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

# LINKED DATA BROWSER: EXTENSIÓN DE HERRAMIENTA EDUCATIVA PARA LA WEB DE DATOS

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN COMPUTACIÓN

RAÚL ALEXANDER CID BASTÍAS

PROFESOR GUÍA:
AIDAN HOGAN

MIEMBROS DE LA COMISIÓN:
JORGE PÉREZ ROJAS
JUAN MANUEL BARRIOS

SANTIAGO DE CHILE
2021

# Resumen

La Web es una enorme plataforma de información, que ha crecido a pasos agigantados durante este milenio. Pero esta información es legible en la medida de que quién la lea sea un humano; las máquinas encargadas de transportar, procesar, y desplegar esta información en la Web no son capaces (aún) de darle sentido al lenguaje natural. En un continuo esfuerzo, la comunidad de la Web ha desarrollado distintos principios, estándares, y tecnologías para dotar a las máquinas la capacidad de entender la información que se encuentra disponible en la Web, dando inicio a una *Web de datos sobre la Web*, la Web Semántica.

El estudio y comprensión de dichas tecnologías es parte de lo abordado en el curso "CC7220 - La Web de Datos". El curso esta diseñado para poner en práctica los conocimientos adquiridos en cátedra a través de actividades que permitan a sus estudiantes experimentar las posibilidades y aplicaciones que presentan estas tecnologías. En instancias pasadas del curso, las herramientas utilizadas en las actividades eran independientes entre sí, lo que no permitía destacar la importancia del conjunto de tecnologías y estándares como un todo.

A partir del 2020, se comienza a utilizar en el curso una herramienta que reúne en un único ambiente la mayor parte de los estándares y tecnologías de la Web Semántica: RDF Playground. Desarrollada por Bastián Inostroza, esta aplicación permite cubrir un amplio espectro de las actividades prácticas del curso, a excepción de aquellas relacionadas con el tópico de Linked Data. De aquí surge la oportunidad de este trabajo de título, al incorporar en RDF Playground los mecanismos necesarios para buscar, visualizar, y explorar documentos de la Web Semántica.

La evaluación de rendimiento del sistema, el Linked Data Browser, nos dice que el tiempo de descarga y modelado de un documento RDF de la Web se mantiene estable bajo los 5 segundos, y que aumenta considerablemente cuando se trata de documentos que contienen del orden de $10^5$ triples. De la misma forma, la visualización de información el grafo tiene un costo asociado a la cantidad de elementos que se agregan en un paso al grafo; mientras más elementos se agreguen a la vez, más se demorará el sistema en renderizar dichos elementos.

Por último, la evaluación de usabilidad de este sistema abarcó una cantidad pequeña de participantes, por lo que sus resultados no tienen una confianza alta. De todas formas, las personas que utilizaron y contestaron la encuesta muestran que el Linked Data Browser de RDF Playground cumple de manera aceptable su función de herramienta educativa para el curso "CC7220 - La Web de Datos"

*"Everyone can have a teacher, in the form of access to the gathered knowledge of the human species" - Isaac Asimov*

# Agradecimientos

Hay un número significante de personas que me han acompañado y apoyado a lo largo de este camino a quienes quiero dedicarles este espacio.

A mi padre Raúl y mi madre Brenda, que sin su apoyo jamás habría llegado hasta este punto. A mi hermano Rubén que me motivó a estudiar esta carrera.

A mis amigos y amigas de antaño, en particular Gabriel y Javier quienes siempre han estado ahí para escucharme y sostenerme, y para disfrutar esta aventura.

A mi familia paterna y materna, que si bien nos vemos pocas veces al año, siempre es un agrado saber que confían en mí.

A mis amigos y amigas de la universidad, que han enriquecido enormemente mi vida a través de sus experiencias de vida. Son demasiadas menciones honrosas; Pablito, Pilar, Pancho, Pepe, Dani, Migue, Pancha, Víctor.

A David, y su familia, por acogerme en su hogar innumerables veces a lo largo de mi paso por Beauchef, tanto para disfrutar de la vida como para sacar adelante nuestras carreras.

A Aidan, por ser el guía de esta memoria. Ha sido un tremendo agrado y una muy buena experiencia trabajar con él.

Y finalmente, agradecerme a mí mismo. Por no haber desertado cuando tuve la oportunidad, por no rendirme pese a las decenas de fracasos que sufrí en la carrera.

Gracias.

# Tabla de Contenido

# Índice de Ilustraciones

# Capítulo 1

# Introducción

La Web lleva funcionando alrededor de 30 años [20]; ha crecido y cambiado considerablemente a lo largo de ese tiempo, desde simples páginas de hipertexto para fines científicos hasta redes sociales con millones de usuarios, sistemas gubernamentales completos, o comercios digitales en expansión.

Aun así, la idea central de la Web no ha cambiado: es una plataforma para consumir y compartir información. La mayor parte de este contenido está creado para consumo humano, para que las personas puedan entenderlo.

¿Pero, y las máquinas? Aquellas que son el medio responsable de enviar, recibir, y desplegar la información, ¿qué saben de esta información? En general, este contenido no se escribe para que las máquinas puedan darle significado o entiendan cosas sobre él.

Hacia finales del S.XX y comienzos del S.XXI, pocos años después del inicio de la Web, el W3C - World Wide Web Consortium - publica las primeras recomendaciones y estándares en torno a generar datos legibles para las máquinas, para que estas puedan realizar un mejor trabajo al procesar e intercambiar información; la Web Semántica [19].

Es válido entonces preguntarse, ¿ayudará a la Web que su información sea cada vez más entendible para las máquinas? ¿Cómo? ¿Por qué? Las respuestas a estas preguntas son parte del contenido del curso `"CC7220 - La Web de Datos"`, del Departamento de Ciencias de la Computación de la Universidad de Chile, dictado por el profesor Aidan Hogan.

Gran parte de las actividades de este curso consisten en laboratorios prácticos donde se utilizan distintas tecnologías relacionadas a la Web Semántica, tales como **RDF** [18], **RDFS** [6], **OWL** [10], **SPARQL** [9], **SHACL** [14], y **ShEx** [17]. Existen pocas herramientas o aplicaciones de enseñanza relacionadas a estas tecnologías, lo que dificulta la organización y elaboración de los laboratorios.

En este contexto, y como trabajo de memoria, Bastián Inostroza desarrolló una herramienta interactiva para apoyar el aprendizaje del curso llamada RDF Playground [13], cuya principal característica es reunir en un ambiente educativo las distintas tecnologías de la Web Semántica. La herramienta se comenzó a utilizar durante el semestre *Primavera 2020*.

La presente memoria extiende la funcionalidad de la aplicación, al añadir un Linked Data Browser (Navegador de Datos Enlazados) que servirá para explorar visualmente datos RDF de la Web a través de un grafo. Con esto, los y las estudiantes del curso podrán ingresar al buscador un Identificador de Recurso Uniforme (de la sigla en ingles **URI**), e interactuar con la información que este recurso tenga, de tal forma que comprendan las buenas prácticas de Linked Data (**LD**) [4].

## 1.1. Problema

En su primera versión, RDF Playground satisface en gran medida los requerimientos que demandan los laboratorios que se realizan en el curso *La Web de Datos*. Estas actividades son fundamentales en el curso; representan un porcentaje importante de la evaluación, cada una está directamente relacionada con un contenido de aprendizaje, y son la instancia que permite afianzar estos contenidos de manera práctica al utilizar diferentes herramientas y aplicaciones que implementan las tecnologías de la Web Semántica.

Pero RDF Playground, al igual que todo software, no está terminado. Siempre se puede encontrar la manera de mejorar un software, ya sea añadiendo nuevas funcionalidades que no estaban consideradas o pensadas necesarias, implementando aquellas que se descartaron en el camino en favor del tiempo, o creando algo totalmente nuevo.

En particular, una de las funcionalidades necesarias para abordar en plenitud las actividades de los laboratorios es la implementación de un Linked Data Browser, herramienta que permite revisar y *navegar* datos enlazados de la Web. Además, el contexto educativo exige que la implementación de esta herramienta permita fortalecer el concepto de LD al ejemplificar cómo puede usarse esta tecnología.

En iteraciones anteriores del curso, se ha utilizado un navegador externo basado en texto que, si bien permite elaborar y realizar los laboratorios, carece de una representación visual de los datos. Pero más importante es que dicha funcionalidad esté integrada en RDF Playground, con el objetivo de reunir las tecnologías de la Web Semántica en un ambiente educativo.

Por lo tanto, el desafío radica en implementar esta herramienta bajo lineamientos pedagógicos que permitan una comprensión de los fundamentos de LD utilizando ejemplos concretos de la Web, manteniendo una interacción humano computador sencilla incorporada en el ambiente de RDF Playground.

## 1.2. Objetivos

### Objetivo General

El objetivo de esta memoria es desarrollar una herramienta digital que permita fundamentalmente dos beneficios, en el contexto del curso `"CC7220 - La Web de Datos"`; ser un

apoyo a la labor educativa en torno al concepto de Linked Data, y contribuir a un proceso de unificación de las tecnologías principales de la Web Semántica en una gran herramienta educativa.

### Objetivos Específicos

1. Implementar un Linked Data Browser sencillo, similar al utilizado actualmente en los laboratorios del curso.
2. Implementar un sistema visual de navegación a base de grafos, dentro de RDF Playground, para el Linked Data Browser.
3. Evaluar y validar el rendimiento y la usabilidad del sistema.
4. Incorporar al sistema funcionalidades que mejoren su usabilidad y rendimiento.

## 1.3. Solución

Para resolver este desafío y cumplir con los objetivos planteados se extendió el sistema RDF Playground, tanto el *frontend* como el *backend*, de manera que la implementación del Linked Data Browser utilice el stack de tecnologías del sistema sin reestructurar su arquitectura. Este stack está conformado por Kotlin [1], Apache Jena [2], y Spring Boot [3] en el *backend*; Vue.js [4] y Vis.js [5] para el *frontend*.

Los cambios importantes en el *backend* apuntan a la capacidad de desreferenciar un URI, modelar la información que contiene, y retornar los datos en un formato que facilite el trabajo de visualización por grafos. En la práctica, esto implica incorporar un método que se encargue de estas peticiones, así como ajustar la funcionalidad que permite formatear el modelo de Jena al lenguaje DOT [6] para que el *frontend* tenga toda la información necesaria al momento de visualizar los datos.

Por su lado, el desarrollo del *frontend* incorpora en la interfaz principal una nueva pestaña, similar a las ya implementadas en la primera versión del sistema, que despliega una sección donde se puede ingresar un URI que el *backend* se encarga de procesar, para finalmente mostrar una interfaz que permite interactuar con los datos visualizados como un grafo.

Entre los desafíos importantes para dicha visualización podemos mencionar los problemas que surgen cuando los documentos RDF que obtenemos de la Web son suficientemente grandes, resultando en grafos difíciles de explorar. Como consecuencia, mientras más infor-

[1] `https://kotlinlang.org/`
[2] `https://jena.apache.org/`
[3] `https://spring.io/projects/spring-boot`
[4] `https://vuejs.org/`
[5] `https://visjs.org/`
[6] `https://graphviz.org/doc/info/lang.html`

mación contenga el documento RDF, su tiempo de descarga y la velocidad del sistema para visualizarlo como grafo aumentan considerablemente.

Para resolver estos desafíos, se optó por una visualización incremental de los datos, es decir, el grafo muestra inicialmente un solo nodo que se corresponde con el URI ingresado. No obstante, la interfaz permite acceder a toda la información del documento e incorporarla gradualmente en el grafo al seleccionar alguna propiedad dentro del documento. Además, se pueden utilizar URI's que se encuentren en el grafo para obtener sus documentos, permitiendo navegar entre los datos. Esta estrategia resuelve tanto el manejo de documentos grandes, como la carga inicial del grafo.

## 1.4. Estructura

La presente memoria se encuentra estructurada de la siguiente forma:

- **Capítulo 2** describe y profundiza los tópicos necesarios para comprender el trabajo descrito en este informe. Aborda los conceptos principales de la Web Semántica y Linked Data, así como las tecnologías y los estándares necesarios para su operatividad. Se describe brevemente las funcionalidades de RDF Playground, y se discuten distintos Linked Data Browsers disponibles en la Web.
- **Capítulo 3**: comprende el desarrollo del LD Browser para RDF Playground. Se describe el problema a resolver, por qué es importante resolverlo, y los desafíos que conlleva. En respuesta, se describe la elaboración de la solución, los aspectos más importantes de su implementación, y las capacidades y funcionalidades añadidas al sistema.
- **Capítulo 4**: evalúa el sistema desarrollado en dos instancias, analizando individualmente el rendimiento del *backend* y del *frontend*, así como la usabilidad del sistema en relación con la experiencia de usuario.
- **Capítulo 5**: concluye la memoria, señalando los objetivos cumplidos y no cumplidos, analizando los resultados de la evaluación, y proponiendo posibles trabajos que pueden realizarse a futuro para mejorar el sistema.

# Capítulo 2

# Marco Teórico

La Web es un sistema de información relativamente joven, que ha crecido y evolucionado considerablemente en estos últimos 20 años. Su popularidad puede atribuirse a múltiples factores: mayor acceso a computadoras durante los años 90 en adelante, el auge de teléfonos *inteligentes* con acceso a internet durante la década del 2000, y el creciente mercado de aparatos electrónicos del mismo tipo.

Si bien estos factores son importantes, el factor más importante en el crecimiento de la Web es su comunidad; las personas que la crearon, las que trabajan en mejorarla continuamente, y aquellas que desean compartir su contenido en ella.

Este capítulo aborda principalmente el trabajo de esta comunidad en el desarrollo de la Web Semántica, y las tecnologías que han diseñado e implementado para realizarla. Luego, se describen las principales funcionalidades de RDF Playground en relación con estas tecnologías. Finalmente se discuten y comparan las características de distintos Linked Data Browsers para determinar cuáles son los requisitos que debe satisfacer la extensión de RDF Playground.

## 2.1. La Web Semántica y Linked Data

Como se mencionó en el capítulo anterior, el objetivo principal de la Web es ser un espacio donde compartir información. Para Berners-Lee, esto significa que la Web es una plataforma donde interactúan tanto máquinas como personas [3].

En la práctica, la información en la Web corresponde principalmente a datos humanamente legibles. Las máquinas pueden leer la información, pero no saben qué están leyendo. Tampoco significa que no puedan hacerlo, simplemente no conocen la semántica que la humanidad le ha dado a los lenguajes naturales.

## RDF

En un esfuerzo dirigido a facilitar el trabajo de las máquinas en procesar la información de la Web, Berners-Lee y el equipo de W3C proponen a fines de la década de 1990 las primeras recomendaciones para construir una Web de Datos, al introducir una hoja de ruta para la evolución de la Web: la Web Semántica [3], basada en el *Resource Description Framework* - RDF [15].

RDF es un marco de trabajo que permite describir información relevante sobre **recursos**. Un recurso puede ser cualquier cosa que podamos nombrar; personas, documentos, países, minerales, teorías, pinturas, estrellas. Para elaborar la descripción de dichos recursos necesitamos dos cosas más: definir qué podemos decir respecto a cada recurso, y una estructura simple con la que trabajar.

Por ejemplo, si queremos describir que *"Javier conoce a Gabriel"*, se debe entender a *Javier* y *Gabriel* como recursos que estamos relacionando. Mientras tanto, *conoce a* describe esta relación que hemos establecido entre los recursos. En el contexto de RDF, esta relación se conoce como **propiedad**, y se consideran un subconjunto de los recursos.


Figura 2.1: Ejemplo de triple en RDF. Grafo dirigido con etiquetas.

Dado este ejemplo, podemos definir cómo es una **declaración** (*statement*) en RDF. Similar a una oración simple de lenguaje natural, una declaración en RDF tiene una estructura con tres elementos (que llamamos **triple**); **sujeto**, **predicado** y **objeto**. Cada declaración nos dice que un recurso en particular (sujeto) tiene una propiedad (predicado) y cuál es su valor (objeto). Utilizando como ejemplo la declaración anterior de *"Javier conoce a Gabriel"*, el triple que lo representa se vería así: (*Javier*, *conoce a*, *Gabriel*), donde *Javier* es el sujeto, *conoce a* el predicado, y *Gabriel* el objeto.

Una de las principales características de RDF, es que lo que hemos llamado triple anteriormente puede ser representado como un **grafo dirigido**, donde el sujeto y el objeto corresponden a nodos, y el predicado es una etiqueta sobre el arco direccionado que los une. La Figura 2.1 ilustra la representación de un triple en un grafo, utilizando el mismo ejemplo.

En el contexto de la Web, la información que podemos incluir en un triple está sujeta a tres tipos de datos: IRIs, Literales, y Nodos blancos.

IRI, del inglés *Internationalized Resource Identifier*, nos ayuda a otorgar de identidad

un recurso para que este pueda ser buscado en la Web. Por ejemplo, el recurso de la Universidad de Chile en la dataset de DBpedia se identifica mediante el siguiente IRI: `https://dbpedia.org/resource/University_of_Chile`. Además, los IRI pueden encontrarse en las tres posiciones de una declaración.

Cabe mencionar que los IRIs corresponden a una extensión del antiguo formato de identificación URI (*Uniform Resource Identifier*), el que estaba limitado a caracteres **ASCII**, mientras que los IRIs permiten la mayor parte de los caracteres del *Universal Character Set*.

Los literales son valores básicos y solo pueden situarse como el objeto de una declaración. Generalmente tienen un tipo de dato asociado, o pueden una simple cadena de caracteres. Cuando tienen un tipo de dato particular, se incluye como parte del literal para que pueda ser procesado a posteriori como tal. En el caso de strings que representan texto en lenguaje natural, suele identificarse el idioma en el que se encuentra escrito.

Por último, los nodos blancos son una utilidad de RDF que permite expresar la existencia de un recurso sin identificarlo mediante un IRI o un literal. Por ejemplo, en el caso de una canción sin autor, podemos utilizar un nodo blanco para expresar que no se conoce quién compuso esa obra.

RDF se caracteriza por ser un modelo de datos lo más general y simple posible. Por supuesto, RDF por si mismo no es suficiente para implementar una Web de Datos apropiadamente; se requieren de otras tecnologías y estándares que resuelvan las distintas aristas que demanda la construcción de la Web Semántica.

## Otros Estándares

La Figura 2.2 ilustra el conjunto de tecnologías que requiere la Web Semántica, más conocido como *Semantic Web Layer Cake*. Existe una extensa variedad de diagramas para representar este *pastel*, y que han ido mutando a lo largo del tiempo, tanto por la distribución de los elementos que contiene, como las tecnologías que se han desarrollado para satisfacer las distintas capas de la arquitectura.

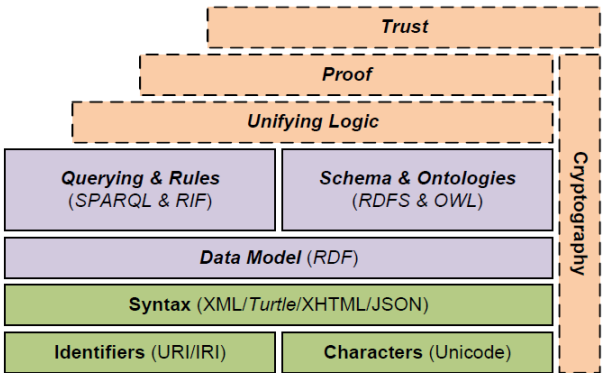



Figura 2.2: Arquitectura de la Web Semántica - *Semantic Web Layer Cake* [11]

En particular, este diagrama agrupa por colores y sección las tecnologías según sus características. En la sección inferior y de color verde, están aquellas que se comparten con

la Web estándar; en la sección central, en lavanda, tecnologías estandarizadas para la Web Semántica; y finalmente en la sección superior de color anaranjado, se encuentran propuestas que no han sido estandarizadas o deben ser implementadas para la completitud de la Web Semántica.

A continuación, se explicarán brevemente los estándares de la Web Semántica que se encuentran incluidos en el conjunto de tecnologías de RDF Playground.

*RDF-Schema* [6] (**RDFS**), es una extensión semántica de RDF. Similar a un lenguaje orientado a objetos, RDFS introduce un sistema de **clases** y **propiedades** que permiten describir grupos de recursos y cuál es la relación que tienen entre sí. Las principales ventajas que otorga este sistema son la capacidad de categorizar recursos, establecer jerarquías de clases y propiedades, y establecer restricciones sobre el dominio y rango de las propiedades.

*Web Ontology Language* [10] (**OWL**), es un lenguaje ontológico. En ciencias de la computación, ontología refiere a una representación formal del conocimiento en un determinado dominio de interés, a través de axiomas, entidades y expresiones. OWL es un vocabulario que extiende RDFS y permite expresar muchísimo más. Por ejemplo, OWL es capaz de expresar que dos IRIs distintos en realidad corresponden al mismo recurso.

Estas dos tecnologías son fundamentales en el desarrollo de una Web de Datos, ya que son las responsables de la capacidad de *razonar* sobre los datos. Así como nuestro razonamiento y entendimiento del mundo nos permite deducir que, si dos personas tienen los mismos padres biológicos, entonces son hermanos, RDFS y OWL establecen los cimientos con los que las máquinas pueden inferir información de la Web de Datos.

*Terse RDF Triple Language* [2] (**Turtle**), es una sintaxis textual que permite transcribir datos RDF de manera simple en formato de texto, y compacta al utilizar atajos y abreviaciones. Se caracteriza por ser una de las sintaxis más amigable para la lectura y escritura humana, y tiene la ventaja de compartir una sintaxis similar al lenguaje de consultas SPARQL.

*SPARQL Protocol and RDF Query Languaje* [9] (**SPARQL**), es un lenguaje de consultas específico para RDF. Similar a SQL en cuanto comparten ciertas características y *keywords* para escribir las consultas. Permite definir prefijos para usar como atajos, especificar el conjunto de datos a consultar, el tipo de consulta, sus patrones y criterios, y modificar/filtrar el resultado final.

*Shapes Constraint Language* [14] (**SHACL**) y *Shape Expressions* [17] (**ShEx**) son las tecnologías incorporadas en RDF Playground; ambas permiten verificar que los grafos RDF cumplan cierta *forma*. SHACL utiliza grafos RDF para expresar las *condiciones* que deben satisfacer los datos, mientras que ShEx utiliza una sintaxis propia y concisa para definirlas. Sus diferencias y similitudes son muchas, lo que escapa al alcance de este trabajo.

## Linked Data

La Web estándar utiliza documentos identificados por URLs, recuperados mediante HTTP, y formateados en un navegador usando HTML, que permite la vinculación con otros docu-

mentos/sitios.

Hasta ahora, hemos discutido distintas tecnologías que permiten describir, definir, consultar y validar datos sobre la Web. Pero, ¿qué Web se puede construir sin interconectar esta información? En efecto, ninguna de estas tecnologías se encarga específicamente sobre como *linkear* los distintos conjuntos de datos RDF que cada sitio Web podría generar.

En 2006, Berners-Lee propuso cuatro principios que abordan esta problemática; principios que deben permitir identificar, describir, localizar, y vincular recursos en la Web de Datos [4]:

1. Usar IRIs para nombrar cosas.
2. Usar HTTP IRIs para que puedan buscarse esos nombres en la Web (desreferenciar).
3. Al buscar un IRI, retornar información relevante en un formato estructurado (documento RDF).
4. Incluir en el documento enlaces a otros IRIs que también sigan estas reglas.

De esta forma, se extiende la Web estándar; se identifican/nombran documentos usando URLs (y IRIs), estos pueden describirse usando HTML (y RDF), pueden recuperarse dichos documentos usando su respectivo identificador, y pueden vincularse a otros documentos.

Sobre como desreferenciar el IRI que identifica el recurso al documento RDF que lo describe, se han propuesto tres métodos para abordar este problema.

El primero – método URL – consiste en utilizar el mismo identificador del documento, lo que generalmente no es una buena idea ya que produce ambigüedad; la máquina no sabrá qué se esta buscando, si es el documento o el recurso.

El segundo – método Hash – utiliza el símbolo *hash* ("#") y permiten diferenciar el identificador del documento RDF, de los recursos que describe. Por lo tanto, al solicitar al servidor el identificador del recurso, este devolverá el documento que lo describe.

El tercero – método Slash – permite definir los identificadores de cada recurso independientemente de la ubicación del documento que lo describe. Se implementa utilizando una redirección de HTTP, lo que agrega una solicitud adicional en comparación a los otros métodos.

Generalmente, los sitios implementan ya sea Hash o Slash, o alguna variante particular de estas dos. Ambos métodos tienen sus ventajas y desventajas sobre el otro, y la elección depende del tipo de contenido que se desea proveer [5].

Por último, se discutirá brevemente el concepto de **Linked Open Data**. *Open Data* corresponde a una práctica, filosófica si se quiere, de que la información debe estar abierta y disponible a todo el mundo. Por su lado, *Linked Data* es una guía para publicar información en la Web. Entonces *Linked Open Data* hace referencia a la intersección de estos dos conceptos, es decir, la idea de publicar Open Data bajo los principios de Linked Data, con la finalidad de mejorar la interoperatividad de estos datos abiertos.

La Figura 2.3 expresa la calidad de la interoperatividad de los datos cuando aspiramos a generar Linked Open Data. Una estrella significa que la información está disponible en la Web. Dos estrellas es que se encuentra estructurada, por ejemplo, en formato Excel.

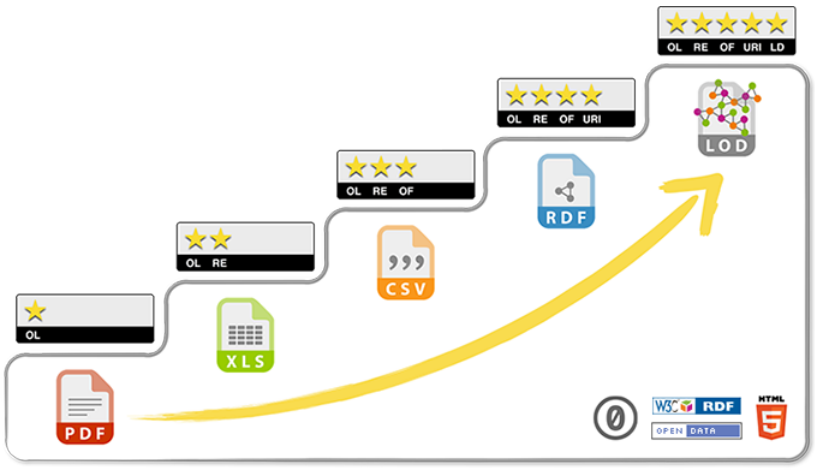



Figura 2.3: Esquema de calidad de Open Data [1]

Tres estrellas corresponde a publicar esta información en un formato estructurado no propietario, como CSV en vez de Excel. Cuatro estrellas significa que se ha utilizado RDF para publicar la información e IRIs para identificar sus recursos.

Finalmente, cinco estrellas indica que la información que hemos creado utilizando RDF se vincula con otra información que también cumple estos estándares.

Aunque se asegura una buena calidad de la información, siguen quedando problemáticas. La necesidad de crear vocabularios específicos para cada conjunto de datos, la veracidad de esta información, contenido de legado que ya está en la Web pero en formatos no compatibles directamente con RDF, o la implementación de aplicaciones que consuman y creen Linked Data, son solo algunos de los problemas que se encuentran abiertos. Resolver estas problemáticas puede despejar el camino para aprovechar el verdadero potencial de la Web Semántica.

## 2.2. Funcionalidades de RDF Playground

RDF Playground[2] ya incorpora varias de las tecnologías de la Web Semántica, las que son utilizadas en los laboratorios del curso "`CC7220 - La Web de Datos`"(ver Figura 2.4). En términos de funcionalidades, RDF Playground puede:

- Escribir un documento RDF usando la sintaxis de Turtle [2].
- Revisar que el documento esté correctamente escrito.

[1] `https://5stardata.info/en/`
[2] `http://rdfplayground.dcc.uchile.cl/`

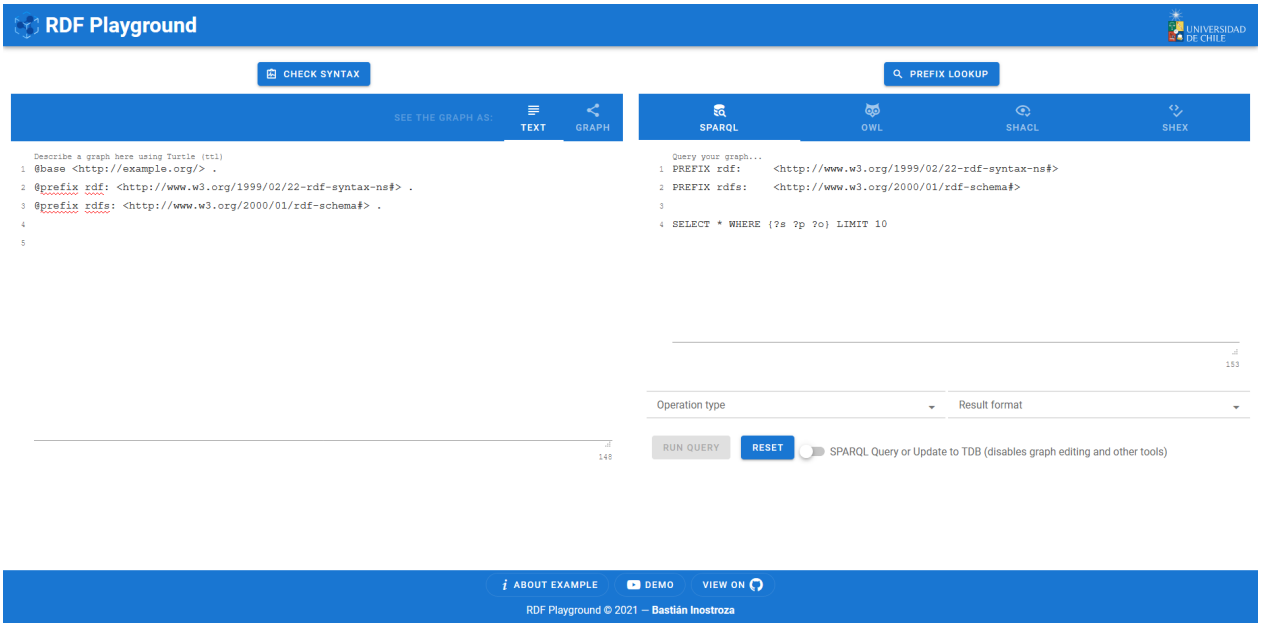



Figura 2.4: Interfaz RDF Playground - 03/01/2021

- Visualizar el documento descrito como un grafo (ver Figura 2.5).
- Realizar consultas al documento utilizando SPARQL [9].
- Razonar sobre el contenido del documento utilizando OWL [10] o RDFS [6].
- Validar/definir reglas sobre el documento utilizando ShEx [17] y SHACL [14].

Es importante recalcar que RDF Playground surge como una iniciativa para construir un ambiente educativo-digital, en el que los y las estudiantes puedan aprender sobre las distintas tecnologías de la Web Semántica, poniendo en práctica todos los conocimientos cubiertos en el curso `"CC7220 - La Web de Datos"`. De aquí surge la necesidad de incluir un Linked Data Browser a RDF Playground.

## 2.3. Linked Data Browsers

La principal característica de un Linked Data Browser (desde ahora **LD Browser**) debe ser permitir la exploración de LD. Al entregarle un IRI al LD Browser, este debe ser capaz de obtener y desplegar la información al desreferenciar el I, además de proveer un mecanismo para explorar recursivamente otros identificadores.

Para el tópico y los laboratorios de Linked Data del curso, se utiliza una herramienta externa; un buscador RDF de la Universidad de Southampton, UK[3]. Si bien, la herramienta cumple con el objetivo de demostrar las capacidades de un LD Browser, la presentación es tosca (ver Figura 2.6) y carece del acercamiento interactivo que caracteriza a RDF Playground.

[3]Q&D RDF Browser - `http://graphite.ecs.soton.ac.uk/browser/`

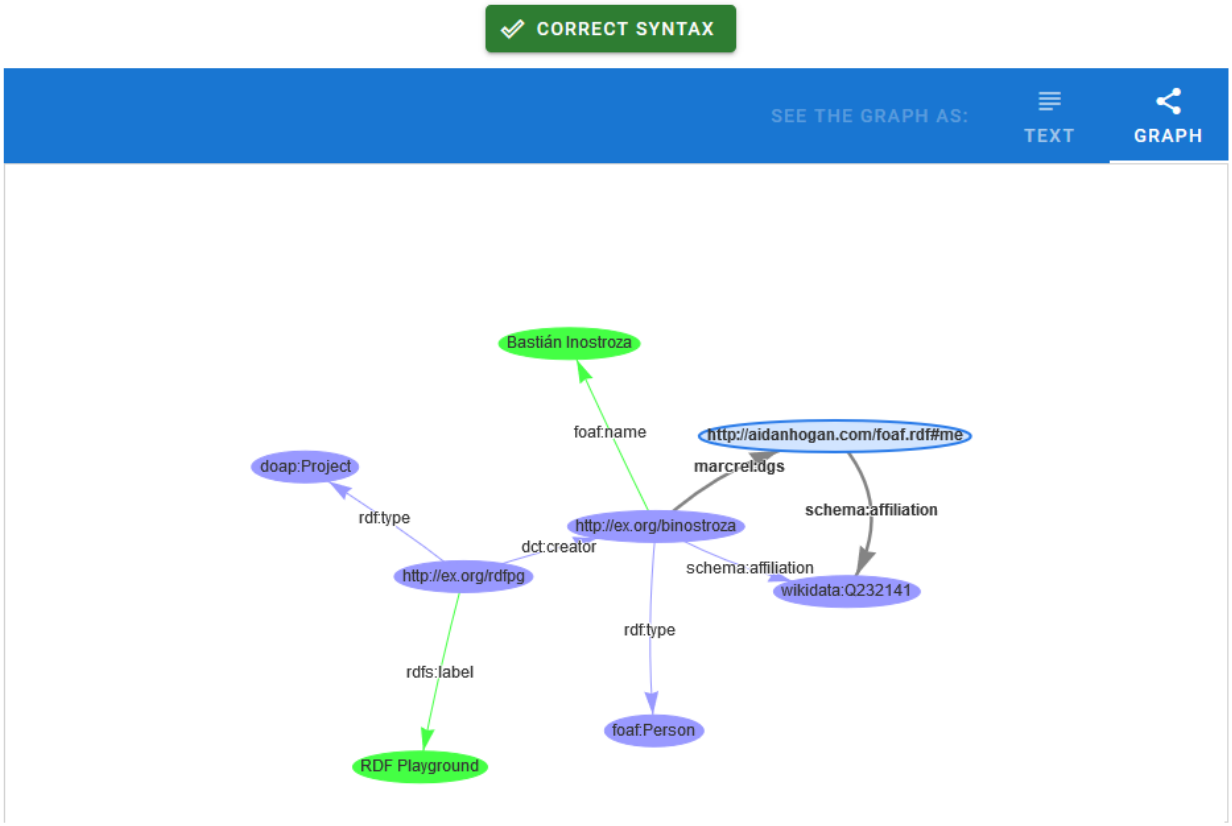



Figura 2.5: Interfaz RDF Playground - 03/01/2021 - Ejemplo de grafo

Por otro lado, debemos entender que la forma de visualizar LD difiere de las típicas visualizaciones de información a las que estamos acostumbrados (tablas, gráficos, texto plano), principalmente porque la estructura de los datos es distinta. Entonces, cobra vital importancia en un LD Browser saber qué es lo que queremos mostrar y cómo.

Un estudio reciente [8] ha comparado distintas herramientas de visualización de LD, mediante distintos casos de uso, y evaluando la funcionalidad de las herramientas frente a estos casos utilizando el dataset de DBpedia.

Sobre los casos de uso, por ejemplo, una distinción importante es el alcance de la búsqueda; ¿Queremos explorar todo el dataset? ¿Solo sobre sus clases? ¿Una instancia en particular? ¿El esquema y la ontología?. Es necesario definir con claridad cuáles serán los escenarios en los que interactuarán usuario y herramienta.

Los casos de uso, los métodos de evaluación, y los distintos tipos de herramientas son explicados con mayor detalle en el libro *Linked Data Visualization: Techniques, Tools, and Big*

[4]Recurso University of Chile en DBpedia - `http://dbpedia.org/resource/University_of_Chile`


Figura 2.6: Resultado de buscar el recurso University_of_Chile[4] en *Quick and Dirty RDF Browser*

*Data* [16], de los mismos autores. Los resultados del estudio indican que hay una gran variedad de acercamientos tanto en la forma de visualizar la información, como de las capacidades y funcionalidades provistas en las herramientas. Algunas logran destacar en varios casos de usos, mientras que otras sobresalen en un caso de uso menos logrado por las demás aplicaciones.

Entre las herramientas evaluadas se encuentran *LodLive*[5], *LODmilla*[6], *LodView*[7]. Notar que la lista completa de herramientas evaluadas en el libro [16] es mucho más extensa, e incluye un desglose de las herramientas por varios parámetros, como el tipo de información y el formato de visualización.

A continuación se describen y muestran las interfaces de estas tres herramientas al exponer un documento RDF, en particular, el recurso de "Santiago" de DBpedia.

Partiendo por *LodView* (Figura 2.7), sus características son bastante similares a las de *Q&D RDF Browser*; la forma de visualizar la información es en base a texto, exponiendo los triples del documento y denotando el predicado y el objeto de las expresiones. Respecto a estética, *LodView* es más legible que la herramienta actual.

*LODmilla* y *LodLive*, en cambio, presentan la información del documento en forma de grafos. Aquí, ya podemos ver un acercamiento más visual a la presentación de la información. *LODmilla* (Figura 2.8) carga el recurso como un único nodo, que al interactuar con él, despliega un componente a la derecha de la interfaz, que permite ir agregando más información al grafo. Al costado izquierdo, hay más opciones que permiten modificar el grafo actual; agregando más nodos, buscar contenido en los nodos visibles y en la vecindad de estos, buscar conexiones en la vecindad, y encontrar caminos entre nodos.

[5]LodLive - `http://en.lodlive.it/`
[6]LODmilla - `http://lodmilla.sztaki.hu/lodmilla/`
[7]LodView - `https://lodview.it`

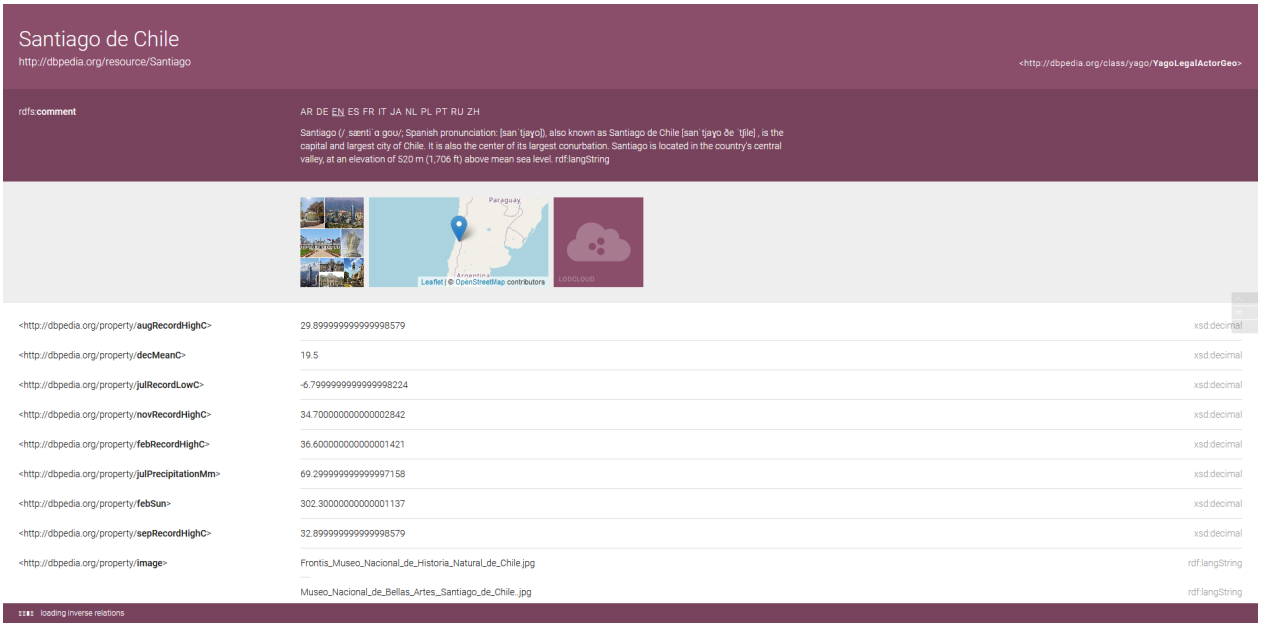



Figura 2.7: Interfaz LodView - Recurso "Santiago"

*LodLive* (Figura 2.9) también carga el recurso buscado como un nodo único, pero a diferencia de *LODmilla*, su información se encuentra alrededor del nodo en forma de pequeños elementos. Al posar el puntero sobre estos elementos, un tooltip indicando el IRI relacionado aparece en la esquina inferior izquierda. Al ir haciendo clic en estos elementos, podemos ir expandiendo el grafo y observar las relaciones que existen entre los distintos nodos. Cada nodo contiene una configuración propia y una descripción, accesibles mediantes sus respectivos iconos. Como opciones adicionales, puede generar relaciones inversas, expandir la visualización automáticamente, recuperar las imágenes en la búsqueda, y obtener los recursos de geolocalización. Además, cuenta con una leyenda de qué significa cada icono/elemento en el grafo.

Respecto a la interacción usuario-herramienta de estos LD Browsers, *LodLive* es el que se siente más fluido, dinámico, e interactivo, por lo que se convierte en una buena alternativa para escoger características que debe tener el LD Browser de RDF Playground. Un único detalle es que ninguno de estos puede buscar recursos en el endpoint de Wikidata, dataset usado en la mayoría de los laboratorios del curso.

El presente trabajo opta por una visualización de LD mediante grafos, por lo tanto, es importante definir las posibles interacciones del usuario con el LD Browser, el tipo de información que se desplegará, y las funcionalidades disponibles para explorar los datos.

Finalmente, la ventaja de integrar un LD Browser con RDF Playground es que permitirá aplicar las otras funcionalidades del sistema sobre los datos recolectados durante la navegación de los documentos LD. De esta forma se podrá consultar, razonar, y validar sobre los datos presentes en el navegador, algo que no permiten los otros LD Browsers discutidos en esta sección.

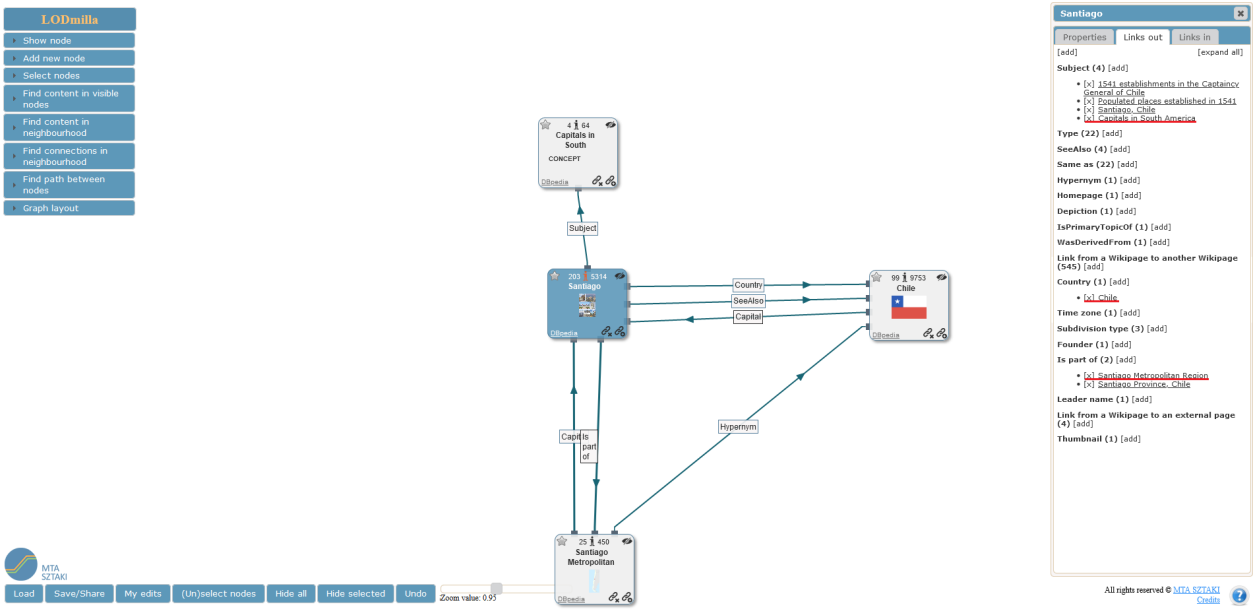



Figura 2.8: Interfaz LODmilla

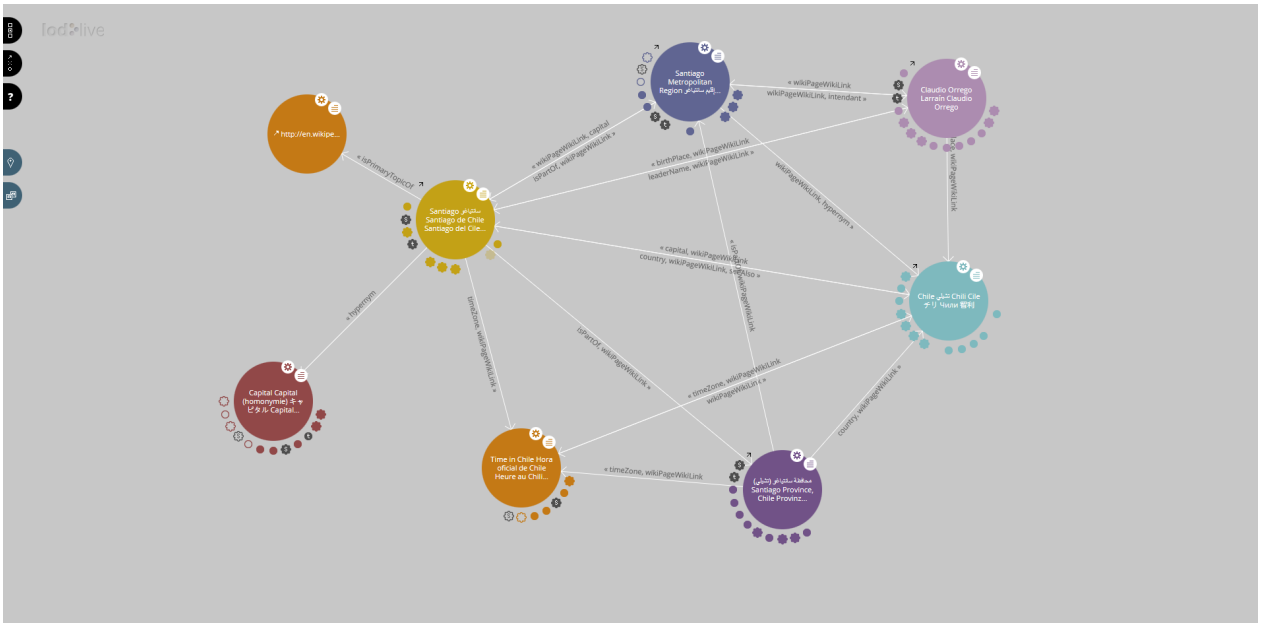



Figura 2.9: Interfaz LodLive

# Capítulo 3

# Desarrollo del Software

La finalidad del presente trabajo consiste en extender RDF Playground para que los y las estudiantes del curso puedan interactuar con datos RDF en la Web. El presente capítulo divide el desarrollo de dicha extensión en dos grandes secciones. En primera instancia, una descripción detallada del problema, y los requisitos que debe satisfacer la solución para resolverlo efectivamente. Y una segunda sección donde se describe la solución implementada y los cambios realizados, así como detalles del sistema que sirven de apoyo para la comprensión de la solución.

## 3.1. Extendiendo RDF Playground

Es claro que existen diferencias entre los procesos de diseñar una aplicación desde cero, y modificar, actualizar, o modificar una que ya existe. El primer proceso permite, en la mayoría de los casos, dibujar sobre una hoja en blanco qué solución es la que mejor se ajusta a los requerimientos del proyecto. Una buena investigación permite seleccionar y utilizar las tecnologías que mejor se adapten a los distintos requisitos de un sistema.

Por el otro lado, al trabajar sobre una aplicación ya construida se tiende a mantener el ecosistema de tecnologías ya dispuesto, sin modificarlo demasiado. Las nuevas mejoras y características que se desarrollen a futuro consideran en su implementación la arquitectura del sistema que se está modificando.

La problemática que aborda este trabajo de título corresponde a esta última categoría. RDF Playground requiere de un LD Browser; debe satisfacer la necesidad de exploración y visualización de LD, como parte de las actividades de aprendizaje del curso. Para cumplir plenamente con esta tarea, su implementación debe considerar la arquitectura y las tecnologías utilizadas para desarrollar RDF Playground.

### 3.1.1. Necesidad y Oportunidad

RDF Playground surge del deseo y la necesidad por reunir en una herramienta educativa las distintas tecnologías que hacen posible la implementación de una Web Semántica. Este proceso de unificación permite eliminar la dependencia de herramientas externas en las actividades prácticas de los laboratorios del curso, permitiendo a sus estudiantes *jugar* y aprender sobre la Web Semántica en un ambiente controlado y desarrollado para ellos/as.

Dado que el uso de este software se enmarca en un contexto de **aprendizaje**, los requisitos que satisface esta herramienta están directamente relacionados con los objetivos de aprendizaje del curso. De ahí el valor agregado de RDF Playground respecto a la *suma* de las distintas herramientas utilizadas en iteraciones anteriores del curso.

Es por eso que para cumplir con las demandas que exigen las actividades prácticas se necesita implementar una forma de visualizar y explorar LD en RDF Playground, añadiendo funcionalidades e interfaces necesarias al sistema, y considerando el carácter educativo de la aplicación.

### 3.1.2. Desafíos

Desarrollar una solución que satisface dichas necesidades, es un camino que presenta varios obstáculos a superar. La naturaleza de estas problemáticas es variada, y en algunos casos quedan fuera del alcance de la solución. A continuación se listan los desafíos principales de este proyecto:

- **Stack de RDF Playground**: Exceptuando las tecnologías básicas de desarrollo Web (JS, HTML, CSS), es la primera oportunidad de trabajar con el lenguaje de programación *Kotlin*, y los frameworks *Vue.js*, *Spring-Boot*, y *Apache Jena*. La carencia de experiencia compromete medianamente el avance de una solución, a cambio de una buena oportunidad de aprendizaje.
- **Documentos RDF - Descarga**: Para acceder a los documentos de la Web, se deben realizar peticiones a distintos servidores que contienen esta información. El sistema de RDF Playground se encontrará supeditado a la respuesta de los distintos servidores que ofrecen documentos RDF. Por lo tanto, mientras más grande sea el tamaño de un documento más tiempo se demorará el servidor en responder la petición, y más tiempo se demorará RDF Playground en procesar su información para ser utilizada en su interfaz por su usuario/a. Notar que el tiempo de descarga del documento RDF depende de la conexión entre cliente-servidor y del tamaño del documento; es independiente de la solución.
- **Documentos RDF - Visualización**: Cada documento RDF publicado contiene una cantidad de información que depende de la persona u organismo que lo publica; son libres de decidir la cantidad de información que desean compartir. El Linked Data Browser debe ser capaz de visualizar todo tipo de documentos; pequeños, medianos, grandes, muy grandes, y gigantes. Para ello, es de vital importancia implementar un

mecanismo que se adapte a la diversidad de documentos RDF. De lo contrario, la visualización por grafo del documento conlleva a dos problemas principalmente: es sumamente difícil explorar un grafo con más de cientos de nodos y aristas, y el tiempo que se demora el sistema en construir gráficamente este grafo. Una posible solución consiste en visualizar el documento de manera incremental: partir desde un nodo central (que corresponde al recurso que describe el documento buscado), e ir agregando paulatinamente las propiedades que deseamos visualizar.

- **Integrar LD**: Uno de los objetivos de aprendizaje del laboratorio correspondiente a Linked Data es observar cómo se puede integrar información de LD en un solo grafo. Es decir, el sistema debe ser capaz de integrar varios documentos RDF en su visualización. Una forma de abordar esto es proporcionando los mecanismo para descargar documentos cuyo identificador se encuentre visible en el grafo.

### 3.1.3. Requisitos del Sistema

Para cumplir con los objetivos y superar los desafíos, la solución propuesta debe extender RDF Playground al incorporarle un LD Browser, reutilizando el ecosistema de tecnologías del sistema.

En primer lugar, se deben identificar las funcionalidades que debe tener dicha extensión, y que correspondan consistentemente con los principios de LD [4]:

- Cualquier recurso que deseemos buscar, y por lo tanto que sea identificable, debe tener un nombre. Necesitamos conocer al menos uno de estos nombres para iniciar una búsqueda.
- Al buscar ese nombre en la Web obtendremos información que lo describe. El LD Browser debe ser capaz de recuperar esa información, y generar un modelo de datos RDF para identificar los distintos *statements* que describen al concepto buscado.
- Este modelo de datos debe ser visualizable por sus usuarios/as, explorando las relaciones descritas del recurso buscado.
- Si en las relaciones que describen al recurso buscado se utilizan otros identificadores, el LD Browser debe ser capaz de ofrecer un mecanismo para solicitar, recuperar, y visualizar la información que este identificador pueda ofrecer.

A partir de esto, se elabora una lista de requerimientos que debe satisfacer el navegador.

#### Requerimientos no Funcionales

- El Linked Data Browser debe ser integrado al conjunto de herramientas de RDF Playground.
- El Linked Data Browser debe responder al usuario en un tiempo suficientemente corto, de no más de 10 segundos para documentos RDF que no sobrepasen las 10000 tuplas.

- El Linked Data Browser soportará la capacidad necesaria de solicitudes para cumplir con su funcionalidad durante las actividades prácticas del curso. Es decir, el sistema debe soportar que al menos 40 personas trabajen concurrentemente en él.
- El Linked Data Browser debe ser sencillo de entender y utilizar para los y las estudiantes del curso `"CC7220 - La Web de Datos"`.

### Requerimientos Funcionales

- El usuario debe poder ingresar un IRI como entrada.
- El sistema debe validar que la entrada dereferencie efectivamente un documento RDF.
- El sistema debe generar un objeto a partir del IRI que contenga la información del recurso.
- El sistema debe visualizar mediante un grafo dicho objeto.
- El sistema debe permitir la interacción del usuario con el grafo.
- La interacción del usuario se define como:
  - Visualizar la información del recurso, total o parcialmente.
  - Seleccionar la información a visualizar en el grafo.
  - Desplazar los elementos del grafo.
  - Revisar información particular de nodos y arcos.
  - Expandir el grafo en función de la información que contiene (exploración incremental) al desreferenciar otras IRIs.

## 3.2. Linked Data Browser

RDF Playground es una aplicación Web compuesta por dos sistemas independientes que interactúan entre sí; un *backend* API REST que implementa la mayor parte de las funcionalidades de la aplicación, y un *frontend* donde ocurre la interacción humano-computador. La Figura 3.1 describe concisamente la arquitectura del sistema.

La implementación de un LD Browser requiere la extensión de estos sistemas para que la aplicación sea capaz de manejar las solicitudes que requieren obtener un documento RDF de la Web a partir de un identificador, desplegar y visualizar esta información como un grafo, explorar su contenido, y expandir el grafo al desreferenciar otros identificadores que se encuentren en el documento.

### 3.2.1. RDF Playground

Para desarrollar correctamente la extensión, el primer paso consiste en identificar y comprender los elementos que conforman RDF Playground, la estructura de ambos sistemas (*backend* y *frontend*), y la forma en que se comunican.

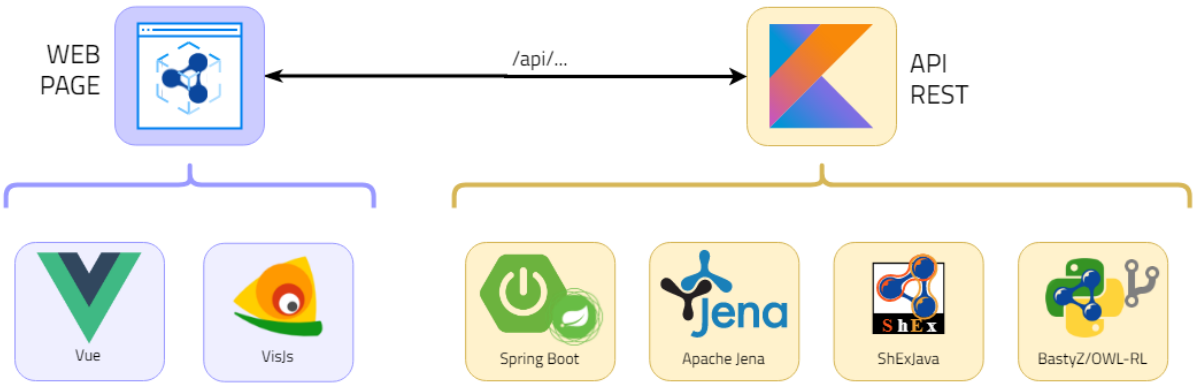



Figura 3.1: Arquitectura del sistema RDF Playground [12]

En primer lugar, el *backend* del sistema se encarga de implementar el grueso del procesamiento de RDF Playground, además de manejar las peticiones enviadas desde el *frontend*.

Está escrito mayormente en **Kotlin**[1], lenguaje de programación moderno y conciso que, entre otras características, opera sobre la máquina virtual de Java, permitiendo la interoperabilidad con aplicaciones, librerías y *frameworks* escritos en Java. Cuando hablemos de Kotlin, nos estaremos refiriendo indirectamente a Java.

De esta manera, es posible utilizar el framework de código abierto para desarrollo de aplicaciones de Web Semántica, **Apache Jena**[2]. Jena ofrece un abanico de APIs que permiten procesar datos RDF e invocar sus funcionalidades directamente desde Kotlin. Además, se utiliza el framework **Spring Boot**[3] para construir la API que expondrá el *backend*. Spring Boot es sumamente práctico para controlar las peticiones HTTP de manera sencilla y transparente.

El *backend* también incluye dos librerías importantes que suplementan ciertas funcionalidades que Jena no provee. Estas corresponden a: **ShexJava**[4] que permite validar datos RDF, y **OWL-RL**[5] que implementa razonamiento tanto para RDFS como OWL2.

Por último, cuenta con una implementación del lenguaje de descripción de grafos **DOT** que permite a Jena transformar sus modelos RDF a un formato que puede manejar el *frontend* para su visualización.

[1] `https://kotlinlang.org/`
[2] `https://jena.apache.org/`
[3] `https://spring.io/projects/spring-boot`
[4] `https://github.com/iovka/shex-java`
[5] `https://github.com/RDFLib/OWL-RL`

Por su lado, el *frontend* es la plataforma de interacción entre estudiante y la aplicación. Está desarrollado utilizando el framework *Vue.js*[6] en conjunto con *Vuetify*[7], haciendo posible una interfaz amigable y sencilla de crear y utilizar. Adicionalmente, la librería *Vis.js*[8] proporciona las funcionalidades necesarias para la creación y visualización de grafos, característica de vital importancia para la implementación del LD Browser.

### 3.2.2. Arquitectura de la solución

La implementación de LD Browser se integra e implementa directamente tanto en el *backend* como el *frontend*, utilizando los lenguajes y tecnologías de RDF Playground. La Figura 3.2 ilustra como está organizada la aplicación en términos de funcionalidad.

Como se aprecia en esta figura, el *backend* se estructura en 5 elementos, que corresponden con las funcionalidades que se mencionaron en la subsección anterior. Dado que Jena provee los mecanismos necesarios para obtener un modelo a partir de una IRI, la implementación del LD Browser en el lado del *backend* corresponde a resolver una petición que trae consigo una IRI. La respuesta a dicha petición, como se detalla más adelante, envía al *frontend* dos versiones del modelo; formateadas en DOT y TTL respectivamente.

La API expone los distintos métodos que posee en cuatro categorías de control: controlador de modelos (`/api/model`), controlador del razonador (`/api/owl`), controlador de base de datos de triples (`/api/tdb/`), y controlador de formas (`/api/shape/`). Siguiendo esta lógica, el controlador implementado para manejar las peticiones del LD Browser quedan bajo el esquema de controlador de modelos (`/api/model`), y se encargará de resolver la IRI de la petición en los modelos que requiere el *frontend*.

Además, se realizan diversas modificaciones a la implementación de DOT. Estas surgen como consecuencia de la dificultades presentes al momento de parsear el documento DOT a la estructura de datos que maneja *Vis* para describir el grafo, y a la visualización de este. En general, se resuelven conflictos respecto a caracteres incompatibles, formato en que se presentan los literales según su *datatype*, e información adicional que pueda contener el documento (como el idioma del literal) que no estaba incorporada en la escritura del modelo en DOT.

En cuanto al *frontend*, la implementación de RDF Playground en *Vue* se encuentra escrita en un único documento, lo que dificulta el mantenimiento y la extensión del código. Se optó por escribir la interfaz del LD Browser como un **componente** propio que se inyecta en el componente principal de la aplicación. Este componente debe incorporar todos los elementos necesarios que permitan al estudiante interactuar con LD al ingresar una IRI.

Para ello, se utiliza el espacio disponible en la sección izquierda de la aplicación para que el usuario pueda ingresar una IRI al sistema y que éste pueda generar una visualización mediante grafos del documento RDF desreferenciado. Una vez generada esta información en el *backend*, el componente del LD Browser despliega una nueva interfaz que permite la interacción con

[6] `https://vuejs.org/`
[7] `https://vuetifyjs.com/`
[8] `https://visjs.org/`

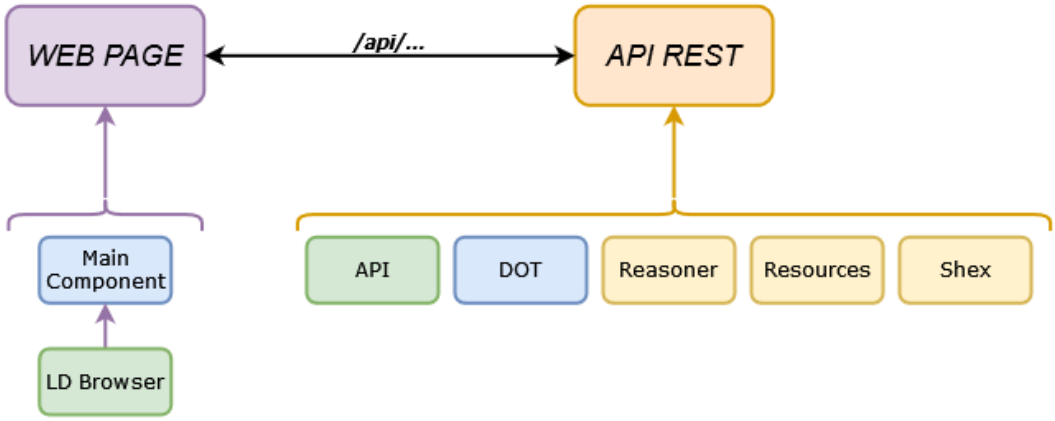



Figura 3.2: Arquitectura de la solución - LD Browser. En verde aquellos elementos en los que se crearon archivos, y en azul aquellos que fueron modificados.

el documento RDF. Es decir, el diseño de este componente organiza la información del LD Browser en dos capas: una que permite realizar las peticiones de documentos RDF a través de IRIs, y una que permite interactuar con este documento.

Esta interacción, grafo mediante, es lo más importante de la segunda capa. Para esto, es fundamental considerar la diversidad de documentos que podemos encontrar y definir una forma de interacción que se adapte a la mayoría de casos. La solución presentada en este trabajo corresponde a una visualización incremental de la información; no se observa la totalidad del grafo, sino que se parte de un grafo con un único nodo al que se le van agregando nodos y arcos. La adición de elementos al grafo esta supeditada a las propiedades de cada documento.

Por otro lado, dado que las IRIs referenciadas en cada documento pueden estar vinculadas a potenciales documentos RDF, la interacción con los nodos IRI del grafo permite añadir nuevos documentos a la lista de propiedades seleccionables para visualizar, agrupadas bajo el nombre de su documento respectivo.

Al abordar el desarrollo de la interfaz de esta forma se mantiene la lógica del LD Browser en su propio componente, y establece un ejemplo de desarrollo de componentes para futuras mejoras en el sistema. En particular, la refactorización del componente principal en múltiples componentes correspondiente a las distintas secciones de RDF Playground.

### 3.2.3. Lógica del servidor

Como se señaló anteriormente, el controlador del LD Browser queda implementado en la sección de controladores de modelo de la API, bajo la dirección (`/api/model/browse`).

Las consultas recibidas por la API se realizan mediante peticiones HTTP con formato

JSON, y cuyas respuestas, en la mayoría de los casos, también suelen usar este formato. La consulta y respuesta que maneja el controlador del LD Browser también sigue este esquema, el que se detalla a continuación:

- **Documento RDF en Web a DOT/Turtle**
  **Petición**: HTTP: `POST` */api/model/browse*
  **Tipo de contenido de la petición**: *application/json*
  **Contenido de la petición**:

| Parámetro | Tipo | Descripción |
|---|---|---|
| url | Texto | Nombre a buscar en la Web |

  **Tipo de contenido de la respuesta**: *application/json*
  **Contenido de la respuesta**:

| Respuesta | Tipo | Descripción |
|---|---|---|
| browse_error | Texto | Primer error encontrado en el proceso. |
| model_dot | Texto | Descripción del grafo en DOT |
| model_ttl | Texto | Descripción del grafo en Turtle |

Jena se encarga internamente de realizar la petición para obtener el documento RDF y transformarlo a un objeto *Model*, que representa un modelo RDF. Este es luego transcrito al formato DOT y Turtle, los que serán enviados al *frontend*.

Respecto al formato DOT y su implementación, se realizaron ajustes desde dos perspectivas. En una primera instancia, los cambios apuntaban al tratamiento de nodos blancos en los documentos RDF de la Web, los que generaban conflicto al momento de parsear el archivo DOT en el *frontend* de la aplicación.

Al resolver dicha problemática, las siguientes modificaciones a la implementación de DOT apuntaban a recuperar más información del documento, en particular describir el idioma y *datatype* de cada literal cuando estuviesen presentes.

### 3.2.4. Interfaz e Interacción

Con Jena a cargo de responder la solicitud HTTP, obteniendo el documento RDF de la Web y devolviendo su representación en un formato amigable para *Vis*, el *frontend* debe encargarse de procesar y desplegar toda esta información de manera coherente con el resto de la aplicación; debe acoplarse a la interfaz de RDF Playground.

A continuación se describe el proceso esperado al buscar un documento RDF en la Web y las distintas acciones que se pueden ejecutar en la interfaz de visualización del grafo, junto a figuras que ilustran la implementación del LD Browser.

El primer paso consiste en asignarle una posición en las pestañas de selección de funcionalidad, optando por la menos cargada (al lado izquierdo de la aplicación). Luego, como se

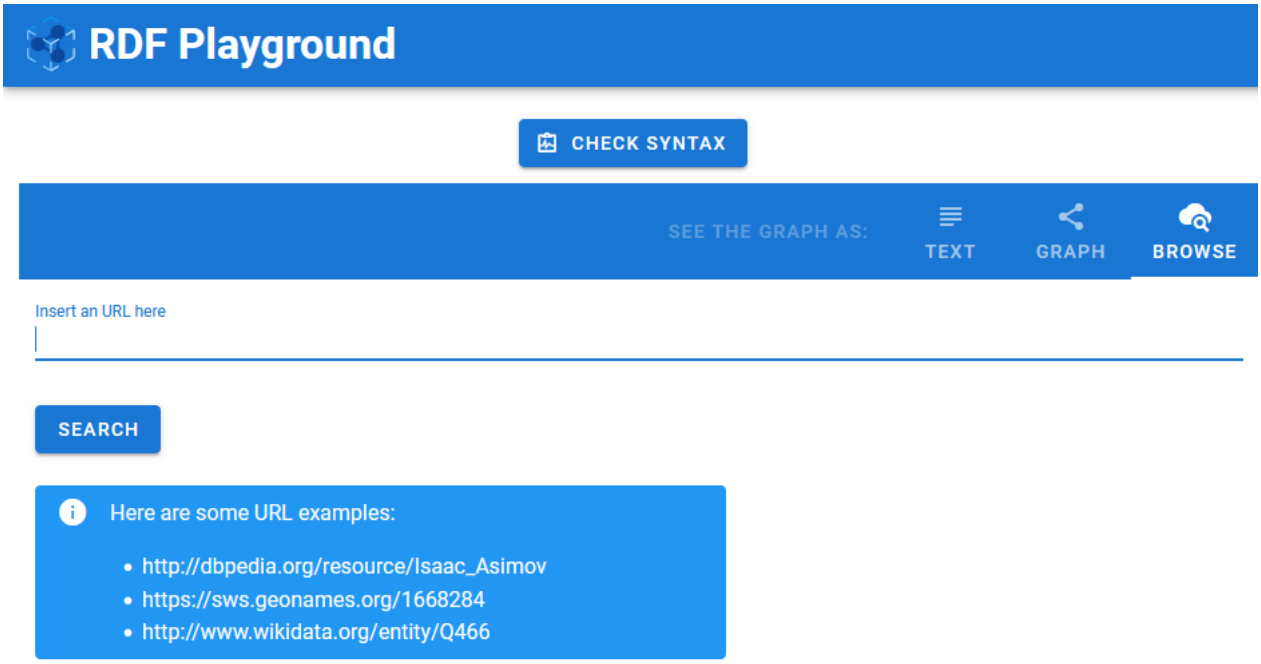



Figura 3.3: Sección del LD Browser en RDF Playground.

aprecia en la Figura 3.3, su sección ofrece la posibilidad de ingresar un URL (IRI) para buscar en la Web, además de incorporar una sección de ejemplos con URLs de distintos datasets.

Una vez decidido el URL a buscar se envía la petición al *backend*, el que responderá con el documento RDF correspondiente en formato DOT. En este punto, el *frontend* toma control del documento y comienza a procesar su información para desplegar un grafo con el *nodo central* del documento, y una lista seleccionable que contiene las propiedades de aquellos documentos que han sido descargados a partir de la URL inicial (que es justamente el nodo central del grafo). La Figura 3.4 ilustra el resultado de este proceso.

Esta interfaz está implementada de manera similar a la visualización en pantalla completa de grafos que RDF Playground ofrece en otras secciones, siendo la principal diferencia la lista de **Documentos y Propiedades** que se encuentra al costado izquierdo del grafo. Como su nombre indica, esta lista permite seleccionar qué información agregar al grafo, a partir de las propiedades que se encuentren en el documento.

Al seleccionar un elemento de la lista, el sistema agregará los arcos correspondientes con esa propiedad junto a los nodos que no se encuentren presentes en el grafo. De igual forma, al deseleccionar un elemento de la lista, se removerán del grafo aquellos arcos que se correspondan con dicha propiedad, y los nodos que queden desconectados del resto del grafo. Se implementa la visualización del documento de esta manera incremental para evitar la sobrecarga inicial del grafo, ya que de lo contrario el sistema tardaría mucho tiempo en desplegar la totalidad de información.

En la Figura 3.5 se observa el efecto en el grafo al seleccionar una propiedad. Adicionalmente, esta figura muestra una de las posibles interacciones con la sección del grafo que permite *Vis*, que consiste en la aparición de ventanas de texto al posar el puntero sobre los

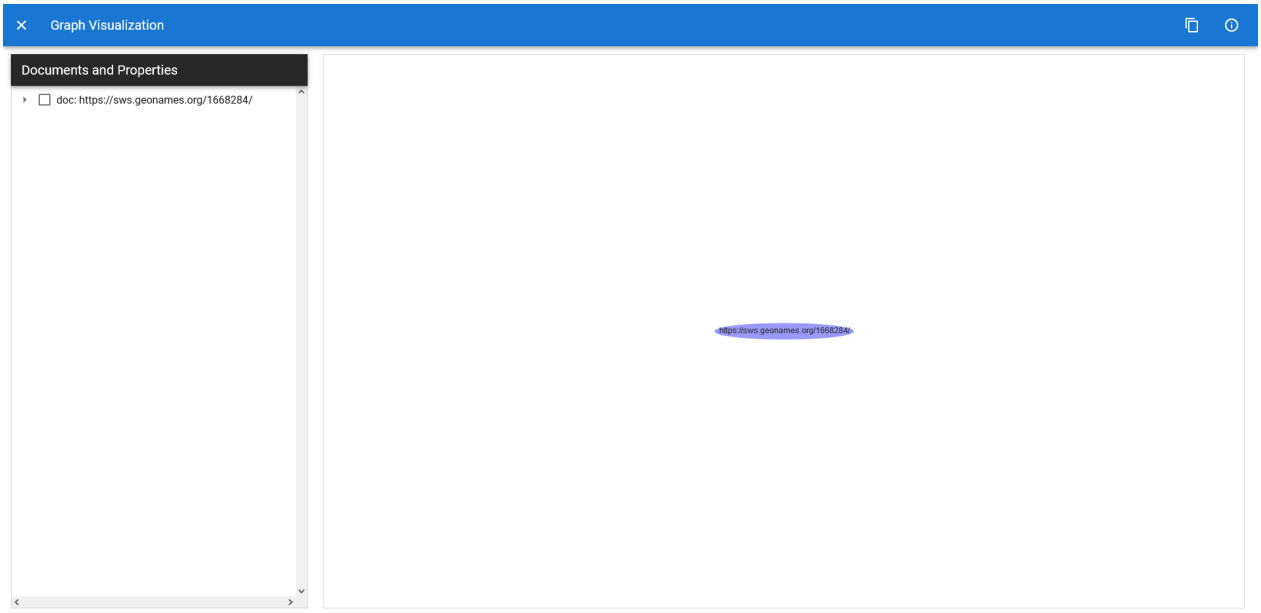



Figura 3.4: Interfaz de exploración de datos RDF del LD Browser.

elementos del grafo, evento conocido como *mouseover* u *on hover*. La información que estas ventanas proporcionen depende del elemento: los arcos indicarán a qué documento pertenecen, literales mostrarán su *datatype* e idioma según disponibilidad (y el texto completo en caso de literales muy extensos), y el resto de nodos su identificador completo.

La otra interacción importante tiene relación con aquellos nodos que potencialmente pueden contener un documento RDF. Al hacer doble-clic sobre alguno de estos nodos, una pequeña ventana emergerá para confirmar que se desea extender el grafo utilizando un determinado URL, correspondiente con el nodo seleccionado. Si Jena encuentra un documento RDF, la lista de Documentos y Propiedades se actualizará, dejando el nuevo documento al final de la lista. La Figura 3.6 expone la ventana que permite explorar más documentos, mientras que la Figura 3.7 muestra el resultado de agregar múltiples documentos.

Finalmente, la barra superior de la interfaz permite volver a la pantalla principal de RDF Playground, copiar en sintaxis *Turtle* los documentos presentes en la lista al portapapeles, y abrir una ventana de ayuda donde se explican brevemente las interacciones posibles dentro del LD Browser. Además, en la sección principal se habilitará un botón que permite regresar a la visualización del grafo.

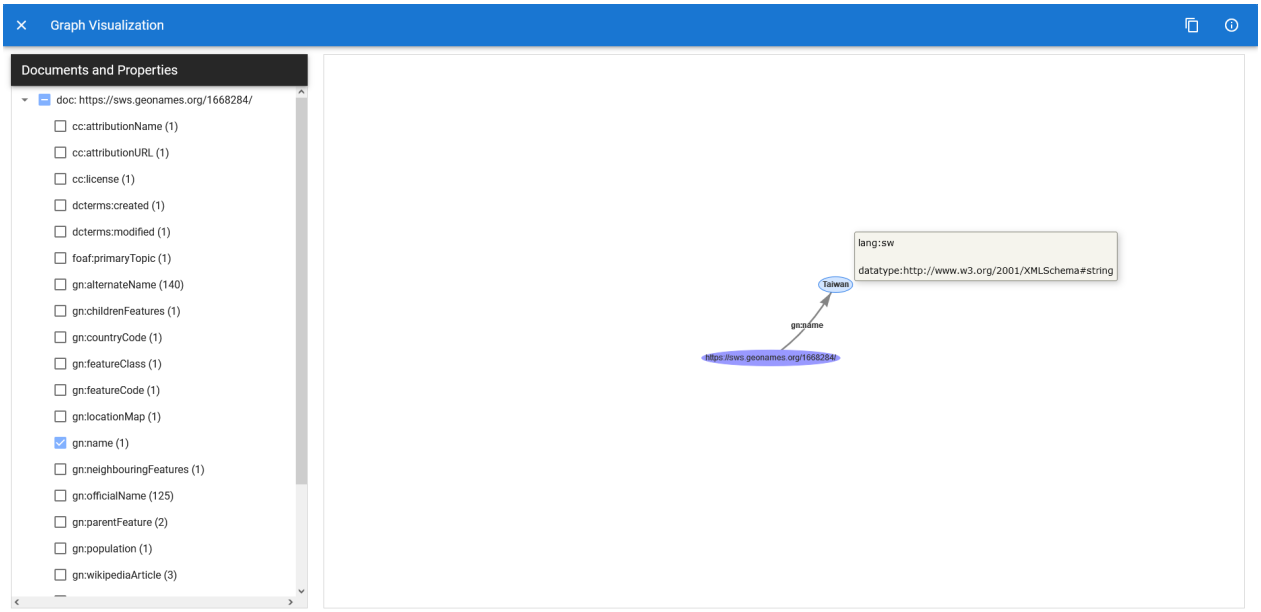



Figura 3.5: Ejemplo de interacción *hover* en nodos.

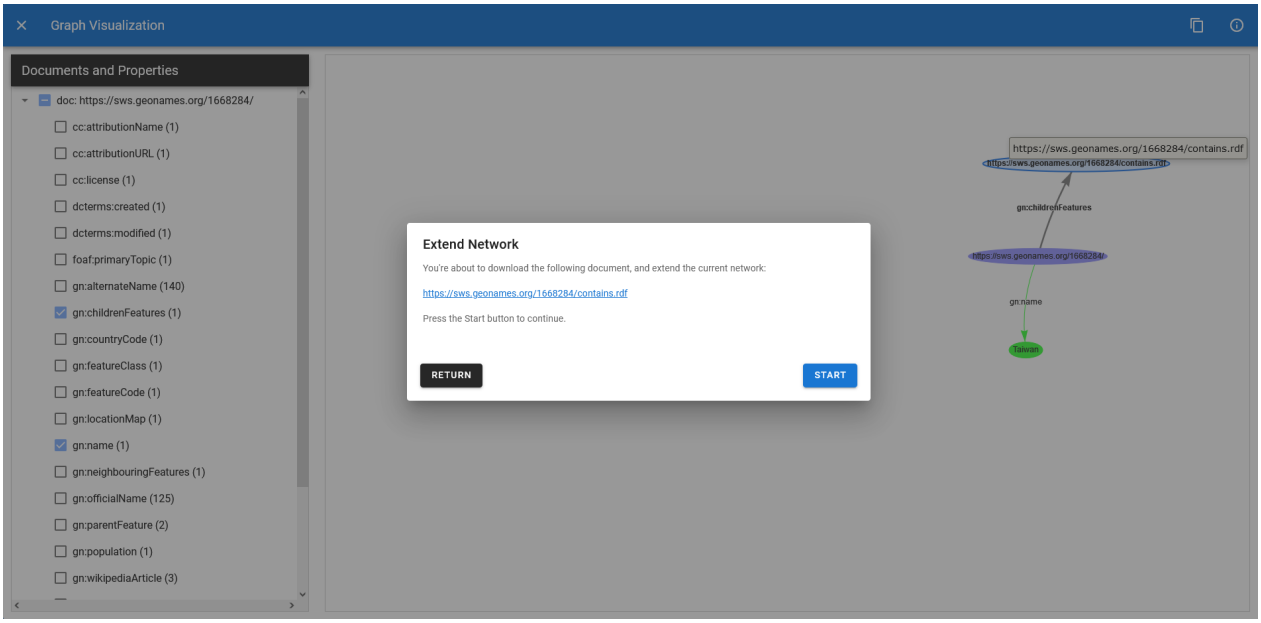



Figura 3.6: Ventana de diálogo para extender el grafo actual.

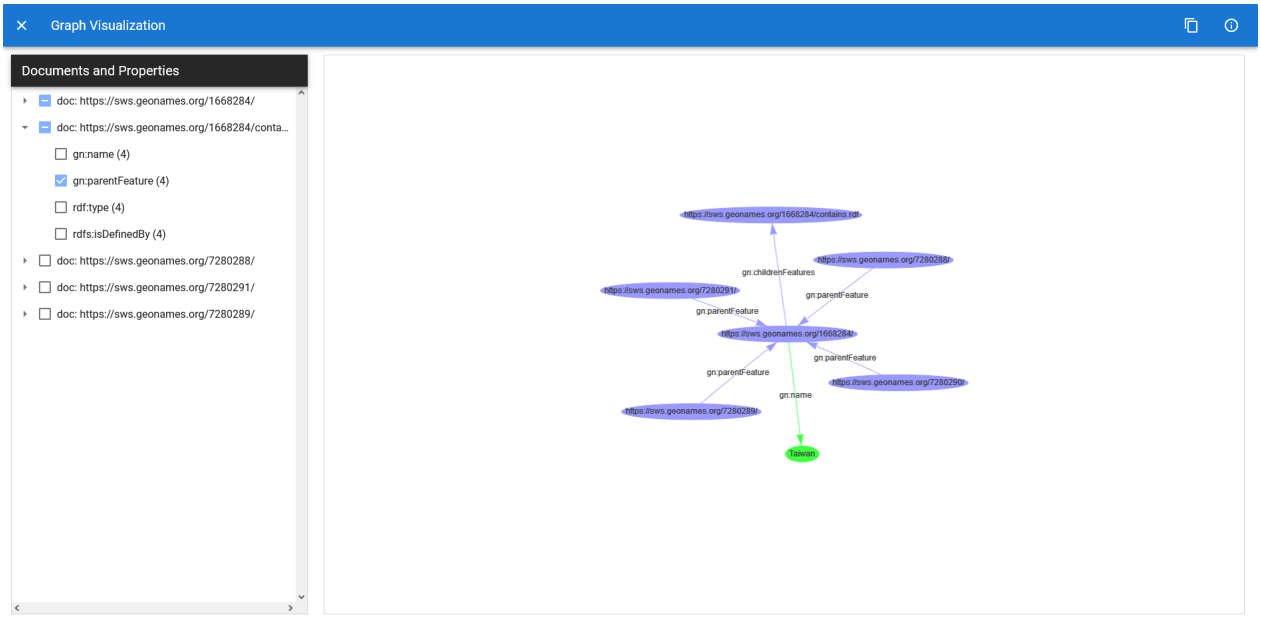



Figura 3.7: Ejemplo de vista al extender el grafo.

# Capítulo 4

# Evaluación del Sistema

Como en todo proyecto, la evaluación de la solución es un factor importante para validar el trabajo realizado. En este caso, dicha evaluación tiene dos aristas principales: el tiempo de respuesta del sistema, y su usabilidad. En esta sección se describe la metodología usada para evaluar estos aspectos, y los resultados de dichas evaluaciones.

## 4.1. Tiempo de Respuesta del Sistema

Para realizar esta evaluación, hay que tomar en consideración las funcionalidades implicadas en la capacidad de respuesta del sistema. En el caso de LD Browser, estas son la descarga y procesamiento del documento RDF, y la carga de elementos al grafo. Es decir, el controlador del LD Browser que expone el *backend*, y la visualización y carga del grafo en el *frontend*.

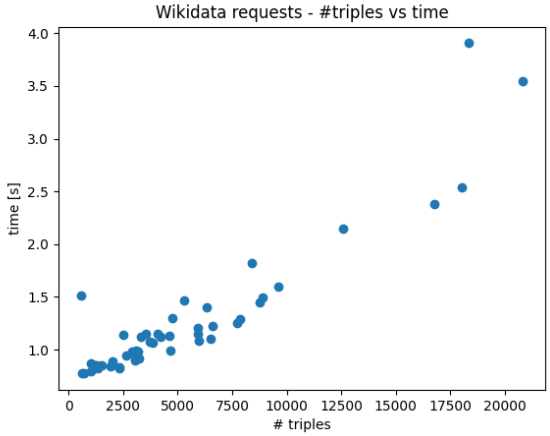
### 4.1.1. Backend

En primer lugar se evaluará el tiempo de respuesta del *backend*; cuánto se demora en desreferenciar y descargar el documento, procesarlo en un modelo de Jena, y transcribir dicho modelo en formato DOT y TTL. Estos tres procesos corresponden a la funcionalidad del controlador del LD Browser.

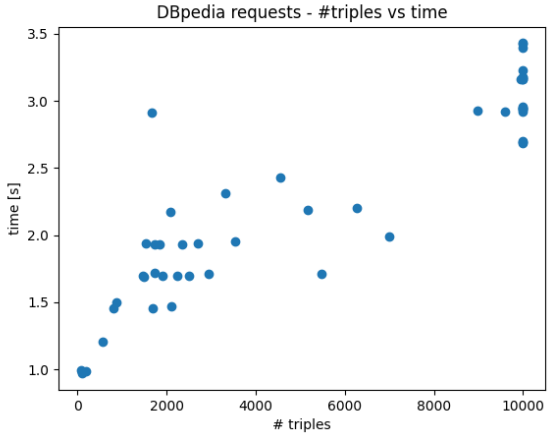
La evaluación consiste en medir el tiempo que demora el sistema en responder peticiones correspondientes a tres conjuntos de IRIs; *DBpedia*, *GeoNames* y *Wikidata* respectivamente. Cada conjunto de IRIs intenta abarcar la mayor cantidad de categorías en el dominio del dataset, es decir, que los documentos correspondan a distintas esferas del conocimiento en el marco de cada dataset.

Para ello, se considera que el factor para discernir estos documentos es la cantidad de información que contienen. En otras palabras, la cantidad de tuplas descritas en el documento. La Figura 4.1 muestra los resultados para los tres datasets distintos. Para cada dataset, se

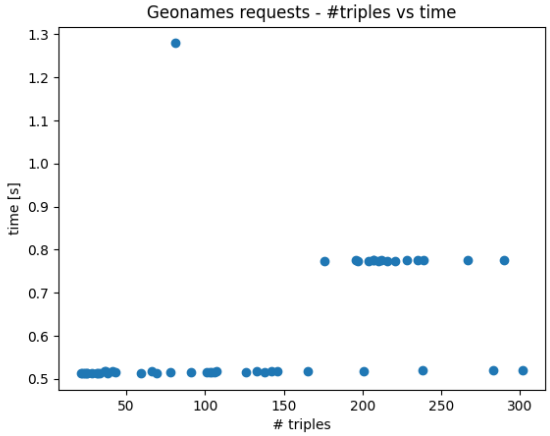
han escogido 50 IRIs (ver Apéndice A) bajo los siguientes criterios:




(a) Wikidata Sample




(b) DBpedia Sample




(c) GeoNames Sample

Figura 4.1: Tiempo de respuesta para distintos datasets según número de triples.

- **Wikidata y DBpedia**: Primero se trata de abarcar distintas categorías de interés (geografía, música, videojuegos, literatura, ciencia, entre otras), y luego buscar recursos que puedan contener más o menos información dependiendo de su naturaleza. Por ejemplo, un país suele contener más información que una de sus ciudades no-capitales.
- **GeoNames**: A diferencia de los datasets anteriores que son multidominio, GeoNames se especializa en información geográfica. El criterio en este caso fue abordar países, ciudades, y lugares geográficos importantes de distintas regiones del mundo.

Los resultados sugieren que puede existir una dependencia lineal entre la cantidad de triples y el tiempo de respuesta del sistema respecto a documentos de *Wikidata* y *DBpedia*. En el caso de *GeoNames* se aprecia que el tiempo es relativamente el mismo para todos los documentos, aunque cabe mencionar que el tamaño de estos es bastante menor que el de los documentos de los otros dos datasets, y abarcan órdenes de magnitud más pequeños (sus documentos no superan las 500 tuplas).

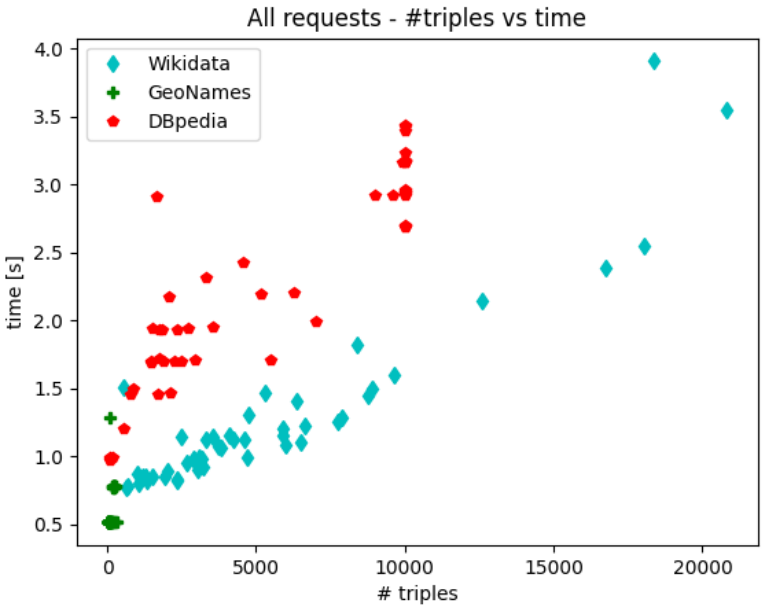



Figura 4.2: Gráfico agregado de los distintos datasets.

Por supuesto, el tiempo de descarga del documento y de la creación del modelo en Jena son factores que pueden variar: la conexión a internet que posea el servidor donde se instale RDF Playground, su capacidad de procesamiento del documento, y la disponibilidad y respuesta del servidor de cada dataset.

Al juntar los resultados en la Figura 4.2 podemos observar que el tiempo de respuesta del sistema aumenta a medida que aumenta la cantidad de triples. Aun cuando se presentan diferencias en tiempo de ejecución para documentos con misma cantidad de tuplas, se puede apreciar una tendencia que indica que al aumentar la cantidad de triples, aumenta el tiempo de respuesta del sistema. Además, la mayor parte de esta selección de IRIs cuenta con una cantidad de tuplas menor o igual a $10^4$ tuplas, siendo las excepciones en el gráfico algunos documentos de *Wikidata*. En general, es un tiempo de respuesta aceptable, y que no excede los 10 segundos.

### 4.1.2. Frontend

Respecto al *frontend*, el tiempo evaluado corresponde a la recarga del grafo cada vez que se selecciona o deselecciona una propiedad, ya que esto implica redefinir nodos y arcos en la red de *Vis* y volver a dibujarla. Como consecuencia de esto, deseleccionar siempre tomará menos tiempo, ya que la red ahora tiene un menor número de elementos.

La evaluación consiste medir el tiempo que se demora el sistema en, dado uno o más documentos, añadir una propiedad con un determinado número de arcos. Luego, seleccionar distintas propiedades de los documentos para abarcar un rango numérico de arcos añadidos al grafo en un paso, donde el paso consiste en hacer seleccionar una propiedad en la lista de documentos y propiedades para agregar sus arcos al grafo. La Figura 4.3 ilustra el

comportamiento de los datos obtenidos.

Se puede observar que hasta los primeros 100 arcos aproximadamente el tiempo crece linealmente, luego en el tramo intermedio hasta los 150 arcos hay bastante diferencia entre distintas propiedades agregadas, y luego el tiempo cae sobre los 150 arcos añadidos. Es difícil decir con certeza qué provoca este comportamiento, pero a continuación se mencionan algunas posibilidades.

En primer lugar, si dos propiedades agregan el mismo número de arcos, estos pueden perfectamente agregar un número distinto de nodos al grafo, variando la cantidad total de elementos añadidos al grafo. Esto podría responder la disparidad que existe entre propiedades que agregan un número similar de arcos y que tienen distintos tiempos de respuesta del sistema. Podemos decir entonces que la *forma* que adopta el grafo al agregar un conjunto de arcos y nodos, y por consiguiente el procesamiento que realiza *Vis-Network* para determinar cómo se desplegará el grafo, no es homogéneo para propiedades con el mismo número de arcos (desde cierta cantidad en adelante).

Otro fenómeno importante que ocurre en la evaluación, superado el umbral de los 100 arcos añadidos, es que el tiempo de respuesta obtenido no es 100 % confiable. Si bien la medición de tiempo se calcula justo antes de que se revisen los arcos y nodos a añadir, y justo después de que se llame a la función para dibujar el grafo, en realidad el navegador sigue procesando el despliegue del grafo en la mayoría de los casos. Es decir, el tiempo de ejecución debiese ser mayor que el expuesto en el gráfico, pasado cierto número de arcos y nodos añadidos. Esto explicaría la abrupta caída en el tiempo de ejecución pasado los 150 arcos, y parte de la diferencia en el rango $[100, 150]$ arcos.

Por último, es poco deseable agregar tantos elementos directamente al grafo, ya que dificulta y perjudica la exploración de los datos. Entonces, y en base a los resultados, se vuelve necesario definir una cantidad límite de arcos que se pueden agregar en cada operación para considerar un comportamiento aceptable del sistema. Se profundiza en posibles soluciones para este problema en el siguiente capítulo.

## 4.2. Usabilidad

Si bien, el tiempo de respuesta es parte de la usabilidad de un sistema, en esta sección se busca evaluar la experiencia de futuros usuarios y usuarias de RDF Playground para determinar su grado de satisfacción con respecto a la aplicación.

### 4.2.1. Metodología de evaluación

Para evaluar la usabilidad del sistema, se utilizará la **Escala de Usabilidad del Sistema** [7] (*SUS*), que ya se ha utilizado anteriormente para evaluar el RDF Playground. Esta metodología consiste en evaluar mediante 10 preguntas, la satisfacción del participante respecto a su experiencia en la realización una tarea asignada en la aplicación. A estas preguntas

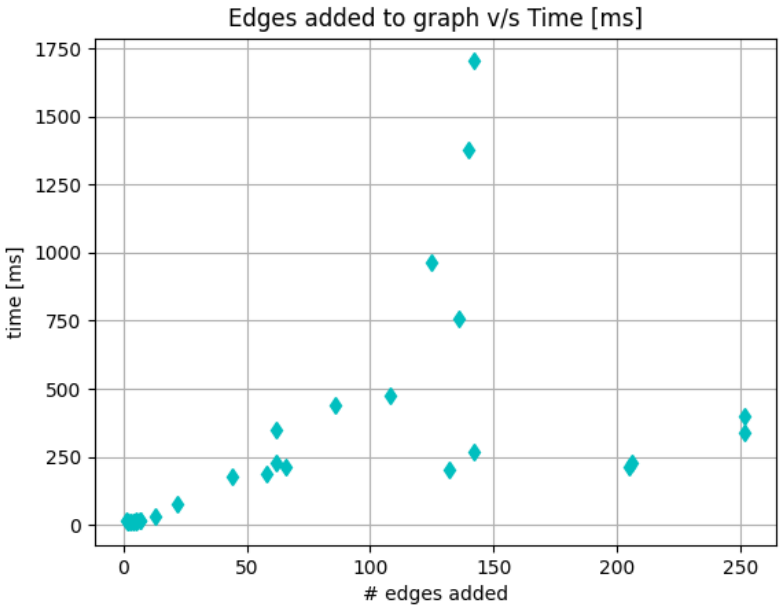



Figura 4.3: Gráfico de la evaluación de rendimiento del *frontend*

se les asigna un puntaje del 1 al 5, desde "*Muy en desacuerdo*"hasta "*Muy de acuerdo*", que luego se ponderan en una escala de puntuación de 0 a 100.

*SUS* es sumamente popular, debido a lo sencillo que es, y la rapidez con que se pueden obtener resultados, permitiendo detectar rápidamente problemas generales de un sistema, pese a que no entrega un diagnóstico preciso de dichos problemas.

Las preguntas de la encuesta se han adaptado levemente a las particularidades del sistema, y son las siguientes:

1. Me gustaría usar frecuentemente este sistema.
2. Encontré este sistema innecesariamente complejo.
3. Creo que el sistema es fácil de usar.
4. Necesitaré ayuda para usar este sistema.
5. Los elementos del sistema se encuentran bien distribuidos.
6. El sistema es inconsistente.
7. Creo que es fácil aprender a utilizar este sistema.
8. Utilizar este sistema fue muy difícil para mi.
9. Siento confianza al utilizar este sistema.
10. El sistema me dificultó el aprendizaje.

| Sentencia a evaluar | Respuestas | | | | | Total | |
|---|---|---|---|---|---|---|---|
| | R1 | R2 | R3 | R4 | R5 | P | DE |
| Me gustaría usar frecuentemente este sistema. | 4 | 3 | 5 | 4 | 4 | 4 | 0.71 |
| Encontré este sistema innecesariamente complejo. | 1 | 2 | 1 | 1 | 3 | 1.6 | 0.89 |
| Creo que el sistema es fácil de usar. | 5 | 4 | 5 | 5 | 3 | 4.4 | 0.89 |
| Necesitaré ayuda para usar este sistema. | 2 | 3 | 2 | 2 | 5 | 2.8 | 1.3 |
| Los elementos del sistema se encuentran bien distribuidos. | 5 | 5 | 4 | 4 | 2 | 4 | 1.22 |
| El sistema es inconsistente. | 1 | 2 | 1 | 1 | 2 | 1.4 | 0.55 |
| Creo que es fácil aprender a utilizar este sistema. | 4 | 4 | 4 | 5 | 4 | 4.2 | 0.45 |
| Utilizar este sistema fue muy difícil para mi. | 1 | 2 | 1 | 1 | 2 | 1.4 | 0.55 |
| Siento confianza al utilizar este sistema. | 3 | 4 | 5 | 4 | 5 | 4.2 | 0.84 |
| El sistema me dificultó el aprendizaje. | 1 | 1 | 4 | 1 | 1 | 1.6 | 1.34 |
| ***Puntaje SUS*** | 87.5 | 75 | 85 | 90 | 62.5 | 80 | 11.32 |

Figura 4.4: Tabla de resultados encuesta de usabilidad SUS - 15/11/2021

El puntaje final de esta encuesta puede expresarse como:

$$\text{SUS}_{\text{score}} = \left( \sum(\text{Puntaje impares} - 1) + \sum(5 - \text{Puntaje pares}) \right) \cdot 2{,}5$$

Este puntaje nos indicará el grado de usabilidad del sistema. Un estudio realizado sobre este tipo de encuesta de usabilidad [1], permite interpretar el puntaje en base a una caracterización cualitativa. Un puntaje SUS $> 50$ corresponde a una usabilidad aceptable, mientras que un SUS $> 68$ es considerado el promedio, una usabilidad estándar.

### 4.2.2. Participantes

El público objetivo corresponde a los y las estudiantes del curso `"CC7220 - La Web de Datos"`, e integrantes del cuerpo docente, tanto del presente año como de anteriores, que se encuentren disponibles para participar de la encuesta. A la fecha de entrega de este informe han participado 5 estudiantes. Se entiende una baja participación debido al carácter opcional del laboratorio de Linked Data, y que la actividad del laboratorio sigue abierta.

### 4.2.3. Evaluación

Pese a la poca representatividad que representan 5 estudiantes respecto a un curso que usualmente tiene 40 estudiantes o más, se ha realizado el calculo para obtener el puntaje SUS, proceso mencionado anteriormente. Los resultados se encuentran en la Tabla/Figura 4.4.

En términos generales, el puntaje SUS de cada participante esta sobre el promedio, indicando un buena buena usabilidad general, con un promedio de puntaje SUS de 80. Por

supuesto, este valor está sobreestimado, lo que se refleja en el valor de la desviación estándar.

En el detalle, la pregunta "Necesitaré ayuda para usar este sistemarecibe el puntaje promedio más bajo en términos de usabilidad. Aunque esto no permite diagnosticar los problemas que el sistema pueda tener, si refleja el hecho que el sistema requiere ajustes en torno a brindar más información del sistema al usuario; que sirva como una guía para el correcto uso del LD Browser.

Respecto a comentarios específicos de los y las participantes, se menciona que falta información retroactiva cuando se cargan datos en el sistema. Por ejemplo, al desreferenciar un nodo dentro del grafo para obtener su documento RDF, la aplicación no alerta que ha efectuado exitosamente el proceso, aunque si alerta cuando encuentra errores. También se menciona que en grafos con muchos elementos se tienden a solapar nodos y arcos, dificultando la visualización de la información y perdiendo claridad.

Pese a los pocos resultados, esta encuesta establece una promesa de que el sistema es potencialmente una buena herramienta de apoyo al aprendizaje, y que sus usuarios finales podrán interactuar con LD en un ambiente controlado y seguro.

## 4.3. Resultados y notas importantes

Tanto la evaluación de rendimiento como de usabilidad arrojan buenos resultados. El tiempo de respuesta del *backend* tiene varios factores que escapan a la aplicación en sí misma, pero que en general es aceptable (o 'no excesivo').

En cuanto al *frontend*, su desempeño requiere una revisión de la implementación de la visualización del grafo, ya que no es deseable que el sistema se comporte de manera no-responsiva frente a una cantidad mediana de elementos agregados. Aunque cabe destacar que para propiedades que agreguen un menor número de elementos, el sistema responde correctamente y un tiempo casi inmediato.

La evaluación de usabilidad arroja que la usabilidad general del sistema es buena, aunque requiere afinar detalles respecto a educar al usuario a utilizar correctamente el LD Browser.

# Capítulo 5

# Conclusión

RDF Playground es una herramienta educativa, por lo que un elemento esencial en su desarrollo es propiciar una interfaz sencilla e interactiva que facilite el aprendizaje de sus estudiantes. La extensión desarrollada en el marco de esta memoria, el LD Browser, tiene como objetivo facilitar y apoyar la labor educativa en el curso `CC7220 - La Web de Datos`.

En cuanto a los objetivos planteados en esta memoria, se logran completar casi en su totalidad, dejando parcialmente completo la evaluación y validación del sistema debido a la poca cantidad de respuestas obtenidas en la encuesta de usabilidad, y la incorporación de funcionalidades que mejoren el rendimiento y usabilidad del sistema a partir de su evaluación.

El LD Browser incorporado a RDF Playground permite abordar el tópico de Linked Data al proporcionar los mecanismos de búsqueda, visualización y exploración de documentos RDF en la Web. Permite ir explorando poco a poco, propiedad a propiedad, el documento RDF de manera que no sobrecargue el grafo con información desde el inicio de la visualización. También es capaz de recuperar documentos cuyos identificadores se encuentren en el documento inicial como parte de una declaración, y añadirlo a la lista de documentos y propiedades para explorarlo en el grafo.

El rendimiento del sistema tiene ciertas limitantes que escapan a esta primera iteración del LD Browser, en particular la descarga del documento RDF de la Web a través de Jena, y la visualización del grafo mediante *Vis*. Respecto a la descarga, el problema surge cuando el documento a descargar cuenta con una cantidad superior a $10^4$ triples, aunque es probable que sean pocos los documentos RDF que sean del orden de los $10^5$ triples o más.

La visualización del grafo parece ser el aspecto más importante en cuanto a la usabilidad del sistema, ya que es en la interacción del usuario con el grafo donde se apoya principalmente la labor educativa. Agregar directamente más de 100 arcos (con sus respectivos nodos) a la vez no parece ser una buena estrategia, tanto por el costo computacional como por la capacidad de exploración, ya que mientras más sobrecargado se encuentre el grafo más difícil resultará explorarlo.

La primera versión extendida de RDF Playground, que incluye el LD Browser, ha sido puesta en producción durante la semana del 13 de septiembre, quedando a disposición de los

y las estudiantes, quienes podrán utilizar el LD Browser durante su laboratorio de Linked Data. La aplicación se encuentra disponible en el sitio usual de RDF Playground: `http://rdfplayground.dcc.uchile.cl`.

Por último, se han implementado algunas correcciones y mejoras al sistema posterior a su puesta en producción, las que están en espera de ser añadidas a producción para evitar diferencias sustanciales en las respuestas a la encuesta de usabilidad, de tal forma que se todas se basen en la misma versión del sistema. Esto debido a que el tiempo límite para completar el laboratorio finaliza el día 17 de noviembre de 2021. Estas mejoras y correcciones incluyen:

- La resolución de un bug que surgía con ciertos nombres de nodos que no podían ser abreviados localmente en el modelo de Jena, y que generaba un grafo con nodos con nombre incompleto.
- Mejora visual del listado de ejemplos.
- La posibilidad de intercambiar el grafo por una visualización del documento RDF en sintáxis TTL.
- La apertura de la ventana de ayuda al ingresar por primera vez al LD Browser en la sesión actual.

## 5.1. Trabajo futuro

Entre posibles mejoras a futuro, pueden elaborarse numerosas soluciones que mejoren el rendimiento de la visualización. Por ejemplo, optimizar la funcionalidad que se encarga de incorporar los datos seleccionados al grafo, y/o expandir la lista de documentos y propiedades para seleccionar individualmente los arcos que se agregan al grafo.

La primera se refiere al hecho de que el sistema determina a partir de los arcos qué nodos deben agregarse al grafo, y luego *redibuja* el grafo con la nueva información. Esto podría solucionarse al utilizar los arreglos especiales que provee *Vis-Dataset*, los que permitirían actualizar instantáneamente el grafo cada vez que se agrega un nodo o un arco al arreglo de *Vis*.

La segunda se enfoca en modificar la interacción del usuario con la lista de documentos y propiedades para otorgar más libertad al momento de seleccionar qué elementos se agregan al grafo en un clic. Es decir, generar más hojas en la lista de tal forma que, para cada propiedad se puedan seleccionar individualmente sus arcos. Un potencial problema que presenta este acercamiento, consiste en que no hay una manera directa de nombrar a los arcos utilizando la información del documento. Se les puede otorgar un identificador local, pero este no tendrá relación con los nodos que relaciona dicho arco.

Otra opción considera la posibilidad de filtrar literales por idioma, los que suelen sobrecargar ciertas propiedades; *esencialmente* son múltiples triples idénticos, pero en idiomas

distintos. También se puede considerar el uso de prefijos externos para abreviar IRIs que sean demasiado extensos, sin necesidad de truncarlos. Por último, algunos comentarios relevantes:

- La visibilidad de los documentos y propiedades en su lista. A veces el nombre de los documentos y/o sus propiedades exceden el ancho de la lista y son truncados automáticamente por *Vuetify*.
- Un bug que ocurre con documentos RDF que poseen un literal de tipo *date* (fecha), cuyo rango se encuentra fuera de la implementación de Jena/Java. Por ejemplo, el documento de la Tierra en Wikidata declara que la Tierra se formó hace 4500 millones de año, y el literal que contiene esa información es de tipo fecha, lo que se transforma en un error del *backend*.
- Sobre el solape que se produce en el grafo cuando hay presentes una cantidad importante de nodos y arcos, puede solucionarse modificando y probando distintas configuraciones en las opciones del grafo.

# Bibliografía


[1] Aaron Bangor, Philip Kortum, and James Miller. Determining What Individual SUS Scores Mean: Adding an Adjective Rating Scale. *J. Usability Studies*, May 2009. https://dl.acm.org/doi/10.5555/2835587.2835589.

[2] David Beckett, Tim Berners-Lee, Eric Prud'hommeaux, and Gavin Carothers. RDF 1.1 Turtle. W3C Recommendation, February 2014. https://www.w3.org/TR/turtle/.

[3] Tim Berners-Lee. Semantic Web. Design Issues, September 1998. https://www.w3.org/DesignIssues/Semantic.

[4] Tim Berners-Lee. Linked Data. Design Issues, July 2006. https://www.w3.org/DesignIssues/LinkedData.

[5] Diego Berrueta and Jon Phipps. Best Practice Recipes for Publishing RDF Vocabularies. W3C Working Group Note, August 2008. http://www.w3.org/TR/swbp-vocab-pub/.

[6] Dan Brickley and R.V. Guha. RDF Schema 1.1. W3C Recommendation, February 2014. https://www.w3.org/TR/rdf-schema/.

[7] John Brooke. SUS: A quick and dirty usability scale. *Usability Eval. Ind.*, 189, November 1995. https://www.researchgate.net/publication/228593520_SUS_A_quick_and_dirty_usability_scale.

[8] Federico Desimoni, Nikos Bikakis, Laura Po, and George Papastefanatos. A Comparative Study of State-of-The-Art Linked Data Visualization Tools. In *VOILA 2020*, October 2020. http://ceur-ws.org/Vol-2778/paper1.pdf.

[9] Steve Harris and Andy Seaborne. SPARQL 1.1 Query Language. W3C Recommendation, March 2013. https://www.w3.org/TR/sparql11-query/.

[10] Pascal Hitzler, Markus Krötzsch, Bijan Parsia, Peter F. Patel-Schneider, and Sebastian Rudolph. OWL 2 Web Ontology Language Primer (Second Edition). W3C Recommendation, December 2012. https://www.w3.org/TR/owl2-primer/.

[11] Aidan Hogan. Linked Data & the Semantic Web Standards. In Andreas Harth, Katja Hose, and Ralf Schenkel, editors, *Linked Data Managment*, pages 3–48. Chapman and Hall/CRC, 2014. https://doi.org/10.1201/b16859.

[12] Bastián Inostroza. Interfaz para el curso La Web de Datos. Universidad de Chile, 2020. http://repositorio.uchile.cl/handle/2250/179645.

[13] Bastían Inostroza. RDF Playground. http://rdfplayground.dcc.uchile.cl/.

[14] Holger Knublauch and Dimitris Kontokostas. Shapes Constraint Language (SHACL). W3C Recommendation, July 2017. https://www.w3.org/TR/shacl/.

[15] Ora Lassila and Ralph R.Swick. RDF Model and Syntax Specification.

[16] Laura Po, Nikos Bikakis, Federico Desimoni, and George Papastefanatos. Linked Data Visualization: Techniques, Tools, and Big Data. *Synthesis Lectures on the Semantic Web: Theory and Technology*, March 2020. http://dx.doi.org/10.2200/S00967ED1V01Y201911WBE019.

[17] Eric Prud'hommeaux, Iovka Boneva, Jose Emilio Labra Gayo, and Gregg Kellogg. Shape Expressions Language 2.1. W3C Final Community Group Report, October 2019. http://shex.io/shex-semantics/.

[18] Guus Schreiber and Yves Raimond. RDF 1.1 Primer. W3C Working Group Note, June 2014. https://www.w3.org/TR/rdf11-primer/.

[19] W3C. Semantic Web. https://www.w3.org/standards/semanticweb/.

[20] W3C. WWW Project History. https://www.w3.org/History.html.

# Apéndice A

# Anexo

## A.1. IRIs utilizados en la evaluación del Sistema

### A.1.1. Wikidata

- http://www.wikidata.org/entity/Q2
- http://www.wikidata.org/entity/Q88888888
- http://www.wikidata.org/entity/Q131007
- http://www.wikidata.org/entity/Q298
- http://www.wikidata.org/entity/Q12585
- http://www.wikidata.org/entity/Q8341
- http://www.wikidata.org/entity/Q1107
- http://www.wikidata.org/entity/Q673
- http://www.wikidata.org/entity/Q196600
- http://www.wikidata.org/entity/Q189382
- http://www.wikidata.org/entity/Q866
- http://www.wikidata.org/entity/Q1
- http://www.wikidata.org/entity/Q323
- http://www.wikidata.org/entity/Q11452
- http://www.wikidata.org/entity/Q937
- http://www.wikidata.org/entity/Q5
- http://www.wikidata.org/entity/Q215627
- http://www.wikidata.org/entity/Q7889

- http://www.wikidata.org/entity/Q160738
- http://www.wikidata.org/entity/Q175173
- http://www.wikidata.org/entity/Q862490
- http://www.wikidata.org/entity/Q420
- http://www.wikidata.org/entity/Q2845
- http://www.wikidata.org/entity/Q3706669
- http://www.wikidata.org/entity/Q729505
- http://www.wikidata.org/entity/Q5741069
- http://www.wikidata.org/entity/Q828
- http://www.wikidata.org/entity/Q21198
- http://www.wikidata.org/entity/Q1466064
- http://www.wikidata.org/entity/Q2013
- http://www.wikidata.org/entity/Q466
- http://www.wikidata.org/entity/Q623
- http://www.wikidata.org/entity/Q9061
- http://www.wikidata.org/entity/Q128309
- http://www.wikidata.org/entity/Q8274
- http://www.wikidata.org/entity/Q1069
- http://www.wikidata.org/entity/Q337535
- http://www.wikidata.org/entity/Q8486
- http://www.wikidata.org/entity/Q15180
- http://www.wikidata.org/entity/Q21980377
- http://www.wikidata.org/entity/Q42177
- http://www.wikidata.org/entity/Q20718
- http://www.wikidata.org/entity/Q884
- http://www.wikidata.org/entity/Q193639
- http://www.wikidata.org/entity/Q60368
- http://www.wikidata.org/entity/Q453156
- http://www.wikidata.org/entity/Q849824
- http://www.wikidata.org/entity/Q820088
- http://www.wikidata.org/entity/Q12567
- http://www.wikidata.org/entity/Q25337
- http://www.wikidata.org/entity/Q362

### A.1.2. DBpedia

- http://dbpedia.org/resource/University_of_Chile
- http://dbpedia.org/resource/Santiago
- http://dbpedia.org/resource/Chile
- http://dbpedia.org/resource/Spanish_language
- http://dbpedia.org/resource/Kathleen_Ferrier_(politician)
- http://dbpedia.org/resource/Politician
- http://dbpedia.org/resource/Jorge_Garc%C3%ADa_Granados
- http://dbpedia.org/resource/Jazz
- http://dbpedia.org/resource/Charlie_Parker
- http://dbpedia.org/resource/Miles_Davis
- http://dbpedia.org/resource/EMI
- http://dbpedia.org/resource/World_of_Warcraft
- http://dbpedia.org/resource/Category:Massively_multiplayer_online_role-playing_games
- http://dbpedia.org/resource/Final_Fantasy
- http://dbpedia.org/resource/Square_Enix
- http://dbpedia.org/resource/Anime
- http://dbpedia.org/resource/Japan
- http://dbpedia.org/resource/Unitary_state
- http://dbpedia.org/resource/Earth
- http://dbpedia.org/resource/Oxygen
- http://dbpedia.org/resource/Chemical_elements
- http://dbpedia.org/resource/Atom
- http://dbpedia.org/resource/Chemistry
- http://dbpedia.org/resource/Branch
- http://dbpedia.org/resource/Person
- http://dbpedia.org/resource/Writer
- http://dbpedia.org/resource/Literature
- http://dbpedia.org/resource/Isaac_Asimov
- http://dbpedia.org/resource/Russian_Soviet_Federative_Socialist_Republic

- http://dbpedia.org/resource/Biochemistry
- http://dbpedia.org/resource/Foundation_(Isaac_Asimov_novel)
- http://dbpedia.org/resource/Science_fiction
- http://dbpedia.org/resource/2001:_A_Space_Odyssey_(film)
- http://dbpedia.org/resource/Stanley_Kubrick
- http://dbpedia.org/resource/The_Bronx
- http://dbpedia.org/resource/United_States
- http://dbpedia.org/resource/North_America
- http://dbpedia.org/resource/Tool_(band)
- http://dbpedia.org/resource/King_Crimson
- http://dbpedia.org/resource/Manga
- http://dbpedia.org/resource/Universe
- http://dbpedia.org/resource/Big_Bang
- http://dbpedia.org/resource/String_theory
- http://dbpedia.org/resource/Computer_science
- http://dbpedia.org/resource/Turing_machine
- http://dbpedia.org/resource/Alan_Turing
- http://dbpedia.org/resource/London
- http://dbpedia.org/resource/Celtic_punk
- http://dbpedia.org/resource/Drum_kit
- http://dbpedia.org/resource/Power_pop
- http://dbpedia.org/resource/Pop_rock

### A.1.3. GeoNames

- https://sws.geonames.org/3871336
- https://sws.geonames.org/3895114
- https://sws.geonames.org/3893894
- https://sws.geonames.org/5128581
- https://sws.geonames.org/5125771
- https://sws.geonames.org/1861060
- https://sws.geonames.org/1850147

- https://sws.geonames.org/11257417
- https://sws.geonames.org/2130038
- https://sws.geonames.org/2128295
- https://sws.geonames.org/2192628
- https://sws.geonames.org/2363254
- https://sws.geonames.org/2661886
- https://sws.geonames.org/2673730
- https://sws.geonames.org/3017382
- https://sws.geonames.org/2988507
- https://sws.geonames.org/2648147
- https://sws.geonames.org/2635167
- https://sws.geonames.org/1814991
- https://sws.geonames.org/1668284
- https://sws.geonames.org/3923974
- https://sws.geonames.org/7729898
- https://sws.geonames.org/6254930
- https://sws.geonames.org/12047392
- https://sws.geonames.org/8261432
- https://sws.geonames.org/7729818
- https://sws.geonames.org/6252001
- https://sws.geonames.org/6255150
- https://sws.geonames.org/6255149
- https://sws.geonames.org/7729892
- https://sws.geonames.org/6295630
- https://sws.geonames.org/357994
- https://sws.geonames.org/360630
- https://sws.geonames.org/361058
- https://sws.geonames.org/1819729
- https://sws.geonames.org/1819727
- https://sws.geonames.org/1819730
- https://sws.geonames.org/1835848

- https://sws.geonames.org/3117735
- https://sws.geonames.org/2921044
- https://sws.geonames.org/3865483
- https://sws.geonames.org/3860259
- https://sws.geonames.org/660013
- https://sws.geonames.org/3144096
- https://sws.geonames.org/3143244
- https://sws.geonames.org/1283416
- https://sws.geonames.org/7500737
- https://sws.geonames.org/8505033
- https://sws.geonames.org/7303837
- https://sws.geonames.org/390903
- https://sws.geonames.org/298795