACM / IEEE International ...	http://sir.unl.edu/portal/index.php	http://sir.csc.ncsu.edu/portal/...	200	FT Softw. Eng. Notes,...	
49	2013 ACM / IEEE International ...	http://guitar.sourceforge.net/	https://guitar.sourceforge.net/	200	n generation, reverse engineering,...	

Figura 6.12: Número de URLs - Caso de estudio 2013

Sin embargo, al filtrar las URLs por código devuelto por el servidor: 200, me encontré con que, en el caso de la revista de 2013, el número de URLs existentes es 44:

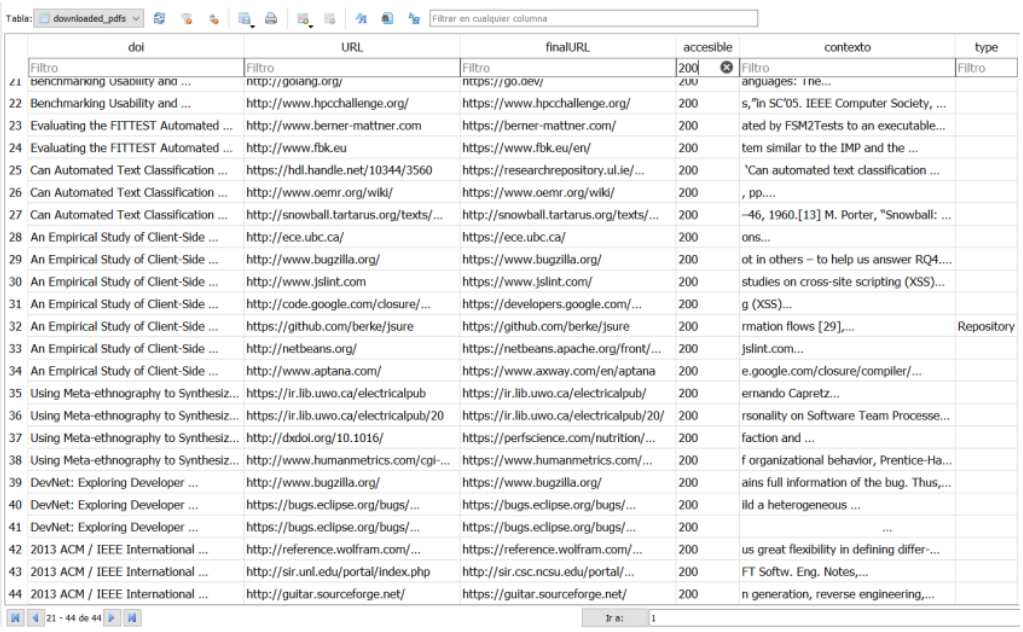


Tabla: downloaded_pdfs — Filtrar en cualquier columna

	doi	URL	finalURL	accesible	contexto	type
	Filtro	Filtro	Filtro	200	Filtro	Filtro
21	Benchmarking Usability and ...	http://goiang.org/	https://go.dev/	200	anguages: The...	
22	Benchmarking Usability and ...	http://www.hpcchallenge.org/	https://www.hpcchallenge.org/	200	s,"in SC'05. IEEE Computer Society, ...	
23	Evaluating the FITTEST Automated ...	http://www.berner-mattner.com	https://berner-mattner.com/	200	ated by FSM2Tests to an executable...	
24	Evaluating the FITTEST Automated ...	http://www.fbk.eu	https://www.fbk.eu/en/	200	tem similar to the IMP and the ...	
25	Can Automated Text Classification ...	https://hdl.handle.net/10344/3560	https://researchrepository.ul.ie/...	200	'Can automated text classification ...	
26	Can Automated Text Classification ...	http://www.oemr.org/wiki/	https://www.oemr.org/wiki/	200	, pp....	
27	Can Automated Text Classification ...	http://snowball.tartarus.org/texts/...	http://snowball.tartarus.org/texts/...	200	–46, 1960.[13] M. Porter, "Snowball: ...	
28	An Empirical Study of Client-Side ...	http://ece.ubc.ca/	https://ece.ubc.ca/	200	ons...	
29	An Empirical Study of Client-Side ...	http://www.bugzilla.org/	https://www.bugzilla.org/	200	ot in others – to help us answer RQ4....	
30	An Empirical Study of Client-Side ...	http://www.jslint.com	https://www.jslint.com/	200	studies on cross-site scripting (XSS)...	
31	An Empirical Study of Client-Side ...	http://code.google.com/closure/...	https://developers.google.com/...	200	g (XSS)...	
32	An Empirical Study of Client-Side ...	https://github.com/berke/jsure	https://github.com/berke/jsure	200	rmation flows [29],...	Repository
33	An Empirical Study of Client-Side ...	http://netbeans.org/	https://netbeans.apache.org/front/...	200	jslint.com...	
34	An Empirical Study of Client-Side ...	http://www.aptana.com/	https://www.axway.com/en/aptana	200	e.google.com/closure/compiler/...	
35	Using Meta-ethnography to Synthesiz...	https://ir.lib.uwo.ca/electricalpub	https://ir.lib.uwo.ca/electricalpub/	200	ernando Capretz...	
36	Using Meta-ethnography to Synthesiz...	https://ir.lib.uwo.ca/electricalpub/20	https://ir.lib.uwo.ca/electricalpub/20/	200	rsonality on Software Team Processe...	
37	Using Meta-ethnography to Synthesiz...	http://dxdoi.org/10.1016/	https://perfscience.com/nutrition/...	200	faction and ...	
38	Using Meta-ethnography to Synthesiz...	http://www.humanmetrics.com/cgi-...	https://www.humanmetrics.com/...	200	f organizational behavior, Prentice-Ha...	
39	DevNet: Exploring Developer ...	http://www.bugzilla.org/	https://www.bugzilla.org/	200	ains full information of the bug. Thus,...	
40	DevNet: Exploring Developer ...	https://bugs.eclipse.org/bugs/...	https://bugs.eclipse.org/bugs/...	200	ild a heterogeneous ...	
41	DevNet: Exploring Developer ...	https://bugs.eclipse.org/bugs/...	https://bugs.eclipse.org/bugs/...	200	...	
42	2013 ACM / IEEE International ...	http://reference.wolfram.com/...	https://reference.wolfram.com/...	200	us great flexibility in defining differ-...	
43	2013 ACM / IEEE International ...	http://sir.unl.edu/portal/index.php	http://sir.csc.ncsu.edu/portal/...	200	FT Softw. Eng. Notes,...	
44	2013 ACM / IEEE International ...	http://guitar.sourceforge.net/	https://guitar.sourceforge.net/	200	n generation, reverse engineering,...	

21 - 44 de 44 — Ir a: 1

Figura 6.13: Número de URLs existentes - Caso de estudio 2013

Por lo tanto, comparando los números obtenidos, se llega a la conclusión de que aproximadamente el 10.20 % de las URLs que se han extraído de los artículos de la revista de 2013 no existen.

Por otro lado, en el año 2023, se puede comprobar, gracias a la base de datos, que se han identificado 220 URLs:

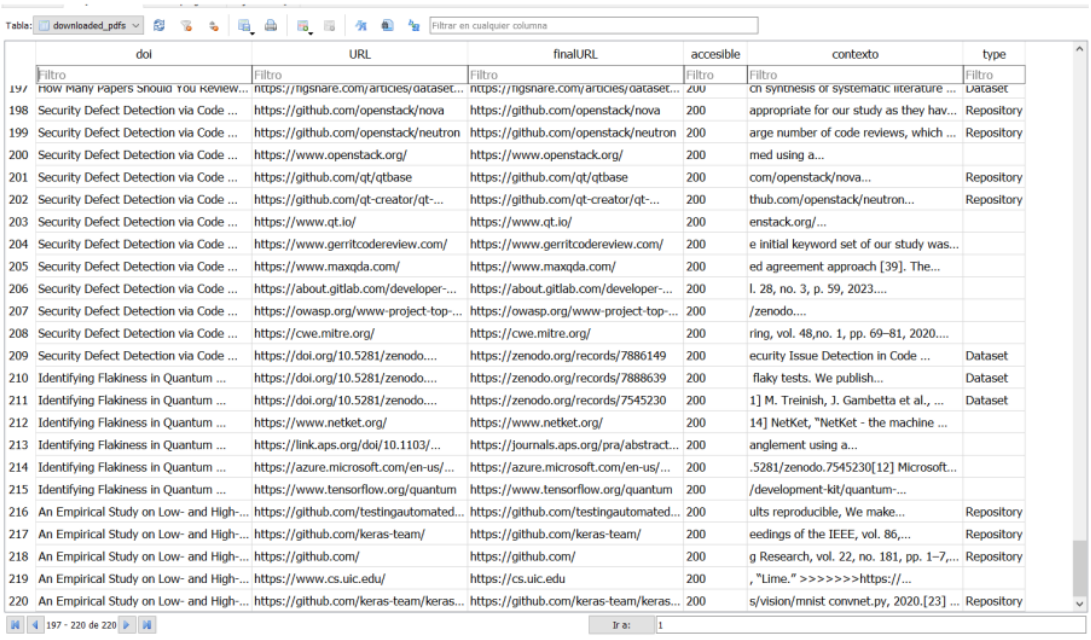


Tabla: downloaded_pdfs — Filtrar en cualquier columna

	doi	URL	finalURL	accesible	contexto	type
	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro
197	How Many Papers Should You Review...	https://rigsnare.com/articles/dataset...	https://rigsnare.com/articles/dataset...	200	ch synthesis or systematic literature ...	Dataset
198	Security Defect Detection via Code ...	https://github.com/openstack/nova	https://github.com/openstack/nova	200	appropriate for our study as they hav...	Repository
199	Security Defect Detection via Code ...	https://github.com/openstack/neutron	https://github.com/openstack/neutron	200	arge number of code reviews, which ...	Repository
200	Security Defect Detection via Code ...	https://www.openstack.org/	https://www.openstack.org/	200	med using a...	
201	Security Defect Detection via Code ...	https://github.com/qt/qtbase	https://github.com/qt/qtbase	200	com/openstack/nova...	Repository
202	Security Defect Detection via Code ...	https://github.com/qt-creator/qt-...	https://github.com/qt-creator/qt-...	200	thub.com/openstack/neutron...	Repository
203	Security Defect Detection via Code ...	https://www.qt.io/	https://www.qt.io/	200	enstack.org/...	
204	Security Defect Detection via Code ...	https://www.gerritcodereview.com/	https://www.gerritcodereview.com/	200	e initial keyword set of our study was...	
205	Security Defect Detection via Code ...	https://www.maxqda.com/	https://www.maxqda.com/	200	ed agreement approach [39]. The...	
206	Security Defect Detection via Code ...	https://about.gitlab.com/developer-...	https://about.gitlab.com/developer-...	200	l. 28, no. 3, p. 59, 2023....	
207	Security Defect Detection via Code ...	https://owasp.org/www-project-top-...	https://owasp.org/www-project-top-...	200	/zenodo....	
208	Security Defect Detection via Code ...	https://cwe.mitre.org/	https://cwe.mitre.org/	200	ring, vol. 48,no. 1, pp. 69–81, 2020....	
209	Security Defect Detection via Code ...	https://doi.org/10.5281/zenodo....	https://zenodo.org/records/7886149	200	ecurity Issue Detection in Code ...	Dataset
210	Identifying Flakiness in Quantum ...	https://doi.org/10.5281/zenodo....	https://zenodo.org/records/7888639	200	flaky tests. We publish...	Dataset
211	Identifying Flakiness in Quantum ...	https://doi.org/10.5281/zenodo....	https://zenodo.org/records/7545230	200	1] M. Treinish, J. Gambetta et al., ...	Dataset
212	Identifying Flakiness in Quantum ...	https://www.netket.org/	https://www.netket.org/	200	14] NetKet, "NetKet - the machine ...	
213	Identifying Flakiness in Quantum ...	https://link.aps.org/doi/10.1103/...	https://journals.aps.org/pra/abstract...	200	anglement using a...	
214	Identifying Flakiness in Quantum ...	https://azure.microsoft.com/en-us/...	https://azure.microsoft.com/en-us/...	200	.5281/zenodo.7545230[12] Microsoft...	
215	Identifying Flakiness in Quantum ...	https://www.tensorflow.org/quantum	https://www.tensorflow.org/quantum	200	/development-kit/quantum-...	
216	An Empirical Study on Low- and High-...	https://github.com/testingautomated...	https://github.com/testingautomated...	200	ults reproducible, We make...	Repository
217	An Empirical Study on Low- and High-...	https://github.com/keras-team/	https://github.com/keras-team/	200	eedings of the IEEE, vol. 86,...	Repository
218	An Empirical Study on Low- and High-...	https://github.com/	https://github.com/	200	g Research, vol. 22, no. 181, pp. 1–7,...	Repository
219	An Empirical Study on Low- and High-...	https://www.cs.uic.edu/	https://cs.uic.edu	200	, "Lime." >>>>>>>https://...	
220	An Empirical Study on Low- and High-...	https://github.com/keras-team/keras...	https://github.com/keras-team/keras...	200	s/vision/mnist convnet.py, 2020.[23] ...	Repository

197 - 220 de 220 — Ir a: 1

Figura 6.14: Número de URLs - Caso de estudio 2023

Sin embargo, al filtrar con el código devuelto por el servidor, con el valor 200, se nos indica que se han verificado correctamente 208 URLs.

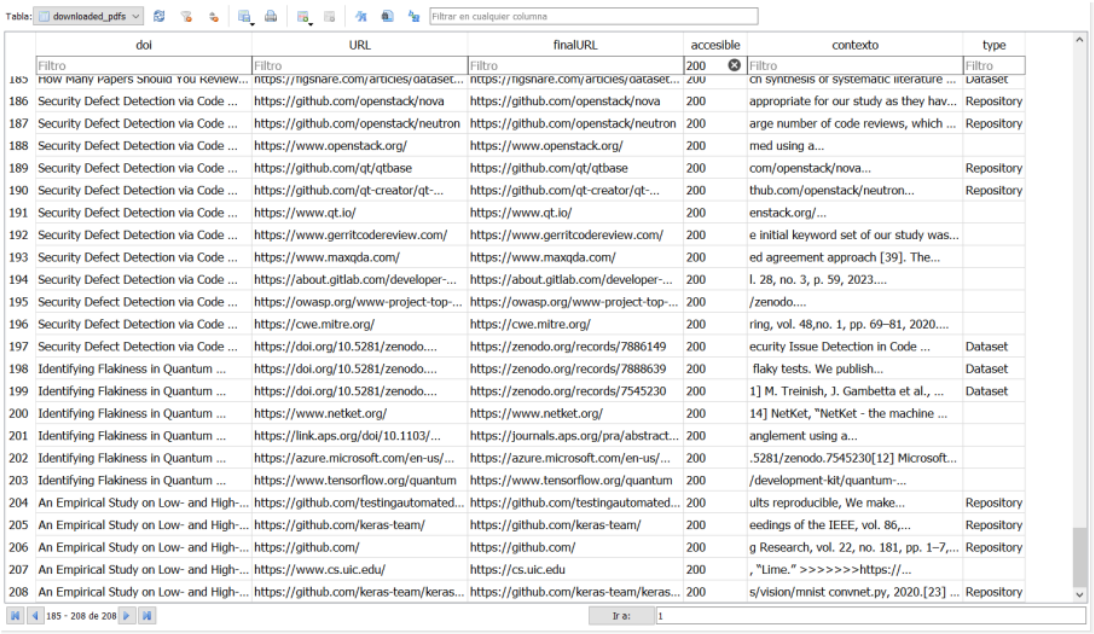


Tabla: downloaded_pdfs — Filtrar en cualquier columna

	doi	URL	finalURL	accesible	contexto	type
	Filtro	Filtro	Filtro	200	Filtro	Filtro
185	How Many Papers Should You Review...	https://figshare.com/articles/dataset...	https://figshare.com/articles/dataset...	200	ch synthesis of systematic literature ...	Dataset
186	Security Defect Detection via Code ...	https://github.com/openstack/nova	https://github.com/openstack/nova	200	appropriate for our study as they hav...	Repository
187	Security Defect Detection via Code ...	https://github.com/openstack/neutron	https://github.com/openstack/neutron	200	arge number of code reviews, which ...	Repository
188	Security Defect Detection via Code ...	https://www.openstack.org/	https://www.openstack.org/	200	med using a...	
189	Security Defect Detection via Code ...	https://github.com/qt/qtbase	https://github.com/qt/qtbase	200	com/openstack/nova...	Repository
190	Security Defect Detection via Code ...	https://github.com/qt-creator/qt-...	https://github.com/qt-creator/qt-...	200	thub.com/openstack/neutron...	Repository
191	Security Defect Detection via Code ...	https://www.qt.io/	https://www.qt.io/	200	enstack.org/...	
192	Security Defect Detection via Code ...	https://www.gerritcodereview.com/	https://www.gerritcodereview.com/	200	e initial keyword set of our study was...	
193	Security Defect Detection via Code ...	https://www.maxqda.com/	https://www.maxqda.com/	200	ed agreement approach [39]. The...	
194	Security Defect Detection via Code ...	https://about.gitlab.com/developer-...	https://about.gitlab.com/developer-...	200	l. 28, no. 3, p. 59, 2023....	
195	Security Defect Detection via Code ...	https://owasp.org/www-project-top-...	https://owasp.org/www-project-top-...	200	/zenodo....	
196	Security Defect Detection via Code ...	https://cwe.mitre.org/	https://cwe.mitre.org/	200	ring, vol. 48,no. 1, pp. 69–81, 2020....	
197	Security Defect Detection via Code ...	https://doi.org/10.5281/zenodo....	https://zenodo.org/records/7886149	200	ecurity Issue Detection in Code ...	Dataset
198	Identifying Flakiness in Quantum ...	https://doi.org/10.5281/zenodo....	https://zenodo.org/records/7888639	200	flaky tests. We publish...	Dataset
199	Identifying Flakiness in Quantum ...	https://doi.org/10.5281/zenodo....	https://zenodo.org/records/7545230	200	1] M. Treinish, J. Gambetta et al., ...	Dataset
200	Identifying Flakiness in Quantum ...	https://www.netket.org/	https://www.netket.org/	200	14] NetKet, "NetKet - the machine ...	
201	Identifying Flakiness in Quantum ...	https://link.aps.org/doi/10.1103/...	https://journals.aps.org/pra/abstract...	200	anglement using a...	
202	Identifying Flakiness in Quantum ...	https://azure.microsoft.com/en-us/...	https://azure.microsoft.com/en-us/...	200	.5281/zenodo.7545230[12] Microsoft...	
203	Identifying Flakiness in Quantum ...	https://www.tensorflow.org/quantum	https://www.tensorflow.org/quantum	200	/development-kit/quantum-...	
204	An Empirical Study on Low- and High-...	https://github.com/testingautomated...	https://github.com/testingautomated...	200	ults reproducible, We make...	Repository
205	An Empirical Study on Low- and High-...	https://github.com/keras-team/	https://github.com/keras-team/	200	eedings of the IEEE, vol. 86,...	Repository
206	An Empirical Study on Low- and High-...	https://github.com/	https://github.com/	200	g Research, vol. 22, no. 181, pp. 1–7,...	Repository
207	An Empirical Study on Low- and High-...	https://www.cs.uic.edu/	https://cs.uic.edu	200	, "Lime." >>>>>>>https://...	
208	An Empirical Study on Low- and High-...	https://github.com/keras-team/keras...	https://github.com/keras-team/keras...	200	s/vision/mnist convnet.py, 2020.[23] ...	Repository

185 - 208 de 208 — Ir a: 1

Figura 6.15: Número de URLs existentes - Caso de estudio 2023

Por lo tanto, comparando los números obtenidos, se llega a la conclusión de que aproximadamente el 5.45 % de las URLs que se han extraído de los artículos de la revista de 2023 no existen.

Una vez que se cuenta con los resultados de las revistas de ambos años, se puede concluir que en 2023 el porcentaje de URLs que ya no están disponibles es bastante inferior al de 2013. Sin embargo, hay que tener en cuenta que 2013 corresponde a la década anterior y todavía no ha pasado ni un año desde 2023. Por ello, considero que, en este caso de estudio, el porcentaje de desaparición de la información en internet es superior en 2023 que en 2013.

Capítulo 7

Análisis de impacto

En este capítulo se trata el impacto de los resultados obtenidos en relación con Objetivos de Desarrollo Sostenible (ODS).

7.1. Educación de calidad (ODS 4)

La aplicación desarrollada ayuda a proporcionar una educación de calidad. Esto se debe a que al poder realizar un análisis de cómo va desapareciendo la información en internet, se pueden tomar medidas para reducir al mínimo dicha desaparición. De esta forma, las bibliografías estarán más completas y se podrá relaizar reproducciones de los trabajo de forma más verídica y sencilla.

7.2. Industria, innovación e infraestructura (ODS 9)

La aplicación promueve una innovación en relación con el almacenamiento de la información, ya que como se han podido ver en los resultados obtenidos, uno de los motivos de desaparición que se han identificado es los sitios web donde se guarda dicha información. De esta forma, se busca elaborar sitios oficiales y verificados para poder almacenar la documentación oficial.

7.3. Paz, justicia e instituciones sólidas (ODS 16)

La aplicación desarrollada contribuye a la transparencia de la información y al poder acceder a ella. Esto ayuda sobre todo a las instituciones, ya que al tener acceso a la información fortalece la capacidad de desarrollo, investigación y toma de decisiones de ellas.

Capítulo 8

Conclusiones y trabajo futuro

En este trabajo de fin de curso se ha investigado cómo desaparece la información que se encuentra almacenada en internet. Para ello, se han analizado artículos de revistas obtenidas de la biblioteca digital DBLP de distintos años. El resultado de este ánalisis abre el horizonte de que cuanto más tiempo transcurre desde la publicación de un artículo, la información citada en él es más propensa a desaparecer.

Gracias a los resultados obtenidos podemos comentar que, se debe de tener en cuenta que, la información que deja de estar disponible se relaciona con encontrarse almacenada en sitios específicos en la red, como repositorios o servicios de cloud. Además, esto nos indica que uno de los motivos de la desaparición, es el hecho de que se almacena la información en servicios personales, en vez de estar disponibles en servicios públicos para que pueda ayudar a mantenerse a través del tiempo con mayor facilidad. También, otro motivo es la eliminación o actualización de dominios, además de redirecciones, que pueden afectar a la contrucción de las URLs.

Sin embargo, pese a los resultados que se han obtenido la aplicación cuenta con una serie de limitaciones que pueden variar el análisis realizado. Una de esas limitaciones es el problema de los denominados *linkrot* que implican la disociación entre las direcciones web y su contenido, que son difíciles de identificar. Otra de las limitaciones importantes es que las direcciones web identifiquen que es una aplicación la que está intentando acceder a ella, y por lo tanto la clasifica como un *bot*, restringindo el acceso. También, existe el problema de identificar de forma correcta el motivo de la desaparición de dicha información.

Para trabajo futuro se debería mejorar el proceso de camuflar la aplicación para que no sea detectada y se pueda reducir al minímo el problema de resultado falso de verificaciones de URLs por acceso denegado. Además, sería muy beneficioso afinar aún más los distintos dominios utilizados para la clasificación de los enlaces en distintas categorías, para realizarlo con más precisión y veracidad.

La realización de este trabajo de fin de curso me ha permitido adquirir conocimientos sobre la manipulación de archivos en formato PDF ya que estos cuentan

con estructuras específicas. Además, he podido enfrentarme a el manejo de enlaces, como por ejemplo las técnicas para extraerlos sin producir una rotura de ellos. También, he podido darme cuenta del problema que supone que la información desaparezca, además de el propio hecho de que va desapareciendo sin que nos demos cuenta.

En conclusión, este proyecto ha ayudado a realizar una investigación y un análisis de cómo desaparece la información en internet y el periodo aproximado de esa desaparicón. Debido a esto espero que se pueda realizar una concienciación de la importancia de mantener la información disponible, y que se tomen medidas para ello.

Bibliografía

[1] Apache Software Foundation. (2024) Apache Maven. [Online]. Available: https://maven.apache.org/

[2] Genbeta. (2011) Introducción a Maven. [Online]. Available: https://www.genbeta.com/desarrollo/introduccion-a-maven

[3] Universidad de La Coruña. (2023) dblp: una base de datos fantástica, gratuita y especializada exclusivamente en computación. [Online]. Available: https://www.udc.es/es/biblioteca.fic/recursos_informacion/bases_de_datos-00001/dblp/

[4] Universidad de Las Palmas de Gran Canaria. (2020) Google Académico: acceso al texto completo ULPGC. [Online]. Available: https://biblioteca.ulpgc.es/google_academico#:~:text=Google%20Acad%C3%A9mico%20es%20un%20buscador,universidades%20y%20otras%20organizaciones%20acad%C3%A9micas.

[5] Universidad Autónoma de Madrid. (2023) Google Académico, paso a paso. [Online]. Available: https://biblioteca.ulpgc.es/google_academico#:~:text=Google%20Acad%C3%A9mico%20es%20un%20buscador,universidades%20y%20otras%20organizaciones%20acad%C3%A9micas.

[6] Daniel Albarracín Morales. (2023) Introducción SerpApi. [Online]. Available: https://www.kaggle.com/code/danielalbarracinm/introducci-n-serpapi

[7] Apache Software Foundation. (2024) Apache PDFBox. [Online]. Available: https://pdfbox.apache.org./

[8] Jsoup. (2023) Jsoup Java HTML Parser. [Online]. Available: https://jsoup.org/

[9] IBM. (2022) Formato JSON (JavaScript Object Notation). [Online]. Available: https://www.ibm.com/docs/es/baw/20.x?topic=formats-javascript-object-notation-json-format

[10] IBM. (2022) Propiedades y conversiones de tipos de datos del formato JSON. [Online]. Available: https://www.ibm.com/docs/es/baw/20.x?topic=format-json-properties-data-type-conversions

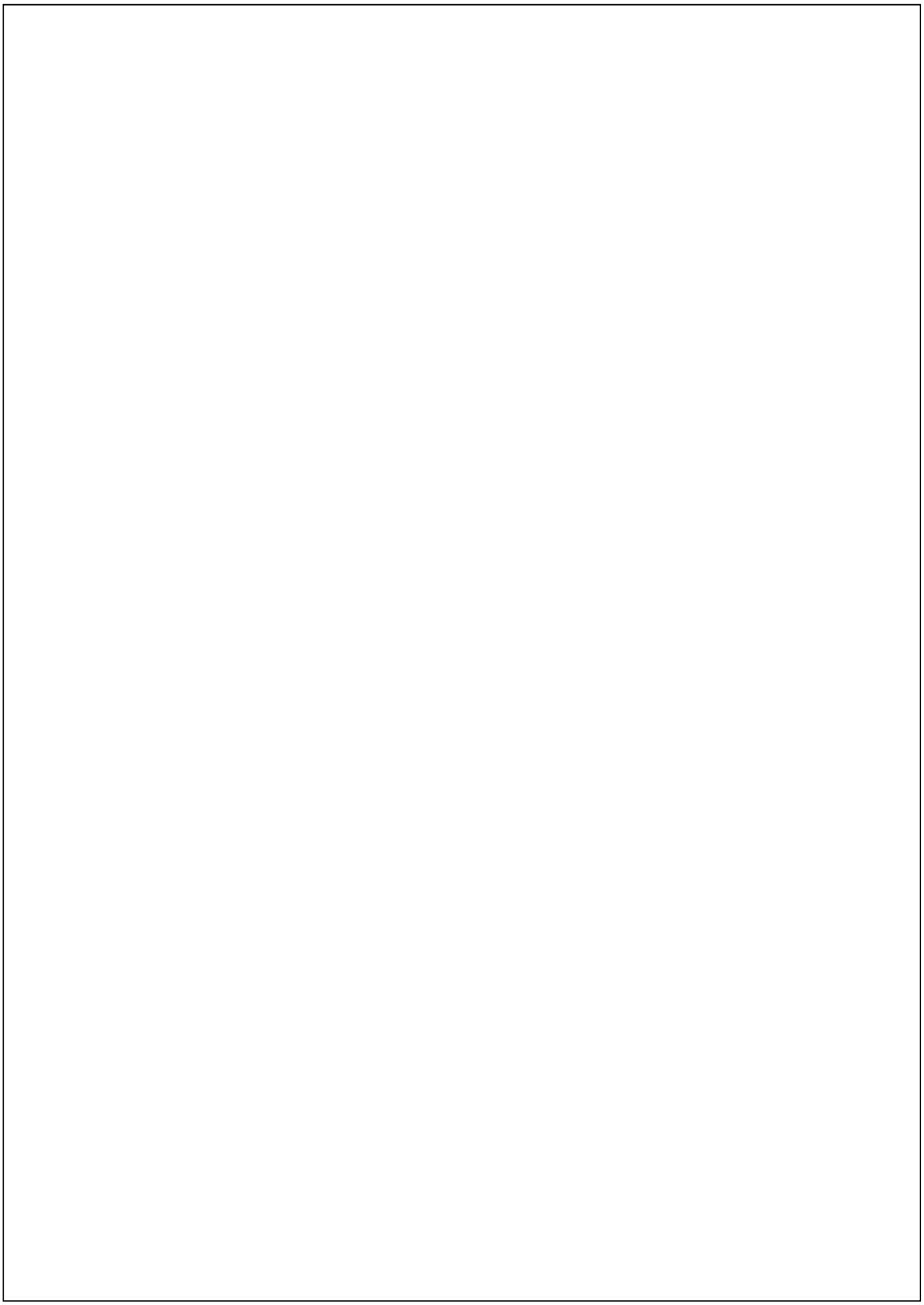
[11] J. Zittrain, K. Albert, and L. Lessig, "Perma: Scoping and addressing the problem of link and reference rot in legal citations," *Legal Information Management*, vol. 14, no. 2, pp. 88–99, 2014.

[12] D. G. Gomes, P. Pottier, R. Crystal-Ornelas, E. J. Hudgins, V. Foroughirad, L. L. Sánchez-Reyes, R. Turba, P. A. Martinez, D. Moreau, M. G. Bertram *et al.*, "Why don't we share data and code? perceived barriers and benefits to public archiving practices," *Proceedings of the Royal Society B*, vol. 289, no. 1987, p. 20221113, 2022.

[13] D. E. Ott, "Reference hygiene and death on the internet–decay, rot, half-life, deterioration, and corruption," *JSLS: Journal of the Society of Laparoscopic & Robotic Surgeons*, vol. 26, no. 1, 2022.

[14] J. Hennessey and S. X. Ge, "A cross disciplinary study of link decay and the effectiveness of mitigation techniques," in *BMC bioinformatics*, vol. 14. Springer, 2013, pp. 1–11.

Anexos



Apéndice A

Especificación de Requisitos

A.1. Introducción

Este documento detalla la Especificación de Requisitos Software (ERS) de una aplicación que extrae y analiza las URLS de pdfs descargados con el fin de conocer si existen o no.

A.1.1. Propósito del proyecto

La realización de esta aplicación tiene el propósito de comprobar si los enlaces utilizados en los artículos, para apoyar o justificar sus conclusiones, siguen existiendo.

Está relacionado con el llamado "Fraude científico". El objetivo principal es estudiar la cantidad de información científica que se pierde en internet, identificando los mecanismos y el tiempo en el que se producen, de forma automatizada.

Objetivos:

- Poder acceder a los PDFs de los artículos que se encuentran en DBLP.
- En los PDFs ser capaz de poder localizar los enlaces a las referencias utilizadas.
- Acceder a esos enlaces para comprobar si existen, y si no existen estimar su motivo de desaparición.

A.1.2. Definiciones, Acrónimos y Abreviaturas

API:	Se trata de: Interfaz de Programación de Aplicaciones.
DOI:	Digital Object Identifier. Se trata de un identificador único y permanente que contienen las publicaciones que se realizan en internet.

A.2. Requisitos Específicos Ciclo 1

En esta sección se enumeran los requisitos que el sistema tendrá que cumplir, tanto funcionales como no funcionales.

A.2.1. Requisitos Funcionales

Línea de comandos

- Req1. La aplicación se deberá lanzar desde la línea de comandos.
- Req2. La aplicación deberá aceptar como entrada un fichero xml.
- Req3. La aplicación deberá un código de éxito(0) si se procesa de forma correcta el fichero xml o un código de fallo(1) en caso contrario.

Obtener los títulos de los artículos desde un fichero xml de DBLP

- Req4. La aplicación deberá extraer los títulos de los artículos presentes en los ficheros xml de entrada de la base de datos DBLP.

Obtener URL de descarga del PDF con SerpApi de Google Scholar

- Req5. La aplicación deberá utilizar SerpApi de Google Scholar para buscar los títulos de los artículos y obtener sus JSON.
- Req6. La aplicación deberá obtener la URL de descarga de los PDF del JSON obtenido.

Descargar PDF desde URL

- Req7. La aplicación deberá descargar el PDF desde la URL obtenida a través de SerpApi.

Analizar el contenido del PDF y comprobar la existencia de las URLs de referencias

- Req8. La aplicación deberá leer el contenido del PDF descargado y extraer las URLs de referencia.
- Req9. La aplicación deberá verificar la existenci de las URLs extraídas sin considerar su contenido.
- Req10. La aplicación deberá asociar los resultados de la verificación con el PDF al que corresponden las URLs.

Caché local para evitar descargas duplicadas de PDFs

- Req11. La aplicación deberá mantener una caché local para evitar las descargas duplicadas de los PDFs.
- Req12. La aplicación deberá identificar a los PDFs por su título.

Almacenar los datos extraídos en una base de datos local

- Req13. La aplicación deberá guardar los datos extraídos en una base de datos local.
- Req14. La aplicación deberá clasificar esos datos por el DOI del pdf, la URL verificada y el resultado de la verificación de la URL.

A.2.2. Requisitos no funcionales

Bloqueo de servidores debido a bots

- Req15. La aplicación deberá implementar una estrategia para evitar que los servidores la consideren un bot.

Fiabilidad en la detección y manejo de errores

- Req16. La aplicación deberá demostrar alta fiabilidad a la hora de detectar y gestionar errores.

Idempotencia del proceso

- Req17. El proceso deberá ser idempotente, lo que quiere decir, que la información asociada a un PDF se sobreescriba si hay alguna modificación.

Manejo de captchas de SerpApi y Google Scholar

- Req18. La aplicación deberá ser capaz de manejar los captchas de forma eficaz generados por SerpApi y Google Scholar. garantizando que no se produzcan bloqueos en el flujo de trabajo automatizado.

Arquitectura JAVA

- Req19. La aplicación contará con una arquitectura JAVA, lo que implica que el diseño y el desarrollo deben estar dentro de los estándares de esta tecnología.

A.3. Requisitos Específicos Ciclo 2

En esta sección se especifican los requisitos que se añaden a la aplicación.

A.3.1. Requisitos Funcionales

Se ha añadido el siguiente requisito:

Exportación de datos

- Req20. La aplicación deberá poder exportar en formato CSV los datos que se encuentran en la base de datos.

A.3.2. Requisitos no funcionales

En cuanto a los requisitos no funcionales se ha añadido el siguiente requisito:

Análisis de datos

- Req21. La aplicación deberá contar con un análisis de los datos exportados en formato CSV.

A.4. Requisitos Específicos Ciclo 3

En esta sección se comentan los requisitos que se han añadido a la aplicación.

A.4.1. Requisitos Funcionales

Descarga del PDF

- Req22. La aplicación deberá mostrar el número de títulos y el número de enlaces encontrados.
- Req23. La aplicación deberá permitir forzar la descarga de un artículo aunque ya se haya descargado previamente.
- Req24. La aplicación deberá bloquear la búsqueda del enlace de descarga de un artículo en formato PDF si ya se ha realizado esa búsqueda con anterioridad.

Contenido del PDF y análisis de las URLs extraídas

- Req25. La aplicación deberá permitir transformar un PDF a texto.
- Req26. La aplicación deberá permitir analizar nuevamente los enlaces encontrados.
- Req27. La aplicación deberá permitir comprobar la accesibilidad de los enlaces directamente en la base de datos.
- Req28. La aplicación deberá permitir eliminar enlaces obsoletos de la base de datos.
- Req29. La aplicación deberá permitir clasificar o categorizar las URLs extraídas.

A.4.2. Requisitos no funcionales

Ejecución

- Req30. La aplicación deberá permitir su ejecución de dos formas. Una de ellas es, a través de la línea de comandos, ejecutando cada clase de forma individual. La otra es, ejecutando una clase que se encarga de activar el flujo de ejecución.

Apéndice B

Diseño del programa

B.1. Ciclo 1

Este ciclo cuenta con dos versiones del programa: la versión con y sin base de datos.

B.1.1. Versión sin base de datos

En esta versión el programa cuenta con las clases:

- LeerXml.java
- ObtenerPdfConSerpApi.java
- DescargarPdfCache.java
- MostrarYComprobarEnlaces.java

A continuación se explicará el propósito de cada clase así como los métodos que incluyen.

LeerXml.java

Esta clase se encarga de extraer el nombre de los títulos de los artículos que se encuentran en el fichero .xml obtenido de DBLP, el cuál, se debe indicar al ejecutarla proprocionando su URL.

Métodos:

- obtenerTitulos: establece una conexión con la URL del fichero .xml de DBLP a través de JSOUP con Jsoup.connect(url). Después, guarda el contenido de la URL en un documento con .get(), sobre el cuál se itera para poder obtener los títulos de los artículos utilizando un forEach. Esto se logra extrayendo el contenido de las etiquetas <title>, suprimiendo el punto final de los títulos y guardándolos en un ArrayList llamado "titulos".

ObtenerPdfConSerpApi.java

Esta clase se encarga de obtener, a partir de los títulos obtenidos en **LeerXml.java**, las URL de descarga de los artículos en PDF, con un formato de salida:
titulo artículo y *URL descarga*.

Métodos:

- obtenerEnlacePDF: se establece una conexión de tipo *OkHttpClient*. Se codifican los títulos en un formato adecuado para poder ser utilizado en la URL de búsqueda de SerpApi de Google Scholar. Si se encuentra una URL para el título del artículo proporcionado, se imprime solo esa URl, y si no, se imprime todo el JSON obtenido gracias a SerpApi, para poder buscar la URL de forma manual.
- obtenerPrimerLink: se encarga de obtener el primer enlace de descarga del artículo, en formato PDF, que aparece en la sección *resources* del JSON de dicho artículo obtenido gracias a SerApi. Este método se utiliza en *obtenerEnlacePDF*.

DescargarPdfCache.java

Esta clase se encarga de descargar, en formato PDF, el artículo gracias al título y a la URL de descarga obtenida de **ObtenerPdfConSerpApi.java**, sólo si no ha sido descargado previamente.

Métodos:

- descargarPDF: lee el contenido presente en la URL utilizando *BufferedInputStream* y después lo escribe en un fichero de destino gracias a *FileOutputStream*. Con esto se logra descargar el PDF desde una URL.
- archivoExiste: este método se encarga de mirar si un archivo ya existe en el destino para evitar descargar el PDF más de una vez. Con *Path* tenemos el destino y comprobamos si ya existe con *Files.exists(path)*.
- limpiarNombreArticulo: se encarga de cambiar los símbolos que pueden causar problemas en los títulos de los artículos por una barra baja.

MostrarYComprobarEnlaces.java

Esta clase se encarga de buscar en el artículo en formato PDF las URL que aparecen como referencias. Una vez que las encuentra intenta entrar en ellas, indicando el código de respuesta del servidor, por lo que nos permite saber si las URLs existen o no.

Métodos:

- en el main se utiliza *PDFTextStripper* para poder buscar en el PDF las URLs que aparecen como referencias.
- verificarExistencia: con JSOUP se establece una conexión con las URLs encontradas, y con *.statusCode()* se obtiene el mensaje devuelto por el servidor.

B.1.2. Versión con base de datos

En esta versión, el programa cuenta con una base de datos para almacenar los resultados obtenidos.

Las clases **LeerXml.java** y **ObtenerPdfConSerpApi.java** se mantienen exactamente igual que en la versión sin base de datos.

En cuanto a la clase **DescargarPdfCache.java** en el *main* se inicializa la base de datos, si no ha sido creada previamente. Se comprueba que no se encuentre el PDF a descargar en la base de datos gracias a *.isNombrePDFExistente(tituloArticulo)*, y si no se encuentra se procede a descargar el PDF y se añade el nombre a la tabla *nombre_pdfs*.

En relación a la clase **MostrarYComprobarEnlaces.java** se añade un método:

- guardarVerificacionEnBaseDeDatos: como su nombre indica, se encarga de guardar las verificaciones de las URLs encontradas en las referencias de los artículos, en la base de datos. Utiliza un objeto *Pattern* que representa el patrón de símbolos de las URLs. Además, cuenta con un objeto *Matcher* que se encarga de encontrar las coincidencias del patrón en el texto. Después, llama al método *verificarExistencia* para obtener el código de respuesta del servidor, y con *.insertPDF(doi, enlace, codigoRespuesta)*, el cuál se verá a continuación, se introducen los datos en la base de datos.

Además, se ha añadido una clase que se encarga de manejar la base de datos llamada **ConexionBaseDeDatos.java**.

ConexionBaseDeDatos.java

Métodos:

- Al inicio se establece que en la tabla *nombre_pdfs* solo se puede introducir una vez el nombre de un artículo. Sin embargo, en la tabla *downloaded_pdfs* se pueden introducir más de una vez, ya que, un PDF puede tener más de una URL como referencia.
- initializeDatabase: se encarga de inicializar las dos tablas de la base de datos : *nombre_pdfs* y *downloaded_pdfs*.
- insertPDF: se encarga de establecer el formato de la tabla *downloaded_pdfs*, utilizando *PreparedStatement* y *Connection*.

- insertNombrePDF: se encarga de establecer el formato de la tabla *nombre_pdfs*, utilizando *PreparedStatement* y *Connection*.
- isNombrePDFExistente: se encarga de consultar a la base de datos si el nombre del PDF ya existe previamente en la tabla *nombre_pdfs*.
- isPDFDownloaded: se encarga de consultar a la base de datos si el nombre del PDF ya existe previamente en la tabla *downloaded_pdfs*.

B.2. Ciclo 2

En este ciclo se han añadido una clase **RunEntireFlow.java** y un script de python, para introducir mejoras en el programa.

En cuanto a la clase **RunEntireFlow.java** se encarga de llamar a las distintas clases que conforman al programa con el objetivo de que se ejecuten de seguido, sin necesidad de ejecutar cada clase de forma individual. Para ello, fue necesario la modificación de las clases para poder enlazar la entrada de una clase con la salida de la anterior.

Flujo de datos entre clases:

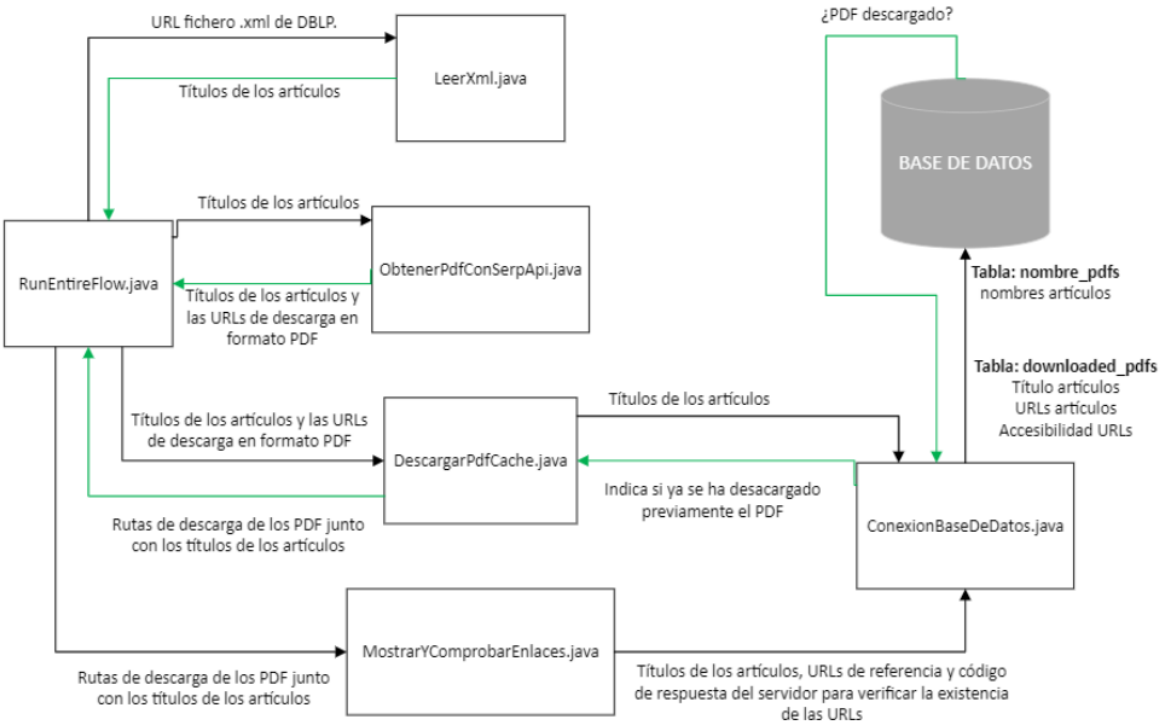


Figura B.1: Diagrama conexión de clases ciclo 2.

Para hacer posible el intercambio de datos entre las clases tal y como se observa en la imagen anterior, ha sido necesario modificar las entradas y salidas de las clases de la manera que se muestra a continuación.

LeerXml.java

La entrada continúa siendo la URL que contiene el fichero .xml de DBLP obtenida desde la línea de comandos. Sin embargo, la salida, en vez de mostrar los títulos de los artículos por pantalla directamente, los va guardando en un array llamado *títulos*.

ObtenerPdfConSerpApi.java

La entrada de esta clase se ha modificado para que coja los nombres de los títulos del array devuelto por **LeerXml.java** en vez de obtenerlos de la línea de comandos. En cuanto a la salida, también se ha modificado, devolviéndose ahora una matriz bidimensinal denominada *titulosYEnlaces* cuyo formato es: *String[título][enlace de descarga PDF del artículo].*

DescargarPdfCache.java

La entrada de esta clase se ha modificado para que utilice los títulos de los artículos, junto con su URL de descarga en formato PDF, obtenidos de **ObtenerPdfConSerpApi.java**. Su salida también ha sido modificada, ya que ahora devuelve una matriz bidimensional denominada *rutasYTitulos* cuyo formato es: *String[rutas de destino para la descarga de los artículos][títulos de los artículos].*

MostrarYComprobarEnlaces.java

La entrada de esta clase se ha modificado para que utilice las rutas de destino y los títulos de los artículos obtenidos de **DescargarPdfCache.java**. Su salida no ha sido modificada, sigue introduciendo en la tabla *downloaded_pdfs*:

- Título del artículo
- URL de referencia
- Código de respuesta del servidor que nos ayuda a identificar si la URL existe o no.

Además, en este ciclo se cuenta con un script de Python, denominado *ExportarCSV*. Este script exporta los datos recogidos en la base de datos para poder realizar un análisis y unas estadísticas de los datos obtenidos.

B.3. Ciclo 3

Representa el último ciclo de desarrollo de la aplicación. En él se modificaron las clases existentes, para poder ejecutar la aplicación tanto con la clase **RunEntireFlow.java** como ejecutar de forma individual las distintas clases que la conforman. Además, se añadió una clase adicional llamada **ClasificarURL.java**.

A continuación, se explican las modificaciones realizadas a las clases existentes, así como se detallará la clase nueva.

LeerXml.java

Se añadió como mejora la posibilidad de mostrar por pantalla el número de títulos que se encuentran en el argumento con el que se ejecuta. Para ello, se utiliza una variable que funciona como contador.

ObtenerPdfConSerpApi.java

Se añadió como mejora una variable contador, para poder mostrar por pantalla el número de enlaces de descarga de un artículo en formato PDF, que se han obtenido gracias a SerpApi de Google Scholar.

Además, se añadió también otra variable de tipo boolean que se encarga de no buscar el enlace de descarga del artículo si ya se ha buscado con anterioridad. Esta mejora es debida a que las búsquedas que se pueden realizar con SerpApi son limitadas por mes, por lo que una vez que se han buscado no es necesario utilizar otra búsqueda para el mismo artículo.

DescargarPdfCache.java

En este caso procedo a detallar los métodos de la clase, ya que han habido mejoras significativas.

- El método *main* en este caso vuelve a ser de la forma *public static void main(String[] args) throws ClassNotFoundException* para poder ejecutar la clase de forma individual.
- Se introduce la casuística de *forzar descarga* la cual, permite volver a descargar los artículos en la caché aunque ya se hubieran descargado previamente.
- Método *chequearDirectorioCache* se encarga de cmprobar si existe el directorio caché, y si no existe lo crea.
- Método *comprobarCacheyDescargarPDF* se encarga de comprobar en la base de datos y en la caché el artículo que se quiere descargar. Además, llama al método *descargarPDF* si no se ha descargado previamente el artículo, o si se indica que se fuerce la descarga.

MostrarYComprobarEnlaces.java

Debido a las mejoras introducidas se expone la estructura de la clase a continuación.

- El método *main* en este caso vuelve también a ser de la forma *public static void main(String[] args) throws SQLException* para poder ejecutar la clase de forma individual.

- Se introducen las casuísticas de *regenerate-text*, *force-link-search* y *use-db-only*.
  - *regenerate-text*: Se vuelve transformar el PDF a texto, se buscan de nuevo los enlaces y se comprueba su accesibilidad.
  - *force-link-search*: Se analizan de nuevo los enlaces de los ficheros, usando el texto existente. Se comprueba su accesibilidad.
  - *use-db-only*: Se comprueba la accesibilidad de los enlaces que están en la base de datos sin utilizar el texto del artículo.
- Método *extraerTodosLosEnlaces* se encarga de extraer todas las URLs que se utilizan como referencias en los artículos.
- Método *extraeEnlacesDeArticulo* se encarga de manipular los enlaces obtenidos. Primero se comprueba si la version .txt del documento ya existe en la cache. A continuación, se eliminan los enlaces existentes en la base de datos. Si no hay un fichero de texto asociado, el PDF corresponde con un nuevo fichero, entonces se obtiene el documento y se transforma a texto. Luego, se marcan todos los "http", ya que, esto permite realizar manipulaciones del texto, como añadir o eliminar, conociendo qué URLs se han procesado y cuáles están pendientes. También, se aplican varias estrategias, en forma de expresiones regulares, para poder maximizar los enlaces correctos. De esta forma, se mejoran los problemas con los matches ya que estos proporcionan muchos falsos negativos por OCR incorrecto o por el formato de los artículos.
- Método *getURLsFromPaperText* se obtienen y se procesan los enlaces del artículo. Para ello, se eliminan los espacios de las URLs para poder procesarlas de forma correcta.
- Método *unmarkURLS* elimina los markers de una cadena de texto solo en posiciones especificadas.
- Método *extractURLContext* extrae el contexto de las URLs obtenidas del artículo.
- Método *generateNewPaperTextWithMarkedHTTP* utiliza un marker para marcar todas apariciones de *http* para poder identificar de forma más eficiente las URLs.
- Para verificar la existencia se ha añadido un agente.
- Método *obtenerEnlaceFinal* se encarga de devolver la URL después de todas las redirecciones que puede tener.

ConexionBaseDeDatos.java

En este caso se han añadido las columnas, a la tabla *downloaded_pdfs*, *finalURL*, *contexto* y *type*. Además, se han añadido los siguientes métodos:

- Método *EliminarEnlacesDeUnArticulo* se encarga de eliminar los enlaces de un artículo que se encuentran almacenados en la tabla *downloaded_pdfs* de la base de datos.
- Método *getAllURLs* extrae las URLs de los artículos que se han identificado y las muestra en una lista.
- Método *saveURLType* guarda en la base de datos, en la tabla *downloaded_pdfs*, la clasificación de las URLs.
- Método *BDExist* indica si la base de datos existe.
- Método *updateResponseCode* indica el código de respuesta que devuelve el servidor al intentar conectar con cada URL para verificar su existencia.

RunEntireFlow.java

Se añadió al flujo de ejecución de la aplicación la nueva clase **ClasificarURL.java**.

ClasificarURL.java

Gracias a un fichero denominado *sites.properties* se clasificam las URLs identificadas.

Contiene un método denominado *getURLType* que categoriza las URLs.

Ejemplo del fichero *sites.properties*:

```
Article = doi\.org/10\.48550,arxiv\.org,academia\.edu,researchgate\.net,sciencedirect\.com,ieeexplore\.ieee\.org,dl\.acm\
Book = taylorfrancis\.com/chapters,aspbooks\.org/publications,books\.google,library\.oapen\.org,academic\
Document = eric\.ed\.gov/\?id=,nvlpubs\.nist\.gov\/nistpubs\/ir,
Regulation = eur-lex\.europa\.eu/legal-content,normlex\.ilo\.org\/dyn\/normlex,uscode\.house\.gov\/view\
Standard = iso\.org/standard,etsi\.org\/deliver\/etsi_ts,w3\.org\/TR\/,astm\.org\/[a-zA-Z]][0-9]{4},webstore\
Dataset = zenodo\.org/records,dataverse\.harvard\.edu/dataset\.xhtml,yareta\.unige\.ch/archives,datadryad\
Cloud = cloud\.box\.com,dropbox\.com/s/,drive\.google\.com/file/d,drive\.google\.com/drive/folders,dropbox\
Repository = \/\/gitlab\.com,\/\/github\.com,sourceforge\.net/projects,code\.google\.com,bitbucket\.com,git\
Social = facebook\.com,instagram\.com,linkedin\.com,pinterest\.com,snapchat\.com,twitter\.com,youtube\.com,reddit\.com,tiktok\
Messaging = whatsapp\.com,
API = \/\/api\.,
```

Figura B.2: Ejemplo fichero *sites.properties*.

Apéndice C

Pruebas

C.1. Ciclo 1

Para este primer ciclo se realizaron una serie de pruebas en base a la versión con base de datos.

C.1.1. Versión con base de datos

Lo primero que hay que tener en cuenta es que es necesario tener instalado en el ordenador donde se va a ejecutar el programa *Maven*. Esto es necesario para que se pueda leer el archivo *pom.xml* y funcionen las dependencias. Además, se miró previamente las URLs presentes en los artículos para verifiar su existencia y así, poder comprobar si el programa realizaba satisfactoriamente su función.

A continuación, se ejecutó la clase **LeerXml.java** con el argumento: `https://dblp.org/search/publ/api?q=toc%3Adb/journals/tse/tse37.bht%3A&h=1000&format=xml`. Como resultado se obtuvieron los títulos de los artículos que se encuentran en la url proporcionada como argumento. Se decidió hacer las pruebas con estos artículos:

1. Frameworks Generate Domain-Specific Languages: A Case Study in the Multimedia Domain.
2. FlowTalk: Language Support for Long-Latency Operations in Embedded Devices.
3. Dynamic QoS Management and Optimization in Service-Based Systems.

Prueba 1

Se ejecutó la clase **ObtenerPdfConSerpApi.java** con el título de un artículo como argumento: *Frameworks Generate Domain-Specific Languages: A Case Study in the Multimedia Domain.*. Como resultado se mostró el título del artículo y una URL de descarga de este en formato PDF:

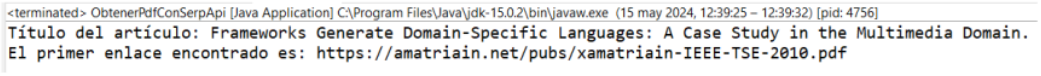


Figura C.1: Resultado obtenido por ObtenerPdfConSerpApi.java - Artículo 1

Posteriormente, se ejecutó la clase **DescargarPdfCache.java** con el título del artículo y su URL de descarga. Como resultado, se descargó el artículo en formato PDF de forma local. Asímismo, se añadió el nombre del artículo descargado a la base de datos, en la tabla *nombre_pdfs*:

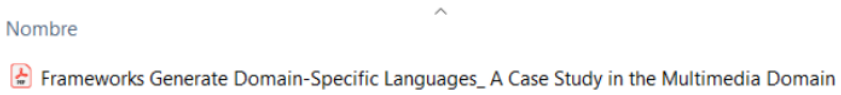


Figura C.2: PDF descargado - Artículo 1

Para poder visualizar la base de datos, se utilizó la aplicación *DB Browser for SQLite*:

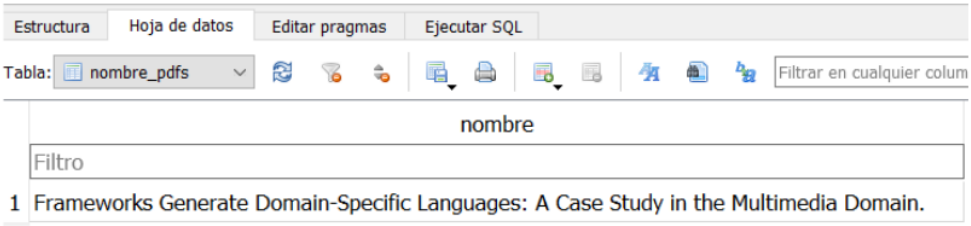


Figura C.3: Tabla nombre_pdfs - Artículo 1

A continuación, se ejecutó la clase **MostrarYComprobarEnlaces.java**, utilizando la ruta de descarga del artículo y su título. Como resultado se introdujo en la base de datos, en la tabla *downloaded_pdfs* el título del artículo, las URLs de referencia y el código devuelto por el servidor al intentar acceder a la URL:

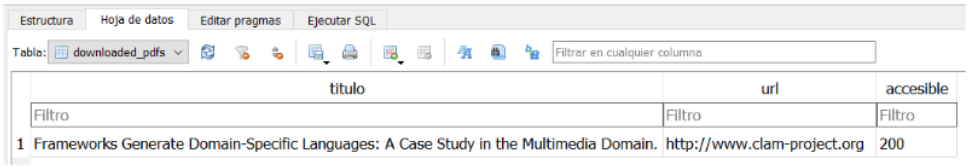


Figura C.4: Tabla downloaded_pdfs - Artículo 1

Una vez realizado el proceso completo, se intentó volver a ejecutar la clase **DescargarPdfCache.java** con el mismo artículo para comprobar que no se descarga, ya que, ya ha sido descargado previamente:

```
El PDF ya ha sido descargado previamente.
```

Figura C.5: Mensaje tras intentar volver a descargar un PDF - Artículo 1

Una vez se terminó de realizar la prueba 1 con el primer artículo, se procedió a realizar la prueba 2.

Prueba 2

Se ejecutó la clase **ObtenerPdfConSerpApi.java** con el título de un artículo como argumento: *FlowTalk: Language Support for Long-Latency Operations in Embedded Devices*. Como resultado se mostró el título del artículo y una URL de descarga de este en formato PDF:

```
<terminated> ObtenerPdfConSerpApi [Java Application] C:\Program Files\Java\jdk-15.0.2\bin\javaw.exe  (15 may 2024, 13:16:49 – 13:17:00) [pid: 14820]
Título del artículo: FlowTalk: Language Support for Long-Latency Operations in Embedded Devices
El primer enlace encontrado es: https://bergel.eu/download/papers/Berg09cFlowtalk.pdf
```

Figura C.6: Resultado obtenido por ObtenerPdfConSerpApi.java - Arículo 2

Luego, se ejecutó la clase **DescargarPdfCache.java** con el título del artículo y su URL de descarga. Como resultado, se descargó el artículo en formato PDF de forma local. Asímismo, se añadió el nombre del artículo descargado a la base de datos, en la tabla *nombre_pdfs*:

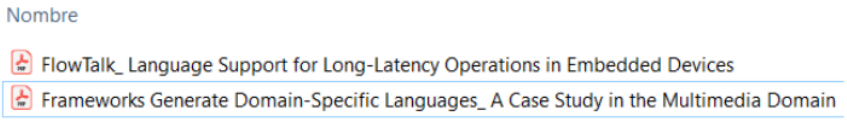


Figura C.7: PDF descargado - Artículo 2

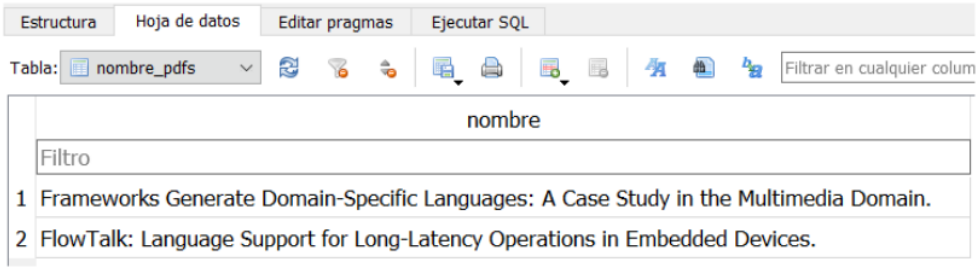
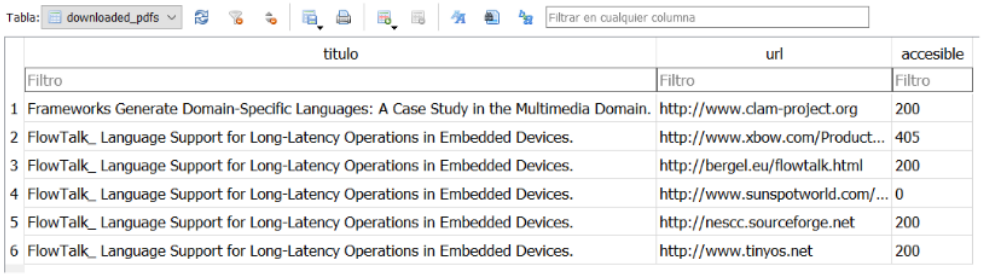


Figura C.8: Tabla nombre_pdfs - Artículo 2

Posteriormente, se ejecutó la clase **MostrarYComprobarEnlaces.java**, utilizando la ruta de descarga del artículo y su título. Como resultado se introdujo en la base de datos, en la tabla *downloaded_pdfs* el título del artículo, las URLs de referencia y el código devuelto por el servidor al intentar acceder a la URL:



	titulo	url	accesible
	Filtro	Filtro	Filtro
1	Frameworks Generate Domain-Specific Languages: A Case Study in the Multimedia Domain.	http://www.clam-project.org	200
2	FlowTalk_ Language Support for Long-Latency Operations in Embedded Devices.	http://www.xbow.com/Product...	405
3	FlowTalk_ Language Support for Long-Latency Operations in Embedded Devices.	http://bergel.eu/flowtalk.html	200
4	FlowTalk_ Language Support for Long-Latency Operations in Embedded Devices.	http://www.sunspotworld.com/...	0
5	FlowTalk_ Language Support for Long-Latency Operations in Embedded Devices.	http://nescc.sourceforge.net	200
6	FlowTalk_ Language Support for Long-Latency Operations in Embedded Devices.	http://www.tinyos.net	200

Figura C.9: Tabla downloaded_pdfs - Artículo 2

Una vez realizado el proceso completo, se intentó volver a ejecutar la clase **DescargarPdfCache.java** con el mismo artículo para comprobar que no se descarga, ya que, ya ha sido descargado previamente:

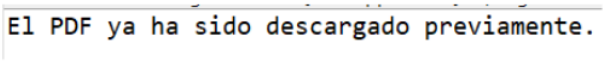


Figura C.10: Mensaje tras intentar volver a descargar un PDF - Artículo 2

Una vez se terminó de realizar la prueba 2 con el segundo artículo, se procedió a realizar la prueba 3.

Prueba 3

Se ejecutó la clase **ObtenerPdfConSerpApi.java** con el título de un artículo como argumento: *Dynamic QoS Management and Optimization in Service-Based Systems*.. Como resultado se mostró el título del artículo y una URL de descarga de este en formato PDF:

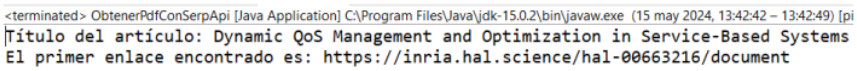


Figura C.11: Resultado obtenido por ObtenerPdfConSerpApi.java - Arículo 3

Luego, se ejecutó la clase **DescargarPdfCache.java** con el título del artículo y su URL de descarga. Como resultado, se descargó el artículo en formato PDF de forma local. Asímismo, se añadió el nombre del artículo descargado a la base de datos, en la tabla *nombre_pdfs*:

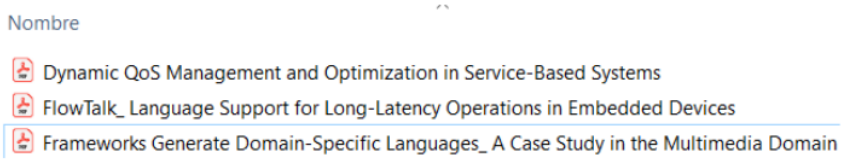
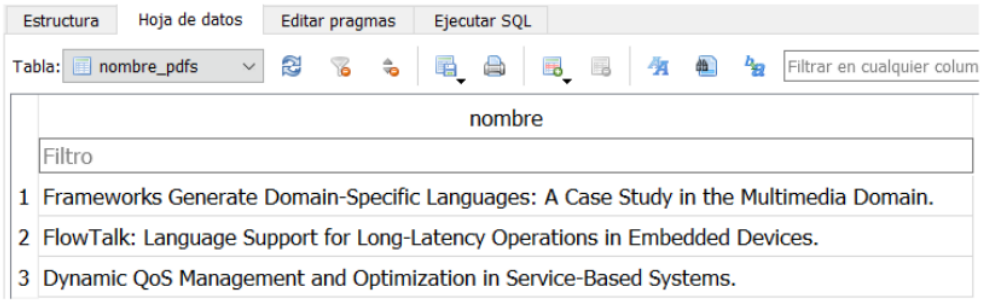


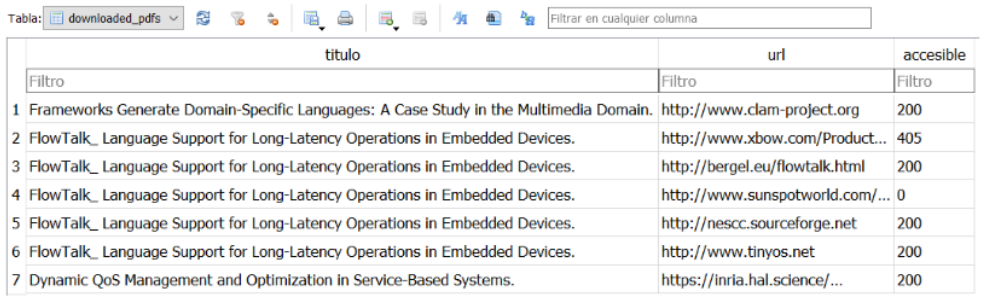
Figura C.12: PDF descargado - Artículo 3



	nombre
	Filtro
1	Frameworks Generate Domain-Specific Languages: A Case Study in the Multimedia Domain.
2	FlowTalk: Language Support for Long-Latency Operations in Embedded Devices.
3	Dynamic QoS Management and Optimization in Service-Based Systems.

Figura C.13: Tabla nombre_pdfs - Artículo 3

Posteriormente, se ejecutó la clase **MostrarYComprobarEnlaces.java**, utilizando la ruta de descarga del artículo y su título. Como resultado se introdujo en la base de datos, en la tabla *downloaded_pdfs* el título del artículo, las URLs de referencia y el código devuelto por el servidor al intentar acceder a la URL:



	titulo	url	accesible
	Filtro	Filtro	Filtro
1	Frameworks Generate Domain-Specific Languages: A Case Study in the Multimedia Domain.	http://www.clam-project.org	200
2	FlowTalk_ Language Support for Long-Latency Operations in Embedded Devices.	http://www.xbow.com/Product...	405
3	FlowTalk_ Language Support for Long-Latency Operations in Embedded Devices.	http://bergel.eu/flowtalk.html	200
4	FlowTalk_ Language Support for Long-Latency Operations in Embedded Devices.	http://www.sunspotworld.com/...	0
5	FlowTalk_ Language Support for Long-Latency Operations in Embedded Devices.	http://nescc.sourceforge.net	200
6	FlowTalk_ Language Support for Long-Latency Operations in Embedded Devices.	http://www.tinyos.net	200
7	Dynamic QoS Management and Optimization in Service-Based Systems.	https://inria.hal.science/...	200

Figura C.14: Tabla downloaded_pdfs - Artículo 3

Una vez realizado el proceso completo, se intentó volver a ejecutar la clase **DescargarPdfCache.java** con el mismo artículo para comprobar que no se descarga, ya que, ya ha sido descargado previamente:

```
El PDF ya ha sido descargado previamente.
```

Figura C.15: Mensaje tras intentar volver a descargar un PDF - Artículo 3

C.2. Ciclo 2

En este ciclo, se añadió una clase denominada **RunEntireFlow.java** la cual, al lanzarla se encarga de ejecutar las distintas clases que conforman el programa.

Prueba 1

Ejecutar la clase **RunEntireFlow.java** con el argumento: `https://dblp.org/search/publ/api?q=conf/icse/2024/workshop&h=3&format=xml`, el cual, contiene un fichero .xml de DBLP. Al ejecutar la clase con la URL especificada, esta llama a la clase **LeerXml.java** que devuelve el nombre de los artículos que contiene el fichero .xml:

```
<<<<<<<<<<LeerXml>>>>>>>>>>
"ECFuzz: Effective Configuration Fuzzing for Large-Scale Systems"
"Enabling Runtime Verification of Causal Discovery Algorithms with Automated Conditional Independence Reasoning"
"Inferring Data Preconditions from Deep Learning Models for Trustworthy Prediction in Deployment"
```

Figura C.16: Salida LeerXml - Ciclo 2

A continuación, llama a **ObtenerPdfConSerpApi.java** la cual, solo muestra por pantalla el JSON del artículo si no encontró ningún enlace para descargar dicho artículo en formato PDF, ya que así, se puede explorar de forma manual. En el caso que encuentre la URL de descarga, llama directamente a **DescargarPdfCache.java** la cual, crea una carpeta denominada *Cache* en la que guarda los artículos descargados en formato PDF.

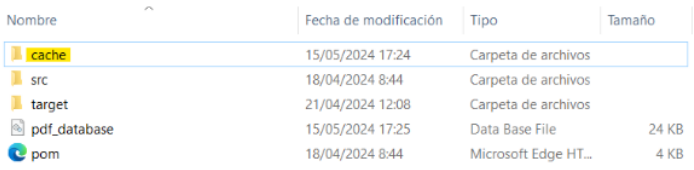


Figura C.17: Carpeta caché creada

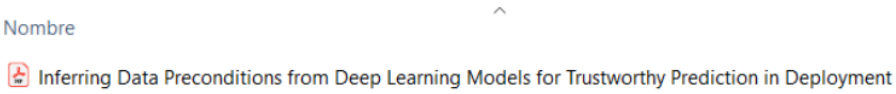


Figura C.18: PDF descargado - Ciclo 2

Además, se introduce en la base de datos, en la tabla *nombre_pdfs* el nombre del PDF descargado. A continuación, llama a **MostrarYComprobarEnlaces.java**, la cual, introduce en la base de datos, en la tabla *downloaded_pdfs* el título del artículo, las URLs que aparecen como referencias y el código del servidor para verificar la existencia de ellas:

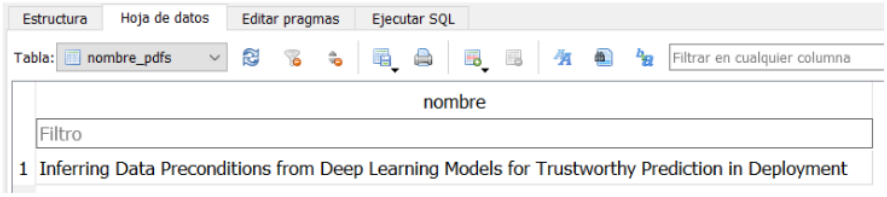
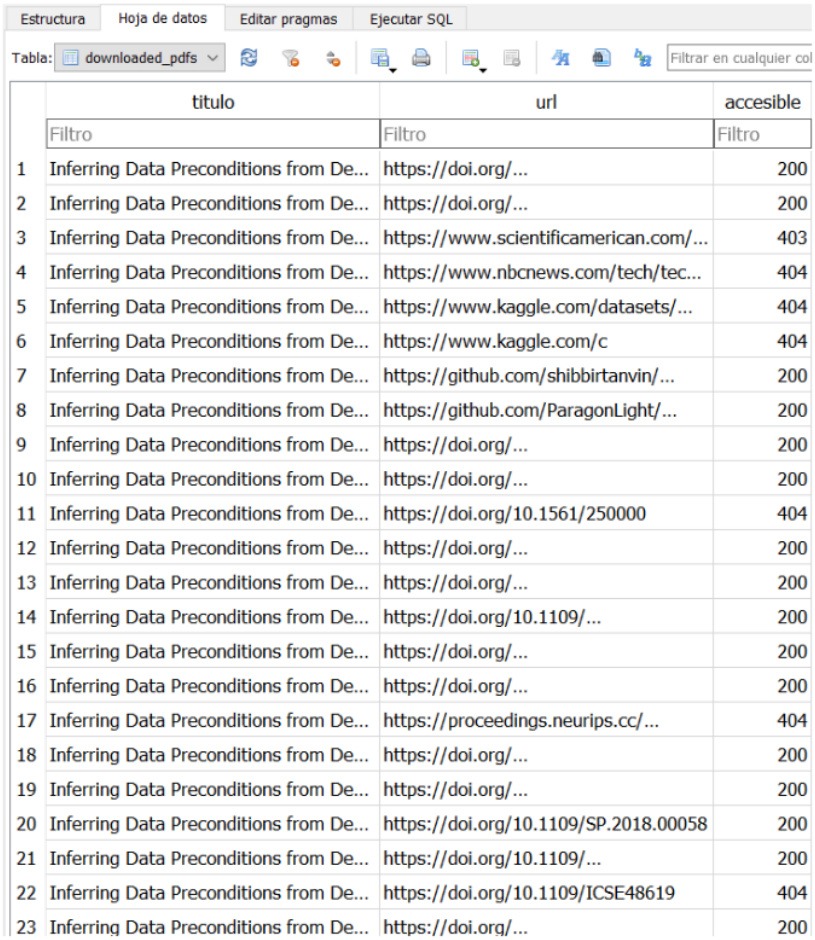


Figura C.19: Tabla nombre_pdfs - Ciclo 2



	titulo	url	accesible
	Filtro	Filtro	Filtro
1	Inferring Data Preconditions from De...	https://doi.org/...	200
2	Inferring Data Preconditions from De...	https://doi.org/...	200
3	Inferring Data Preconditions from De...	https://www.scientificamerican.com/...	403
4	Inferring Data Preconditions from De...	https://www.nbcnews.com/tech/tec...	404
5	Inferring Data Preconditions from De...	https://www.kaggle.com/datasets/...	404
6	Inferring Data Preconditions from De...	https://www.kaggle.com/c	404
7	Inferring Data Preconditions from De...	https://github.com/shibbirtanvin/...	200
8	Inferring Data Preconditions from De...	https://github.com/ParagonLight/...	200
9	Inferring Data Preconditions from De...	https://doi.org/...	200
10	Inferring Data Preconditions from De...	https://doi.org/...	200
11	Inferring Data Preconditions from De...	https://doi.org/10.1561/250000	404
12	Inferring Data Preconditions from De...	https://doi.org/...	200
13	Inferring Data Preconditions from De...	https://doi.org/...	200
14	Inferring Data Preconditions from De...	https://doi.org/10.1109/...	200
15	Inferring Data Preconditions from De...	https://doi.org/...	200
16	Inferring Data Preconditions from De...	https://doi.org/...	200
17	Inferring Data Preconditions from De...	https://proceedings.neurips.cc/...	404
18	Inferring Data Preconditions from De...	https://doi.org/...	200
19	Inferring Data Preconditions from De...	https://doi.org/...	200
20	Inferring Data Preconditions from De...	https://doi.org/10.1109/SP.2018.00058	200
21	Inferring Data Preconditions from De...	https://doi.org/10.1109/...	200
22	Inferring Data Preconditions from De...	https://doi.org/10.1109/ICSE48619	404
23	Inferring Data Preconditions from De...	https://doi.org/...	200

Figura C.20: Tabla downloaded_pdfs - Ciclo 2

24	Inferring Data Preconditions from De...	https://doi.org/...	200
25	Inferring Data Preconditions from De...	https://doi.org/10.1145/3546947	200
26	Inferring Data Preconditions from De...	https://doi.org/...	200
27	Inferring Data Preconditions from De...	https://doi.org/...	200
28	Inferring Data Preconditions from De...	https://proceedings.mlr.press/v80/...	200
29	Inferring Data Preconditions from De...	https://doi.org/10.1109/...	200
30	Inferring Data Preconditions from De...	https://doi.org/10.1145/3290354	200
31	Inferring Data Preconditions from De...	https://doi.org/...	200
32	Inferring Data Preconditions from De...	https://doi.org/...	200
33	Inferring Data Preconditions from De...	https://doi.org/...	200
34	Inferring Data Preconditions from De...	https://doi.org/10.1109/TDSC....	200
35	Inferring Data Preconditions from De...	https://doi.org/10.1	404
36	Inferring Data Preconditions from De...	https://doi.org/...	200

Figura C.21: Tabla downloaded_pdfs - Ciclo 2

Como se puede observar, se utilizó un artículo que contaba con un amplio número de URLs, en concreto 36, para poder comprobar si aún con un número elevado de ellas, el programa lograba verificar su existencia. El resultado fue satisfactorio.

tfg_etsiinf_Sara.pdf

ORIGINALITY REPORT

7%	7%	1%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	hdl.handle.net Internet Source	1%
2	oa.upm.es Internet Source	1%
3	dblp.dagstuhl.de Internet Source	<1%
4	docplayer.es Internet Source	<1%
5	issuu.com Internet Source	<1%
6	es.scribd.com Internet Source	<1%
7	ftp.riken.jp Internet Source	<1%
8	repository.unipiloto.edu.co Internet Source	<1%
9	libros.catedu.es Internet Source	<1%

10	www.researchgate.net Internet Source	<1 %
11	www.dropbox.com Internet Source	<1 %
12	www.jove.com Internet Source	<1 %
13	Janaina Sánchez García. "Desarrollo y caracterización de nuevas harinas de lenteja y quinoa fermentadas con Pleurotus ostreatus", Universitat Politecnica de Valencia, 2023 Publication	<1 %
14	view.genial.ly Internet Source	<1 %
15	thesisfiocruz.bvs.br Internet Source	<1 %
16	xavier.amatriain.net Internet Source	<1 %
17	dbpedia.org Internet Source	<1 %
18	repositori.uji.es Internet Source	<1 %
19	www.dmssupport.com Internet Source	<1 %
20	www.lua.inf.puc-rio.br Internet Source	<1 %

21	"Video juegos como herramienta de evaluación y comparación de aprendizaje con distintos formatos de respuesta", Pontificia Universidad Catolica de Chile, 2016 Publication	<1 %
22	1library.co Internet Source	<1 %
23	doczz.es Internet Source	<1 %
24	dspace.uib.es Internet Source	<1 %
25	pdfcookie.com Internet Source	<1 %
26	repositorio.ug.edu.ec Internet Source	<1 %
27	www.bibvirtual.ucb.edu.bo Internet Source	<1 %
28	(Carlinda Leite and Miguel Zabalza). "Ensino superior: inovação e qualidade na docência", Repositório Aberto da Universidade do Porto, 2012. Publication	<1 %
29	erevistas.publicaciones.uah.es Internet Source	<1 %
30	espaciojapon.com	

	Internet Source	<1%
31	experienceleague.adobe.com Internet Source	<1%
32	mx6.bumeran.com Internet Source	<1%
33	repositorio.uleam.edu.ec Internet Source	<1%
34	silo.tips Internet Source	<1%
35	www.ajpdsoft.com Internet Source	<1%
36	www.etse.urv.es Internet Source	<1%
37	www.intersystems.es Internet Source	<1%
38	Orlando Micolini, Luis Orlando Ventre, Mauricio Ludemann, Juan Ignacio Resiale Viano, Christopher Carlos Bien. "Case Study of Reactive and Embedded System Design Modeled with Petri Nets", 2018 IEEE International Conference on Automation/XXIII Congress of the Chilean Association of Automatic Control (ICA-ACCA), 2018 Publication	<1%

39	agorafir.org Internet Source	<1 %
40	docplayer.dk Internet Source	<1 %
41	documentosboletinoficial.buenosaires.gob.ar Internet Source	<1 %
42	europa.eu Internet Source	<1 %
43	hormigaanalitica.blogspot.com Internet Source	<1 %
44	idus.us.es Internet Source	<1 %
45	inlab.fib.upc.edu Internet Source	<1 %
46	openaccess.uoc.edu Internet Source	<1 %
47	riunet.upv.es Internet Source	<1 %
48	scindeks-clanci.ceon.rs Internet Source	<1 %
49	www.coursehero.com Internet Source	<1 %
50	www.idgl.com Internet Source	<1 %

51	www.scribd.com Internet Source	<1%
52	Angela Periche Santamaría. "STEVIA Y OTROS EDULCORANTES SALUDABLES EN LA FORMULACION DE GOLOSINAS FUNCIONALES: IMPLICACIONES TECNOLÓGICAS Y DE CALIDAD", Universitat Politecnica de Valencia, 2014 Publication	<1%
53	Guilherme Victor Selicani. "Desenvolvimento de transdutores ultrassônicos de alta potência para emissão em ar.", Universidade de Sao Paulo, Agencia USP de Gestao da Informacao Academica (AGUIA), 2021 Publication	<1%
54	www.sinnaps.com Internet Source	<1%
55	pazante.com Internet Source	<1%

Exclude quotes	On	Exclude matches	Off
Exclude bibliography	On		

Este documento esta firmado por

	**Firmante**	CN=tfgm.fi.upm.es, OU=CCFI, O=ETS Ingenieros Informaticos - UPM, C=ES
	Fecha/Hora	Mon Jun 03 12:16:08 CEST 2024
	Emisor del Certificado	EMAILADDRESS=camanager@etsiinf.upm.es, CN=CA ETS Ingenieros Informaticos, O=ETS Ingenieros Informaticos - UPM, C=ES
	Numero de Serie	561
	Metodo	urn:adobe.com:Adobe.PPKLite:adbe.pkcs7.sha1 (Adobe Signature)