



Universidad Politécnica
de Madrid



**Escuela Técnica Superior de
Ingenieros Informáticos**

Grado en Ingeniería Informática

Trabajo Fin de Grado

**Creación de una Web Demostradora
sobre Construcción de APIs REST a
partir de Ontologías y Grafos de
Conocimientos**

Autor: Jose Elías Silva Manrriquez

Tutor(a): Oscar Corcho García

Madrid, <<mes año>>

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: Creación de una Web Demostradora sobre construcción de APIs REST
a partir de Ontologías y Grafos de Conocimientos

Mes Año

Autor: Jose Elías Silva Manriquez

Tutor:

Oscar Corcho García
Departamento de Inteligencia Artificial
ETSI Informáticos
Universidad Politécnica de Madrid

Resumen

Durante los últimos años el uso de Grafos de Conocimiento (Knowledge Graphs) se ha incrementado tanto por organizaciones públicas y privadas. Actualmente podemos distinguir, por un lado, los ingenieros ontológicos que diseñan estos grafos de conocimientos y, por otro, los desarrolladores de aplicaciones que consumen sus contenidos. Aunque la adopción de estos grafos de conocimiento ha incrementado, siguen siendo difíciles de consumir por parte de los desarrolladores de aplicaciones. Las tecnologías web basadas en API para hacer que los datos de los grafos sean más accesibles para los desarrolladores pueden presentar dificultades a la hora de utilizarlos. Hay muchas APIs que son de utilidad hoy en día, pero debido a que los proyectos ya tienen sus años, es necesario realizar una investigación sobre el estado de estas, así como su funcionamiento. Este problema se plantea en una investigación por parte de la profesora Paola Espinoza [23], cotutora de este proyecto fin de grado.

El objetivo de este proyecto, en grandes rasgos, es facilitar el uso de estas APIs a través de contenedores Docker. Se detallará cada uno de los pasos de configuración esenciales para el uso de estas aplicaciones. Esto evitará contratiempos, al utilizar las APIs por primera vez, a futuros desarrolladores que necesiten utilizarlas para consumir los grafos de conocimiento.

Abstract

In recent years the use of Knowledge Graphs has increased by both public and private organizations. Currently we can distinguish, on the one hand, the ontological engineers who design these knowledge graphs and, on the other, the application developers who consume their content. Although the adoption of these knowledge graphs has increased, they are still difficult for application developers to consume. API-based web technologies to make graph data more accessible to developers can be difficult to use. There are many APIs that are useful today, but since the projects are already old, it is necessary to carry out an investigation on their status, as well as their operation. This problem arises in an investigation by professor Paola Espinoza [23], co-tutor of this final degree project.

The objective of this project, broadly speaking, is to facilitate the use of these APIs through Docker containers. Each of the essential configuration steps for the use of these applications will be detailed. This would avoid setbacks, when using the APIs for the first time, for future developers who need to use them to consume the knowledge graphs.

Índice de contenido

1	Introducción y Objetivos.....	7
1.1	Introducción.....	7
1.2	Objetivos	8
1.3	Estructura de la Memoria.....	8
2	Estado del Arte	10
2.1	Tecnologías	10
2.1.1	Pubby	10
2.1.2	Puelia.....	11
2.1.3	Basil	11
2.1.4	Trellis.....	11
2.1.5	Solid	11
2.1.6	Ontology2GraphQL	11
3	Desarrollo	12
3.1	Entorno Docker	12
3.1.1	Introducción e instalación.....	12
3.1.2	Comandos Docker	13
3.1.3	Comandos Dockerfile	14
3.2	Pubby.....	15
3.2.1	Pasos de configuración.....	15
3.2.2	Pasos de ejecución	16
3.2.2.1	Dockerfile	16
3.2.2.2	Ejecución.....	17
3.2.3	Ejemplos	17
3.2.3.1	Configuración con endpoint de Wikipedia	18
3.2.3.2	Configuración cargando un RDF	19
3.2.4	Análisis	20
3.2.4.1	Documentación.....	20
3.2.4.2	Mantenibilidad.....	20
3.2.4.3	Interfaz	20
3.2.5	Complicaciones	20
3.2.5.1	Maquina local	20
3.2.5.2	Dockerfile	21
3.2.6	Propuesta de mejora	22
3.2.7	Conclusión.....	22
3.3	Basil.....	23
3.3.1	Pasos de configuración.....	23
3.3.1.1	Creación de usuario.....	23
3.3.1.2	Creación de API	23

3.3.1.3	Uso y gestión de API.....	24
3.3.2	Pasos de ejecución	26
3.3.2.1	Dockerfile	26
3.3.2.2	Ejecución.....	27
3.3.3	Ejemplos	28
3.3.3.1	Creación de API Películas.....	29
3.3.3.2	Creación de API parametrizada y con documentación	30
3.3.4	Análisis.....	31
3.3.4.1	Documentación.....	31
3.3.4.2	Mantenibilidad.....	31
3.3.4.3	Interfaz	31
3.3.5	Complicaciones	31
3.3.5.1	Máquina local	31
3.3.5.2	Dockerfile	35
3.3.6	Propuesta de mejora	36
3.3.7	Conclusión.....	36
3.4	Puelia.....	37
3.5	Trellis.....	41
3.5.1	Pasos de configuración.....	41
3.5.2	Pasos de ejecución	43
3.5.3	Ejemplos.....	44
3.5.4	Análisis.....	44
3.5.4.1	Documentación.....	44
3.5.4.2	Mantenibilidad.....	44
3.5.4.3	Interfaz	44
3.5.5	Conclusión.....	44
3.5.6	Propuesta de mejora	44
3.6	Solid.....	45
3.6.1	Pasos de configuración.....	45
3.6.2	Pasos de ejecución	45
3.6.3	Ejemplos.....	45
3.6.4	Puntos débiles.....	45
3.6.5	Conclusión.....	45
3.6.6	Propuesta de mejora	45
3.7	Ontology2GraphQL.....	46
3.7.1	Pasos de configuración.....	46
3.7.2	Pasos de ejecución	46
3.7.3	Ejemplos.....	46
3.7.4	Puntos débiles.....	46

3.7.5	Conclusión.....	46
3.7.6	Propuesta de mejora	46
4	Resultados y conclusiones	48
5	Análisis de Impacto	49
6	Bibliografía	50
7	Anexos.....	51

Índice de ilustraciones

Ilustración 1: Esquema funcionamiento Pubby	10
Ilustración 2: Aplicación Docker.....	12
Ilustración 3: Extensión Docker Visual Studio Code	13
Ilustración 4: Visual Studio Code.....	13
Ilustración 5: Dockerfile para ejecución de Pubby	16
Ilustración 6: Ejemplo Pubby (endpoint dbpedia)	18
Ilustración 7: Ejemplo Pubby (navegando por endpoint dbpedia).....	19
Ilustración 8: Ejemplo Pubby (carga de RDF)	20
Ilustración 9: Error Pubby por configuración obsoleta.....	21
Ilustración 10: Error Dockerfile (version JDK)	21
Ilustración 11: Creación de API con Basil.....	23
Ilustración 12: Ejemplo de consulta parametrizada Basil.....	23
Ilustración 13: Puntos finales de API con Basil	24
Ilustración 14: Dockerfile (Basil)	26
Ilustración 15: Fichero script.sh (Basil).....	27
Ilustración 16: Fichero run.sh (Basil)	28
Ilustración 17: Ejecución servidor MySQL y aplicación Basil.....	28
Ilustración 18: Ejemplo Basil (API películas)	30
Ilustración 19: Ejemplo Basil (API parametrizada con documentación).....	31
Ilustración 20: Error creación proyecto con Maven I (Basil)	32
Ilustración 21: Error creación proyecto con Maven II (Basil).....	32
Ilustración 22: Error creación proyecto con Maven III (Basil).....	33
Ilustración 23: Error lanzamiento de aplicación Basil (MySQL)	33
Ilustración 24: Error lanzamiento de aplicación Basil (Javax).....	34
Ilustración 25: Solución error Javax	34
Ilustración 26: Navegador lanzamiento Basil.....	34
Ilustración 27: Error mysql-server	35
Ilustración 28: Error conexión mysql-server desde Dockerfile (Basil).....	35
Ilustración 29: Test Java conexión mysql-server	36
Ilustración 30: Error php 7 Puelia.....	38
Ilustración 31: simplegraph.class.php Puelia	38
Ilustración 32: Error configuración incompleta Basil	38
Ilustración 33: Error php en lda-cache.class.php (Puelia)	39
Ilustración 34: lda-cache.class.php (Puelia)	39
Ilustración 35: Error php en index.php (Puelia).....	40
Ilustración 36: index.php (Puelia).....	40

1 Introducción y Objetivos

1.1 Introducción

Durante los últimos años el uso de Grafos de Conocimiento (Knowledge Graphs) se ha incrementado tanto por organizaciones públicas y privadas. Actualmente podemos distinguir, por un lado, los ingenieros ontológicos que diseñan estos grafos de conocimientos y, por otro, los desarrolladores de aplicaciones que consumen sus contenidos. Aunque la adopción de estos grafos de conocimiento ha incrementado, siguen siendo difíciles de consumir por parte de los desarrolladores de aplicaciones. Las tecnologías web basadas en API para hacer que los datos de los grafos sean más accesibles para los desarrolladores pueden presentar dificultades a la hora de utilizarlos. Hay muchas APIs que son de utilidad hoy en día, pero debido a que los proyectos ya tienen sus años, es necesario realizar una investigación sobre el estado de estas, así como su funcionamiento. Este problema se plantea en una investigación por parte de la profesora Paola Espinoza [23], cotutora de este proyecto fin de grado.

El objetivo de este proyecto, en grandes rasgos, es facilitar el uso de estas APIs a través de contenedores Docker. Se detallará cada uno de los pasos de configuración esenciales para el uso de estas aplicaciones. Esto evitaría contratiempos, al utilizar las APIs por primera vez, a futuros desarrolladores que necesiten utilizarlas para consumir los grafos de conocimiento.

[EN PROCESO]

1.2 Objetivos

OB 1. – Investigación de tecnologías Linked Data

Se realizará un estudio sobre las tecnologías Linked Data propuestas. Principalmente se detallarán las diferentes configuraciones que puedan soportar así como los requerimientos que debe de tener para su funcionamiento.

OB 2. – Preparación y despliegue de contenedores Docker

Las herramientas desarrolladas hace años pueden presentar ciertos problemas de versiones y falta de mantenibilidad. Se desarrollará contenedores Docker a través de Dockerfiles para poder lanzar la aplicaciones de una manera sencilla. Se detallará cada uno de los pasos para poder lanzar las aplicaciones con éxito así como los resultados esperados.

OB 3. – Implementación de ejemplos

Sera necesario implementar un par de ejemplos para demostrar el funcionamiento de las herramientas. Estos ejemplos tendrán los requerimientos mínimos para que las personas que no tengan conocimientos sobre ello puedan entenderlos.

OB 4. - Elaboración de memoria

Se desarrollará a cabo durante todo el proceso de investigación y pruebas de las tecnologías planteadas. Se documentará cada uno de los pasos dados, dificultades encontradas y realizando un análisis de los aspectos más relevantes de las herramientas estudiadas.

1.3 Estructura de la Memoria

Con el propósito de ayudar a entender el trabajo realizado y la estructura de la memoria se detallará cada uno de los apartados con una breve explicación.

- **Introducción y Objetivos:** Primer capítulo donde se expone el contexto del problema planteado y la solución a realizar detallando los objetivos en los que se basa el desarrollo del trabajo.
- **Estado del Arte:** Introducción de las tecnologías propuestas
- **Desarrollo:** Capitulo con el mayor peso que contiene todo el desarrollo realizado sobre las tecnologías. Se explica los pasos de configuración a seguir y como lanzar las aplicaciones con Dockerfile. Se realiza un análisis sobre la documentación aportada, mantenibilidad e interfaz de la aplicación. También se detalla las complicaciones en el transcurso del desarrollo así como unas propuestas de mejora.
 - Pasos de configuración
 - Pasos de ejecución
 - Ejemplos
 - Análisis
 - ❖ Documentación

- ❖ Mantenibilidad
 - ❖ Interfaz
 - Complicaciones
 - Máquina local
 - Dockerfile
 - Propuesta de mejora
 - Conclusión
-
- **Resultados y conclusiones:** Conclusión del trabajo realizado.
 - **Análisis de Impacto:** Análisis del impacto del trabajo, vinculándolo con los Objetivos de Desarrollo Sostenible
 - **Bibliografía:** Referencias externas consultadas durante la investigación para realizar el trabajo.
 - **Anexos:** Documentos adicionales.

2 Estado del Arte

2.1 Tecnologías

2.1.1 Pubby

Gran parte de los datos de la Web Semántica se encuentran dentro de triplestores y solo se puede acceder a ellos enviando consultas SPARQL a un punto final SPARQL. Es difícil conectar la información de estas almacenes de RDF con otras fuentes de datos externas [5].

En RDF, los recursos se identifican mediante URI. Los URI utilizados en la mayoría de los conjuntos de datos SPARQL no son desreferenciables, esto significa que no se puede acceder a ellos en un navegador web semántico, ya que devuelven errores 404 (Not Found), o utilizan esquemas URI no desreferenciables, como en la etiqueta URI `tag:dbpedia.org,2007:Berlin`. [1]

Pubby es una aplicación web Java que permite navegar al usuario por el contenido de un punto final de forma sencilla desde el navegador sin la necesidad de realizar consultas SPARQL.

Al configurar un servidor Pubby para un punto final SPARQL, configurará un mapeo que traduzca esos URI en URI desreferenciables manejados por Pubby.

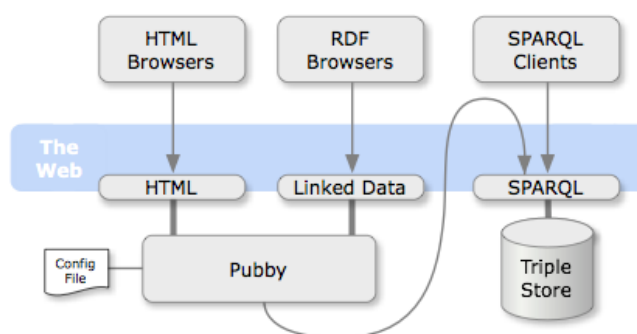


Ilustración 1: Esquema funcionamiento Pubby

Pubby manejará las solicitudes a los URI mapeados conectándose al punto final SPARQL, solicitándole información sobre el URI original y devolviendo los resultados al cliente. También maneja varios detalles de la interacción HTTP, como la redirección 303 requerida por la Arquitectura Web y la negociación de contenido entre HTML, RDF / XML y descripciones de Turtle del mismo recurso. Incluye una extensión de metadatos para agregar metadatos a los datos proporcionados.

2.1.2 Puelia

Puelia es una implementación PHP de la especificación de la API Linked Data [2]. Es una aplicación que maneja las solicitudes entrantes leyendo archivos de configuración rdf o turtle (en /api-config-files/) y convirtiendo esas solicitudes en consultas SPARQL que se utilizan para recuperar datos RDF de los puntos finales SPARQL declarados en los archivos de configuración. Los datos RDF obtenidos del punto final SPARQL se puede retornar al usuario en una serie de opciones de formato, incluidos los formatos turtle, rdf / xml y “simples” json/xml.

2.1.3 Basil

Basil está diseñado como un sistema de middleware que media entre los puntos finales SPARQL y las aplicaciones mediante la construcción de API web a partir de consultas SPARQL almacenadas [3].

Para la generación de una API se debe de incluir un punto final SPARQL y la consulta SPARQL. Tanto los parámetros de entrada como la forma de los datos de salida se especifican dentro de la propia consulta SPARQL. Finalmente, Basil genera las rutas de la API para recuperar los datos y la documentación de la especificación Swagger de la API.

2.1.4 Trellis

Trellis es un servidor LDP modular que enfatiza tanto la escalabilidad como la conformidad con los estándares web [4].

La arquitectura de Trellis admite la ampliación horizontal de un sistema, tanto para admitir grandes cantidades de datos, como para la redundancia de datos y altas cargas de servidor

2.1.5 Solid

Community Solid Server es un software abierto para proporcionar a las personas su propio Solid Pod.

Proporcionará a los desarrolladores un entorno para crear y probar nuevas aplicaciones de Solid.

Su arquitectura modular permite probar nuevas ideas en el lado del servidor y así dar forma al futuro de Solid.

2.1.6 Ontology2GraphQL

Aplicación web que genera un esquema GraphQL y su correspondiente servicio GraphQL a partir de una determinada Ontología RDF [29]. Para ello, la ontología para la representación de datos debe anotarse manualmente con un Metamodelo GraphQL (GQL), que incluye varias clases que representan los tipos GraphQL que componen un esquema GraphQL (por ejemplo, objeto, lista, enumeración, entre otros). Por lo tanto, cada

El elemento de ontología se asigna a una clase GQL (una ontología class es una instancia de una clase de objeto GQL).

Tiene memoria de desarrollo

3 Desarrollo

3.1 Entorno Docker

3.1.1 Introducción e instalación

Docker nos permite crear imágenes apartir del nucleo del sistema operativo que necesitemos y apartir de ello crear contenedores para poder ejecutar nuestras aplicaciones. Todas las herramientas que se explicaron anteriormente necesitaran instalar varios programas y mediante dockerfiles, ficheros con los que se crean las imágenes, podemos realizar todas esas instalaciones de una manera conjunta y sencilla.

Para obtener Docker debemos de acceder a la página oficial de Docker [6] y seguir los pasos de instalación según el sistema operativo que tengamos. Para el desarrollo práctico y prueba de herramientas se ha utilizado Docker para Windows 10 Pro.

Una vez instalado la aplicación Docker tendremos una interfaz similar a la siguiente, donde podremos ver los contenedores e imágenes que tenemos en nuestro sistema. Tendremos que crearnos una cuenta en Docker Hub [7] para acceder al repositorio de imágenes, que se suelen usar como base para los dockerfile, necesario para lanzar nuestras aplicaciones.

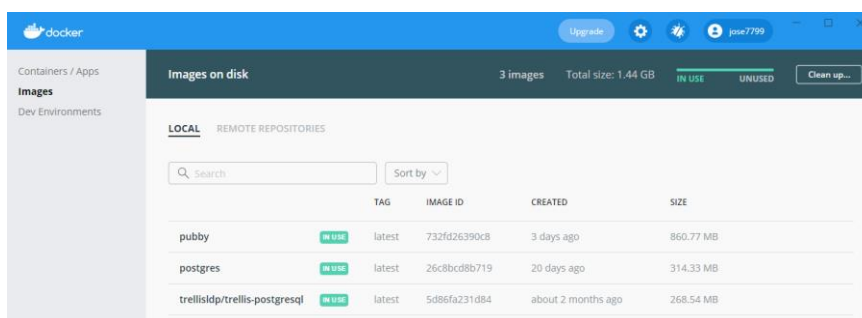


Ilustración 2: Aplicación Docker

Una vez creada nuestra cuenta, regresamos a nuestra aplicación para iniciar sesión. Desde un terminal cmd de nuestro equipo ya podremos tener acceso a las funcionalidades de Docker para probar nuestros Dockerfile. Por mayor comodidad recomiendo instalarnos Visual Studio Code [8] o editares similares para poder observar de forma más sencilla los Dockerfile, ficheros de configuración y el terminal de ejecución. Además si optamos por Visual Studio Code, hay la posibilidad de instalarnos la extensión de Microsoft Docker que nos permite, dentro del mismo entrono de desarrollo, gestionar cada una de nuestras imágenes y contenedores. Al igual que en la aplicación Docker, será necesario iniciar sesión para tener acceso al repositorio de imágenes Docker.



Ilustración 3: Extensión Docker Visual Studio Code

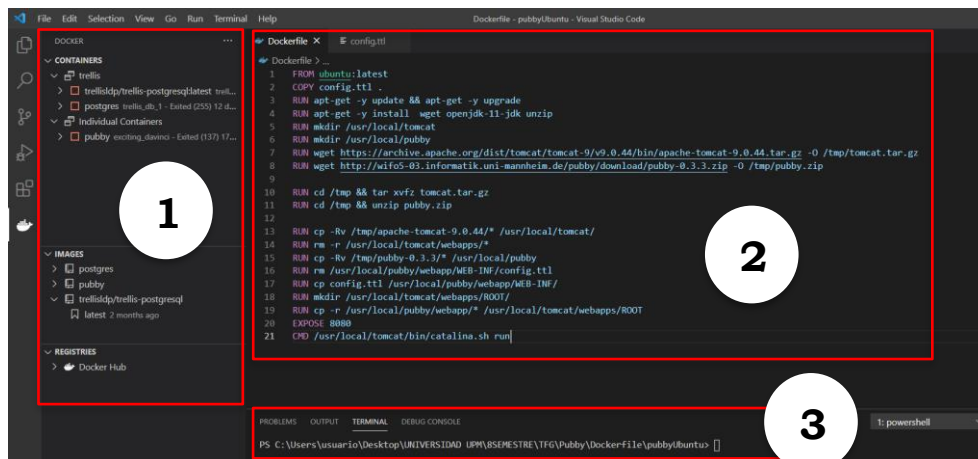


Ilustración 4: Visual Studio Code

Podemos observar en la Ilustración 4 la distribución del espacio de nuestro entorno de desarrollo teniendo la extensión Docker.

- Zona 1: Acceso a nuestros contenedores e imágenes creadas. En la opción “Registros” podremos iniciar sesión en Docker Hub.
- Zona 2: Donde se nos abrirá nuestros Dockerfile y ficheros de configuración
- Zona 3: Terminal donde ejecutaremos nuestros comandos Docker para la creación de imágenes y lanzamientos de contenedores. Para abrir esta zona tendremos que dirigirnos a la barra superior y seleccionar “Terminal”-“New Terminal”.

3.1.2 Comandos Docker

Los comandos Docker más frecuentes que utilizaremos para probar las herramientas serán:

```
docker build -t nombreimagen .
```

Creación de la imagen a partir de nuestro Dockerfile, sustituyendo “nombreimagen” por el nombre que queramos para nuestra imagen. Este nombre deberá de estar todo en minúsculas y se recomienda un nombre relacionado con la herramienta que se está probando. El “.” del final indica que realice la búsqueda del Dockerfile en el directorio actual de trabajo.

```
docker run -d -p 8080:8080 nombreimagen
```

Lanzar nuestra imagen “nombreimagen” en el contenedor en segundo plano(-dp) y creando un mapeo entre el puerto 8080 del host y el puerto 880 del contenedor. Sin el mapeo de puertos, no podríamos acceder a la aplicación. En

ocasiones necesitaremos cambiar la ejecución en segundo plano por una sesión iterativa con nuestro contenedor, para ello cambiamos “-d” por “-it”

```
-v "$pwd/DirTrabajo:/rutaimagen " nombreimagen
```

Si queremos añadir un volumen para tener cambios actualizables entre una carpeta de nuestro directorio actual de trabajo y un directorio de la imagen necesitamos añadir la opción:

```
docker-compose up
```

Permite iniciar todos los servicios de un fichero docker-compose.yml. Este fichero nos permite definir y ejecutar aplicaciones Docker de varios contenedores.

3.1.3 Comandos Dockerfile

```
FROM imagen
```

Permite coger como base una imagen del repositorio de DockerHub.

```
COPY directorio-host directorio-imagen
```

Permite copiar el contenido de un directorio o de un fichero de nuestro equipo al directorio que especifiquemos dentro de la imagen base.

```
RUN comando
```

Ejecuta los comandos que se lanzarían dentro de nuestra imagen. Debe de ser acorde al sistema operativo que use la imagen base.

```
EXPOSE puerto
```

Indica a Docker que el servicio del contenedor se puede conectar a través del puerto que se pasa como argumento.

```
CMD ejecutable parametro1 parametro2
```

Ejecuta un comando o ejecutable una vez que el contenedor se haya inicializado

3.2 Pubby

3.2.1 Pasos de configuración

En el archivo config.ttl podemos determinar nuestra configuración de la aplicación Pubby.

En primer lugar debemos de especificar los prefijos que vamos a utilizar. Los que se utilizan comúnmente son:

```
@prefix conf: <http://richard.cyganiak.de/2007/pubby/config.rdf#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
```

También necesitamos determinar, el nombre del proyecto (como título de las páginas), la URI relacionada al título (como página de inicio) y la web base donde está instalada la web Pubby, en nuestro caso, “http://localhost:8080/”:

```
conf:projectName "Project Name";
conf:projectHomepage <project_homepage_url.html>;
conf:webBase <http://localhost:8080/>;
```

Las configuraciones de un dataset están englobadas mediante los corchetes. Dentro debemos de especificar los datos a exponer a través de un endpoint SPARQL y el prefijo URI común que identifica a los recursos:

```
conf:dataset [
conf: sparqlEndpoint < sparql_endpoint_url >;
conf: datasetBase < dataset_uri_prefix >;
];
```

Tenemos la opción de cargar un recurso RDF en vez de usar un endpoint SPARQL, para ello sustituimos la línea de configuración “sparqlEndpoint” por:

```
conf: loadRDF < data1.rdf >, < data1.rdf >, ...;
```

Con los anteriores pasos podríamos tener una configuración sencilla lista para utilizar. Las siguientes configuraciones que se describen me parecieron de gran interés para usarlas. Podemos encontrar todas las configuraciones disponibles en la página oficial de Pubby [1]

Si queremos establecer una página de inicio debemos de especificar una URI de conjunto de datos y no una URI web mapeada en la configuración “indexResource”:

```
conf:indexResource <dataset_uri>;
```

Si el valor de las siguientes propiedades RDF esta presente en el conjunto de datos, se utilizará como etiquetas (labelProperty), descripción textual (commentProperty) o como una URL de imagen (imageProperty):

```
conf:labelProperty ex:property1, ...;
```

Por defecto, rdfs:label, dc:title, foaf:name.

```
conf:commentProperty ex:property1, ...;
```

Por defecto, rdfs:comment, dc:description.

```
conf:imageProperty ex:property1, ...;
Por defecto, foaf:depiction..
```

Si queremos utilizar las declaraciones de prefijo, de un documento RDF, en la salida podemos utilizar la configuración “usePrefixFrom” para vincularlo. Si por el contrario queremos utilizar los prefijos de nuestro fichero config.ttl lo dejamos vacío:

```
conf: usePrefixesFrom < archivo.rdf >;
```

Las siguientes configuración forman parte del conjunto de reglas que se podrían poner dentro de cada dataset[]

Si los datos de interés no se encuentran en el gráfico predeterminado del conjunto de datos SPARQL, sino dentro de un gráfico con nombre, entonces su nombre debe especificarse en la configuración “sparqlDefaultGraph”

```
conf: sparqlDefaultGraph < sparql_default_graph_name >;
```

Si queremos que una declaración owl:sameAs de la forma <web_uri> owl:sameAs <dataset_uri> este presente en la salida de nuestros datos vinculados, debemos de poner “true” en la configuración “addSameAsStatements”

```
conf: addSameAsStatements "true"/"false" ;
```

3.2.2 Pasos de ejecución

3.2.2.1 Dockerfile

Nuestro fichero Dockerfile nos instalara todo lo necesario para que podamos ejecutar nuestra aplicación Pubby, sin importar el sistema operativo desde donde se lance. En este caso instalará en nuestra imagen resultante:

- Imagen base: ubuntu:16.04
- Comandos Ubuntu
 - wget: para descargar recursos
 - unzip: para descomprimir los recursos en formato zip
- Apache Tomcat 9.0.44
- JDK-11
- pubby-0.3.3

```
Dockerfile > ...
1 FROM ubuntu:20.04
2 COPY ./Configuracion/config.ttl .
3 RUN apt-get -y update && apt-get -y upgrade
4 RUN apt-get -y install wget openjdk-11-jdk unzip
5 RUN mkdir /usr/local/tomcat
6 RUN mkdir /usr/local/pubby
7 RUN wget https://archive.apache.org/dist/tomcat/tomcat-9/v9.0.44/bin/apache-tomcat-9.0.44.tar.gz -O /tmp/tomcat.tar.gz
8 RUN wget http://wifo5-03.informatik.uni-mannheim.de/pubby/download/pubby-0.3.3.zip -O /tmp/pubby.zip
9
10 RUN cd /tmp && tar xvfz tomcat.tar.gz
11 RUN cd /tmp && unzip pubby.zip
12
13 RUN cp -Rv /tmp/apache-tomcat-9.0.44/* /usr/local/tomcat/
14 RUN rm -r /usr/local/tomcat/webapps/*
15 RUN cp -Rv /tmp/pubby-0.3.3/* /usr/local/pubby
16 RUN cp config.ttl /usr/local/pubby/webapp/WEB-INF/
17 RUN mkdir /usr/local/tomcat/webapps/ROOT/
18 RUN cp -r /usr/local/pubby/webapp/* /usr/local/tomcat/webapps/ROOT
19 EXPOSE 8080
```

Ilustración 5: Dockerfile para ejecución de Pubby

La aplicación Tomcat despliega las aplicaciones que se encuentren dentro de la carpeta “webapp”. Ya viene con carpetas por defecto, se eliminarán ya que queremos que nuestra aplicación Pubby se ejecute desde una URI raíz **http://localhost:8080/**. Se vuelve a crear una carpeta ROOT, donde se mete la carpeta “webapp” de nuestra aplicación Pubby. Pero antes se sustituye la configuración “config.ttl” que viene por defecto en la ruta “pubby/webapp/WEB-INF” por nuestro fichero de configuración del directorio actual de trabajo. Finalmente se indica que el contenedor se puede conectar a través del puerto 8080.

3.2.2.2 Ejecución

Entendido el funcionamiento de nuestro Dockerfile, procedemos a generar la imagen:

```
docker build -t pubby .
```

Esperamos a que el proceso termine. Esto puede demorar unos minutos ya que se debe de descargar e instalar bastante paquetes de datos. Una vez tengamos nuestra imagen debemos de ejecutar el comando:

```
docker run -it -p 8080:8080 -v "$(pwd)/Configuracion:/usr/tmp" pubby
```

Esto permite una comunicación entre el contenedor y el host por el puerto 8080. Además usa un volumen para copiar el contenido de nuestra carpeta “Configuración” en la ruta “usr/tmp” de nuestra imagen de forma actualizable. Lo ejecutamos de forma iterativa “-t” ya que necesitaremos ejecutar comandos en nuestro contenedor. Una vez lanzado el anterior comando se nos abrirá un Shell Bash donde debemos de poner el siguiente comando para lanzar el servlet Tomcat:

```
/usr/local/tomcat/bin/catalina.sh run
```

Con esto ya tendríamos la aplicación Pubby en funcionamiento. Podemos observar el resultado en la ruta **http://localhost:8080/**

Si deseamos cambiar nuestra configuración bastaría con modificar el fichero “config.ttl” de nuestro directorio de trabajo en host y realizar los siguientes pasos:

- Modificar el fichero config.ttl en nuestro host que se encuentra en el directorio “/Configuracion”
- Parar la ejecución de Catalina (Tomcat) con ctrl+C en nuestro Shell Bash
- Ejecutar el siguiente comando para actualizar la configuración en Tomcat :

```
cp /usr/tmp/config.ttl /usr/local/tomcat/webapps/ROOT/WEB-INF/config.ttl
```

- Lanzar de nuevo el servlet Tomcat:

```
/usr/local/tomcat/bin/catalina.sh run
```

3.2.3 Ejemplos

Tras lanzar la aplicación Pubby podemos realizar los siguientes ejemplos cambiando el contenido del fichero config.ttl, que tenemos en el directorio actual de trabajo, por las configuraciones “configEj1.ttl” o “configEj2” de nuestra carpeta del proyecto correspondientes al ejemplo 1 y 2

3.2.3.1 Configuración con endpoint de Wikipedia

En el siguiente ejemplo podemos observar que se utilizara como endpoint DBpedia y se buscará todo los recursos que coincidan con la URI “/resource/Wikipedia”.

```
@prefix conf: <http://richard.cyganiak.de/2007/pubby/config.rdf#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix skos: <http://www.w3.org/2004/02/skos/core#> .
@prefix geo: <http://www.w3.org/2003/01/geo/wgs84_pos#> .
@prefix dbpedia: <http://localhost:8080/resource/> .
@prefix p: <http://localhost:8080/property/> .
@prefix prvTypes: <http://purl.org/net/provenance/types#> .
@prefix doap: <http://usefulinc.com/ns/doap#> .
<> a conf:Configuration;
    conf:projectName "DBpedia.org";
    conf:projectHomepage <http://dbpedia.org>;
    conf:webBase <http://localhost:8080/>;
    conf:usePrefixesFrom <>;
    conf:defaultLanguage "es";
    conf:indexResource <http://dbpedia.org/resource/Wikipedia>;

    conf:dataset [
        conf:sparqlEndpoint <https://dbpedia.org/sparql>;
        conf:sparqlDefaultGraph <http://dbpedia.org>;
        conf:datasetBase <http://dbpedia.org/resource/>;
    ];
.
```



Wikipedia es una enciclopedia libre, poliglota y editada de manera colaborativa. Es administrada por la Fundación Wikimedia, una organización sin ánimo de lucro cuya financiación está basada en donaciones. Sus más de 50 millones de artículos en 300 idiomas han sido redactados en conjunto por voluntarios de todo el mundo, lo que suma más de 2000 millones de ediciones, y permite que cualquier persona pueda sumarse al proyecto para editarlos, a menos que la página se encuentre protegida contra vandalismos para evitar problemas o disputas.

Property	Value
abstract	<ul style="list-style-type: none">A Wikipédia é um projeto de enciclopédia multilíngua de licença livre, baseado na web e escrito de maneira colaborativa. O projeto encontra-se sob administração ... more ^(pt)Ciclipéid, ghréasán-bhunaithe, iteangach, is ea an Vicipéid (i.e. Wikipedia agus na fionscraimh atá ag gabháil leis) agus tá an méid a bhfuil inti ar ... more ^(ga)Die Wikipedia [ˈvɪkiˈpeːdi̯a] () ist ein am 15. Januar 2001 gegründetes gemeinnütziges Projekt zur Erstellung einer Enzyklopädie in zahlreichen Sprachen ... more ^(de)La Viquipèdia (Wikipedia en anglès, mantleu del hawaiaikiwiki – pronunciat amb variació lliure /ˈvikiˈviki/ o /ˈvikiˈwiki/– que vol dir «molt ràpid», ... more ^(ca)Wikipedia (angle Wikipedia [dɪkɪpiˈdiːə]) estas interreta, plurlingva enciklopedio kun enhavo laŭ la koncepto de libera verko. Ĝin verkadas volontuloj surbaze ... more ^(eo)Wikipedia () atau () WIK-ī-PEE-dee-a) adalah proyek ensiklopedia multibahasa dalam jaringan yang bebas dan terbuka, yang dijalankan oleh Wikimedia ... more ^(id)Wikipedia () wik-ih-PEE-dee-a or () wik-ee-PEE-dee-a, abbreviated as WP) is a multilingual online encyclopedia created and maintained as an open collaboration ... more ^(en)Wikipedia (engelskt uttal: [ˈwɪki pi di ə] () eller [ˈwɪki pi di a] (); svenskt uttal: [ˈvɪkɪpe dɛa]), brukar ofta förkortas WP, är en wiki och en mångspråkig ... more ^(sv)Wikipedia (pronuncia:) è un'enciclopedia online a contenuto libero, collaborativa, multilingue e gratuita, nata nel 2001, sostenuta e ospitata dalla Wikimedia ... more ^(it)Wikipedia eduki askeko entziklopedia bat da, lankidetzaz editatua, eleantiza, Interneten argitaratua, Wikimedia Fundazioa irabazi asmorik gabeko erakundeak ... more ^(eu)Wikipedia es una enciclopedia libre, poliglota y editada de manera colaborativa. Es administrada por la Fundación Wikimedia, una organización sin ánimo ... more ^(es)Wikipedia is een meertalige internetencyclopedie, die door auteurs op vrijwillige basis wordt geschreven. Wikipedia wordt gepubliceerd onder een vrije ... more ^(nl)Wikipedia – wielojęzyczna encyklopedia internetowa działająca zgodnie z zasadą otwartej treści. Funkcjonuje w oparciu o oprogramowanie MediaWiki (haw. ... more ^(pl)Wikipedia (anglicky a v mnoha dalších jazycích Wikipedia) je mnohojazyčná online encyklopedie vytvořená a udržovaná jako projekt otevřené spolupráce se ... more ^(cs)Wikipedia est une encyclopédie universelle, collaborative et multilingue, créée par Jimmy Wales et Larry Sanger le 15 janvier 2001, gérée par wiki dans ... more ^(fr)Η Βικιπαίδεια (αγγλικά: Wikipedia, ρωσικά: Википедия) είναι διεθνής, πολυγλώσσια, ψηφιακή, διαδικτυακή, ελεύθερου περιεχομένου, εγκυκλοπαίδεια, που βασίζεται ... more ^(el)Вікіпедія (англ. Wikipedia, МФА: [ˈwɪki pi di ə] или [ˈwɪki pi di a]) — общедоступная многоязычная универсальная интернет-энциклопедия со свободным ... more ^(ru)Wikipédia (англ. Wikipedia, МФА: [ˈwɪki pi di ə]) — загальнодоступна вільна багатомовна онлайн-енциклопедія, якою опікується неприбуткова організація «Фонд ... more ^(uk)ويكيپيديا (arab. [ˈwɪkiːbiːdiːja] و [ˈwɪkiːpiːdiːa])، وتسمى لغة دواي "مِقالَة الحرّة"؛ wikipi di a)؛ "وتعني لغة دواي "مِقالَة الحرّة" ... more ^(ar)ウィキペディア (英: Wikipedia) は、ウィキメディア財団が運営しているインターネット百科事典である。コピーレフトなライセンスのもと、サイトにアクセス可能な誰もが無料で自由に編集に参加できる。世界の各言語で展開されている。「ウィキペディア (Wikipedia)」という名前は、ウェブブラウザ上でウェブページを編集することができる「ウィキ (Wiki)」というシステムを使用した「百科事典」(英: Encyclopedia) であることに由来する造語である。設立者の1人であるラリー・サンガーにより命名された。(ja)维基百科 (英語: Wikipedia, /ˈwɪki pi di ə/ 或 /ˈwɪki pi di ə/) 是网络百科全书项目, 特点是自由内容、自由编辑。目前是全球网络上最大且最受大眾歡迎的參考工具书, 名列全球十大最受歡迎的網站。維基百科目前由非營利組織維基媒體基金會負責營運。Wikipedia是是成詞, 分別取自於網站核心技术「Wiki」以及英文中文百科全書之意的「encyclopedia」。(zh)위키백과(Wiki百科, IPA: [ˈwikibɐkʰwɑ], [ˈɣɕibɐkʰwɑ] ()) 혹은 위키미디어(영어: Wikipedia 위키피디어[1], IPA: [ˈwɪki pi di ə] ())는 누구나 자유롭게 쓸 수 있는 다언어판 인터넷 백과사전이다. ... more ^(ko)
accessdate	2015-04-05 ()
alexa	(en)
	13 ()
align	right (en)
audio	−01:15 ()
author	<ul style="list-style-type: none"><http://localhost:8080/Jimmy_Wales><http://localhost:8080/Larry_Sanger>

Ilustración 6: Ejemplo Pubby (endpoint dbpedia)

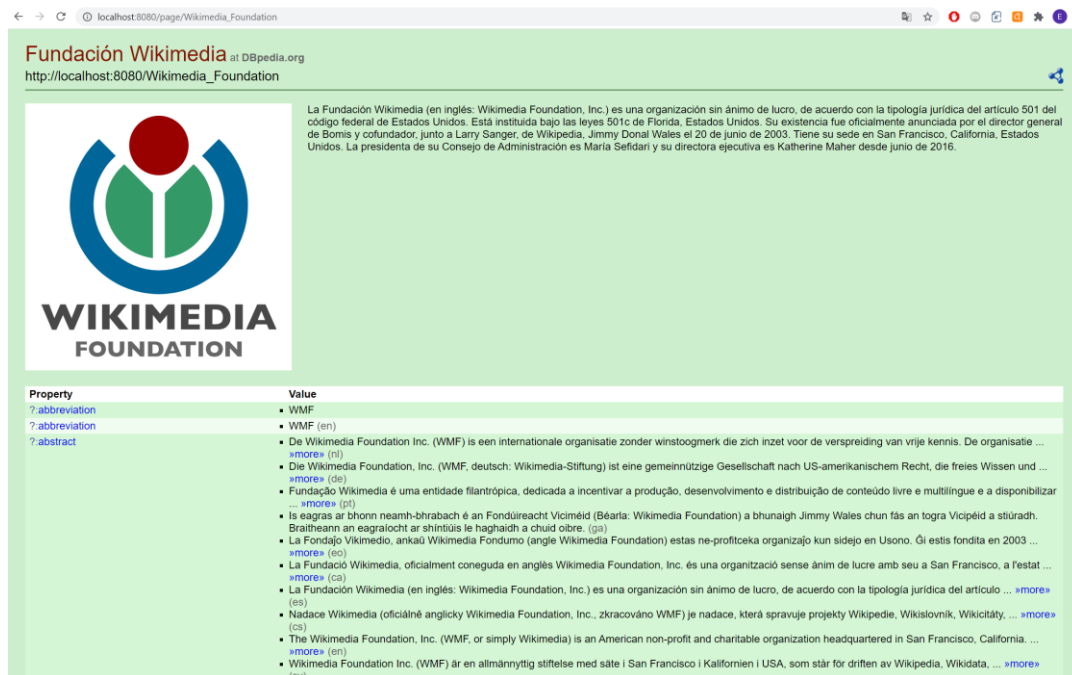


Ilustración 7: Ejemplo Pubby (navegando por endpoint dbpedia)

3.2.3.2 Configuración cargando un RDF

En este ejemplo podemos observar cómo se carga 2 ficheros RDF, de propiedad de Richar Cyganiak, para exponer los recursos a través de la aplicación Pubby. Además se usan los prefijos de estos ficheros en la salida.

```
# Ejemplo de configuración que carga algunos ficheros estáticos RDF de Richard
Cyganiak.
# Asumiendo que Pubby está corriendo en http://localhost:8080/
@prefix conf: <http://richard.cyganiak.de/2007/pubby/config.rdf#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
<> a conf:Configuration;
    conf:projectName "Richard Cyganiak's Homepage";
    conf:projectHomepage <http://richard.cyganiak.de/>;
    conf:webBase <http://localhost:8080/>;
    conf:dataset [
        conf:datasetBase <http://richard.cyganiak.de/>;
        conf:loadRDF <http://richard.cyganiak.de/foaf.rdf>;
        conf:loadRDF <http://richard.cyganiak.de/cygri.rdf>;
    ];
    conf:usePrefixesFrom <http://richard.cyganiak.de/foaf.rdf>;
    conf:usePrefixesFrom <http://richard.cyganiak.de/cygri.rdf>;
    conf:labelProperty rdfs:label, dc:title, foaf:name;
    conf:indexResource <http://richard.cyganiak.de/foaf.rdf#cygri>;
    .
```

Property	Value
vcard:BDAY	1979-07-15
foaf:account	<ul style="list-style-type: none"> <http://beta.plazes.com/user/cygri/> <http://del.icio.us/cygri> <http://flickr.com/people/cygri/> <http://twitter.com/cygri> <http://www.dopplr.com/traveller/cygri>
foaf:based_near	<ul style="list-style-type: none"> <http://dbpedia.org/resource/Galway> [2 anonymous resources]
foaf:currentProject	[3 anonymous resources]
foaf:depiction	<ul style="list-style-type: none"> <http://localhost:8080/richard-2002.jpg> <http://localhost:8080/richard-2003.jpg> <http://localhost:8080/richard.jpg> <http://statcvs.sourceforge.net/statcvs-developers.jpg>
is foaf:depicts of	<ul style="list-style-type: none"> <http://localhost:8080/richard-2002.jpg> <http://localhost:8080/richard-2003.jpg> <http://localhost:8080/richard.jpg> <http://statcvs.sourceforge.net/statcvs-developers.jpg>
owl:differentFrom	<http://www.abeservices.com.au/people/rhancock/foaf.rdf#rhancock>
foaf:firstName	Richard
foaf:gender	male

Ilustración 8: Ejemplo Pubby (carga de RDF)

3.2.4 Análisis

3.2.4.1 Documentación

La documentación proporcionada en la página oficial de Pubby [1] solo muestra la nomenclatura que debe tener la configuración. No podemos encontrar ejemplos funcionales, más allá del fichero de configuración secundario “conf-myfoaf.ttl” o documentación más detallada para poder construir nuestro fichero de configuración correctamente. Además podemos encontrarnos con las instrucciones de descarga e instalación para nuestra aplicación.

3.2.4.2 Mantenibilidad

Archivo de configuración por defecto tiene un punto endpoint obsoleto “http://dbpedia.openlinksw.com:8890/sparql” lo que produce un error en su despliegue.

La última versión de la aplicación es del 2011. Podemos concluir que esta aplicación no presenta actualizaciones desde hace mucho tiempo por lo que no cumple el aspecto de mantenibilidad.

3.2.4.3 Interfaz

Podemos observar que esta herramienta presenta un interfaz bastante amigable e intuitiva que nos permite interactuar con los recursos clicando directamente sobre el enlace. Además nos permite recuperar imágenes de los recursos a los que accedemos lo que mejora la estética de la página.

3.2.5 Complicaciones

3.2.5.1 Maquina local

Una vez instalado las aplicaciones necesarias y sin cambiar la configuración que venía por defecto, lanzando la aplicación me salió el siguiente error:

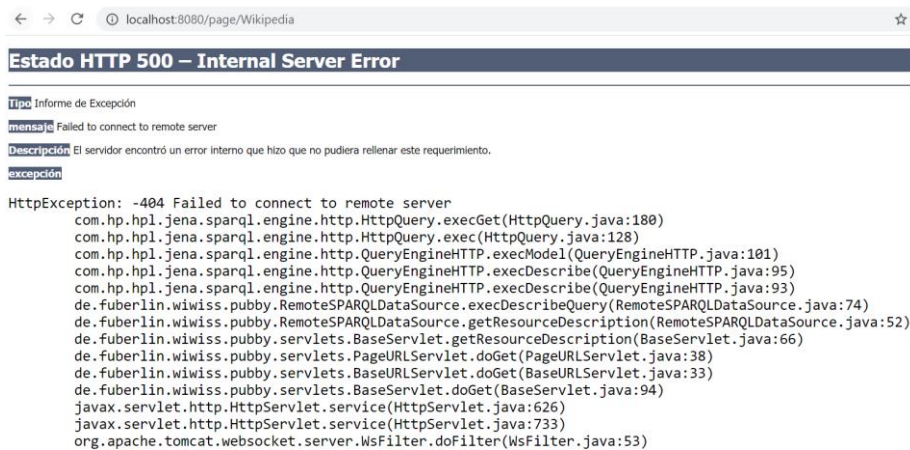


Ilustración 9: Error Pubby por configuración obsoleta

Para verificar que el error se debe por el fichero “config.ttl” decidí reemplazar el contenido por el de la configuración secundaria “config-myfoaf.ttl”. Después de este cambio la aplicación se lanzó correctamente y pude comprobar de esta forma que el error se hallaba en ese fichero. Con la ayuda de mi cotutora pudimos identificar que el endpoint era incorrecto y cambiándolo por el de “dbpedia/sparql” funciono correctamente.

3.2.5.2 Dockerfile

3.2.5.2.1 Version JDK

Replicando los pasos realizadas en la maquina local para el lanzamiento de Pubby y corrigiendo el error anterior con el fichero de configuración obtuve el siguiente error al lanzar el contenedor:

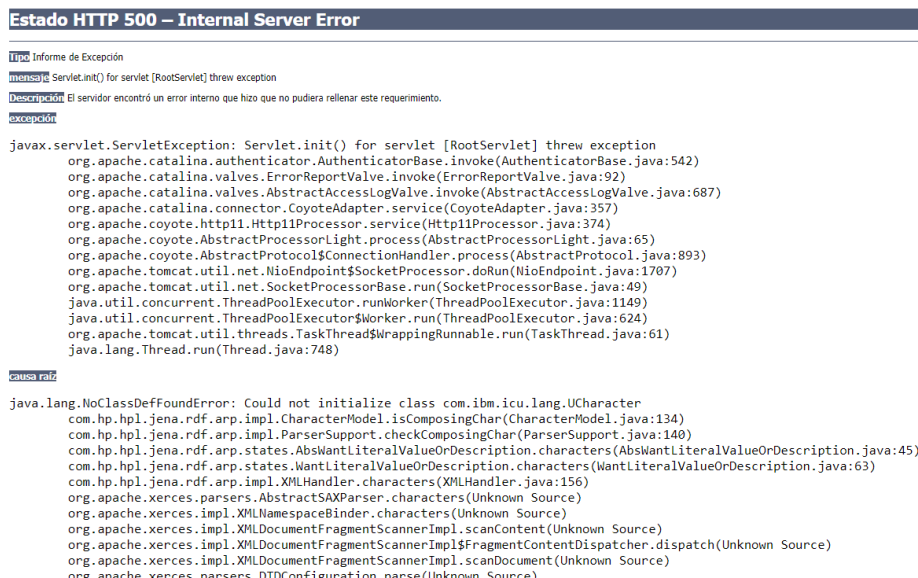


Ilustración 10: Error Dockerfile (version JDK)

Realizando una investigación y comparación con las versiones que tenía en la maquina local pude localizar el error. La solución se hallaba en utilizar una versión JDK superior a la que estaba probando en su momento. Con una versión open-jdk-11 (para sistemas Ubuntu) la aplicación Pubby pudo funcionar correctamente usando el Dockerfile.

3.2.5.2.2 *Copiar ficheros del host al contenedor (síncrono)*

En la primera version de mi Dockerfile, se debía de generar una nueva imagen y contenedor para probar los cambios realizados en el fichero “config.ttl”. Esto era ineficiente ya que para cambios pequeños se consumía demasiados recursos. Para que los cambios se guardaran de forma síncrona mientras el contenedor se está ejecutando se decidió usar volúmenes. La creación del volumen se indica en el comando que nos permite lanzar el contenedor, indicando la ruta del fichero en nuestro host y la ruta donde se almacenara en la imagen.

3.2.6 Propuesta de mejora

Se debería de especificar las versiones JDK y Tomcat con las que se ha verificado que la herramienta funciona correctamente, así como el sistema operativo. Esto permitiría a futuros desarrolladores a conocer el contexto de la aplicación y evitar contratiempos con las versiones.

Propondría que los ficheros de configuración se revisarán cada cierto tiempo para así evitar el error al lanzar la herramienta por primera vez. Esto puede causar confusión a desarrolladores con poca experiencia en el manejo de ficheros .ttl. Además añadiría configuraciones de ejemplo más completas con sus correspondientes comentarios explicativos.

3.2.7 Conclusión

Pubby es una herramienta que permite realizar modificaciones en la configuración de una manera sencilla pudiendo visualizar los cambios de una manera rápida. Esto permite mayor facilidad a los desarrolladores a probar sus diferentes configuraciones y familiarizarse con ellas. Al utilizarse URI no desreferenciables en la mayoría de los conjuntos de datos SPARQL, Pubby es la solución para poder acceder a estos recursos ya que mapea estas URI y las hace desreferenciables. Podremos navegar clicando por todos los recursos, tipos y relaciones de una manera sencilla pudiendo visualizar imágenes de nuestros recursos.

3.3 Basil

3.3.1 Pasos de configuración

Por si solo Basil no tiene incluido una interfaz gráfica. Para poder configurar y ejecutar nuestras APIS debemos de hacerlo mediante un terminal Shell Bash usando el comando `curl` [9].

3.3.1.1 Creación de usuario

Debemos de tener un usuario para crear nuestras APIS. Entonces lo primero que debemos de hacer es crearnos uno. Usaremos un JSON similar al siguiente(ej: user.json):

```
{
  username:myuser,
  password:mypassword,
  password2:mypassword2,
  email:example@example.org
}
```

Subimos nuestro usuario mediante método POST a nuestra ruta `/basil/users` indicando el tipo de archivo subido en “Content-Type”:

```
curl -v -X POST http://localhost:8080/basil/users -d @<user.json> --header
"Content-Type: application/json"
```

3.3.1.2 Creación de API

Nuestra API realiza y guarda el resultado de una consulta SPARQL a un endpoint (ej:DBpedia SPAQL). Para ello debemos de crearnos un fichero (ej: query.sparql) en donde guardemos nuestra consulta y después ejecutar el siguiente comando, indicando las credenciales de nuestro usuario, para generar nuestra API:

```
curl -u <username>:<password> -X PUT "http://localhost:8080/basil" -H "X-
Basil-Endpoint: <endpoint>" -T <query.sparql>
```

La respuesta debe de ser un “Created” (HTTP 201), indicándonos el identificador de nuestra API como se muestra en la siguiente ilustración:

```
$ curl -u jose:mypassword -X PUT "http://localhost:8080/basil" -H "X-Basil-Endpoint: http://dbpedia.org/sparql" -T query6.s
parql
{"message":"Created","location":"http://localhost:8080/basil/ubgc5zn319o8"}
```

Ilustración 11: Creación de API con Basil

Tenemos la posibilidad de tener variables SPARQL parametrizadas en nuestro fichero asignando el valor de la variable en la URI de nuestra petición [14].

```
curl http://localhost:8080/basil/<code_api>/api?nameparam="value"
```

```
$ curl http://localhost:8080/basil/ubgc5zn319o8/api?entity="http://dbpedia.org/resource/Rome"
| Concept | | Type |
|-----|
| <http://dbpedia.org/resource/Rome> | "c" |
| <http://dbpedia.org/resource/Category:Metropolitan_City_of_Rome_Capital> | "c" |
| <http://dbpedia.org/resource/Category:World_Heritage_Sites_in_Italy> | "c" |
| <http://dbpedia.org/resource/Category:New_Testament_cities> | "c" |
| <http://dbpedia.org/resource/Category:Populated_places_established_in_the_8th_century_BC> | "c" |
```

Ilustración 12: Ejemplo de consulta parametrizada Basil

Dentro de nuestro SPARQL podemos establecer diferentes tipos de nuestra variable parametrizada:

- `?_nameparam`: Especifica el nombre del parámetro obligatorio de la API, incorporando el valor como literal simple.

- `?__nameparam`: Indica que el nombre del parámetro es opcional.
- `?_nameparam_iri`: La variable se sustituye por el valor del parámetro como IRI.
- `?_nameparam_integer`: El valor del parámetro se considera literal y el tipo de datos XSD 'entero'.
- `?_nameparam_prefix_datatype`: El valor del parámetro se considera literal y el tipo de datos 'prefijo: tipo de datos'.

Podemos acceder a todas las APIs creadas por nuestro usuario:

```
curl http://localhost:8080/basil/users/<username>/apis
```

Si deseamos eliminar nuestra API:

```
curl -v -u <username>:<password> -X DELETE
"http://localhost:8080/basil/<code_api>"
```

3.3.1.3 Uso y gestión de API

Si realizamos una consulta `curl -v` a nuestra API veremos que obtendremos la siguiente información:

```
$ curl -v http://localhost:8080/basil/lji7aecfg8t9y
* Trying ::1...
* TCP_NODELAY set
* Connected to localhost (::1) port 8080 (#0)
> GET /basil/lji7aecfg8t9y HTTP/1.1
> Host: localhost:8080
> User-Agent: curl/7.55.1
> Accept: */*
>
< HTTP/1.1 303 See Other
< Date: Wed, 05 May 2021 16:27:47 GMT
< X-Basil-API: http://localhost:8080/basil/lji7aecfg8t9y/api
< X-Basil-Alias: http://localhost:8080/basil/lji7aecfg8t9y/alias
< X-Basil-Spec: http://localhost:8080/basil/lji7aecfg8t9y/spec
< X-Basil-Direct: http://localhost:8080/basil/lji7aecfg8t9y/direct
< X-Basil-View: http://localhost:8080/basil/lji7aecfg8t9y/view
< X-Basil-Docs: http://localhost:8080/basil/lji7aecfg8t9y/docs
< X-Basil-Swagger: http://localhost:8080/basil/lji7aecfg8t9y/api-docs
< X-Basil-Creator: jose
< Location: http://localhost:8080/basil/lji7aecfg8t9y/spec
< Content-Length: 0
< Server: Jetty(9.4.35.v20201120)
<
* Connection #0 to host localhost left intact
```

Ilustración 13: Puntos finales de API con Basil

Cada uno de los puntos finales se puede usar para: acceder, crear o modificar las características de la API usando métodos HTTP: PUT (crear), POST (reemplazar), GET (leer), DELETE (eliminar).

3.3.1.3.1 Endpoint /api

Podemos acceder al resultado de la consulta de nuestra API:

```
curl http://localhost:8080/basil/<code_api>/api
```

Podemos cambiar el formato de salida a json, xml o csv especificando el formato deseado en la ruta: `/api.<formato>`

3.3.1.3.2 Endpoint /alias

Acceder a nuestras APIs por el identificador puede ser algo tedioso. Asignar un Alias a nuestras APIs puede facilitarnos la identificación. Para ello primero creamos un .txt (ej: alias.txt) con el nombre que queremos identificar a nuestra

API. Realizamos una petición PUT sobre el endpoint `/alias` con nuestro anterior fichero para crear nuestro identificador.

```
curl -u <username>:<password> -X PUT
http://localhost:8080/basil/<code_api>/alias -T <alias.txt>
```

3.3.1.3.3 Endpoint `/spec`

Podemos obtener nuestro SPARQL de nuestra API:

```
curl "http://localhost:8080/basil/s50nwb9679dd/spec"
```

Si queremos actualizar la consulta de nuestra API:

```
curl -v -u <username>:<password> -X PUT
"http://localhost:8080/basil/<code_api>/spec" -H "X-Basil-Endpoint:
http://dbpedia.org/sparql" -T <query.sparql>
```

3.3.1.3.4 Endpoint `/views`

Podemos definir una representación HTML en los resultados de nuestras consultas de nuestra API. Hace posible adaptar la salida de una API web a aplicaciones que, por ejemplo, quieran incrustar dichos fragmentos en páginas web sin más procesamiento [10]

Lo primero es crear un fichero `.tmpl` (ej: `list.tmpl`) con el formato que queramos darle a nuestra vista. Un ejemplo de formato de salida con el motor de plantillas Mustache [14] que lista nuestros resultados en HTML:

```
<ul>
{{#items}}
  <li><a href="{{Concept}}">{{Concept}}</a> (<small>{{Type}}</small></li>
{{/items}}
</ul>
```

Subimos nuestro `.tmpl` a la ruta `/views/<name_view>` especificando el nombre de nuestra vista, el tipo de formato de salida (`type`) y como procesar la plantilla indicada (`Content-type`):

```
curl -u <username>:<password> -X PUT
"http://localhost:8080/basil/<code_api>/view/<name_view>?type=text/html" -H
"Content-type: template/mustache" -T <list.tmpl>
```

Si queremos obtener todas las vistas de nuestra API:

```
curl http://localhost:8080/basil/<code_api>/view
```

Para eliminar una vista:

```
curl -u <username>:<password> -X DELETE
http://localhost:8080/basil/<code_api>/view/<name_view>
```

3.3.1.3.5 Endpoint `/docs`

Si queremos documentar nuestra API podemos crear un fichero `.txt` (ej: `doc.txt`) con toda la información relevante y subirlo a la ruta `/docs`

```
curl -u <username>:<password> -X PUT
http://localhost:8080/basil/<api_code>/docs -H "X-Basil-Name: Concepts of
entity" -T <doc.txt>
```

Para acceder a la documentación:

```
curl http://localhost:8080/basil/<code_api>/docs
```

3.3.1.3.6 Endpoint /api-docs

Permite acceder a la especificación de la API Swagger. El resultado es una especificación en JSON.

```
curl http://localhost:8080/basil/<code_api>/api-docs
```

3.3.2 Pasos de ejecución

3.3.2.1 Dockerfile

Nuestro Dockerfile instalará en nuestra imagen resultante:

- Imagen base: ubuntu:20.04
- Comandos Ubuntu
 - wget: permite descargar recursos
 - unzip: permite descomprimir los recursos en formato zip
 - nano: permite tener un editor de texto en consola
 - curl: para realizar las interacciones con nuestro localhost:8080
- Maven
- mysql-server
- mysql-connector-java 8.0.24
- JDK-11
- basil-0.8.0

```
Dockerfile > ...
1 FROM ubuntu:20.04
2
3 #FICHEROS PARA REALIZAR LAS PRUEBAS
4 #-Ejemplo 1-
5 COPY ./Ejemplos/Ejemplo1/alias.txt .
6 COPY ./Ejemplos/Ejemplo1/query1.sparql .
7 COPY ./Ejemplos/Ejemplo1/user1.json .
8
9 #-Ejemplo 2-
10 COPY ./Ejemplos/Ejemplo2/query2.sparql .
11 COPY ./Ejemplos/Ejemplo2/user2.json .
12 #-----
13 COPY script.sh .
14 COPY basil.ini .
15 COPY run.sh .
16 COPY pom.xml .
17 RUN apt-get -y update && apt-get -y upgrade
18 RUN apt-get -y install wget openjdk-11-jdk unzip maven
19 RUN apt-get -y install nano curl
20 RUN wget https://github.com/basilapi/basil/archive/refs/tags/v0.8.0.zip -O /tmp/basil.zip
21 RUN cd /tmp && unzip basil.zip
22 RUN cp pom.xml /tmp/basil-0.8.0/server/
23 RUN cp basil.ini /tmp/basil-0.8.0/
24 RUN cd /tmp/basil-0.8.0 && mvn clean install
25 RUN apt-get -y install mysql-server
26 RUN wget https://dev.mysql.com/get/Downloads/Connector-J/mysql-connector-java_8.0.24-1ubuntu18.04_all.deb
27 RUN apt install ./mysql-connector-java_8.0.24-1ubuntu18.04_all.deb
28 RUN export CLASSPATH=/usr/share/java/mysql-connector-java-8.0.24.jar
29 RUN ./script.sh
30 EXPOSE 8080
```

Ilustración 14: Dockerfile (Basil)

Se copian del directorio de Ejemplos los ficheros que se utilizan para realizar los ejemplos del siguiente apartado.

Se descarga del repositorio GitHub del autor [11] los ficheros de la aplicación Basil. El fichero de configuración “basil.ini” se reemplaza por el que se dispone en el directorio actual de trabajo para modificar los parámetros de conexión con la base de datos. De la misma forma, se reemplaza el fichero /server/pom.xml por el “pom.xml” ya que se necesita agregar las dependencias Javax (a partir de java 8 es necesario). Usamos Maven para la construcción del proyecto java a

partir del fichero pom.xml de la raíz del proyecto. Esto nos genera “basil-server-0.8.0.jar”

```
# MySQL Database
ds = com.mysql.jdbc.jdbc2.optional.MysqlDataSource
ds.serverName = localhost
ds.port = 3306
ds.user = root
ds.password = jose
ds.databaseName = basil
ds.useSSL=false
ds.verifyServerCertificate=false
```

Ilustración 14: Fichero basil.ini (Basil)

```
<dependency>
  <groupId>javax.activation</groupId>
  <artifactId>activation</artifactId>
  <version>1.1.1</version>
</dependency>

<dependency>
  <groupId>javax.xml.bind</groupId>
  <artifactId>jaxb-api</artifactId>
  <version>2.3.0</version>
</dependency>

<dependency>
  <groupId>com.sun.xml.bind</groupId>
  <artifactId>jaxb-core</artifactId>
  <version>2.3.0</version>
</dependency>

<dependency>
  <groupId>com.sun.xml.bind</groupId>
  <artifactId>jaxb-impl</artifactId>
  <version>2.3.0</version>
</dependency>
```

Ilustración 15: Fichero pom.xml (Basil)

La aplicación Basil necesita un servidor base de datos para guardar toda la información relacionada con las APIs. Se instala el servidor mysql-server y el conector java 8.0.24 para utilizar la API JDBC. A través del fichero “basil.ini” podemos establecer conexión y así permitir la ejecución de operaciones sobre la base de datos con las aplicaciones java que utiliza Basil.

Se ejecuta un script que inicializa el servidor, crea la base de datos “basil” y las tablas correspondientes.

```
script.sh
1  service mysql start
2  echo "ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY 'jose';">usuario.txt
3  echo "CREATE DATABASE basil;">crear.sql
4  mysql --host=localhost --user=root --password=jose < usuario.txt
5  mysql --host=localhost --user=root --password=jose < crear.sql
6  mysql --host=localhost --user=root --password=jose --database=basil < ./tmp/basil-0.8.0/db.sql
```

Ilustración 15: Fichero script.sh (Basil)

Finalmente se indica que el contenedor se puede conectar a través del puerto 8080.

3.3.2.2 Ejecución

Entendido el funcionamiento de nuestro Dockerfile, procedemos a generar la imagen:

```
docker build -t basil .
```

Esperamos a que el proceso termine. Esto puede demorar unos minutos ya que se debe de descargar e instalar bastante paquetes de datos. Una vez tengamos nuestra imagen debemos de ejecutar el comando:

```
docker run -it -p 8080:8080 basil
```

Esto permite una comunicación entre el contenedor y el host por el puerto 8080. Se nos abrirá un Shell Bash donde deberemos arrancar el servidor mysql (no prestamos atención al warning que aparece):

```
service mysql start
```

Una vez arrancado el servidor ya podremos ejecutar la aplicación Basil con nuestro script (Ilustración 16). Lo ejecutaremos con el parámetro “&” para ponerlo en segundo plano:

```
./run.sh &
```

Este script internamente indica que se desea utilizar nuestro fichero “basil.ini” como Configuración, la salida se especifica en el fichero “log4j2.xml” y que se utiliza la aplicación java “basil-server-0.8.0.jar” expuesto por el puerto 8080.

```
run.sh
1  java -jar -Dbasil.configurationFile=./tmp/basil-0.8.0/basil.ini
2  -Dlog4j.configurationFile=./tmp/basil-0.8.0/server/src/test/resources/log4j2.xml
3  |./tmp/basil-0.8.0/server/target/basil-server-0.8.0.jar -p 8080
```

Ilustración 16: Fichero run.sh (Basil)

```
root@23c843aa6b70:/# service mysql start
* Starting MySQL database server mysqld
su: warning: cannot change directory to /nonexistent: No such file or directory

root@23c843aa6b70:/# ./run.sh &
[1] 294
root@23c843aa6b70:/# #1: welcome to the world's helthiest food
WARNING: sun.reflect.Reflection.getCallerClass is not supported. This will impact performance.
#2: basil is starting on port 8080
#3: resources setup
[INFO ] 2021/5/11:21:13:7:981 BasilApplication.java:61 - Initializing context.
[WARN ] 2021/5/11:21:13:8:622 MySQLStore.java:102 - To Upgrade: 0
[WARN ] 2021/5/11:21:13:8:646 MySQLStore.java:133 - Upgrade completed.
[INFO ] 2021/5/11:21:13:8:652 BasilApplication.java:82 - Setting query executor: class io.github.basilapi.basil.invoke.DirectExecutor
#4: enjoy
^C
root@23c843aa6b70:/# ps -u
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.2  0.0  4232  3472 pts/0    Ss   21:12   0:00 /bin/bash
root      294  0.0  0.0  4232  2052 pts/0    S    21:13   0:00 /bin/bash
root      295  124  1.3 7518392 177020 pts/0    Sl   21:13   0:26 java -jar -Dbasil.configurationFile=./tmp/basil-0.8.0/basil.ini -Dlog4j.c
root      343  0.0  0.0  5896  2880 pts/0    R+   21:13   0:00 ps -u
root@23c843aa6b70:/#
```

Ilustración 17: Ejecución servidor MySQL y aplicación Basil

Una vez que ejecutemos el anterior código, aunque lo hayamos ejecutado en segundo plano, la consola se nos quedará esperando después de la línea “#4: enjoy”. Debemos de obtener el control de la consola con “ctrl+c”. Podemos comprobar con el comando “ps -u” que nuestro proceso sigue funcionando.

Con esto ya tendríamos la aplicación Basil en funcionamiento y lista para usar. Esta versión no dispone de una interfaz gráfica. Para crear y gestionar nuestras APIs se deberá de realizar a través del comando “curl”[9]. Los ficheros que necesitamos lo podemos crear a través del comando “nano nombrefichero.extension” mediante nuestra terminal. En el apartado “3.4.1.Pasos de configuración” podemos encontrar todos los comandos necesarios para crear, consultar, actualizar y eliminar nuestras APIs.

3.3.3 Ejemplos

Una vez lanzada la aplicación podemos probar los siguientes ejemplos lanzando los comandos ,desde el terminal Shell Bash en la ruta raíz, en el orden correspondiente.

Una vez realizados los ejemplos podemos comprobar que los datos se han ido guardado en nuestra base de datos.

1. Debemos de identificarnos con las credenciales de nuestro servidor mysql
mysql --host=localhost --user=root --password=jose

Se nos abrirá un prompt “mysql>” en donde podremos ejecutar sentencias SQL.

2. Observamos las bases de datos existentes.

```
mysql> show databases;
```

3. Seleccionamos la base de datos basil.

```
mysql> use basil;
```

4. Observamos las tablas existentes.

```
mysql> show tables;
```

5. Con esta sentencia podemos seleccionar todo el contenido de la tabla especificada, en este caso de la tabla users.

```
mysql> select * from users;
```

3.3.3.1 Creación de API Películas

El objetivo es construir una API sencilla, pudiendo distinguirla con un alias identificativo, que permita buscar las películas de un determinado director.

1. Creamos nuestro usuario mediante el fichero “user1.json” :

```
{
  username:jose,
  password:contra123,
  email:jose@gmail.com
}
```

```
curl -v -X POST http://localhost:8080/basil/users -d @user1.json --header
"Content-Type: application/json"
```

2. Nuestro fichero “query1.sparql” nos permitirá buscar las películas de director Peter Jackson usando como Endpoint DBpedia. Subimos nuestro fichero identificándonos con nuestro usuario.

```
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbr: <http://dbpedia.org/resource/>
select ?peliculas
WHERE {
  ?peliculas dbo:director dbr:Peter_Jackson .
}
```

```
curl -u jose:contra123 -X PUT "http://localhost:8080/basil" -H "X-Basil-
Endpoint: http://dbpedia.org/sparql" -T query1.sparql
```

3. Se utiliza el fichero alias.txt para proporcionar el nombre identificativo (“películas”) de nuestra API. Registramos este identificativo en el punto final /alias

```
curl -u jose:contra123 -X PUT http://localhost:8080/basil/<code_api>/alias
-T <alias.txt>
```

4. Ahora podremos lanzar una consulta GET a nuestra API para obtener nuestro resultado

```
curl http://localhost:8080/basil/peliculas/api
```



```

[DEBUG] 2021/5/12:18:42:47:944 ApiResource.java:196 - Acceptable media types: [*/*]
[TRACE] 2021/5/12:18:42:47:945 ApiResource.java:150 - API execution. Prepare response.
[TRACE] 2021/5/12:18:42:47:976 ApiResource.java:163 - API execution. Return response.
-----
| películas |
|-----|
| <http://dbpedia.org/resource/Braindead_(film)> |
| <http://dbpedia.org/resource/King_Kong_(2005_film)> |
| <http://dbpedia.org/resource/Production_of_The_Lord_of_the_Rings_film_series> |
| <http://dbpedia.org/resource/Bad_Taste> |
| <http://dbpedia.org/resource/Forgotten_Silver> |
| <http://dbpedia.org/resource/The_Beatles:_Get_Back> |
| <http://dbpedia.org/resource/The_Frighteners> |
| <http://dbpedia.org/resource/The_Hobbit:_An_Unexpected_Journey> |
| <http://dbpedia.org/resource/The_Hobbit:_The_Battle_of_the_Five_Armies> |
| <http://dbpedia.org/resource/The_Hobbit:_The_Desolation_of_Smaug> |
| <http://dbpedia.org/resource/The_Hobbit_(film_series)> |
| <http://dbpedia.org/resource/The_Lord_of_the_Rings:_The_Fellowship_of_the_Ring> |
| <http://dbpedia.org/resource/The_Lord_of_the_Rings:_The_Return_of_the_King> |
| <http://dbpedia.org/resource/The_Lord_of_the_Rings:_The_Two_Towers> |
| <http://dbpedia.org/resource/The_Lord_of_the_Rings_(film_series)> |
| <http://dbpedia.org/resource/The_Lovely_Bones_(film)> |
| <http://dbpedia.org/resource/The_Valley_(1976_film)> |
| <http://dbpedia.org/resource/Crossing_the_Line_(2008_film)> |
| <http://dbpedia.org/resource/Heavenly_Creatures> |
| <http://dbpedia.org/resource/Meet_the_Feebles> |
| <http://dbpedia.org/resource/They_Shall_Not_Grow_Old> |
|-----|

```

Ilustración 18: Ejemplo Basil (API películas)

3.3.3.2 Creación de API parametrizada y con documentación

El objetivo es crear una API que permita tener parámetros en la URI para realizar las consultas y tener una breve documentación.

1. Se debería de tener creado el usuario en la aplicación como en el ejemplo 1. Usaremos las credenciales de dicho usuario.
2. El fichero “query2.sparql” es muy similar al anterior, pero podemos limitar el número de películas en la salida con una variable parametrizada.

```

PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbr: <http://dbpedia.org/resource/>
select ?películas
WHERE {
  ?películas dbo:director dbr:Peter_Jackson .
} LIMIT ?_limitador_integer

```

```

curl -u jose:contra123 -X PUT "http://localhost:8080/basil" -H "X-Basil-Endpoint: http://dbpedia.org/sparql" -T query2.sparql

```

3. Se utiliza el fichero alias2.txt para proporcionar el nombre identificativo (“películas-filtro”) de nuestra API. Registramos este identificativo en el punto final /alias

```

curl -u jose:contra123 -X PUT http://localhost:8080/basil/<code_api>/alias -T <alias2.txt>

```

4. Usamos el fichero “doc.txt” como documentación de nuestra API. Lo subimos al punto final correspondiente /docs

```

curl -u jose:contra123 -X PUT http://localhost:8080/basil/películas-filtro/docs -H "X-Basil-Name: Concepts of entity" -T doc.txt

```

Podemos acceder a la documentación realizando una petición GET a /docs:


```
curl http://localhost:8080/basil/peliculas-filtro/docs
```

5. Ahora podremos lanzar una consulta GET a nuestra API. Limitaremos la salida a 2 películas

```
curl http://localhost:8080/basil/peliculas-filtro/api?limitador=2
```

```
[DEBUG] 2021/5/13:18:31:55:764 ApiResource.java:196 - Acceptable media types: [*/*]
[TRACE] 2021/5/13:18:31:55:768 ApiResource.java:150 - API execution. Prepare response.
[TRACE] 2021/5/13:18:31:55:819 ApiResource.java:163 - API execution. Return response.
-----
| películas |
|-----|
| <http://dbpedia.org/resource/Braindead_(film)> |
| <http://dbpedia.org/resource/King_Kong_(2005_film)> |
|-----|
```

Ilustración 19: Ejemplo Basil (API parametrizada con documentación)

3.3.4 Análisis

3.3.4.1 Documentación

Existe documentación bastante completa en el Wiki del repositorio GitHub Basil [9] para poder realizar todas las funciones CRUD (Create, Read, Update and Delete) con nuestras APIs. En la zona de Code del repositorio nos especifican los pasos a dar para ejecutar nuestra aplicación. Podemos encontrar un pequeño fallo con el último paso ya que no se especifica en que carpeta debemos de situarnos para lanzar el comando. Después de lanzar la aplicación no se especifica que la herramienta no tiene interfaz gráfica y que los procesos hay que realizarlos siguiendo el tutorial “curl” (lo encontramos en la sección Wiki) pudiendo causar confusión al interesado en un primer momento.

3.3.4.2 Mantenibilidad

Podemos observar que los ficheros del repositorio están siendo actualizados por uno de los contribuidores [13]. Basil está pensado para versiones inferiores a Java 1.8 ya que no se incluyen las dependencias Javax. Conversando con el desarrollador se ha comentado que se tendrá en cuenta esto para futuras versiones. Además, se contesta a las dudas planteadas sobre la herramienta, siendo las respuestas bastante rápidas por experiencia propia. Podemos determinar que esta herramienta, al contar con un proyecto activo, satisface el punto de mantenibilidad.

3.3.4.3 Interfaz

Esta herramienta no dispone de por sí sola una interfaz gráfica. Para los usuarios que no cuenten con un sistema operativo Ubuntu les puede ser tedioso ir creando y modificando los diferentes ficheros. Además, los resultados que se presentan por la consola cuesta distinguirlos. Para realizar cambios sencillos puede ser algo laborioso tener que ejecutar líneas largas “curl” pudiendo cometer errores. Existe una version con interfaz [12] con funciones muy limitadas no desarrollada en esta memoria.

3.3.5 Complicaciones

3.3.5.1 Máquina local

3.3.5.1.1 Creación del proyecto con Maven

Una vez descargado los ficheros del repositorio de Basil en GitHub [11] es necesario construir el proyecto Java con Maven. Para ello hay que situarse en la raíz de los ficheros descargados de GitHub donde se encuentra “pom.xml”.

En las instrucciones proporcionadas en este repositorio nos indica que podemos generar el proyecto java ejecutando las pruebas (mvn clean install) o omitiéndolas (mvn install -DskipTests).

En primer lugar se intento con la primera opción. Esto, como se puede observar en la siguiente ilustración, provoco un error. Mirando el resultado de los test, se pensó que el error fue ocasionado por algo interno de las pruebas. Por lo tanto se intentó omitir las pruebas.

```
Tests in error:
  ConfigTest.test:44 » Configuration java.io.FileNotFoundException: C:\Users\usu...

Tests run: 27, Failures: 0, Errors: 1, Skipped: 0

[INFO] -----
[INFO] Reactor Summary for BASIL :: Reactor 0.8.0-SNAPSHOT:
[INFO] BASIL :: Parent ..... SUCCESS [ 15.325 s]
[INFO] BASIL :: Rendering ..... SUCCESS [01:58 min]
[INFO] BASIL :: SPARQL ..... SUCCESS [ 8.999 s]
[INFO] BASIL :: Core ..... FAILURE [01:17 min]
[INFO] BASIL :: Store ..... SKIPPED
[INFO] BASIL :: REST ..... SKIPPED
[INFO] BASIL :: Server ..... SKIPPED
[INFO] BASIL :: Reactor ..... SKIPPED
[INFO] -----
[INFO] BUILD FAILURE
[INFO] -----
[INFO] Total time: 03:40 min
[INFO] Finished at: 2021-04-06T18:23:33+02:00
[INFO] -----
[ERROR] Failed to execute goal org.apache.maven.plugins:maven-surefire-plugin:2.19:test (default-test) on project core: There are test failures.
[ERROR] Please refer to C:\Users\usuario\tecnologia 3\core\target\surefire-reports for the individual test results.
[ERROR] -> [Help 1]
[ERROR] To see the full stack trace of the errors, re-run Maven with the -e switch.
[ERROR] Re-run Maven using the -X switch to enable full debug logging.
[ERROR] For more information about the errors and possible solutions, please read the following articles:
[ERROR] [Help 1] http://cwiki.apache.org/confluence/display/MAVEN/MojoFailureException
[ERROR] After correcting the problems, you can resume the build with the command
[ERROR] mvn <args> -rf :core

C:\Users\usuario\tecnologia 3>
```

Ilustración 20: Error creación proyecto con Maven I (Basil)

En esta ocasión el proyecto fue generado correctamente (ilustración 21). Una vez que ya se tenía el servidor MySQL listo (con la base de datos 'basil', tablas y conector jdbc) y el fichero basil.ini con las parámetros de conexión con la base de datos se intentó lanzar la aplicación. Se ejecuto la instrucción del último paso desde el directorio actual. Esto provoco una serie de errores ya que no se estaba ejecutando desde el directorio correcto y por lo tanto los ficheros que se especificaban en los parámetros eran erróneos (ilustración 22).

```
[INFO] --- maven-install-plugin:2.4:install (default-install) @ basil-server ---
[INFO] Installing C:\Users\usuario\tecnologia 3\server\target\basil-server-0.8.0-SNAPSHOT.jar to C:\path\io\github\basilapi\basil-server\0.8.0-SNAPSHOT\basil-server-0.8.0-SNAPSHOT.jar
[INFO] Installing C:\Users\usuario\tecnologia 3\server\pom.xml to C:\path\io\github\basilapi\basil-server\0.8.0-SNAPSHOT\basil-server-0.8.0-SNAPSHOT.pom
[INFO] -----
[INFO] Building BASIL :: Reactor 0.8.0-SNAPSHOT [8/8]
[INFO] -----
[INFO] --- maven-install-plugin:2.4:install (default-install) @ reactor ---
[INFO] Installing C:\Users\usuario\tecnologia 3\pom.xml to C:\path\io\github\basilapi\reactor\0.8.0-SNAPSHOT\reactor-0.8.0-SNAPSHOT.pom
[INFO] -----
[INFO] Reactor Summary for BASIL :: Reactor 0.8.0-SNAPSHOT:
[INFO] BASIL :: Parent ..... SUCCESS [ 1.012 s]
[INFO] BASIL :: Rendering ..... SUCCESS [ 4.898 s]
[INFO] BASIL :: SPARQL ..... SUCCESS [ 0.768 s]
[INFO] BASIL :: Core ..... SUCCESS [ 3.028 s]
[INFO] BASIL :: Store ..... SUCCESS [ 4.478 s]
[INFO] BASIL :: REST ..... SUCCESS [ 4.525 s]
[INFO] BASIL :: Server ..... SUCCESS [01:12 min]
[INFO] BASIL :: Reactor ..... SUCCESS [ 0.156 s]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 01:33 min
[INFO] Finished at: 2021-04-06T18:29:17+02:00
[INFO] -----

C:\Users\usuario\tecnologia 3>
```

Ilustración 21: Error creación proyecto con Maven II (Basil)

```

C:\Users\usuario\tecnologia 3>java -jar -Obasil.configurationFile=..\basil.ini -Dlog4j.configurationFile=.\server\src\test\resources\log4j2.xml -Dserver\target\basil-server-0.8.0-SNAPSHOT.jar -p 8080
picked up JAVA OPTIONS: -Xms128M
#1: welcome to the world's helthiest food
WARNING: sun.reflect.Reflection.getCallerClass is not supported. This will impact performance.
#2: basil is starting on port 8080
#3: resources setup
[ERROR] 2021/4/6:19:31:23:319 EnvironmentLoader.java:152 - Shiro environment initialization failed
org.apache.shiro.util.InstantiationException: Unable to instantiate class [io.github.basilapi.basil.server.BasilServerEnvironment]
    at org.apache.shiro.util.ClassUtils.newInstance(ClassUtils.java:183) ~[basil-server-0.8.0-SNAPSHOT.jar:?]
    at org.apache.shiro.web.env.EnvironmentLoader.determineWebEnvironment(EnvironmentLoader.java:265) ~[basil-server-0.8.0-SNAPSHOT.jar:?]
    at org.apache.shiro.web.env.EnvironmentLoader.createEnvironment(EnvironmentLoader.java:287) ~[basil-server-0.8.0-SNAPSHOT.jar:?]
    at org.apache.shiro.web.env.EnvironmentLoader.initEnvironment(EnvironmentLoader.java:139) [basil-server-0.8.0-SNAPSHOT.jar:?]
    at org.apache.shiro.web.env.EnvironmentLoaderListener.contextInitialized(EnvironmentLoaderListener.java:58) [basil-server-0.8.0-SNAPSHOT.jar:?]
    at org.eclipse.jetty.server.handler.ContextHandler.callContextInitialized(ContextHandler.java:1868) [basil-server-0.8.0-SNAPSHOT.jar:?]
    at org.eclipse.jetty.servlet.ServletContextHandler.callContextInitialized(ServletContextHandler.java:572) [basil-server-0.8.0-SNAPSHOT.jar:?]
    at org.eclipse.jetty.server.handler.ContextHandler.contextInitialized(ContextHandler.java:997) [basil-server-0.8.0-SNAPSHOT.jar:?]
    at org.eclipse.jetty.servlet.ServletHandler.initialize(ServletHandler.java:746) [basil-server-0.8.0-SNAPSHOT.jar:?]
    at org.eclipse.jetty.servlet.ServletContextHandler.startContext(ServletContextHandler.java:279) [basil-server-0.8.0-SNAPSHOT.jar:?]
    at org.eclipse.jetty.webapp.WebAppContext.startWebapp(WebAppContext.java:1457) [basil-server-0.8.0-SNAPSHOT.jar:?]
    at org.eclipse.jetty.webapp.WebAppContext.startContext(WebAppContext.java:1422) [basil-server-0.8.0-SNAPSHOT.jar:?]
    at org.eclipse.jetty.server.handler.ContextHandler.doStart(ContextHandler.java:911) [basil-server-0.8.0-SNAPSHOT.jar:?]
    at org.eclipse.jetty.servlet.ServletContextHandler.doStart(ServletContextHandler.java:288) [basil-server-0.8.0-SNAPSHOT.jar:?]
    at org.eclipse.jetty.webapp.WebAppContext.doStart(WebAppContext.java:524) [basil-server-0.8.0-SNAPSHOT.jar:?]
    at org.eclipse.jetty.util.component.AbstractLifecycle.start(AbstractLifecycle.java:73) [basil-server-0.8.0-SNAPSHOT.jar:?]
    at org.eclipse.jetty.util.component.ContainerLifecycle.start(ContainerLifecycle.java:169) [basil-server-0.8.0-SNAPSHOT.jar:?]
    at org.eclipse.jetty.server.Server.start(Server.java:423) [basil-server-0.8.0-SNAPSHOT.jar:?]

```

Ilustración 22: Error creación proyecto con Maven III (Basil)

Una vez ubicado el directorio donde se debía de lanzar la instrucción me dio errores relacionados con el fichero “basil-server-0.8.0-SNAPSHOT.jar” que se estaba intentado ejecutar (ilustración 23). Llegados a este punto y tras un razonamiento se decidió volver a generar este .jar con la primera opción pensado que no se había generado correctamente. Volviendo a probar aparece el mismo error que en la ilustración 20. Profundizando en los log que volcaba el terminal, se pudo hallar el origen del error. Esto era debido a que el nombre del directorio “tecnologia 3” no se estaba leyendo correctamente en las pruebas por el espacio. Cambiando el nombre del directorio se pudo solucionar el problema y todas las pruebas pasaron correctamente.

3.3.5.1.2 Lanzamiento de aplicación

Lanzando la aplicación con el fichero .jar generado correctamente y desde la ubicación apropiada dio un error al intentar conectarse con la base de datos (ilustración 24). Revisando el fichero de configuración “basil.ini” se observó que se estaba utilizando el puerto 8889, modificándolo por el 3306 (puerto por defecto de MySQL) se solucionó el error.

```

#1: welcome to the world's helthiest food
WARNING: sun.reflect.Reflection.getCallerClass is not supported. This will impact performance.
#2: basil is starting on port 8080
#3: resources setup
[INFO ] 2021/4/6:20:47:56:294 BasilApplication.java:61 - Initializing context.
[ERROR] 2021/4/6:20:48:0:674 MySQLStore.java:78 - FATAL: Upgrade 0.3 -> 0.4.0 failed
java.io.IOException: com.mysql.jdbc.exceptions.jdbc4.CommunicationsException: Communications link failure

The last packet sent successfully to the server was 0 milliseconds ago. The driver has not received any packets from the server.
    at io.github.basilapi.basil.store.mysql.MySQLStore.migrate_0_3_0_4_0(MySQLStore.java:97) ~[basil-server-0.8.0-SNAPSHOT.jar:?]
    at io.github.basilapi.basil.store.mysql.MySQLStore.<init>(MySQLStore.java:76) [basil-server-0.8.0-SNAPSHOT.jar:?]
    at io.github.basilapi.basil.server.BasilApplication.contextInitialized(BasilApplication.java:75) [basil-server-0.8.0-SNAPSHOT.jar:?]
    at org.eclipse.jetty.server.handler.ContextHandler.callContextInitialized(ContextHandler.java:1068) [basil-server-0.8.0-SNAPSHOT.jar:?]
    at org.eclipse.jetty.servlet.ServletContextHandler.callContextInitialized(ServletContextHandler.java:572) [basil-server-0.8.0-SNAPSHOT.jar:?]
    at org.eclipse.jetty.server.handler.ContextHandler.contextInitialized(ContextHandler.java:997) [basil-server-0.8.0-SNAPSHOT.jar:?]
    at org.eclipse.jetty.servlet.ServletHandler.initialize(ServletHandler.java:746) [basil-server-0.8.0-SNAPSHOT.jar:?]

```

Ilustración 23: Error lanzamiento de aplicación Basil (MySQL)

El siguiente error “java.lang.NoClassDefFoundError. javax/activation/DataSource” es debido a que Basil no incluía el paquete de Javax. Contactando con el desarrollador pude saber que la aplicación Basil fue desarrollado para Java 1.8. No lo incluyeron ya que esta version de Java ya contenía las dependencias Javax. Para versiones superiores de Java era necesario incluir las estas dependencias para su funcionamiento. Para solucionar el error se incluyeron las dependencias que se pueden observar en la ilustración 26.

```

Picked up _JAVA_OPTIONS: -Xmx512M
#1: welcome to the world's healthiest food
WARNING: sun.reflect.Reflection.getCallerClass is not supported. This will impact performance.
#2: basil is starting on port 8080
#3: resources setup
[INFO ] 2021/4/12:18:23:4:621 BasilApplication.java:61 - Initializing context.
[WARN ] 2021/4/12:18:23:5:153 MySQLStore.java:102 - To Upgrade: 0
[WARN ] 2021/4/12:18:23:5:167 MySQLStore.java:133 - Upgrade completed.
[INFO ] 2021/4/12:18:23:5:172 BasilApplication.java:82 - Setting query executor: class io.github.basilapi.basil.invoke.DirectExecutor
abr. 12, 2021 6:23:13 P.M. org.glassfish.jersey.internal.Errors logErrors
WARNING: The following warnings have been detected: WARNING: HK2 service reification failed for [org.glassfish.jersey.message.internal.DataSourceProvider] with an exception:
MultiException stack 1 of 2
java.lang.NoClassDefFoundError: javax/activation/DataSource
    at java.base/java.lang.Class.getDeclaredConstructors0(Native Method)
    at java.base/java.lang.Class.privateGetDeclaredConstructors(Class.java:3296)
    at java.base/java.lang.Class.getDeclaredConstructors(Class.java:2515)
    at org.jvnet.hk2.internal.Utilities$.run(Utilities.java:1352)

```

Ilustración 24: Error lanzamiento de aplicación Basil (Javax)

```

<dependency>
  <groupId>javax.activation</groupId>
  <artifactId>activation</artifactId>
  <version>1.1.1</version>
</dependency>

<dependency>
  <groupId>javax.xml.bind</groupId>
  <artifactId>jaxb-api</artifactId>
  <version>2.3.0</version>
</dependency>

<dependency>
  <groupId>com.sun.xml.bind</groupId>
  <artifactId>jaxb-core</artifactId>
  <version>2.3.0</version>
</dependency>

<dependency>
  <groupId>com.sun.xml.bind</groupId>
  <artifactId>jaxb-impl</artifactId>
  <version>2.3.0</version>
</dependency>

```

Ilustración 25: Solución error Javax

3.3.5.1.3 Salida de errores

Algunos errores, comentados en los anteriores apartados, excedían el tamaño del buffer de cmd de Windows y no se podía observar las primeras líneas que identificaban el error. Esto fue un problema ya que no podía saber el tipo de error. Ampliar el tamaño del buffer del cmd no fue suficiente. Se intentó redirigir los errores a un fichero pero solo me cogía el resultado del proceso (#1welcome.....#2starting...#3done...#4 enjoy), sin los errores, como si todo hubiera ido bien. Supuse que el proceso terminaba, se guardaba en el fichero y después ocurría el error. La solución que halle fue utilizar el terminal Git Bash que ya tenía instalado en mi equipo. De esta forma pude ampliar mucho más el buffer y así ver las primeras líneas donde se podía observar el origen del error.

3.3.5.1.4 Resultado del lanzamiento

Una vez que obtuve lo que se esperaba que el terminal devolviese no sabía si se había lanzado correctamente ya que al entrar en localhost:8080 me aparecía lo siguiente:

Directory: /		
Name ↑	Last Modified	Size
swagger-ui/	16 may. 2021 14:50:50	

Ilustración 26: Navegador lanzamiento Basil

En las instrucciones del repositorio GitHub [11] no se indica que es lo que debería aparecer en localhost:8080. Para saber que pasos a dar y comprobar que lo que había hecho hasta ese momento era correcto contacté con el desarrollador [13]. Me informó que todo estaba bien configurado y que lo que me salía en el navegador del localhost:8080 era lo adecuado. Me indicó que para usar la aplicación siguiera el tutorial de Curl [9] que se encontraba en la Wiki

del repositorio. Mi error por mi parte fue no revisar la Wiki, ya que no estaba muy familiarizado con GitHub en ese entonces.

3.3.5.2 Dockerfile

3.3.5.2.1 Actualización del repositorio

Para ejecutar la aplicación encontré , en el directorio /server, un fichero que se llama “run.sh” que me ejecuta el último comando indicado en las instrucciones para lanzar la aplicación. Me funcionaba correctamente en mi maquina local con la version 0.8.0 SNAPSHOT. Cuando estaba realizando el Dockerfile vi que se había actualizado a la version 0.8.0 y al ejecutar el script “run.sh” del repositorio lanzaba un error ya que el fichero .jar que se indicaba en el comando era distinto al que se generaba con Maven. Para solucionar esto deje de usar el script “run.sh” que viene con el repositorio y me cree otro con el ejecutor java puesto correctamente. La versión que probé en local ya no existía en el repositorio. Informándome comprendí que SNAPSHOT es una version que esta en desarrollo y cuando está finalizado se sube a los releases del repositorio. La version que utilice finalmente fue el reléase 0.8.0.

3.3.5.2.2 Base de datos

Uno de los problemas al utilizar la base de datos en Ubuntu desde un Dockerfile fue tener arrancado el servidor mysql. Para realizar las operaciones sobre la base de datos primero había que iniciar el servidor. Al realizar **RUN service mysql start** y después autenticarse me saltaba el siguiente error:

```
ERROR 2002 (HY000): Can't connect to local MySQL server through socket '/var/run/mysqld/mysqld.sock' (2)
```

Ilustración 27: Error mysql-server

Pude observar que esto se debía a se estaba autenticándose en un servidor que no estaba arrancado. Para solucionar esto, después de varias pruebas, pude realizar las operaciones sobre el servidor metiendo todas las líneas de código relacionados con mysql en un script y ejecutándolo con un **RUN ./script.sh**. Creo que esto se debe a que a la hora de lanzar el comando, se realiza todo esto en un mismo proceso, permitiendo al servidor mantenerse arrancado.

Para establecer la conexión intente usar el comando **apt-get install libmysql-java pensando** y añadir la ruta al CLASSPATH. Al ejecutar el Dockerfile obtuve un fallo de conexión:

```
root@9034a8bb16f5:/tmp/basil-master/server# ./run.sh
#1: welcome to the world's helthiest food
WARNING: sun.reflect.Reflection.getCallerClass is not supported. This will impact performance.
#2: basil is starting on port 8080
#3: resources setup
[INFO ] 2021/5/6:16:55:37:469 BasilApplication.java:61 - Initializing context.
[ERROR] 2021/5/6:16:55:37:821 MySQLStore.java:78 - FATAL: Upgrade 0.3 -> 0.4.0 failed
java.io.IOException: com.mysql.jdbc.exceptions.jdbc4.CommunicationsException: Communications link failure

The last packet sent successfully to the server was 0 milliseconds ago. The driver has not received any packets from the server.
```

Ilustración 28: Error conexión mysql-server desde Dockerfile (Basil)

Tras varios intentos con soluciones usando libmysql-java no se pudo establecer conexión. Para saber que el error era solo por el conector decidí crearme un fichero básico java para probar la conexión:


```

import java.sql.*;
class connect{
    public static void main (String args[]){
        try{
            Connection con=DriverManager.getConnection("jdbc:mysql://localhost:3306","root","jose");
            System.out.println("Sucess...");
            con.close();
        }
        catch(Exception e){
            System.out.println(e);
        }
    }
}

```

Ilustración 29: Test Java conexión mysql-server

El problema se solucionó descargando Connector J (formato .deb) desde el navegador con wget, instalando el fichero y añadiendo al CLASSHPATH.

3.3.5.2.3 Fichero pom.xml del directorio server

El Dockerfile se hizo funcionar correctamente con los ficheros que se habían descargado del repositorio en la máquina local. Posteriormente se pidió que los archivos se debían de descargar del repositorio desde el Dockerfile. Copie el fichero “pom.xml” de la ruta /server de mi máquina local a la imagen, al incluir ya las dependencias javax. Al lanzar la aplicación me volvió a salir un error relacionado con la conexión a la base de datos. No entendía lo que pasaba ya que contenía exactamente con lo que me había funcionado anteriormente. Localice unas dependencias para el conector mysql que no hacían falta ya que ya se había solucionado los problemas de conexión en el Dockerfile. Volví a probar la aplicación y me seguía saliendo el mismo error. Después de un tiempo me di cuenta de que para que los cambios fueran efectivos había que volver a generar el proyecto Java con los cambios realizados. Con esto se solucionó el problema y se pudo ejecutar la aplicación con éxito desde el Dockerfile.

3.3.6 Propuesta de mejora

Se debería de especificar en el repositorio las versiones de java con las que puede funcionar la aplicación. Indicando también las dependencias Javax que se deben de incluir para versiones superiores.

Se necesita incluir una interfaz gráfica a este proyecto ya que es muy tedioso realizar todas las operaciones desde el terminal. Por ejemplo cuando se crea una API y queremos coger el identificador, es necesario tener una precisión con el ratón para coger exactamente el ID. Además, cuando obtenemos los datos en formato JSON no se distinguen cada uno de los campos.

El fichero “run.sh” del directorio server se debería de actualizar con la versión .jar correspondiente de la aplicación. Además, sería muy buena idea especificar en las instrucciones de lanzamiento utilizar este script.

3.3.7 Conclusión

Basil es una herramienta con mucho potencial que permite crear APIS y tener todo el control sobre ellas. Almacenando el contenido en nuestra propia base de datos permite tener un registro del contenido generado y poder acceder de una manera sencilla a los datos. Poder controlar la vista de los datos (/views) nos da la posibilidad de relacionar los datos obtenidos con fragmentos HTML para incrustar dichos fragmentos, por ejemplo en páginas web, sin más procesamiento. Aunque realizar todas las operaciones desde el terminal pueda resultar tedioso, si tenemos apuntado en un bloc de notas los comandos esenciales, modificar nuestro SPARQL o cambiar nuestro endpoint no resulta complicado.

3.4 Puelia

Lamentablemente esta herramienta no ha podido ser lanzada con éxito por errores php que aparecen en la salida de la interfaz. Se ha intentado contactar con el usuario que se encargaba de realizar la mayoría de commits del repositorio[16] para solventar el problema pero no ha habido ninguna respuesta.

En las indicaciones del repositorio Google Code [17] se especifica que se puede utilizar php 5.2.12 o versiones superiores. Empecé intentándolo con la version php 7.4.19 Thread Safe (TS) que, por sus características, es ideal para usarlo en entorno Windows con Apache, además de incluir la librería “php7apache2_4.dll” que permite las conexiones php con Apache. Lo probé con la version Apache 2.4 ya que en las indicaciones no se especificaba una versión en concreto a utilizar.

Se tuvo que realizar una serie de cambios en el fichero de configuración (httpd.conf) para que Apache pudiera trabajar con ficheros php.

Se añadieron las siguientes líneas a la configuración:

```
AddType application/x-httpd-php .php
LoadModule php5_module C:/php/php5apache2_4.dll
AddHandler application/x-httpd-php .php
PHPIniDir "C:/php"
```

También se agregó “index.php” en el modulo que permite distinguir los ficheros que usa el servidor como índice al inicializarlo:

```
<IfModule dir_module>
    DirectoryIndex index.html index.php
</IfModule>
```

Además, siguiendo las indicaciones, se habilitó el mod_rewrite y reconocimiento de ficheros .htaccess. Para ello se descomenta la siguiente línea:

```
LoadModule rewrite_module modules/mod_rewrite.so
```

Y se declara la directiva AllowOverride a all para que cualquier directiva incluida en un fichero .htaccess se tenga en cuenta. Esto se declara en el conjunto <Directory /> para que nuestra declarativa anterior sea válida desde el directorio raíz y sus subdirectorios.

```
<Directory />
    AllowOverride all
    #Require all denied
</Directory>
```

Una vez completada toda la configuración que debería tener el servidor Apache se continuó con la configuración php. Nuestro php debería de tener activadas las extensiones curl, xml y memcache. Para las dos primeras librerías solo era necesario descomentar del fichero php.ini-development las correspondientes extensiones:

```
extension=curl
extension=xmlrpc
```

La librería memcache se tuvo que descargar de la red [18] y añadir la librería al directorio /ext de nuestro php. Además de agregar la extensión en php.ini-development:

```
extension=memcache
```

Para descargar el código hay que hacerlo desde la sección “Source” de Google Code [19] ya que la sección “Downloads” [20] esta caído:

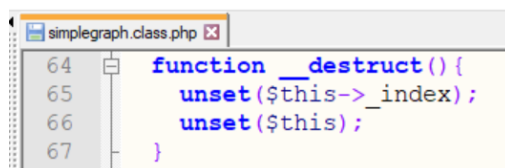
```
401: Anonymous caller does not have storage.objects.get access to the Google Cloud Storage object.
```

Solo hay un link que descarga todo el repositorio. No se puede descargar un release en concreto. Estaba un poco confundido ya que había muchos ficheros y no se especificaba que carpeta de debe de desplegar en Apache. Con ayuda de mi cotutor pude entender la organización de los ficheros y la localización de los releases. Probé “release-2010-06-10” y “release-2010-08-24” de la carpeta “tags”. Los ficheros se desplegaron en la raíz de la carpeta httdocs de Apache (previamente vaciada).

Al lanzar la aplicación el siguiente error:

Fatal error: Cannot unset \$this in C:\Apache24\htdocs\puelia\lib\moriarty\simplegraph.class.php on line 66

Ilustración 30: Error php 7 Puelia



```
64 function __destruct() {
65     unset($this->_index);
66     unset($this);
67 }
```

Ilustración 31: simplegraph.class.php Puelia

Esto error fue debido a la version del php. A partir de php 7 es redundante el uso de `unset($this)` para especificar que es ese el objeto a destruir, ya que método sabe que cuando es invocado es para destruir ese objeto [21]. Eliminando esta línea se pudo solucionar el error aunque se cargó otra página indicándonos que aún nos queda pasos de configuración:

Ilustración 32: Error configuración incompleta Basil

Compruebe con `php -m` que las librerías no estaban incluidas en php. No era suficiente con descomentar las extensiones de `php.ini-development`. Era necesario cambiar el nombre de este fichero por “`php.ini`” y ejecutar `php --ini` para instalar las anteriores librerías. Ejecutando de nuevo la aplicación salió el siguiente error:

Strict Standards: Declaration of PueliaGraph::get_label() should be compatible with SimpleGraph::get_label(\$resource_uri, \$capitalize = false, \$use_qnames = false) in C:\Apache24\htdocs\graphs\pueliagraph.class.php on line 67

Strict Standards: Declaration of ConfigGraph::add_rdf() should be compatible with SimpleGraph::add_rdf(\$rdf = false, \$base = "") in C:\Apache24\htdocs\graphs\configgraph.class.php on line 694

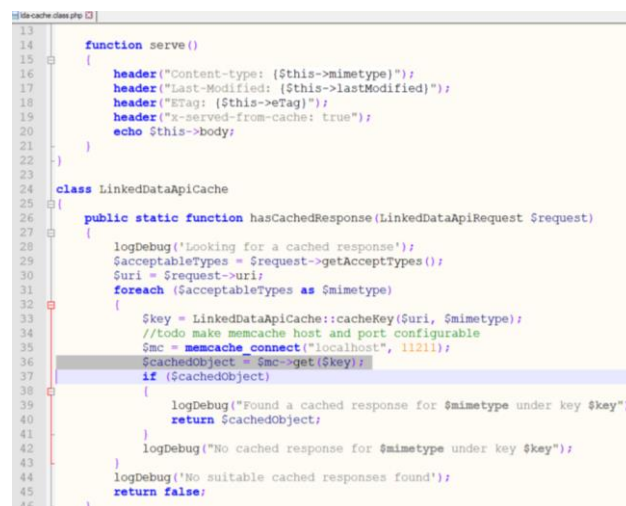
Warning: strftime(): It is not safe to rely on the system's timezone settings. You are *required* to use the date.timezone setting or the date_default_timezone_set() function. In case you used any of those methods and you are still getting this warning, you most likely misspelled the timezone identifier. We selected the timezone 'UTC' for now, but please set date.timezone to select your timezone. in C:\Apache24\htdocs\lib\log4php\src\main\php\layouts\LoggerLayoutTTCC.php on line 179

Notice: memcache_connect(): in C:\Apache24\htdocs\lda-cache.class.php on line 35

Warning: memcache_connect(): in C:\Apache24\htdocs\lda-cache.class.php on line 35

Fatal error: Call to a member function get() on a non-object in C:\Apache24\htdocs\lda-cache.class.php on line 36

Ilustración 33: Error php en lda-cache.class.php (Puelia)



```

13
14
15 function serve()
16 {
17     header("Content-type: {$this->mimetype}");
18     header("Last-Modified: {$this->lastModified}");
19     header("ETag: {$this->eTag}");
20     header("X-served-from-cache: true");
21     echo $this->body;
22 }
23
24 class LinkedDataApiCache
25 {
26     public static function hasCachedResponse(LinkedDataApiRequest $request)
27     {
28         logDebug('Looking for a cached response');
29         $acceptableTypes = $request->getAcceptTypes();
30         $uri = $request->uri;
31         foreach ($acceptableTypes as $mimetype)
32         {
33             $key = LinkedDataApiCache::cacheKey($uri, $mimetype);
34             //todo make memcache host and port configurable
35             $mc = memcache_connect("localhost", 11211);
36             $cachedObject = $mc->get($key);
37             if ($cachedObject)
38             {
39                 logDebug("Found a cached response for $mimetype under key $key");
40                 return $cachedObject;
41             }
42             logDebug("No cached response for $mimetype under key $key");
43         }
44         logDebug('No suitable cached responses found');
45         return false;
46     }

```

Ilustración 34: lda-cache.class.php (Puelia)

Para intentar corregir el error se utilizó el fichero “logs”, del directorio /log, para comprobar los valores que obtenían las variables que daban conflicto. Después de un periodo de investigación y una serie de pruebas no se consiguió solventar el problema. Con el objetivo de que el flujo continuara y ver si había más errores, se simuló la variable `$cachedObject` a true. Al volver a ejecutar la aplicación se obtuvo el siguiente error:

Strict Standards: Declaration of PueliaGraph::get_label() should be compatible with SimpleGraph::get_label(\$resource_uri, \$capitalize = false, \$use_qnames = false) in C:\Apache24\htdocs\graphs\pueliagraph.class.php on line 67

Strict Standards: Declaration of ConfigGraph::add_rdf() should be compatible with SimpleGraph::add_rdf(\$rdf = false, \$base = "") in C:\Apache24\htdocs\graphs\configgraph.class.php on line 694

Warning: strftime(): It is not safe to rely on the system's timezone settings. You are *required* to use the date.timezone setting or the date_default_timezone_set() function. In case you used any of those methods and you are still getting this warning, you most likely misspelled the timezone identifier. We selected the timezone 'UTC' for now, but please set date.timezone to select your timezone. in C:\Apache24\htdocs\lib\log4php\src\main\php\layouts\LoggerLayoutTTC.php on line 179

Notice: memcache_connect(): in C:\Apache24\htdocs\lda-cache.class.php on line 35

Warning: memcache_connect(): in C:\Apache24\htdocs\lda-cache.class.php on line 35

Warning: strftime(): It is not safe to rely on the system's timezone settings. You are *required* to use the date.timezone setting or the date_default_timezone_set() function. In case you used any of those methods and you are still getting this warning, you most likely misspelled the timezone identifier. We selected the timezone 'UTC' for now, but please set date.timezone to select your timezone. in C:\Apache24\htdocs\lib\log4php\src\main\php\layouts\LoggerLayoutTTC.php on line 179

Warning: strftime(): It is not safe to rely on the system's timezone settings. You are *required* to use the date.timezone setting or the date_default_timezone_set() function. In case you used any of those methods and you are still getting this warning, you most likely misspelled the timezone identifier. We selected the timezone 'UTC' for now, but please set date.timezone to select your timezone. in C:\Apache24\htdocs\lib\log4php\src\main\php\layouts\LoggerLayoutTTC.php on line 179

Warning: strftime(): It is not safe to rely on the system's timezone settings. You are *required* to use the date.timezone setting or the date_default_timezone_set() function. In case you used any of those methods and you are still getting this warning, you most likely misspelled the timezone identifier. We selected the timezone 'UTC' for now, but please set date.timezone to select your timezone. in C:\Apache24\htdocs\lib\log4php\src\main\php\layouts\LoggerLayoutTTC.php on line 179

Fatal error: Call to a member function serve() on a non-object in C:\Apache24\htdocs\index.php on line 58

Ilustración 35: Error php en index.php (Puelia)

```
58 $Response->serve();
59 if ($Response->cacheable)
60 {
61     LinkedDataApiCache::cacheResponse($Request, $Response);
62 }
```

Ilustración 36: index.php (Puelia)

Antes de intentar solucionar este fichero se verificó que las configuraciones php y Apache estaban correctamente. Se volvió a comprobar los valores de las variables a través de logs. Tras varias búsquedas para solventar el problema no se consiguió ninguna solución. Ante esta situación, pensando que era debido a la versión php 7, se probó con una version cercana al lanzamiento de los releases(php 5.5.23 TS). Se volvió a instalar las librerías curl, xml y memcache. Desafortunadamente después de repetir todos los pasos anteriores volvió aparecer exactamente el mismo error que se muestra en la Ilustración 33.

En conclusión, a pesar de que se ha seguido todos los pasos expuestos en la documentación de la aplicación se ha producido un error php. La falta de conocimiento manejando este lenguaje fue un gran hándicap. La exploración de este lenguaje en el tiempo disponible para esta herramienta no fue suficiente. Se probó diferentes versiones de php y memcache sin ningún éxito. Ha dificultado mucho el hecho de que esta herramienta no esta siendo mantenida y no hay soporte ante estos problemas. Además, se ha navegado por los asuntos abiertos en la sección “Issues” del Google Code [22] para encontrar posibles soluciones sin éxito. Se debería de especificar en la documentación cómo realizar las configuraciones requeridas en php y Apache, incluyendo las versiones exactas, para así solventar futuros problemas.

3.5 Trellis

Trellis es un [servidor LDP](#) modular que enfatiza tanto la escalabilidad como la conformidad con los [estándares web](#) .

La [arquitectura](#) de Trellis admite la ampliación horizontal de un sistema, tanto para admitir grandes cantidades de datos, como para la redundancia de datos y altas cargas de servidor

[Los recursos se gestionan](#) mediante una API HTTP RESTful, siguiendo la [especificación de Linked Data Platform](#) . Las operaciones de creación, modificación y eliminación se asignan a métodos HTTP estándar que un cliente puede utilizar para interactuar con los recursos del servidor.

Un cliente HTTP también puede recuperar el estado histórico de un recurso en cualquier momento arbitrario. Esto se hace siguiendo los modelos de interacción de la [especificación Memento](#) . Para obtener más información y ejemplos de cómo un cliente interactúa con las versiones de un recurso, consulte el [documento de control de versiones](#) del [recurso](#)

3.5.1 Pasos de configuración

El mecanismo principal para crear, actualizar y eliminar recursos está definido por la especificación [W3C Linked Data Platform](#) (LDP). Para aquellos que no están familiarizados con LDP, hay disponible un [manual introductorio](#) .

En resumen, todos los recursos se pueden administrar con métodos HTTP estándar: GET para recuperar, POST para crear, PATCH para modificar recursos RDF, PUT para reemplazar contenido, DELETE para eliminar contenido. Además HEAD y OPTIONS son muy útiles para determinar las capacidades de un recurso determinado.

Recuperando recursos

```
GET / repositorio / recurso HTTP / 1.1
```

En general, cuando se desea **una** representación RDF, un cliente debe incluir un Accept encabezado en la solicitud:

```
Accept: application/ld+json;q=1.0, text/turtle;q=0.5
```

Representación de recursos

Al usar un **Prefer** encabezado en la solicitud, es posible manipular qué triples incluir o excluir de esa representación.

```
Prefer:
return=representation;
include="http://www.trellisldp.org/ns/trellis#PreferAudit"
```

Creando recursos

Utilizando POST:

Cuando un cliente crea un recurso, se recomienda que la solicitud incluya un Link encabezado que defina el tipo de recurso LDP que se creará junto con el Content-Type de la solicitud:

```
POST /container/ HTTP/1.1
Link: <http://www.w3.org/ns/ldp#RDFSSource>; rel="type"
Content-Type: application/ld+json
Slug: resource
```

Utilizando PUT:

También es posible crear recursos con PUT. Sin embargo, si lo hace, es posible que se produzca un gráfico de recursos desconectado, ya que los contenedores intermedios no se generarán automáticamente. Si un cliente desea evitar esta situación, el cliente debe usar solo POST para la creación de recursos.

```
POST /container/resource HTTP/1.1
Link: <http://www.w3.org/ns/ldp#RDFSSource>; rel="type"
Content-Type: application/ld+json
```

Modificar recursos

Utilizando PATCH

Los recursos RDF se pueden modificar in situ con el PATCH método HTTP . Trellis admite el application/sparql-update tipo de contenido para esto, que es un mecanismo basado en estándares para modificar documentos RDF. Los recursos binarios no permiten PATCH el flujo de bits de recursos, pero sí permiten PATCH modificar la descripción RDF correspondiente.

```
PATCH /container/resource-1 HTTP/1.1
Content-Type: application/sparql-update
If-Match: "aeb02953618b318fde4a1b162f138b77b981c71c"
```

Utilizando PUT:

Al modificar un recurso existente con PUT o PATCH, se recomienda que un cliente use un If-Match encabezado junto con el identificador ETag para el recurso. Nota : el valor ETag proporcionado debe ser un ETag fuerte, incluso si el recurso generalmente responde con un valor ETag débil.

Es posible modificar el tipo de recurso LDP PUT siempre que el nuevo tipo sea un subtipo del recurso existente. Para hacer esto, Link se necesitaría proporcionar un encabezado apropiado . Trellis responderá con un error si el nuevo tipo de LDP no es un subtipo del recurso existente.

```
PUT / container / resource-1 HTTP / 1.1
Tipo de contenido: application / ld + json
If-Match: "aeb02953618b318fde4a1b162f138b77b981c71c"
```

Eliminando recursos

Un recurso se puede eliminar con el DELETE método HTTP . Vale la pena señalar que no se admiten eliminaciones recursivas, por lo que una DELETE operación en un recurso que contiene recursos secundarios dará como resultado la eliminación del recurso de destino,

ELIMINAR / contenedor / recurso-1 HTTP / 1.1

3.5.2 Pasos de ejecución

43

3.5.3 Ejemplos

```
Jose Silva@PORTATIL-99 ~
$ curl http://localhost/
@prefix as: <https://www.w3.org/ns/activitystreams#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix ldp: <http://www.w3.org/ns/ldp#> .
@prefix skos: <http://www.w3.org/2004/02/skos/core#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix acl: <http://www.w3.org/ns/auth/acl#> .
@prefix vcard: <http://www.w3.org/2006/vcard/ns#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix dc: <http://purl.org/dc/terms/> .

<http://localhost/> rdf:type ldp:BasicContainer .

Jose Silva@PORTATIL-99 ~
$ curl --location --request PUT 'http://localhost/foo/' \
> /C

Jose Silva@PORTATIL-99 ~
$ curl --location --request PUT 'http://localhost/foo/' \
> --header 'Link: <http://www.w3.org/ns/ldp#BasicContainer>; rel="type" ' \
> --header 'Content-Type: text/turtle' \
> --data-raw '@prefix dcterms: <http://purl.org/dc/terms/> .
> @prefix ldp: <http://www.w3.org/ns/ldp#> .
>
> <> a ldp:Container, ldp:BasicContainer;
> dcterms:title "Foo Container" ;
> dcterms:description "Container description." .
> '

Jose Silva@PORTATIL-99 ~
$ curl http://localhost/foo/
@prefix as: <https://www.w3.org/ns/activitystreams#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix ldp: <http://www.w3.org/ns/ldp#> .
@prefix skos: <http://www.w3.org/2004/02/skos/core#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix acl: <http://www.w3.org/ns/auth/acl#> .
@prefix vcard: <http://www.w3.org/2006/vcard/ns#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix dc: <http://purl.org/dc/terms/> .

<http://localhost/foo/>
  rdf:type ldp:BasicContainer ;
  dc:title "Foo Container" ;
  dc:description "Container description." .
```

3.5.4 Análisis

3.5.4.1 Documentación

3.5.4.2 Mantenibilidad

3.5.4.3 Interfaz

3.5.5 Complicaciones

3.5.5.1 Máquina local

```
Jose Silva@PORTATIL-99 ~
$ curl --location --request PUT 'http://localhost/prueba/' \
> --header 'Link: <http://www.w3.org/ns/ldp#BasicContainer>; rel="type"' \
> --header 'Content-Type: text/turtle' \
> --data-raw '@prefix dcterms: <http://purl.org/dc/terms/> .
> @prefix ldp: <http://www.w3.org/ns/ldp#> .
>
> <> a ldp:Container, ldp:BasicContainer;
> dcterms:tittle "Foo Container";
> dcterms:description "Container description" .
> '
curl: (3) Port number ended with ' '
```

3.5.5.2 Dockerfile

3.5.6 Propuesta de mejora

3.5.7 Conclusión

3.6 Solid

3.6.1 Pasos de configuración

3.6.2 Pasos de ejecución

3.6.3 Ejemplos

3.6.4 Puntos débiles

3.6.5 Conclusión

3.6.6 Propuesta de mejora

3.7 Ontology2GraphQL

3.7.1 Pasos de configuración

3.7.2 Pasos de ejecución

3.7.3 Ejemplos

3.7.4 Puntos débiles

3.7.5 Conclusión

3.7.6 Propuesta de mejora

4 Resultados y conclusiones

Resumen de resultados obtenidos en el TFG. Y conclusiones personales del estudiante sobre el trabajo realizado.

5 Análisis de Impacto

6 Bibliografía

- [1]: <http://wifo5-03.informatik.uni-mannheim.de/pubby/>
- [2]: <https://code.google.com/archive/p/puelia-php/>
- [3]: <https://github.com/basilapi/basil/wiki/Introduction>
- [4]: <https://github.com/trellis-ldp/trellis/wiki>
- [5]: <https://datos.gob.es/es/blog/pubby-y-lodi-abriendo-los-datos-enlazados-los-humanos>
- [6]: <https://docs.docker.com/get-docker/>
- [7]: <https://hub.docker.com/>
- [8]: <https://code.visualstudio.com/download>
- [9]: <https://github.com/basilapi/basil/wiki/cURL-tutorial>
- [10]: <https://github.com/basilapi/basil/wiki/Views>
- [11]: <https://github.com/basilapi/basil>
- [12]: <https://github.com/basilapi/pesto>
- [13]: <https://github.com/enridaga>
- [14]: <https://github.com/basilapi/basil/wiki/SPARQL-variable-name-convention-for-WEB-API-parameters-mapping>
- [15]: [https://es.wikipedia.org/wiki/Mustache_\(motor_de_plantillas\)](https://es.wikipedia.org/wiki/Mustache_(motor_de_plantillas))
- [16]: <https://code.google.com/archive/p/puelia-php/source/default/commits>
- [17]: <https://code.google.com/archive/p/puelia-php/wikis/GettingStarted.wiki>
- [18]: https://github.com/nono303/PHP-memcache-dll/blob/master/vc15/x64/ts/php-7.4.x_memcache.dll
- [19]: <https://code.google.com/archive/p/puelia-php/source/default/source>
- [20]: <https://code.google.com/archive/p/puelia-php/downloads>
- [21]: <https://es.stackoverflow.com/questions/115553/fatal-error-cannot-unset-this-php-7-1>
- [22]: <https://code.google.com/archive/p/puelia-php/issues>
- [23]: **Tesis Paola Espinoza**

7 Anexos