



Universidad Politécnica  
de Madrid



**Escuela Técnica Superior de  
Ingenieros Informáticos**

Grado en Ingeniería Informática

Trabajo Fin de Grado

**Creación de contenedores Docker sobre  
Construcción de APIs REST a partir de  
Ontologías y Grafos de Conocimientos**

Autor: Jose Elías Silva Manrriquez

Tutor(a): Oscar Corcho García

Madrid, Junio 2021

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

Junio 2021

*Autor:* Jose Elías Silva Manriquez

*Tutor:*

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

# Resumen

Durante los últimos años la generación de datos en la red se ha incrementado. Esto ha traído consigo un gran aumento en el uso de grafos de conocimiento tanto por organizaciones públicas y privadas para el desarrollo de aplicaciones. Actualmente se puede distinguir, por un lado, los ingenieros ontológicos que diseñan estos grafos de conocimientos y, por otro, los desarrolladores de aplicaciones que consumen sus contenidos a través de Interfaces de Programación de Aplicaciones (APIs).

De acuerdo con la tesis de mi cotutora Paola Espinoza [23], quien ha realizado una investigación sobre la situación actual, se determina que hay una brecha entre los desarrolladores de aplicaciones y los ingenieros ontológicos. Según este estudio hay una serie de limitaciones en las especificaciones basadas en las APIs existentes, las tecnologías para el consumo de los grafos de conocimiento, generación automática de las APIs, control del estado actual y pruebas existentes.

El objetivo de este proyecto es proporcionar herramientas que montan estas APIs a través de contenedores Docker. Estos contenedores podrán ser lanzados desde cualquier sistema operativo, lo que permite usar las APIs en cualquier entorno. También se detallará cada uno de los pasos de configuración esenciales para el uso de estas herramientas, así como el estado de estas. Se incluirá ejemplos funcionales que permitan probar la API de una forma sencilla. La finalidad es crear un puente entre los desarrolladores y los ingenieros ontológicos, ahorrando tiempo de investigación y problemas con versiones del entorno.

# Abstract

During the last years the generation of data in the network has increased. This has brought about a great increase in the use of knowledge graphs by both public and private organizations for the development of applications. 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 through Application Programming Interfaces (APIs).

According with my co-tutor Paola's thesis [23], who has carried out research on the current situation, it is determined that there is a gap between application developers and ontological engineers. According to this study, there are a series of limitations in the specifications based on the existing APIs, the technologies for the consumption of the knowledge graphs, automatic generation of the APIs, control of the current state and existing tests.

The objective of this project is to provide tools that mount these APIs through Docker containers. These containers can be launched from any operating system, which allows using the APIs in any environment.

It will detail each of the steps essential configuration using these tools as well as the status of these. It will include functional examples that will test the API in a simple way. The purpose is to create a bridge between developers and ontological engineers, saving research time and problems with versions of the environment.

# Índice de contenido

<b>1</b>	<b>Introducción.....</b>	<b>7</b>
1.1	Motivación.....	7
1.2	Objetivos .....	7
1.3	Estructura de la Memoria.....	8
<b>2</b>	<b>Estado del Arte .....</b>	<b>9</b>
2.1	Introducción.....	9
2.1.1	Grafos de conocimiento .....	9
2.1.2	Web Semántica .....	9
2.1.3	Linked Data y Datos abiertos .....	10
2.2	Tecnologías .....	10
2.2.1	Pubby .....	10
2.2.2	Puelia.....	11
2.2.3	Basil .....	11
2.2.4	Trellis.....	11
2.2.4.1	LDP.....	11
<b>3</b>	<b>Desarrollo .....</b>	<b>13</b>
3.1	Metodología .....	13
3.2	Herramientas utilizadas .....	14
3.2.1	Entorno Docker.....	14
3.2.1.1	Introducción e instalación.....	14
3.2.1.2	Comandos Docker.....	15
3.2.1.3	Comandos Dockerfile .....	16
3.2.2	Comando CURL .....	17
3.2.2.1	Instalación.....	17
3.2.2.2	Opciones.....	17
3.3	Herramientas desarrolladas .....	18
3.3.1	Pubby .....	18
3.3.1.1	Configuración .....	18
3.3.1.2	Pasos de ejecución .....	19
3.3.1.3	Ejemplos.....	20
3.3.1.4	Análisis.....	23
3.3.1.5	Complicaciones.....	24
3.3.1.6	Propuesta de mejora .....	25
3.3.1.7	Conclusión.....	25
3.3.2	Basil .....	26
3.3.2.1	Configuración .....	26
3.3.2.2	Pasos de ejecución .....	29
3.3.2.3	Ejemplos.....	31

3.3.2.4	Análisis.....	34
3.3.2.5	Complicaciones.....	34
3.3.2.6	Propuesta de mejora .....	39
3.3.2.7	Conclusión.....	39
3.3.3	Puelia.....	40
3.3.4	Trellis.....	44
3.3.4.1	Configuración .....	44
3.3.4.2	Pasos de ejecución .....	45
3.3.4.3	Ejemplos.....	47
3.3.4.4	Análisis.....	50
3.3.4.5	Complicaciones.....	51
3.3.4.6	Propuesta de mejora .....	51
3.3.4.7	Conclusión.....	52
<b>4</b>	<b>Conclusiones .....</b>	<b>53</b>
4.1	Resultados .....	53
4.2	Conclusiones personales .....	53
4.3	Líneas futuras .....	54
<b>5</b>	<b>Análisis de Impacto .....</b>	<b>55</b>
<b>6</b>	<b>Bibliografía .....</b>	<b>56</b>

# Índice de ilustraciones

Ilustración 1: Esquema funcionamiento Pubby[1] .....	10
Ilustración 2: Tipos de LDPR.[34] .....	12
Ilustración 3: Aplicación Docker.....	14
Ilustración 4: Extensión Docker Visual Studio Code .....	15
Ilustración 5: Visual Studio Code.....	15
Ilustración 6: Dockerfile Pubby .....	19
Ilustración 7: Ejemplo 1. Endpoint dbpedia (Pubby).....	21
Ilustración 8: Ejemplo 1. Navegación por endpoint dbpedia (Pubby).....	22
Ilustración 9: Ejemplo 2. Carga RDF (Pubby) .....	23
Ilustración 10: Error configuración obsoleta (Pubby).....	24
Ilustración 11: Error version JDK en Dockerfile (Pubby) .....	24
Ilustración 12: Creación de API (Basil) .....	26
Ilustración 13: Ejemplo de consulta parametrizada (Basil) .....	26
Ilustración 14: Puntos finales de API (Basil) .....	27
Ilustración 15: Dockerfile Basil .....	29
Ilustración 16: Fichero script.sh (Basil).....	30
Ilustración 17: Fichero run.sh (Basil) .....	31
Ilustración 18: Ejecución servidor MySQL y aplicación (Basil).....	31
Ilustración 19: Ejemplo 1. API películas (Basil) .....	33
Ilustración 20: Ejemplo 2. API parametrizada con documentación (Basil).....	34
Ilustración 21: Error creación proyecto con Maven I (Basil).....	35
Ilustración 22: Error creación proyecto con Maven II (Basil).....	35
Ilustración 23: Error creación proyecto con Maven III (Basil).....	36
Ilustración 24: Error lanzamiento de aplicación Basil (MySQL) .....	36
Ilustración 25: Error lanzamiento de aplicación por Javax (Basil) .....	36
Ilustración 26: Solución error Javax (Basil).....	37
Ilustración 27: Navegador lanzamiento Basil.....	37
Ilustración 28: Error mysql-server (Basil) .....	38
Ilustración 29: Error conexión mysql-server desde Dockerfile (Basil).....	38
Ilustración 30: Test Java conexión mysql-server .....	38
Ilustración 31: Error php 7 (Puelia).....	41
Ilustración 32: Fichero simplegraph.class.php (Puelia).....	41
Ilustración 33: Error configuración incompleta (Puelia) .....	41
Ilustración 34: Error php en lda-cache.class.php (Puelia) .....	42
Ilustración 35: Fichero lda-cache.class.php (Puelia) .....	42
Ilustración 36: Error php en index.php (Puelia).....	42
Ilustración 37: index.php (Puelia).....	43
Ilustración 38: Docker-compose.yml Trellis.....	46

Ilustración 39: Ejecución exitosa Trellis .....	46
Ilustración 40: Contenedores trellisldap y postgres (Trellis).....	47
Ilustración 41: Petición curl localhost (Trellis) .....	47
Ilustración 42: Ejemplo 1. Resultados (Trellis) .....	48
Ilustración 43. Ejemplo 2 Resultados (Trellis) .....	50
Ilustración 44: Error curl Trellis.....	51
Ilustración 45: Error credenciales Trellis.....	51



# 1 Introducción

## 1.1 Motivación

Vivimos en una era digital donde nuestra sociedad genera y consume una gran cantidad de datos. En la última década esto se ha incrementado sustancialmente y ha permitido incrementar el uso de grafos de conocimiento en el desarrollo de aplicaciones en diferentes áreas. Los grafos de conocimientos no es más que un sencilla forma de representar relaciones entre entidades y que permite establecer vínculos semánticos [45]. Su uso se ha extendido tanto a organizaciones públicas y privadas. Empresas como Google o Microsoft utilizan los grafos de conocimiento para mejorar sus motores de búsqueda [45]. También portales de países, como España o Reino unido lo utilizan para las administraciones públicas [23].

Actualmente se puede distinguir, por un lado, los ingenieros ontológicos que diseñan estos grafos de conocimientos y, por otro, los desarrolladores de aplicaciones que consumen sus contenidos a través de Interfaces de Programación de Aplicaciones (APIs). De acuerdo con la tesis de la cotutora Paola Espinoza [47], quien ha realizado una investigación[23] sobre la situación actual, se determina que hay una brecha entre los desarrolladores de aplicaciones y los ingenieros ontológicos. Esto trae consigo retos a la hora de utilizar las APIs para consumir los grafos de conocimientos, al existir limitaciones con las especificaciones, generación automática de las APIs, estado actual y pruebas existentes .

La tesis recoge una serie de APIs que tienen las limitaciones comentadas anteriormente. Para el desarrollo de este trabajo mis cotutores, Daniel Garijo [46] y Paola Espinoza, seleccionaron Pubby, Puelia, Basil y Trellis como APIs a desarrollar para solucionar esas limitaciones. El objetivo es proporcionar herramientas que montan estas APIs a través de contenedores Docker, permitiendo usarlas desde cualquier entorno. Incluyendo documentación de las configuraciones y ejemplos.

La motivación principal fue el reto de interaccionar con distintos entornos de desarrollo, utilizar APIs en el área de grafos de conocimiento y el uso del entorno Docker, ya que me parecía una herramienta con mucho potencial que nunca había utilizado. Además, antes de involucrarme con el proyecto, asignaturas como Web Semantic y Sistemas Orientados a Servicios me permitieron tener una base sobre grafos de conocimientos, Linked Data y desarrollo de aplicaciones con servicios externos o APIs Java.

## 1.2 Objetivos

### **OB 1. – Investigación de tecnologías Linked Data**

Realizar un estudio sobre las APIs Linked Data propuestas. Principalmente será necesario detallar 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**

Desarrollar contenedores Docker a través de Dockerfiles para poder lanzar las APIs de una manera sencilla y reduciendo el consumo de recursos. Detallar

cada uno de los pasos para poder lanzar las aplicaciones con éxito así como los resultados esperados.

### **OB 3. – Implementación de ejemplos**

Implementar un par de ejemplos para demostrar el funcionamiento de las herramientas. Estos ejemplos tendrán los requerimientos mínimos para que las personas sin conocimiento previo puedan entenderlos.

### **OB 4. - Elaboración de memoria**

Documentar cada uno de los pasos dados, dificultades encontradas y realizar un análisis de la documentación, mantenibilidad , interfaz y una serie de propuestas de mejora para cada herramienta.

## **1.3 Estructura de la Memoria**

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

- **Introducción:** 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:** Se realiza una breve introducción de conceptos básicos para entender el contexto del proyecto y del funcionamiento de las APIs.
- **Desarrollo:** Capítulo con el mayor peso que contiene todo el desarrollo realizado sobre las tecnologías. Se incluye explicación de la metodología seguida para el desarrollo y la explicación de las herramientas utilizadas Docker y Curl. Se explica las diferentes configuraciones que tienen las APIs y como lanzar estas herramientas 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. Cada herramienta desarrollada tiene la siguiente estructura:
  - Configuración
  - Pasos de ejecución
  - Ejemplos
  - Análisis
    - ❖ Documentación
    - ❖ Mantenibilidad
    - ❖ Interfaz
  - Complicaciones
    - Máquina local
    - Dockerfile
  - Propuesta de mejora
  - Conclusión
- **Conclusiones:** Se incluye los resultados que se ha obtenido, conclusiones personales de todo el trabajo realizado y las líneas futuras que podría darse al proyecto.
- **Análisis de Impacto:** Análisis del impacto del trabajo, vinculándolo con los Objetivos de Desarrollo Sostenible
- **Bibliografía:** Referencias externas consultadas durante la investigación

## 2 Estado del Arte

### 2.1 Introducción

Los grafos de conocimiento, Web Semántica, Linked Data y datos abiertos son conceptos necesarios para comprender el funcionamiento de las APIs de las que se habla en este proyecto.

#### 2.1.1 Grafos de conocimiento

Los grafos de datos es una manera de representar la relación entre entidades permitiendo establecer vínculos semánticos entre datos y metadatos (datos acerca de los datos [48]). Se puede recorrer los distintos nodos que se han formado a través de estos vínculos usando lógicas de razonamiento. Permite gestionar la información de una manera ordenada pudiendo clasificar los datos, describir sus propiedades o añadir descripciones semánticas. Con esto se consigue que los grafos de datos se vuelven grafos de conocimiento [45].

#### 2.1.2 Web Semántica

El Consorcio WWW [58], también conocido como World Wide Web Consortium (W3C), genera recomendaciones y estándares que aseguran el crecimiento de la World Wide Web a largo plazo [49]. W3C creó las primeras especificaciones para la Web Semántica, tecnología que facilita la comunicación entre diversas entidades usando modelos bien definidos, con la finalidad de evitar ambigüedades en las comunicaciones. Esto facilita el desarrollo de aplicaciones que utilicen diversas fuentes de datos [50]. Esta tecnología está constituida por 3 bloques: un modelo de datos estándar, un conjunto de vocabularios de referencia y un protocolo estándar de consulta[50].

El modelo de datos o infraestructura para la descripción de recursos (RDF) permite crear grafos de conocimientos globales usando protocolos y lenguajes de la Web. Es posible describir cualquier objeto, ya sea real o abstracto (persona, coche, sentimiento, color...) en múltiples idiomas a través de tripletas con la siguiente estructura: <sujeito> <predicado> <objeto>. Los RDF se localizan a través de identificadores web (URIs) del tipo: <http://... /recurso>. De esta se consigue que los grafos de vuelvan universales ya que se pueden acceder desde cualquier lugar de la red

La Web Semántica necesita un conjunto de vocabularios para facilitar la comunicación de los metadatos [51]. Se utiliza lo que se conoce como ontologías para especificar un concepto dentro de un determinado dominio de interés [54]. Cada vocabulario u ontología se identifica por una URI. Por ejemplo la ontología FOAF que permite describir personas, se accede a través de la URI “http://xmlns.com/foaf/0.1/”. Cada una de las clases y propiedades se pueden acceder concatenando a la URI el nombre de la respectiva clase o propiedad.

SPARQL es el lenguaje de consulta de la Web Semántica[55]. Al igual que SQL permite realizar consultas en bases de datos, SPARQL de la misma forma, con otra sintaxis, permite consultar datos almacenados en los conjuntos de tripletas RDF.

### 2.1.3 Linked Data y Datos abiertos

Linked Data o datos enlazados describe un método de publicación de datos estructurados para que puedan ser interconectados y más útiles. Permite mostrar, intercambiar y conectar datos a través de URI desreferenciables[52] en la Web ( a través de URL).

Apoyándose en la definición que ofrece Opendatahandbook [53], los datos abiertos “son datos que pueden ser utilizados, reutilizados y redistribuidos libremente por cualquier persona, y que se encuentran sujetos, cuando más, al requerimiento de atribución y de compartirse de la misma manera en que aparecen”

Existen varias plataformas de datos enlazados, como puede ser DBpedia [56] (extracción de datos de Wikipedia) o Datos.bne (Biblioteca Nacional de España) [57], que son muy utilizados para la extracción de datos siguiendo las reglas de Web Semantic. No todas las plataformas de datos enlazados son abiertas, ni todas las plataformas abiertas tienen datos enlazados.

## 2.2 Tecnologías

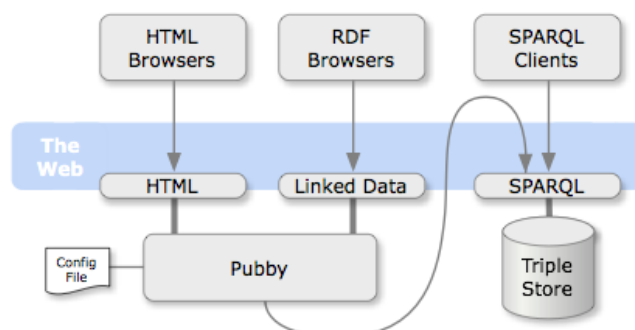
### 2.2.1 Pubby

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[1]*

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.

### **2.2.2 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. 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.

### **2.2.3 Basil**

Basil esta diseñado para montar APIs web sobre puntos finales SPARQL [3]. Realiza el papel de un sistema middleware que permite la comunicación entre puntos finales y sentencias SPARQL almacenadas como recursos. También permite incluir en la URI de la API parámetros para dar valor a las variables dentro de las consultas SPARQL. Finalmente se genera todas las rutas de las APIs para poder realizar operaciones CRUD (crear, recuperar, actualizar y eliminar). Todo esto se realiza organizando los recursos por usuarios, pudiendo tener varios usuarios dentro del servicio.

### **2.2.4 Trellis**

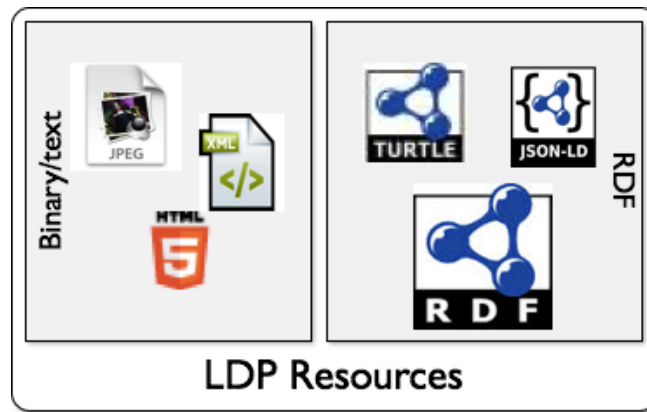
Trellis es un servidor modular que hospeda plataforma de datos vinculados (LDP) cumpliendo con los estándares web [38]. Ideal para proyectos donde hay gran cantidad de datos y altas cargas en el servidor. Su arquitectura permite ampliación horizontal del sistema, proporcionando escalabilidad [4].

Se puede acceder a la API a través de método HTTP siguiendo las especificaciones de LDP [36]:

- POST: crear recurso
- PUT: actualizar recurso (no conjunto de cambios) y crear si no existe
- PATCH: actualizar recurso(conjunto de cambios)
- GET: obtener recurso
- DELETE: eliminar recurso
- OPTIONS: obtener opciones de comunicación existentes de un recurso destino

#### **2.2.4.1 LDP**

Un servidor que hospeda recursos de plataforma de datos vinculados (LDPR) puede administrar dos tipos de LDPR [35]: aquellos recursos cuyo estado se representa mediante un RDF (LDP-RS) y aquellos que utilizan otros formatos no RDF (LDP-NRS) como archivos HTML, imágenes, otros archivos binarios, etc.



*Ilustración 2: Tipos de LDP R.[34]*

Los contenedores (LDPC) son recursos que almacenan otros recursos. Pueden almacenar otros contenedores, RDF o datos binarios. Existen 3 diferentes tipos de contenedores, que presentan características muy similares pero con pequeñas diferencias [36,37].

- Contenedor Básico : Define un enlace simple a recursos de información
- Contenedor Directo: Se agrega el concepto de membresía , permitiendo la flexibilidad de elegir qué forma toman los triples. Se puede especificar dos atributos adicionales, membershipResource y hasMemberRelation, que el servidor LDP añadirá de forma automática a los recursos creados. De esta forma, accediendo a membershipResource se accede a la tripleta creada. Se tendrá como sujeto este último atributo, como predicado hasMemberRelation y como predicado el recurso agregado.
- Contenedor Indirecto: Similar al contenedor directo, también es capaz de tener membresía. La diferencia recae en que se puede especificar el sujeto, el predicado y el objeto del nuevo triple que se genera.

## 3 Desarrollo

### 3.1 Metodología

Para el desarrollo de cada una de las herramientas se ha decidido seguir los siguientes pasos. La primera toma de contacto se realiza desde el sistema nativo, en este caso Windows 10 Pro, en donde se instala cada una de las aplicaciones externas que requiere cada API (Apache, Tomcat, MySQL...). Se asegura que en el sistema nativo la herramienta funcione correctamente, probándolo con configuraciones básicas. Se documenta las dificultades que se hayan podido tener hasta este punto, como puede ser falta de documentación, errores de configuración de las aplicaciones externas o de la misma API. Posteriormente se procede a realizar los correspondientes ficheros Dockerfile (se detalla su significado y uso en el siguiente apartado). Como imagen base para todos los Dockerfile se ha decidido utilizar Ubuntu, versión 20.04, con la finalidad de probar la API en un diferente entorno al nativo. En cada Dockerfile se ha instalado los menores paquetes posibles para poder realizar versiones más ligeras de cada una de las herramientas. Una vez finalizado el Dockerfile, que permite desplegar la API, se ha vuelto a probar con las configuraciones que ha funcionado en el entorno nativo. Se ha documentado cada uno de los errores que haya podido suceder en el proceso de generación del Dockerfile.

Una vez que se ha tenido desarrollada la herramienta de forma completa se ha decidido, con los datos disponibles y con la experiencia misma tras el desarrollo, realizar un breve análisis. En este análisis, para cada una de las APIs, se ha valorado la documentación, mantenibilidad e interfaz. En cuanto a la documentación, se ha especificado los posibles errores o mejoras que presenta la documentación proporcionada por los repositorios oficiales. Para el aspecto de mantenibilidad se ha comprobado si las herramientas presentan actualizaciones recientes, si se ha encontrado alguna versión que presenta error de compatibilidad con las aplicaciones externas (versiones actuales), así como el soporte que existe por parte de su comunidad. El último elemento que se ha evaluado son las ventajas e inconvenientes de la interfaz de la APIs.

Además, se ha finalizado con una serie de propuestas y una conclusión de la herramienta.

Todos los Dockerfile, ficheros de ejemplos, pasos de lanzamiento se ha decidido subir al repositorio GitHub TFG-APIsREST-LinkedData [40]

## 3.2 Herramientas utilizadas

Se especifican las herramientas que se han utilizado durante el desarrollo del proyecto. No se especifican las aplicaciones externas que se han tenido que instalar para cada API (Apache, Tomcat, MySQL...)

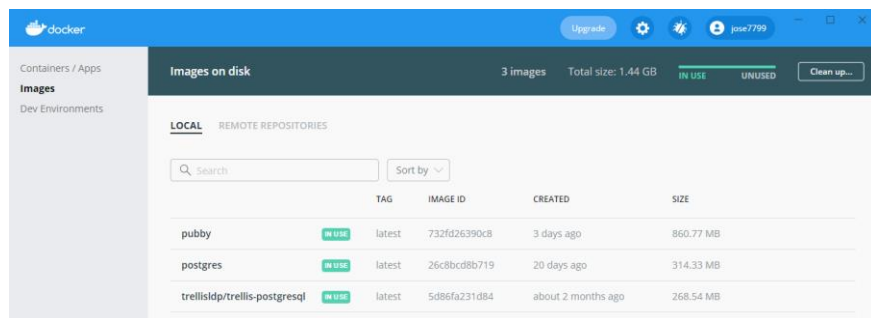
### 3.2.1 Entorno Docker

#### 3.2.1.1 Introducción e instalación

Docker es una herramienta software que nos permite crear, probar e implementar aplicaciones asegurándonos, sea cual sea el entorno donde se requiera usar la aplicación desarrollada, que siempre funcionará. Esto es posible ya que Docker se basa en contenedores que poseen todo lo necesario (núcleo sistema operativo, código, librerías...) para ser ejecutados en cualquier entorno. Estos contenedores se crean apartir de “imágenes”, versiones ligeras del cotenedor, que tienen el mismo contenido. A su vez estas imágenes se generán a partir de ficheros que se conocen como Dockerfiles. Estos ficheros contienen todas las instrucciones necesarias para montar la aplicación, desde el sistema operativo a usar, instalacion de librerías, instalacion de aplicaciones, conexiones externas....Los Dockerfiles pueden usar a su vez imágenes como base para las aplicaciones. Para poder montar las APIs, mencionadas anteriormente, es necesario realizar varios pasos de configuración e instalación de aplicaciones. Usando Dockerfiles se ahorra todos estos pasos y se evita problemas con las versiones de aplicaciones que se tenga instaladas.

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

Una vez instalado la aplicación Docker se tendrá una interfaz similar a la siguiente ilustración, donde se puede observar los contenedores e imágenes que tiene el sistema. Se debe de crear una cuenta en Docker Hub [7] para acceder al repositorio de imágenes, que se utilizan como base en los Dockerfiles.



*Ilustración 3: Aplicación Docker*

Una vez creada la cuenta, se tiene que iniciar sesión para tener acceso de las imágenes del repositorio Docker Hub. Desde un terminal CMD del equipo ya se puede tener acceso a las funcionalidades de Docker para probar los Dockerfile. Por mayor comodidad se recomienda instalar 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 se opta por Visual Studio Code, hay la posibilidad instalar la extensión de Microsoft Docker (Ilustración 4) que permite, dentro del mismo entrono de desarrollo, gestionar cada una de



las imágenes y contenedores. Al igual que en la aplicación Docker, será necesario iniciar sesión para tener acceso al repositorio de imágenes Docker.



Ilustración 4: Extensión Docker Visual Studio Code

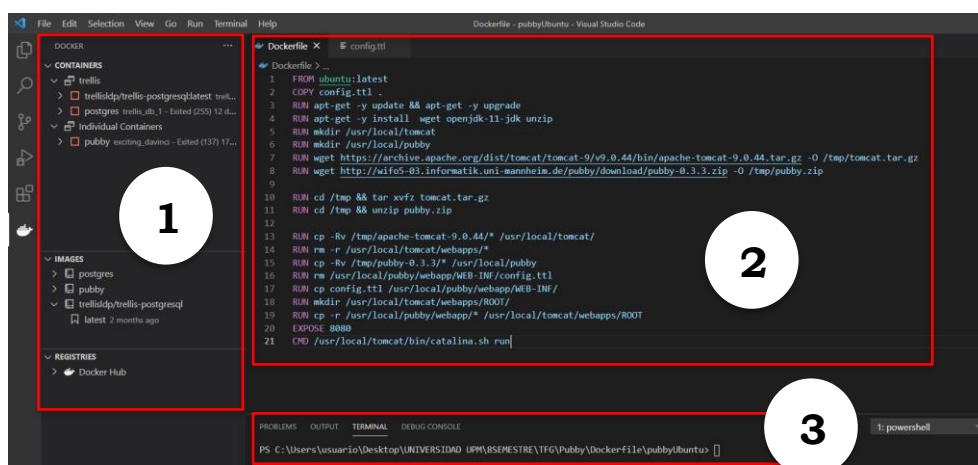


Ilustración 5: Visual Studio Code

Se puede observar en la Ilustración 5 la distribución del espacio del entorno de desarrollo teniendo la extensión Docker.

- Zona 1: Acceso a los contenedores e imágenes creadas. En la opción “Registres” se puede iniciar sesión en Docker Hub.
- Zona 2: Donde se abre los Dockerfile y ficheros de configuración
- Zona 3: Terminal donde se ejecuta los comandos Docker para la creación de imágenes y lanzamientos de contenedores. Para abrir esta zona hay que dirigirse a la barra superior y seleccionar “Terminal”-“New Terminal”.

### 3.2.1.2 Comandos Docker

Los comandos Docker que se deberán utilizar para probar las herramientas:

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

Crea la imagen a partir del Dockerfile, sustituyendo “nombreimagen” por el nombre que se quiera dar a la 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 realiza la búsqueda del Dockerfile en el directorio actual de trabajo.

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

Genera el contenedor a partir de la imagen “nombreimagen”. Se indica que el proceso se ejecute en segundo plano(-d) y creando un mapeo entre el puerto

8080 del host y el puerto 8080 del contenedor. Sin el mapeo de puertos, no se podría acceder a la aplicación desde el Host. En ocasiones se necesita cambiar la ejecución en segundo plano por una sesión interactiva con el contenedor utilizando “-d” por “-it”

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

Permite añadir un volumen para tener cambios actualizables entre una carpeta del directorio actual de trabajo y un directorio de la imagen.

```
docker-compose up
```

Inicia todos los servicios de un fichero docker-compose.yml. Este fichero contiene las instrucciones para lanzar aplicaciones Docker de varios contenedores.

### 3.2.1.3 Comandos Dockerfile

Las instrucciones más frecuentes que se utilizan para los Dockerfiles:

```
FROM imagen
```

Permite coger como base una imagen del repositorio de Docker Hub.

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

Permite copiar el contenido de un directorio o de un fichero del equipo Host al directorio que se especifique dentro de la imagen base.

```
RUN comando
```

Ejecuta los comandos que se lanzan 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

### 3.2.2 Comando CURL

Curl es una herramienta de línea de comandos que permite realizar solicitudes HTTP. Permite probar las APIs sin la necesidad de tener una aplicación web montada.

#### 3.2.2.1 Instalación

Se ha utilizado el terminal de Git Bash durante el desarrollo del proyecto para realizar las interacciones con las APIs. Se puede utilizar otros terminales instalando los correspondientes paquetes de Curl. Para instalar Git Bash:

1. Visitar la página : <https://git-scm.com/downloads>
2. Elegir el sistema en la lista e instalar la configuración descargada
3. Registrar la ruta del directorio instalable en las variables de entorno.
4. Abrir Git Bash

#### 3.2.2.2 Opciones

Las opciones [24,25] más destacadas para utilizar la herramienta curl con las APIs son:

- **-X, --request:** especificar el método de solicitud cuando se comunica con el servidor HTTP (POST, PUT, GET, DELETE...).
- **-I, --head :** recuperar solo los encabezados.
- **-i, --include:** incluir los encabezados de respuesta HTTP en la salida.
- **-H:** Especifica cualquier contenido de encabezado adicional para incluir en la solicitud HTTP, generalmente incluyen el tipo de contenido de los datos. Ejemplo: `-H "Content-Type: application/json"`. También se puede especificar el directorio donde se realiza la operación con "Slug: directorio".
- **-s, --silent :** indica que no muestre información de progreso o error.
- **-v, --verbose:** lo contrario de `--silent` .
- **-d:** permite introducir datos.
- **--data-raw:** publica datos de manera similar a `-d`, pero sin la interpretación especial del carácter `@`.

## 3.3 Herramientas desarrolladas

### 3.3.1 Pubby

#### 3.3.1.1 Configuración

En el archivo config.ttl se determina la configuración de la aplicación Pubby.

En primer lugar se debe de especificar los prefijos que se va a emplear. 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 se necesita 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 este 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 se debe 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 >;
];
```

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

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

Con los anteriores pasos ya se tendría una configuración sencilla lista para utilizar. Las siguientes configuraciones que se describen se consideran de bastante utilidad. Se puede encontrar todas las configuraciones disponibles en la página oficial de Pubby [1]

Si se quiere establecer una página de inicio se debe 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 está 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 se quiere utilizar las declaraciones de prefijo de un documento RDF, en la salida, se puede utilizar la configuración “usePrefixFrom” para vincularlo. Si por el contrario se quiere utilizar los prefijos del fichero config.ttl lo se debe de dejar 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 >;
```

Para que declaraciones owl:sameAs de la forma <web\_uri> owl: sameAs <dataset\_uri> esté presente en la salida de los datos vinculados, se debe de poner “true” en la configuración “addSameAsStatements”:

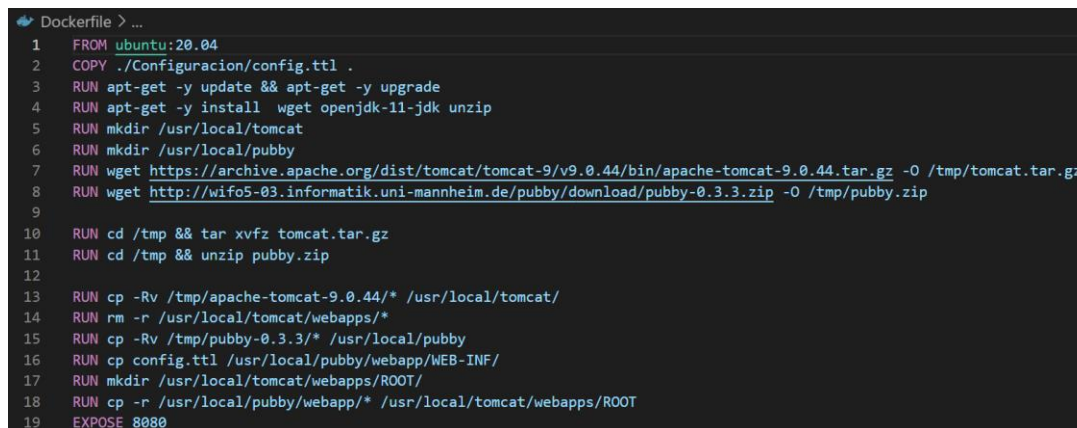
```
conf: addSameAsStatements "true"/"false" ;
```

### 3.3.1.2 Pasos de ejecución

#### 3.3.1.2.1 Dockerfile

Dockerfile instalará los siguientes recursos:

- Imagen base: Ubuntu: 20.04
- 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 6: Dockerfile Pubby*

La aplicación Tomcat despliega las aplicaciones que se encuentren dentro de la carpeta “webapp”. Se eliminan las que vienen por defecto para que la 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 la aplicación Pubby. Pero antes se sustituye la configuración “config.ttl”, que viene por defecto en la ruta “pubby/webapp/WEB-INF”, por el fichero de configuración del directorio

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

### 3.3.1.2.2 Ejecución

Entendido el funcionamiento del Dockerfile, se procede a generar la imagen:

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

Se espera a que el proceso termine. Esto puede demorar unos minutos ya que se debe de descargar e instalar los paquetes de datos. Una vez generada la imagen se debe 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 la carpeta “Configuración” en la ruta “usr/tmp” de la imagen. Se debe de ejecutar de forma iterativa “-t” ya que es necesario ejecutar comandos dentro del contenedor. Una vez lanzado el anterior comando se abrirá un Shell Bash donde se debe de poner el siguiente comando para lanzar el servlet Tomcat:

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

Con esto ya se tendría la aplicación Pubby en funcionamiento. Se puede observar el resultado en la ruta <http://localhost:8080/>

Si se quiere cambiar la configuración bastaría con modificar el fichero “config.ttl” del directorio de trabajo en Host y realizar los siguientes pasos:

- Modificar el fichero config.ttl del Host que se encuentra en el directorio “/Configuracion”
- Parar la ejecución de Catalina (Tomcat) con ctrl+C en el 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
```

### 3.3.1.3 Ejemplos

Tras lanzar la aplicación Pubby se puede realizar los siguientes ejemplos cambiando el contenido del fichero config.ttl, que se tiene en el directorio actual de trabajo, por las configuraciones “configEj1.ttl” o “configEj2” de la carpeta del proyecto [40] correspondientes al ejemplo 1 y 2

#### 3.3.1.3.1 Configuración con endpoint de Wikipedia

En el siguiente ejemplo se puede observar que se utiliza como endpoint DBpedia y se busca 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/> .
```

*Ilustración 7: Ejemplo 1. Endpoint dbpedia (Pubby)*



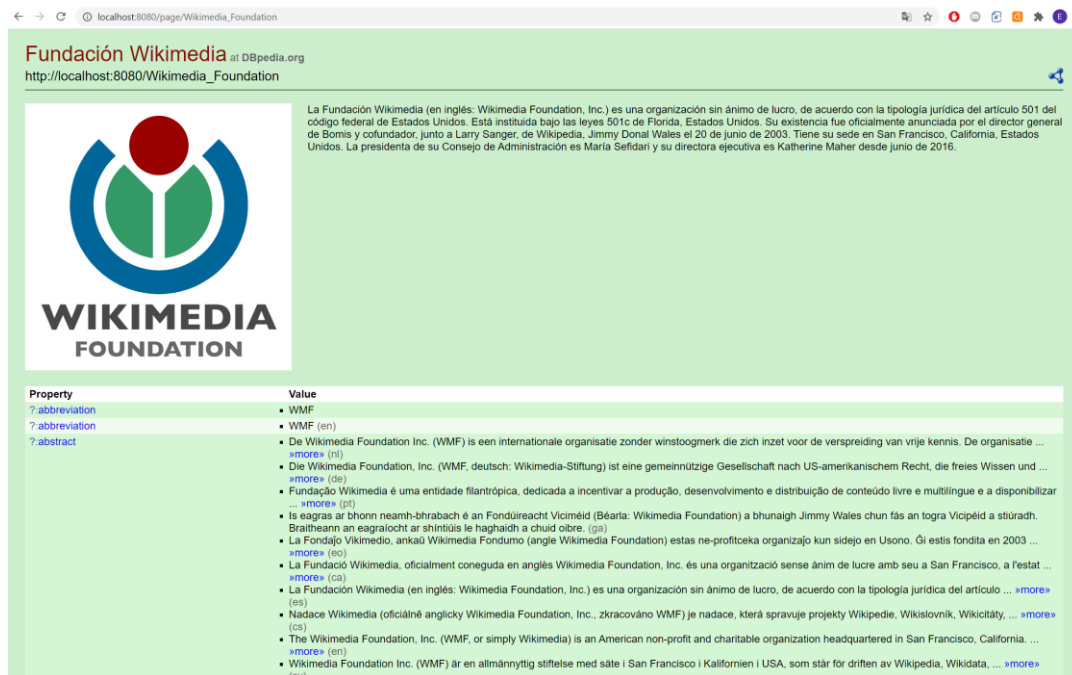


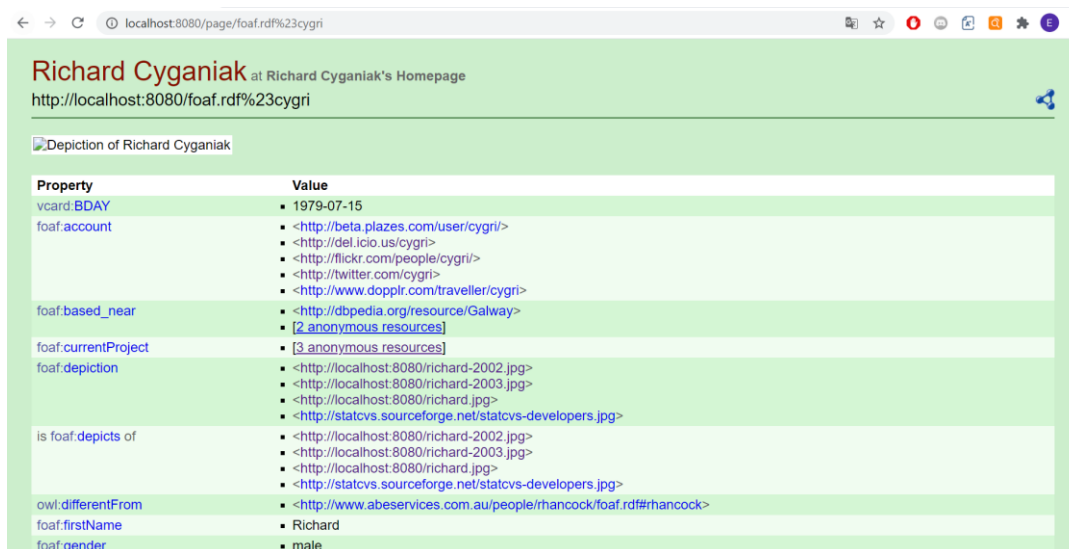
Ilustración 8: Ejemplo 1. Navegación por endpoint dbpedia (Pubby)

### 3.3.1.3.2 Configuración cargando un RDF

En este ejemplo se puede observar cómo se carga 2 ficheros RDF, de propiedad de Rachar 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>;
    .
```





Richard Cyganiak at Richard Cyganiak's Homepage  
http://localhost:8080/foaf.rdf%23cygri

Depiction of Richard Cyganiak

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

*Ilustración 9: Ejemplo 2. Carga RDF (Pubby)*

### 3.3.1.4 Análisis

#### 3.3.1.4.1 Documentación

La documentación proporcionada en la página oficial de Pubby [1] solo muestra la nomenclatura que debe tener la configuración. No se encuentra ejemplos funcionales, más allá del fichero de configuración secundario “conf-myfoaf.ttl”, y documentación más detallada para poder construir el fichero de configuración correctamente. Las instrucciones que se encuentran sobre la descarga e instalación están bastante completas.

#### 3.3.1.4.2 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. La última participación de la comunidad se remonta a 2016. Al no existir actualizaciones reciente y muy poca interacción de la comunidad en la actualidad se puede determinar que no cumple con el aspecto de mantenibilidad.

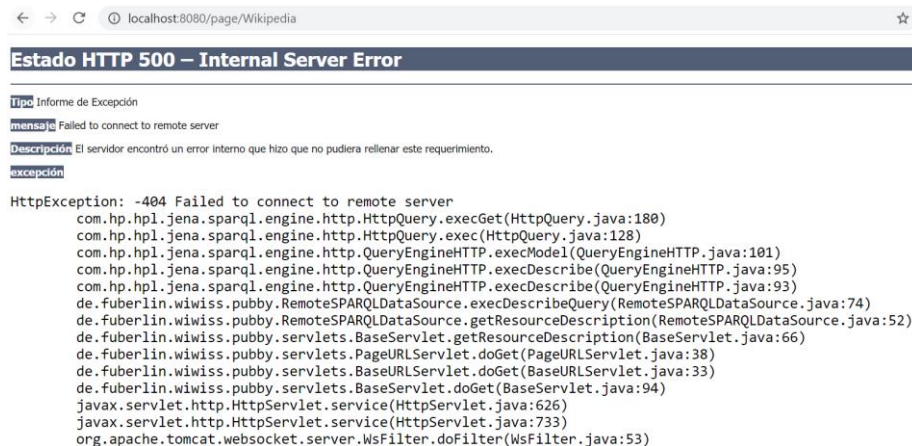
#### 3.3.1.4.3 Interfaz

Se puede observar que esta herramienta presenta un interfaz bastante amigable e intuitiva que permite interaccionar con los recursos clicando directamente sobre el enlace. Además, permite recuperar imágenes de los recursos a los que se accede lo que mejora la estética de la página.

### 3.3.1.5 Complicaciones

#### 3.3.1.5.1 Maquina local

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



```
Estado HTTP 500 – Internal Server Error

Tipo Informe de Excepción
Mensaje Failed to connect to remote server
Descripción El servidor encontró un error interno que hizo que no pudiera rellenar este requerimiento.
Excepción
HttpException: -404 Failed to connect to remote server
com.hp.hpl.jena.sparql.engine.http.HttpQuery.execGet(HttpQuery.java:180)
com.hp.hpl.jena.sparql.engine.http.HttpQuery.exec(HttpQuery.java:128)
com.hp.hpl.jena.sparql.engine.http.QueryEngineHTTP.execModel(QueryEngineHTTP.java:101)
com.hp.hpl.jena.sparql.engine.http.QueryEngineHTTP.execDescribe(QueryEngineHTTP.java:95)
com.hp.hpl.jena.sparql.engine.http.QueryEngineHTTP.execDescribe(QueryEngineHTTP.java:93)
de.fuberlin.wiwi.pubby.RemoteSPARQLDataSource.execDescribeQuery(RemoteSPARQLDataSource.java:74)
de.fuberlin.wiwi.pubby.RemoteSPARQLDataSource.getResourceDescription(RemoteSPARQLDataSource.java:52)
de.fuberlin.wiwi.pubby.servlets.BaseServlet.getResourceDescription(BaseServlet.java:66)
de.fuberlin.wiwi.pubby.servlets.PageURLServlet.doGet(PageURLServlet.java:38)
de.fuberlin.wiwi.pubby.servlets.BaseURLServlet.doGet(BaseURLServlet.java:33)
de.fuberlin.wiwi.pubby.servlets.BaseServlet.doGet(BaseServlet.java:94)
javax.servlet.http.HttpServlet.service(HttpServlet.java:626)
javax.servlet.http.HttpServlet.service(HttpServlet.java:733)
org.apache.tomcat.websocket.server.WsFilter.doFilter(WsFilter.java:53)
```

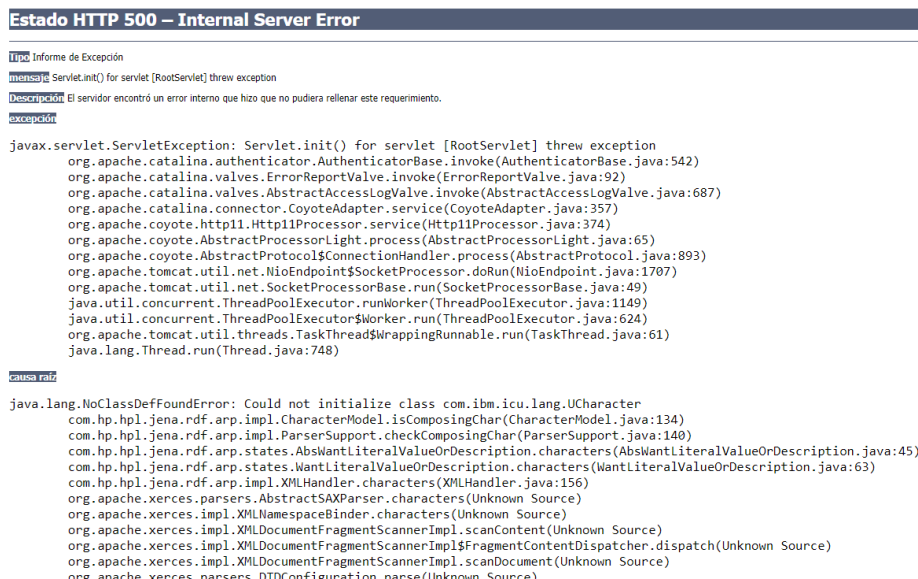
Ilustración 10: Error configuración obsoleta (Pubby)

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

#### 3.3.1.5.2 Dockerfile

##### 3.3.1.5.2.1 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 se obtuvo el siguiente error al lanzar el contenedor:



```
Estado HTTP 500 – Internal Server Error

Tipo Informe de Excepción
Mensaje Servlet.init() for servlet [RootServlet] threw exception
Descripción El servidor encontró un error interno que hizo que no pudiera rellenar este requerimiento.
Excepción
javax.servlet.ServletException: Servlet.init() for servlet [RootServlet] threw exception
org.apache.catalina.authenticator.AuthenticatorBase.invoke(AuthenticatorBase.java:542)
org.apache.catalina.valves.ErrorReportValve.invoke(ErrorReportValve.java:92)
org.apache.catalina.valves.AbstractAccessLogValve.invoke(AbstractAccessLogValve.java:687)
org.apache.catalina.connector.CoyoteAdapter.service(CoyoteAdapter.java:357)
org.apache.coyote.http11.Http11Processor.service(Http11Processor.java:374)
org.apache.coyote.AbstractProcessorLight.process(AbstractProcessorLight.java:65)
org.apache.coyote.AbstractProtocol$ConnectionHandler.process(AbstractProtocol.java:893)
org.apache.tomcat.util.net.NioEndpoint$SocketProcessor.doRun(NioEndpoint.java:1707)
org.apache.tomcat.util.net.SocketProcessorBase.run(SocketProcessorBase.java:49)
java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)
java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)
org.apache.tomcat.util.threads.TaskThread$WrappingRunnable.run(TaskThread.java:61)
java.lang.Thread.run(Thread.java:748)

Causa raíz
java.lang.NoClassDefFoundError: Could not initialize class com.ibm.icu.lang.UCharacter
com.hp.hpl.jena.rdf.arp.impl.CharacterModel.isComposingChar(CharacterModel.java:134)
com.hp.hpl.jena.rdf.arp.impl.ParserSupport.checkComposingChar(ParserSupport.java:140)
com.hp.hpl.jena.rdf.arp.states.AbsWantLiteralValueOrDescription.characters(AbsWantLiteralValueOrDescription.java:45)
com.hp.hpl.jena.rdf.arp.states.WantLiteralValueOrDescription.characters(WantLiteralValueOrDescription.java:63)
com.hp.hpl.jena.rdf.arp.impl.XMLHandler.characters(XMLHandler.java:156)
org.apache.xerces.parsers.AbstractSAXParser.characters(Unknown Source)
org.apache.xerces.impl.XMLNamespaceBinder.characters(Unknown Source)
org.apache.xerces.impl.XMLDocumentFragmentScannerImpl.scanContent(Unknown Source)
org.apache.xerces.impl.XMLDocumentFragmentScannerImpl$FragmentContentDispatcher.dispatch(Unknown Source)
org.apache.xerces.impl.XMLDocumentFragmentScannerImpl.scanDocument(Unknown Source)
org.apache.xerces.parsers.DTDConfiguration.parse(Unknown Source)
```

Ilustración 11: Error version JDK en Dockerfile (Pubby)

Realizando una investigación y comparación con las versiones que se tenía en la maquina local se pudo localizar el error. La solución se hallaba en utilizar

una versión JDK superior a la que se estaba probando en su momento. Con una versión open-jdk-11 (para sistemas Ubuntu) la aplicación Pubby pudo funcionar correctamente usando el Dockerfile.

#### 3.3.1.5.2.2 Copiar ficheros del host al contenedor (síncrono)

En la primera versión del 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 guarden de forma síncrona, mientras el contenedor se está ejecutando, se decidió usar volúmenes. La creación del volumen se indica en el comando que permite lanzar el contenedor, indicando la ruta del fichero en el Host y la ruta donde se almacena en la imagen.

#### 3.3.1.6 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 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.

#### 3.3.1.7 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 desreferenciados 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 desreferenciados. Se puede navegar clicando por todos los recursos, tipos y relaciones de una manera sencilla pudiendo visualizar las imágenes de los recursos.

### 3.3.2 Basil

#### 3.3.2.1 Configuración

Por si solo Basil no tiene incluido una interfaz gráfica. Para poder configurar y ejecutar las APIs se debe de realizar mediante un terminal Shell Bash usando el comando curl [9], explicado en los primeros apartados de esta memoria.

##### 3.3.2.1.1 Creación de usuario

Se debe de tener un usuario para crear APIs. Lo primero que se debe de hacer es crear un usuario. Se debe de utilizar un JSON similar al siguiente(ej: user.json):

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

Se sube el usuario mediante método POST a la 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"
```

##### 3.3.2.1.2 Creación de API

La API realiza y guarda el resultado de una consulta SPARQL a un endpoint (ej:DBpedia SPAQL). Para ello es necesario tener un fichero (ej: query.sparql) en donde guardar las consultas SPARQL y después ejecutar el siguiente comando, indicando las credenciales del usuario, para generar la 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), indicándo el identificador de la 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 12: Creación de API (Basil)*

Se tiene la posibilidad de tener variables SPARQL parametrizadas asignando el valor de la variable en la URI de la 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 13: Ejemplo de consulta parametrizada (Basil)*

Dentro del SPARQL se puede establecer diferentes tipos de variables parametrizadas:

- `?_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'.

Se puede acceder a todas las APIs creadas por un determinado usuario:

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

Para eliminar una determinada API:

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

### 3.3.2.1.3 *Uso y gestión de API*

Realizando una consulta `curl -v` a una API se obtiene la siguiente información:

```
$ curl -v http://localhost:8080/basil/1ji7aecfg8t9y
* Trying ::1...
* TCP_NODELAY set
* Connected to localhost (::1) port 8080 (#0)
> GET /basil/1ji7aecfg8t9y 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/1ji7aecfg8t9y/api
< X-Basil-Alias: http://localhost:8080/basil/1ji7aecfg8t9y/alias
< X-Basil-Spec: http://localhost:8080/basil/1ji7aecfg8t9y/spec
< X-Basil-Direct: http://localhost:8080/basil/1ji7aecfg8t9y/direct
< X-Basil-View: http://localhost:8080/basil/1ji7aecfg8t9y/view
< X-Basil-Docs: http://localhost:8080/basil/1ji7aecfg8t9y/docs
< X-Basil-Swagger: http://localhost:8080/basil/1ji7aecfg8t9y/api-docs
< X-Basil-Creator: jose
< Location: http://localhost:8080/basil/1ji7aecfg8t9y/spec
< Content-Length: 0
< Server: Jetty(9.4.35.v20201120)
* Connection #0 to host localhost left intact
```

*Ilustración 14: Puntos finales de API (Basil)*

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

### **Endpoint /api**

Para acceder al resultado de una API:

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

Se puede cambiar el formato de salida a JSON, XML o CSV especificando el formato deseado en la ruta: `/api.<formato>`

### **Endpoint /alias**

Acceder a las APIs por el identificador puede ser algo tedioso. Asignarles un Alias facilita su identificación. Se debe de tener un fichero .txt con el nombre que con el que se quiera identificar a la API. Se realiza una petición PUT sobre el endpoint `/alias` con el anterior fichero para crear el identificador.

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

### Endpoint /spec

Se puede obtener el SPARQL de la API:

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

Para actualizar la consulta de la 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>
```

### Endpoint /views

Se puede definir una representación HTML de los resultados que genera las APIs. Hace posible adaptar la salida para aplicaciones que, por ejemplo, quieran incrustar dichos fragmentos en páginas web sin más procesamiento [10]

Es necesario tener un fichero “.tmpl” (ej: list.tmpl) con el formato de la vista. Un ejemplo de formato de salida con el motor de plantillas Mustache [14] que lista los resultados en HTML:

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

Se sube el fichero “.tmpl” a la ruta /views/<name\_view> especificando el nombre de la 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>
```

Para 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>
```

### Endpoint /docs

Hay la posibilidad de subir documentación a la API. Se deberá de tener 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
```

## 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
```

### 3.3.2.2 Pasos de ejecución

#### 3.3.2.2.1 Dockerfile

Dockerfile instalará los siguientes recursos:

- 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 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 15: Dockerfile Basil*

Se copian del directorio /Ejemplos los ficheros que se utilizan en los ejemplos que se describe en el 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). Se utiliza Maven para la construcción del proyecto java a



partir del fichero pom.xml de la raíz del proyecto. Esto 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” se puede 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 16: Fichero script.sh (Basil)

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

### 3.3.2.2.2 Ejecución

Entendido el funcionamiento del Dockerfile, se procede a generar la imagen:

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

Se espera a que el proceso termine. Esto puede demorar unos minutos ya que se debe de descargar e instalar paquetes de datos. Una vez se tenga la imagen se debe 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 abrirá un Shell Bash donde se debe de arrancar el servidor mysql (no prestar atención al warning que aparece):

```
service mysql start
```

Una vez arrancado el servidor ya se puede ejecutar la aplicación Basil con el script (Ilustración 16). Se debe de ejecutar con el parámetro “&” para ponerlo en segundo plano:



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

Este script internamente indica que se desea utilizar el 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 17: Fichero run.sh (Basil)*

Una vez ejecutado el anterior código, aunque se haya ejecutado en segundo plano, la consola se quedará esperando después de la línea “#4: enjoy”. Se debe de obtener el control de la consola con “ctrl+c”. Se puede comprobar con el comando “ps -u” que el proceso sigue funcionando.

```
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 ] 2021/5/11:21:13:7981 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 18: Ejecución servidor MySQL y aplicación (Basil)*

Con esto ya se tendría la aplicación Basil funcionando 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]. Para crear ficheros dentro del contenedor se puede utilizar el comando “nano nombrefichero.extension”. En el apartado “3.4.1. Pasos de configuración” se puede encontrar todos los comandos necesarios para crear, consultar, actualizar y eliminar las APIs.

### 3.3.2.3 Ejemplos

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

Para comprobar que los datos se han ido guardado en la base datos, tras realizar los ejemplos, se pueden seguir los siguientes pasos.

1. Identificarnos en el servidor mysql:

```
mysql --host=localhost --user=root --password=jose
```

Se abrirá un prompt “mysql>” en donde se puede ejecutar ejecutar sentencias SQL.

2. Observar las bases de datos existentes:

```
mysql> show databases;
```

3. Seleccionar la base de datos basil:

```
mysql> use basil;
```

4. Observar las tablas existentes:

```
mysql> show tables;
```

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

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

### 3.3.2.3.1 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. Crear el 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. El fichero “query1.sparql” permite buscar las películas de director Peter Jackson usando como endpoint DBpedia. Subir el fichero identificando el usuario creado anteriormente.

```
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 la API. Registrar 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 se puede lanzar una consulta GET a la API para obtener el 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 19: Ejemplo 1. API películas (Basil)*

### 3.3.2.3.2 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. Usar las credenciales de dicho usuario.
2. El fichero “query2.sparql” es muy similar al anterior, pero pudiendo 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 la API. Registrar este identificativo en el punto final /alias

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

4. Se utiliza el fichero “doc.txt” como documentación de la API. Subir 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
```

Se puede acceder a la documentación realizando una petición GET a /docs:

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

5. Ahora se puede lanzar una consulta GET a la API. Se limita 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 20: Ejemplo 2. API parametrizada con documentación (Basil)*

### 3.3.2.4 Análisis

#### 3.3.2.4.1 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 la API. Se puede encontrar un pequeño fallo con el último paso ya que no se especifica en que carpeta situarse 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” (localizado en la sección Wiki) pudiendo causar confusión al interesado en un primer momento.

#### 3.3.2.4.2 Mantenibilidad

Se puede 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. Se puede determinar que esta herramienta, al contar con un proyecto activo, satisface el punto de mantenibilidad.

#### 3.3.2.4.3 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 versión con interfaz [12] con funciones muy limitadas no desarrollada en esta memoria.

### 3.3.2.5 Complicaciones

#### 3.3.2.5.1 Máquina local

##### 3.3.2.5.1.1 Creación del proyecto con Maven

Descargados 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 se puede

generar el proyecto java ejecutando las pruebas (mvn clean install) u omitiéndolas (mvn install -DskipTests).

Intentando con la primera opción no se pudo generar el proyecto java correctamente. Como se puede observar en la siguiente ilustración, el error fue provocado porque no se pasó todos los test. Se pensó que podría haber algún error internamente en los test y que eso ha generado el error. Por lo tanto se intentó con la opción 2, generando el proyecto java omitiendo 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] -----
[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]
[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]
[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]
[ERROR] After correcting the problems, you can resume the build with the command
[ERROR] mvn <args> -rf :core

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

*Ilustración 21: Error creación proyecto con Maven I (Basil)*

En esta ocasión el proyecto fue generado correctamente (Ilustración 22). 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 provocó 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.

```
[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] -----
[INFO] BASIL :: Parent ..... SUCCESS [ 1.012 s]
[INFO] BASIL :: Rendering ..... SUCCESS [ 4.898 s]
[INFO] BASIL :: SPARQL ..... SUCCESS [ 0.765 s]
[INFO] BASIL :: Core ..... SUCCESS [ 3.628 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 22: Error creación proyecto con Maven II (Basil)*

```

C:\Users\usuario\Documents> 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.InstanceException: 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.determineWebEnv(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:1869) [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.callContextInitialized(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.doStart(ServletContextHandler.java:379) [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 23: Error creación proyecto con Maven III (Basil)*

Una vez ubicado el directorio donde se debía de lanzar la instrucción se obtuvo 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 21. 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.

### 3.3.2.5.1.2 Lanzamiento de aplicación

Lanzando la aplicación con el fichero .jar, generado correctamente y desde la ubicación apropiada, se obtuvo 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:1066) [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.servlet.ServletContextHandler.doStart(ServletContextHandler.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 24: Error lanzamiento de aplicación Basil (MySQL)*

El siguiente error “java.lang.NoClassDefFoundError. javax/...” es debido a que Basil no incluía el paquete de Javax. Contactando con el desarrollador se pudo saber que la aplicación Basil fue desarrollado para Java 1.8. No lo incluyeron ya que esta versión 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: -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
[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 25: Error lanzamiento de aplicación por Javax (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>

```

*Ilustración 26: Solución error Javax (Basil)*

#### 3.3.2.5.1.3 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 se 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 se cogía el resultado del proceso (#1welcome.....#2starting...#3done...#4 enjoy), sin los errores, como si todo hubiera ido bien. Se supuso que el proceso terminaba, se guardaba en el fichero y después ocurría el error. La solución que se halló fue utilizar el terminal Git Bash que ya se tenía instalado en el equipo. De esta forma se pudo ampliar mucho más el buffer y así ver las primeras líneas donde se podía observar el origen del error.

#### 3.3.2.5.1.4 Resultado del lanzamiento

Una vez que se obtuvo lo que se esperaba, según las instrucciones, no se supo si se había lanzado correctamente ya que al entrar en localhost:8080 aparecía lo siguiente:



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

*Ilustración 27: Navegador lanzamiento Basil*

En las instrucciones del repositorio GitHub [11] no se indica que es lo que debería aparecer en localhost:8080. Para saber qué pasos a dar y comprobar que lo que se ha hecho hasta ese momento es correcto se contactó con el desarrollador [13]. Se concluye que todo estaba bien configurado y que el resultado del navegador localhost:8080 es lo adecuado. Se indicó que para usar la aplicación se siguiera el tutorial de Curl [9] que se encontraba en la Wiki del repositorio. El error por mi parte fue no revisar la Wiki, no estaba muy familiarizado con GitHub en ese entonces.

### 3.3.2.5.2 Dockerfile

#### 3.3.2.5.2.1 Actualización del repositorio

Para ejecutar la aplicación se encontró , en el directorio /server, un fichero que se llama “run.sh” que ejecuta el último comando, indicado en las instrucciones del repositorio Basil [11], para lanzar la aplicación. Funcionaba correctamente

en la maquina local con la versión 0.8.0 SNAPSHOT. Cuando se estaba realizando el Dockerfile se observó que se había actualizado a la versión 0.8.0 y al ejecutar el script “run.sh”, del directorio /server, 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 se dejó de usar el script “run.sh” que viene con el repositorio y se creó otro con el ejecutor java puesto correctamente. La versión que se probó en local ya no existía en el repositorio. Se comprendió que SNAPSHOT es una versión que está en desarrollo y cuando está finalizado se sube a los releases del repositorio. La versión que se utilizó finalmente fue el reléase 0.8.0.

### 3.3.2.5.2.2 Base de datos

Uno de los problemas al utilizar la base de datos en Ubuntu desde un Dockerfile fue mantener 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 saltaba el siguiente error:

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

*Ilustración 28: Error mysql-server (Basil)*

Se pudo observar que esto se debía a se estaba autenticándose en un servidor que no estaba arrancado. Para solucionar esto, después de varias pruebas, se pudo 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 al lanzar el comando se realiza todo esto en un mismo proceso permitiendo al servidor mantenerse arrancado.

Para establecer la conexión se intentó usar el comando `apt-get install libmysql-java` y añadir la ruta de la librería al CLASSPATH. Al ejecutar el Dockerfile se obtuvo 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 29: 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 se decidió crear 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 30: 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.

#### 3.3.2.5.2.3 Fichero pom.xml del directorio server

El Dockerfile funcionó correctamente copiando 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. Se copió el fichero “pom.xml”, de la ruta /server, de la máquina local a la imagen, al incluir ya las dependencias Javax. Al lanzar la aplicación se volvió a obtener un error relacionado con la conexión a la base de datos. No se entendía lo que pasaba ya que los ficheros contenían exactamente con lo había funcionado anteriormente. Se localizó unas dependencias para el conector MySQL que no hacían falta ya que ya se había solucionado los problemas de conexión en el Dockerfile. Se volvió a probar la aplicación y se seguía obteniendo el mismo error. Después de un tiempo, se descubrió que para hacer los cambios efectivos había que volver a generar el proyecto Java. Con esto se solucionó el problema y se pudo ejecutar la aplicación con éxito desde el Dockerfile.

#### 3.3.2.6 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 se quiere coger el identificador, es necesario tener una precisión con el ratón para coger exactamente el ID. Además, cuando se obtiene 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.

#### 3.3.2.7 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 resultantes nos da la posibilidad de relacionar dichos datos 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 se apunta en un bloc de notas los comandos esenciales, modificar los SPARQL o cambiar los endpoint no resulta complicado.

### 3.3.3 Puelia

Lamentablemente esta herramienta no ha podido ser probada 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. Al principio se intentó con la versión 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. Se probó con la versión Apache 2.4 ya que en las indicaciones no se especifica 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 módulo 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
```

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 la declarativa anterior sea válida en el directorio raíz y sus subdirectorios.

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

Una vez completada toda la configuración que debe tener el servidor Apache se continuó con la configuración php. Se debe 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 del php. Además, se agregó la extensión en php.ini-development:

extension=memcache

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

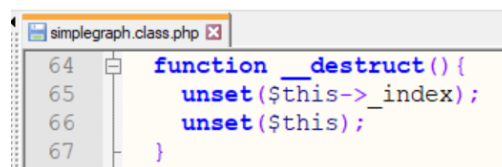
401: Anonymous caller does not have storage.objects.get access to the Google Cloud Storage object.

Solo hay un link que descarga todo el repositorio. No se puede descargar un release en concreto. Esto causó confusión ya que hay muchos ficheros y no se especifica 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. Se probó “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 se obtuvo el siguiente error:

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

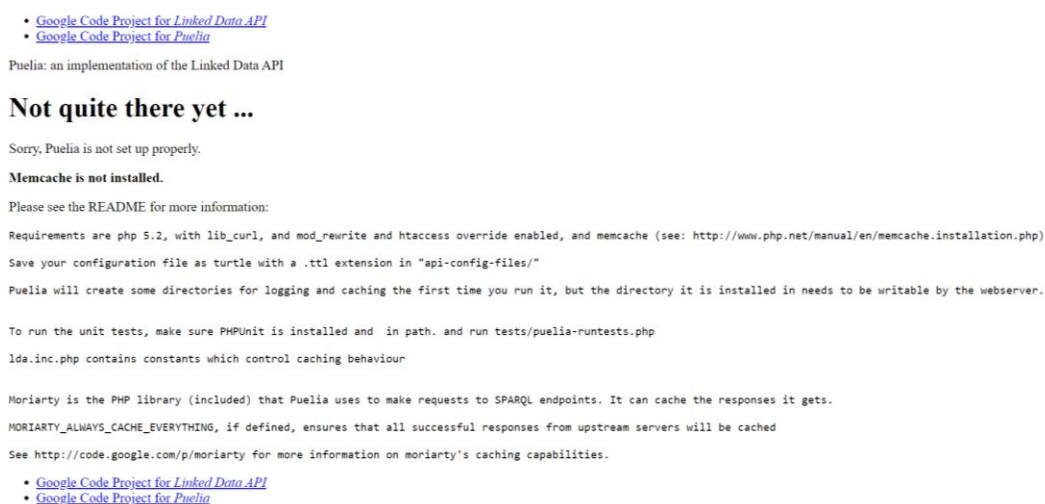
*Ilustración 31: Error php 7 (Puelia)*



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

*Ilustración 32: Fichero simplegraph.class.php (Puelia)*

Esto error fue debido a la versión 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:



• [Google Code Project for Linked Data API](#)  
• [Google Code Project for Puelia](#)

Puelia: an implementation of the Linked Data API

**Not quite there yet ...**

Sorry, Puelia is not set up properly.

**Memcache is not installed.**

Please see the README for more information:

Requirements are php 5.2, with lib\_curl, and mod\_rewrite and htaccess override enabled, and memcache (see: <http://www.php.net/manual/en/memcache.installation.php>)

Save your configuration file as turtle with a .ttl extension in "api-config-files/"

Puelia will create some directories for logging and caching the first time you run it, but the directory it is installed in needs to be writable by the webserver.

To run the unit tests, make sure PHPUnit is installed and in path. and run tests/puelia-runtests.php

lda.inc.php contains constants which control caching behaviour

Moriarty is the PHP library (included) that Puelia uses to make requests to SPARQL endpoints. It can cache the responses it gets.

MORIARTY\_ALWAYS\_CACHE\_EVERYTHING, if defined, ensures that all successful responses from upstream servers will be cached

See <http://code.google.com/p/moriarty> for more information on moriarty's caching capabilities.

• [Google Code Project for Linked Data API](#)  
• [Google Code Project for Puelia](#)

*Ilustración 33: Error configuración incompleta (Puelia)*

Se comprobó 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 se obtuvo el siguiente error:

**Strict Standards:** Declaration of `PueliaGraph::get_label()` should be compatible with `SimpleGraph::get_label($resource_uri, $scapitalize = false, $suse_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

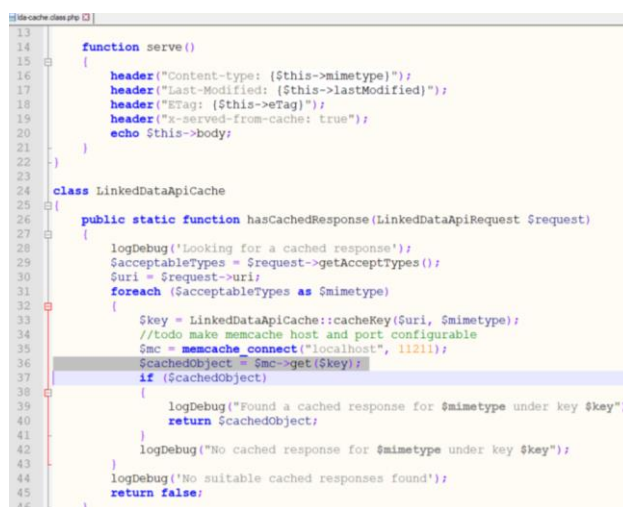
**Warning:** `strftime()`: It is not safe to rely on the system's timezone settings. You are \*required\* to use the `date.timezone` setting or the `date_default_timezone_set()` function. In case you used any of those methods and you are still getting this warning, you most likely misspelled the timezone 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 34: Error `php` en `lda-cache.class.php` (Puelia)*



```

13 function serve()
14 {
15     header("Content-type: {$this->mimetype}");
16     header("Last-Modified: {$this->lastModified}");
17     header("Etag: {$this->eTag}");
18     header("X-served-from-cache: true");
19     echo $this->body;
20 }
21 }
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 35: Fichero `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, $scapitalize = false, $suse_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

**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

**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

**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

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

*Ilustración 36: Error `php` en `index.php` (Puelia)*

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

*Ilustración 37: 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 versión 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 36.

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 está 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 pero 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 este tipo de problemas.

### 3.3.4 Trellis

#### 3.3.4.1 Configuración

Como se comentó en la sección del estado del arte, los recursos de esta herramienta se gestionan con métodos HTTP estándar: POST (crear), PUT (modificar/crear), GET (obtener), PATCH (conjunto de modificaciones) DELETE (eliminar). Además, OPTIONS se utilizan para determinar las capacidades de un recurso determinado.

##### 3.3.4.1.1 Obtener recursos

Se puede obtener los diferentes tipos de recursos creados, contenedores RDF y no RDF. La sintaxis básica para obtener el recurso:

```
curl -X GET <URL>
```

Para especificar el formato del resultado de la petición. Por ejemplo si se quiere obtener un RDF se utiliza:

```
curl -X GET <URL> -H "Accept: text/turtle"
```

También se puede obtener resultados en:

- image/jpeg
- application/ld+json
- application/sparql-update

Para especificar los triples a incluir o excluir en la representación se puede utilizar la opción "prefer" [44]. Por ejemplo:

```
curl -X GET <URL> -H "Prefer: return=representation; include= "<recurso >" "
```

##### 3.3.4.1.2 Crear recursos

Utilizando POST:

Al crear los recursos del LDP se debe de incluir en el encabezado un Link que especifique el tipo de recurso que es:

- http://www.w3.org/ns/ldp#BasicContainer
- http://www.w3.org/ns/ldp#DirectContainer
- http://www.w3.org/ns/ldp#IndirectContainer
- http://www.w3.org/ns/ldp#RDFSsource
- http://www.w3.org/ns/ldp#Resource

También se debe de especificar el tipo de dato con "Content-Type" que se incluye en la petición y el fichero donde está alojado con "-data-binary@<recurso>". Los tipos de datos que soporta son:

- text/turtle
- image/jpeg
- application/ld+json
- application/sparql-update

Además, se debe de especificar la ruta de creación del recurso con la opción "Slug"

Un ejemplo ilustrativo que incluye todas las opciones mencionadas anteriormente:

```
curl -X POST <URL> -H"Slug:<Directorio>" -H"Link:<<url-tipo-recurso>>;rel=\"type\""-H"Content-Type: <tipo-dato>" --data-binary@<recurso>
```

#### Utilizando PUT:

También es posible crear recursos con PUT siguiendo la mismas indicaciones que en POST. Pero es posible que al crear los recursos con PUT se produzcan desconexiones en la asociación de los recursos ya que los contenedores intermedios no se generan de forma automática[39] . Por ello, es recomendable siempre usar POST

#### **3.3.4.1.3 Modificar recursos**

##### Utilizando PATCH

Para modificar documentos RDF se utiliza el tipo “application/sparql-updatetipo”. No se puede utilizar PATH para modificar recursos binarios.

```
curl -X PATCH <URL> -H“Content-Type: <tipo-dato>” --data-binary@<recursoRDF>
```

#### Utilizando PUT:

Para realizar las modificaciones es necesario que el nuevo tipo de recurso sea subtipo del recurso que se esta modificando. En caso de que no sean del mismo tipo, Trellis lanza un error.

```
curl -X PUT <URL> -H“Content-Type: <tipo-dato>” --data-binary@<recurso>
```

#### **3.3.4.1.4 Eliminar recursos**

Para eliminar recursos se utiliza el método HTTP DELETE. Trellis no permite realizar eliminaciones de forma recursiva, lo que significa que el directorio se eliminará pero la jerarquía de directorios que contiene seguirán estando disponibles. Esto puede causar una desconexión de los recursos relacionados a partir de este directorio. Para evitar este suceso se debe de recorrer toda la jerarquía de directorios, del directorio a eliminar, asegurándonos que son eliminados también.

```
curl -X DELETE <URL>
```

#### **3.3.4.2 Pasos de ejecución**

##### **3.3.4.2.1 Docker-compose**

Esta herramienta para funcionar de manera completa necesita, por un lado, el contenedor de la aplicación (trellisldp), y por otro, el contenedor de la base de datos del servidor (postgres). Estos contenedores se pueden obtener del repositorio GitHub de Trellis [26]. Se utiliza un docker-compose.yml para definir las instrucciones que permite ejecutar ambos contenedores a la vez. Después de crear un docker-compose.yml se encontró uno ya existente en el repositorio de Wiki [27], utilizando este finalmente para el lanzamiento de la aplicación. Se tuvo que realizar modificaciones en las credenciales para que pueda funcionar correctamente.







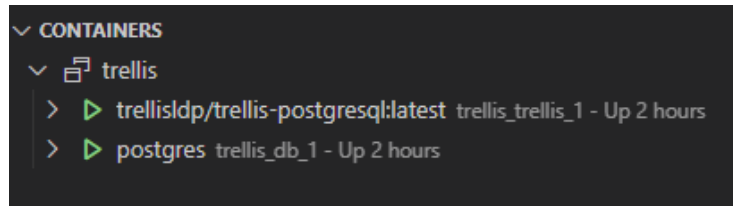


Ilustración 40: Contenedores trellisldp y postgres (Trellis)

Ya se podría utilizar la API de Trellis a través del comando curl desde una terminal. La instalación del terminal y uso del comando curl se especifica en el apartado 3.2. Los comandos necesarios para interactuar con la API se explican en el apartado 3.6.1.

```

Jose Silva@PORTATIL-99 MINGW64 ~/pruebasTrellis
$ curl http://localhost:8080
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           %             %             Dload  Upload   Total   Spent    Left   Speed
100  590    0  590    0    0   3470      0  --:--:-- --:--:-- --:--:-- 3532@f
ams#> .
@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:8080/>
  rdf:type ldp:BasicContainer .

```

Ilustración 41: Petición curl localhost (Trellis)

Se puede comprobar, realizando una petición a localhost:8080, que el servidor Trellis ya viene por defecto con un contenedor LDP “BasicContainer”. Los detalles de qué es un contenedor y un LDP se encuentran en la sección 2.1.4.

### 3.3.4.3 Ejemplos

#### 3.3.4.3.1 Creación de recursos LDP-RS y LDP-NRS (contenedor básico)

El objetivo es almacenar recursos del usuario Pablo. En este caso el usuario tiene información sobre su persona y una foto de perfil. Es necesario utilizar para cumplir estos campos un recurso RDF y una imagen (no RDF) respectivamente.

1. Creación contenedor básico para el usuario Pablo:

```

curl -s http://localhost:8080 -XPOST -H"Slug: pablo" -H"Link:
<http://www.w3.org/ns/ldp#BasicContainer>; rel=\"type\" \" -H"Content-Type:
text/turtle" --data-binary @recursos/pablo.ttl

```

Contenido del fichero pablo.ttl:

```

PREFIX ldp: <http://www.w3.org/ns/ldp#>
PREFIX dc: <http://purl.org/dc/terms/>
<> a ldp:BasicContainer ;
  dc:title "Contenedor Pablo" .

```

2. Creación recurso RDF con la información del usuario:

```

curl -s http://localhost:8080/pablo/ -XPOST -H"Slug: informacion" -H"Link:
<http://www.w3.org/ns/ldp#Resource>; rel=\"type\" \" -H"Content-Type:
text/turtle" --data-binary @recursos/informacion.ttl

```

Contenido del fichero informacion.ttl:

```
@prefix dc: <http://purl.org/dc/terms/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

<> a foaf:PersonalProfileDocument;
    foaf:primaryTopic <#me> ;
    dc:title 'Informacion Pablo' .

<#me> a foaf:Person;
    foaf:name 'Pablo';
    foaf:lastName 'Sanchez';
    foaf:interest 'Le apasiona la ingeniería y el futbol' .
```

3. Creación un recurso no RDF, de tipo imagen, para el avatar del usuario:

```
curl -s http://localhost:8080/pablo/ -XPOST -H"Slug: avatar" -H"Link:
<http://www.w3.org/ns/ldp#Resource>; rel=\"type\"" -H"Content-Type:
image/jpg" --data-binary @recursos/avatar.jpg
```

4. Realizando una petición Get a /pablo/ se puede observar que se ha agregado enlaces a los recursos creados con “ldp:contains”:

```
$ curl http://localhost:8080/pablo/
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload   Total   Spent    Left     Speed
100  770    0  770    0    11846    0 --:--:-- --:--:-- --:--:-- 12222@p
@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:8080/pablo/>
    rdf:type      ldp:BasicContainer ;
    dc:title      "Contenedor Pablo" ;
    ldp:contains  <http://localhost:8080/pablo/informacion> ;
    ldp:contains  <http://localhost:8080/pablo/avatar> .
```

*Ilustración 42: Ejemplo 1. Resultados (Trellis)*

### **3.3.4.3.2 Ejemplo 2. Uso de contenedores básicos y directos**

El objetivo es almacenar recursos de una editorial. Esta editorial almacena publicaciones que contienen RDF libro. Además, usando la propiedad de contenedor directo, se indica la creación de tripletas con los atributos membershipResource y hasMemberRelation en el recurso /editorial/escritores

1. Creación contenedor básico para la editorial:

```
curl -s http://localhost:8080 -XPOST -H"Slug: edits" -H"Link:
<http://www.w3.org/ns/ldp#BasicContainer>; rel=\"type\"" -H"Content-Type:
text/turtle" --data-binary @recursos/editorial.ttl
```

Contenido del fichero editorial.ttl:

```
PREFIX ldp: <http://www.w3.org/ns/ldp#>
PREFIX dc: <http://purl.org/dc/terms/>

<> a ldp:BasicContainer ;
    dc:title "Contenedor Editorial" .
```

2. Creación contenedor directo de una publicación de la editorial:

```
curl -s http://localhost:8080/editorial -XPOST -H"Slug: publicacion1" -
H"Link: <http://www.w3.org/ns/ldp#DirectContainer>; rel=\"type\" \" -H"Content-
Type: text/turtle" --data-binary @recursos/publicacion.ttl
```

Contenido del fichero publicacion.ttl:

```
PREFIX ldp: <http://www.w3.org/ns/ldp#>
PREFIX dc: <http://purl.org/dc/terms/>
PREFIX p: <http://purl.org/saws/ontology#>
<> a ldp:DirectContainer ;
    dc:title "Contenedor Publicacion" ;
    ldp:membershipResource </editorial/escritores> ;
    ldp:hasMemberRelation p:hasWrite .
```

3. Creación contenedor básico de escritores donde se guardan las relaciones que se especifican en el contenedor directo:

```
curl -s http://localhost:8080/editorial -XPOST -H"Slug: escritores" -H"Link:
<http://www.w3.org/ns/ldp#BasicContainer>; rel=\"type\" \" -H"Content-Type:
text/turtle" --data-binary @recursos/escritores.ttl
```

Contenido del fichero escritores.ttl:

```
PREFIX ldp: <http://www.w3.org/ns/ldp#>
PREFIX dc: <http://purl.org/dc/terms/>
<> a ldp:BasicContainer ;
    dc:title "Contenedor Escritores" .
```

4. Creación recurso RDF del libro que se encuentra en publicación:

```
curl -s http://localhost:8080/editorial/publicacion1 -XPOST -H"Slug: libro1"
-H"Link: <http://www.w3.org/ns/ldp#Resource>; rel=\"type\" \" -H"Content-Type:
text/turtle" --data-binary @recursos/libro.ttl
```

Contenido del fichero libro.ttl:

```
@prefix dc: <http://purl.org/dc/terms/> .
@prefix bo: <http://example/books/#> .
<> a bo:Book;
    bo:primaryTopic <#info> ;
    bo:title 'Señor de los anillos' .

<#info> a bo:Book;
    bo:description 'Narra las aventuras de unos Hobbits para destruir un
anillo'.
```

5. Realizando una petición GET a /editorial/escritores/ se puede observar que se ha agregado una tripleta al recurso escritores. Como predicado tiene el

atributo `hasMemberRelation` “`hasWrite`”, especificado en el contenedor directo, y como objeto el libro creado:

```
$ curl http://localhost:8080/editorial/escritores/
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           %         0         0      0 16469   0 --:--:-- --:--:-- --:--:-- 17170@p
100  807    0  807    0  0 16469   0 --:--:-- --:--:-- --:--:-- 17170@p
ref: 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:8080/editorial/escritores/>
  rdf:type ldp:BasicContainer ;
  dc:title "Contenedor Escritores" .

<http://localhost:8080/editorial/escritores>
  <http://purl.org/saws/ontology#hasWrite> <http://localhost:8080/editori
al/publicacion1/libro1> .
```

*Ilustración 43. Ejemplo 2 Resultados (Trellis)*

### 3.3.4.4 Análisis

#### 3.3.4.4.1 Documentación

La información que se proporciona en el repositorio GitHub de Trellis [26] es bastante completa. Se dispone en el repositorio de un Wiki [4], de un grupo de discusión [29] y de un Java Docs [30] de la API. Además, en el directorio raíz del repositorio [31] se puede encontrar otros repositorios relacionados con Trellis. Por ejemplo, existe un repositorio que permite realizar pruebas [32] al servidor Trellis creando diferentes tipos de contenedores y otro que permite crear vocabularios [33].

#### 3.3.4.4.2 Mantenibilidad

Se puede comprobar que la herramienta cuenta con actualizaciones constantes y recientes. Hay varias versiones de contenedores Trellis disponibles en Docker Hub [28]. El grupo de discusión tiene temas de conversación recientes y con respuestas bastantes rápidas por parte de la comunidad, pudiendo comprobar esto último por experiencia propia. Por todo ello se puede concluir que Trellis satisface el criterio de mantenibilidad.

#### 3.3.4.4.3 Interfaz

Trellis se gestiona exclusivamente desde el terminal de comandos a través de solicitudes HTTP con el comando “`curl`”. Al igual que Basil, esto puede ser algo tedioso ya que gestionar todo desde el terminal puede provocar errores por pequeños fallos en las instrucciones. Esto se puede mejorar si se utiliza editores de texto y solo se usa el terminal para ejecutar estos ficheros. Trellis presenta una pequeña interfaz básica donde solo se puede observar las tripletas de los contenedores creados y navegar por los enlaces.

### 3.3.4.5 Complicaciones

#### 3.3.4.5.1 Máquina local

Al realizar pruebas de la API se observó que el mínimo error de sintaxis produce errores en el comando curl. Se debe de distinguir bien comillas simples y dobles, las barras “\” deben estar bien orientadas, los puntos finales situados correctamente...Por experiencia, es mejor utilizar un fichero “.sh” con algún editor de textos (Notepad o Visual Studio Code) para visualizar mejor los comandos. Después ejecutarlo desde el terminal con ./nombrefichero.sh

```
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:title "Foo Container";
> dcterms:description "Container description" .
>
curl: (3) Port number ended with ' '
```

*Ilustración 44: Error curl Trellis*

Al realizar las pruebas, al eliminar el contenedor principal, no se realizaba ninguna acción y seguía apareciendo. La solución fue eliminar cada uno de los recursos que contenía el contenedor para así poder eliminar de forma exitosa el contenedor principal.

#### 3.3.4.5.2 Docker-compose

Utilizar el docker-compose.yml que viene en la documentación del Wiki de Trellis [27] provocó un error de autenticación. Al no localizar las credenciales se contactó con uno de los desarrolladores. Se me proporcionó unas credenciales estándares para lanzar la aplicación. Poniendo los datos proporcionados en los campos “database”, “username” y “password” la aplicación se lanzó correctamente.

```
2021-05-28 11:27:01,850 INFO [io.agr.pool] (main) DataSource 'default': Initial size smaller than min. Connections will be created when necessary
2021-05-28 11:27:01,895 INFO [org.fly.cor.int.lic.VersionPrinter] (main) Flyway Community Edition 7.7.0 by Redgate
2021-05-28 11:27:02,072 UTC [33] FATAL: password authentication failed for user "changeme"
2021-05-28 11:27:02,072 UTC [33] DETAIL: Role "changeme" does not exist.
Connection matched pg_hba.conf line 99: "host all all all md5"
```

*Ilustración 45: Error credenciales Trellis*

### 3.3.4.6 Propuesta de mejora

Observando toda la documentación que se puede obtener acerca del funcionamiento de la herramienta así como el contexto de LDP pienso que puede ser algo complicado de entender para nuevos interesados. La información está bastante dispersa, aunque esto no sea malo, creo que sería de gran utilidad resumir toda la información (LDP, ejemplos y Trellis) en un apartado de la Wiki para facilitar la iniciación con la herramienta.

La herramienta debería de permitir eliminar de forma recursiva existiendo recursos en su interior. Aunque esto puede ser algo peligroso ya que eliminaría todo el contenido, se podría distinguir con algún tipo de encabezado para evitar eliminar contenido por error.

### **3.3.4.7 Conclusión**

Trellis es una herramienta con mucho potencial ideal para almacenar una gran cantidad de datos. La forma de organizar los recursos, al principio, es algo complejo de entender, pero luego se comprende bastante bien su estructura. Es muy útil para relacionar diferentes tipos de datos, de la realidad o abstractos, a través de recursos RDF, binarios, imágenes, HTML.... Otro punto positivo es su flexibilidad de definir la formación de tripletas y la generación automática de estas en los recursos definidos. La información que se posee para utilizar la herramienta es muy amplia, sigue las especificaciones de plataforma de datos enlazados [36] (LDP) y los estándares HTTP [39]. Se ha intentado resumir las especificaciones más relevantes de esta herramienta para facilitar el entendimiento de su uso, usando ejemplos básicos que representan recursos de la vida real. Además, es una herramienta que cuenta con bastante soporte y que presenta actualizaciones bastantes recientes lo que permite mejorar el uso de la API y el potencial.

## 4 Conclusiones

### 4.1 Resultados

La creación de herramientas que permite generar el entorno necesario para el lanzamiento de las APIs REST propuestas, a partir de ontologías y grafos de conocimientos, han sido el resultado del desarrollo del proyecto. Estas herramientas han sido desarrolladas utilizando contenedores Docker, simplificando el lanzamiento de estas APIs y utilizando la menor cantidad de recursos posibles para su funcionamiento. Se ha podido desarrollar herramientas de lanzamiento para las APIs Pubby, Basil y Trellis. No se ha logrado desarrollar la API Puelia debido a errores documentados en la sección 3.3.3.

Con el desarrollo de las herramientas se ha proporcionado la visión de un desarrollador que está interesado en utilizar las APIs por primera vez, reflejando los problemas que puede encontrarse una persona sin experiencia. Se ha documentado las configuraciones necesarias para el funcionamiento de las APIs de una forma sencilla y que permite una mejor comprensión inicial.

Además se ha obtenido un estudio de cada API desarrollada, realizando un análisis, reflejando el estado de su documentación, mantenibilidad, interfaz e incluyendo una serie de propuestas de mejora.

### 4.2 Conclusiones personales

El desarrollo de este proyecto me ha proporcionado una visión y una experiencia de lo que significa realizar una investigación real en el ámbito informático. Me ha servido tanto para completar mi formación académica de ingeniero informático, como en lo personal. Realizar una investigación de herramientas que puedan presentar una serie de problemas, así como entender las diferentes configuraciones y los distintos entornos que necesitan para su funcionamiento, no es tarea fácil desde mi perspectiva. Sobre todo me quedo con la satisfacción de resolver los problemas que se me han ido presentando, excluyendo la herramienta Puelia, y del aprendizaje sobre los errores cometidos. La resolución de problemas es una cualidad importante ya que, de acuerdo con lo dicho por mi cotutor Daniel Garijo, en el mundo profesional siempre surgirá problemas que pueden no tener una solución a primera vista y que se debe de “hincar codos” para hallar la solución de estos.

También me ha permitido adquirir habilidades de búsqueda de información e interpretación de las guías que proporcionan los repositorios de las APIs estudiadas, así como las fuentes externas. Todo esto unido la elaboración de una memoria de investigación con todos los apartados requeridos, me ha permitido mejorar mis habilidades expresivas y de documentación.

Me hubiera gustado desarrollar más herramientas que desplieguen APIs Linked Data, pero no se ha podido debido a las diversas complicaciones y al límite de tiempo. Estas complicaciones fueron un aporte positivo para mi aprendizaje.

En definitiva, estoy bastante satisfecho con el trabajo realizado y con lo aprendido durante el desarrollo. El aprendizaje de los errores que ha habido por mi parte, y la experiencia de utilizar diferentes entornos de desarrollo y aplicaciones, ha incrementado mi perspectiva sobre las tecnologías existentes.

### **4.3 Líneas futuras**

Creo que las APIs en el campo de Linked Data es una de las herramientas más potentes y útiles para un futuro, que nos puede ayudar a desarrollar diversas aplicaciones que pueden ser utilizadas en diferentes áreas (sanidad, entretenimiento, información...). Siguiendo la dinámica del proyecto, creo que es una buena idea Dockerizar el lanzamiento de APIs, ya que aún existe muchas que necesitan una revisión de su estado al tener varios años desde su creación. De esta forma se puede incentivar su uso por parte de nuevos desarrolladores ya que permite probar y entender su funcionamiento de una manera más sencilla.



## 5 Análisis de Impacto

Analizando los 17 objetivos [41] de desarrollo sostenible que se han comprometido 193 países, incluyendo España, este proyecto se puede relacionar con algunos de ellos. Este proyecto no tiene un gran valor significativo en relación con los objetivos que se describen después. Existe una pequeña, muy pequeña, relación que, a lo mejor con mayor desarrollo de este proyecto y cogiendo la idea principal (ahorro de tiempo), puede causar un impacto mayor al actual. Como ya se ha comentado anteriormente, uno de los objetivos de este proyecto es realizar herramientas que desplieguen las APIs de una manera conjunta, aportando las configuraciones más relevantes. Esto permite que el usuario interesado pueda ahorrarse mucho tiempo de investigación y preparación del entorno de lanzamiento de las APIs por primera vez. Este ahorro de tiempo podría traducirse en ahorro de energía eléctrica al reducir el uso del equipo informático, en comparación de si no contará con las herramientas que proporciona este proyecto. Por ello se podría relacionar con el objetivo 7 “Energía asequible y no contaminante” [42]. Más concretamente se podría asociar a la meta 7.3 “Eficiencia energética”, que tiene como finalidad “duplicar la tasa mundial de mejora de la eficiencia energética hasta 2030”. Claramente esta meta es muy global y se habla de una gran cantidad de energía, que no tiene ni punto de comparación con lo que consume un equipo informático. Pero si se extrae la idea general, usar contenedores Docker usando versiones lo más ligeras posibles, cuando se necesite entornos de trabajo que requiera muchas aplicaciones externas, usando los contenedores, el consumo de energía sería algo más eficiente. También puede relacionarse con el objetivo 8 “Trabajo decente y crecimiento económico” [43]. Observando todas las metas que tiene este objetivo se podría asociar con la meta 8.2 “Diversificación, tecnología e innovación”. Esta meta intenta incrementar la productividad económica modernizando los medios tecnológicos y la innovación. Ocurre exactamente lo mismo que lo explicado anteriormente, el proyecto actual no tiene mucho peso con lo descrito pero la idea fundamental, uso de contenedores Docker y aportación de documentación para ayudar a los desarrolladores, puede surgir nuevas ideas tecnológicas que cause mayor productividad en la economía.

## 6 Bibliografía

[1]: Web oficial Pubby:

<http://wifo5-03.informatik.uni-mannheim.de/pubby/>

[2]: Repositorio Google Code Puelia :

<https://code.google.com/archive/p/puelia-php/>

[3]: Repositorio GitHub Basil. Sección Wiki, introducción:

<https://github.com/basilapi/basil/wiki/Introduction>

[4]: Repositorio GitHub Trellis. Sección Wiki, introducción:

<https://github.com/trellis-ldp/trellis/wiki>

[5]: Web datos.gob “Pubby y LODI, abriendo los datos enlazados a los humanos”:

<https://datos.gob.es/es/blog/pubby-y-lodi-abriendo-los-datos-enlazados-los-humanos>

[6]: Web oficial Docker. Sección instalación aplicación:

<https://docs.docker.com/get-docker/>

[7]: Web oficial Docker Hub:

<https://hub.docker.com/>

[8]: Web oficial Visual Studio Code. Sección instalación aplicación:

<https://code.visualstudio.com/download>

[9]: Repositorio GitHub Basil. Sección Wiki, tutorial curl:

<https://github.com/basilapi/basil/wiki/cURL-tutorial>

[10]: Repositorio GitHub Basil. Sección Wiki, views:

<https://github.com/basilapi/basil/wiki/Views>

[11]: Repositorio GitHub Basil:

<https://github.com/basilapi/basil>

[12]: Repositorio GitHub Basil aplicación Pesto(interfaz Basil):

<https://github.com/basilapi/pesto>

[13]: Repositorio GitHub desarrollador Basil:

<https://github.com/enridaga>

[14]: Repositorio GitHub Basil. Sección Wiki, parametrizar SPARQL:

<https://github.com/basilapi/basil/wiki/SPARQL-variable-name-convention-for-WEB-API-parameters-mapping>

[15]: Web Wikipedia. Sección motor plantilla Mustache:

[https://es.wikipedia.org/wiki/Mustache\\_\(motor\\_de\\_plantillas\)](https://es.wikipedia.org/wiki/Mustache_(motor_de_plantillas))

[16]: Repositorio Google Code Puelia. Sección commits:

<https://code.google.com/archive/p/puelia-php/source/default/commits>

[17]: Repositorio Google Code Puelia. Sección Lanzamiento API:

<https://code.google.com/archive/p/puelia-php/wikis/GettingStarted.wiki>

- [18]: Repositorio GitHub Memcache:  
[https://github.com/nono303/PHP-memcache-dll/blob/master/vc15/x64/ts/php-7.4.x\\_memcache.dll](https://github.com/nono303/PHP-memcache-dll/blob/master/vc15/x64/ts/php-7.4.x_memcache.dll)
- [19]: Repositorio Google Code Puelia. Sección recursos:  
<https://code.google.com/archive/p/puelia-php/source/default/source>
- [20]: Repositorio Google Code Puelia. Sección descargas:  
<https://code.google.com/archive/p/puelia-php/downloads>
- [21]: Web Stackoverflow. Cuestión “fatal error cannot unset this php 7.1”:  
<https://es.stackoverflow.com/questions/115553/fatal-error-cannot-unset-this-php-7-1>
- [22]: Repositorio Google Code Puelia. Sección issues:  
<https://code.google.com/archive/p/puelia-php/issues>
- [23]: Paola Espinoza, 2021. Tesis “Crossing the Chasm Between Ontology Engineering and Application Development: A Survey”.
- [24]: Web Simplifyingtech “how to use curl command”:  
<https://simplifyingtech371899608.wordpress.com/2020/10/26/how-to-use-curl-command-get-post-put-deletehead/>
- [25]: Web oficial Curl:  
<https://curl.se/docs/manpage.html>
- [26]: Repositorio GitHub Trellis:  
<https://github.com/trellis-ldp/trellis>
- [27]: Repositorio GitHub Trellis. Sección Wiki, Dockerizando Trellis:  
<https://github.com/trellis-ldp/trellis/wiki/Dockerized-Trellis>
- [28]: Web Docker Hub. Docker de Trellis:  
[https://hub.docker.com/r/trellisldp/trellis/tags?page=1&ordering=last\\_updated](https://hub.docker.com/r/trellisldp/trellis/tags?page=1&ordering=last_updated)
- [29]: Web Grupos Google Trellis:  
<https://groups.google.com/forum/#!forum/trellis-ldp>
- [30]: Web oficial Trellis. Sección Java Docs:  
<https://www.trellisldp.org/docs/trellis/current/apidocs/>
- [31]: Repositorio GitHub General Trellis:  
<https://github.com/trellis-ldp>
- [32]: Repositorio GitHub Pruebas Trellis:  
<https://github.com/trellis-ldp/trellis-docker-tests>
- [33]: Repositorio GitHub Vocabulario Trellis:  
<https://github.com/trellis-ldp/trellis-vocabulary>
- [34]: Imagen sobre recursos RDF y N-RDF de w3.org:  
<https://www.w3.org/TR/ldp/images/ldpr1.png>
- [35]: Web oficial w3.org. Estándares de plataforma de datos enlazados (básico):  
<https://www.w3.org/TR/ldp-primer/>

- [36]: Web oficial w3.org. Estándares de plataforma de datos enlazados:  
<https://www.w3.org/TR/ldp/>
- [37]: Repositorio GitHub Héctor Correa. Diferencias contenedores LDP:  
<https://gist.github.com/hectorcorrea/dc20d743583488168703>
- [38]: Repositorio GitHub Trellis. Sección Wiki, estándares web:  
<https://github.com/trellis-ldp/trellis/wiki/Web-Standards>
- [39]: Repositorio GitHub Trellis. Sección Wiki, administrar recursos:  
<https://github.com/trellis-ldp/trellis/wiki/Resource-Management>
- [40]: Repositorio GitHub del Trabajo Fin de Grado:  
<https://github.com/Josesilva99/TFG-APIsREST-LinkedData>
- [41]: Web oficial agenda2030. Sección 17 objetivos:  
<https://www.agenda2030.gob.es/objetivos/home.htm>
- [42]: Web oficial agenda2030. Sección objetivo 7:  
<https://www.agenda2030.gob.es/objetivos/objetivo7.htm>
- [43]: Web oficial agenda2030. Sección objetivo 8:  
<https://www.agenda2030.gob.es/objetivos/objetivo8.htm>
- [44]: Web oficial w3.org. Estándares de plataforma de datos enlazados, sección “prefer-parameters”:  
<https://www.w3.org/TR/ldp/#prefer-parameters>
- [45]: Web fundacionctic.org “Transformación social y grafos de conocimientos”:  
<https://www.fundacionctic.org/es/actualidad/transformacion-social-y-grafos-de-conocimiento>
- [46]: Daniel Garijo, cotutor del proyecto. Perteneciente al Instituto de Ciencias de la Información, Universidad del Sur de California, CA, Estados Unidos
- [47]: Paola Espinoza, cotutora del proyecto. Perteneciente al grupo de Ingeniería en Ontología, Universidad Politécnica de Madrid, MAD, España
- [48]: Web gnos.com “Datos enlazados y grafos”:  
<https://www.gnos.com/datos-enlazados-grafos>
- [49]: Wikipedia, World Wide Web Consortium:  
[https://es.wikipedia.org/wiki/World\\_Wide\\_Web\\_Consortium](https://es.wikipedia.org/wiki/World_Wide_Web_Consortium)
- [50]: Web Ceweb.br “Web Semantica Capitulo 4” Sección Web Semantic:  
<https://ceweb.br/guias/web-semantica/es/capitulo-4/>
- [51]: Web Ceweb.br “Web Semantica Capitulo 6”:  
<https://ceweb.br/guias/web-semantica/es/capitulo-6/>
- [52]: Web Semantizandolaweb.wordpress.com “URIs desreferenciabiles”:  
<https://semantizandolaweb.wordpress.com/2011/12/01/uris-para-nombrar-recursos/>
- [53]: Web Opendatahandbook.org “Datos Abiertos”  
<https://opendatahandbook.org/guide/es/what-is-open-data/>
- [54]: Web Ceweb.br “Web Semantica Capitulo 4” Sección ontologías:  
<https://ceweb.br/guias/web-semantica/es/capitulo-4/#capitulo-4-sh6>

[55]: Web Ceweb.br “Web Semantica Capitulo 4” Sección SPARQL:  
<https://ceweb.br/guias/web-semantic/es/capitulo-4/#capitulo-4-sh7>

[56]: Web oficial DBpedia:  
<https://www.dbpedia.org/>

[57]: Web oficial Datos.bne:  
<https://datos.bne.es/inicio.html>

[58]: Web oficial W3C:  
<https://www.w3c.es/>