UNIVERSIDAD NACIONAL DE EDUCACIÓN A DISTANCIA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

Proyecto de Fin de Grado en Ingeniería en Tecnologías de la Información

DESARROLLO DE UN SISTEMA DE ETIQUETADO Y CONSULTA SEMÁNTICOS PARA USUARIOS DE LA UNED

MIGUEL EXPÓSITO MARTÍN

Dirigido por: JOSÉ LUIS FERNÁNDEZ VINDEL

Co-dirigido por: RAFAEL MARTÍNEZ TOMÁS

Curso: 2017-2018: 2ª Convocatoria

DESARROLLO DE UN SISTEMA DE ETIQUETADO Y CONSULTA SEMÁNTICOS PARA USUARIOS DE LA UNED

Proyecto de Fin de Grado de modalidad oferta específica

Realizado por: Miguel Expósito Martín

Dirigido por: José Luis Fernández Vindel

Co-dirigido por: Rafael Martínez Tomás

Tribunal calificador

Presidente: D/Dª.

Secretario: D/Dª.

Vocal: D/Dª.

Fecha de lectura y defensa:

Calificación:

Agradecimientos

A mi familia, por soportarme.

Resumen

El presente proyecto tiene como objeto el desarrollo de un sistema de *playground* para SPARQL y RDF que habrá de servir como herramienta educativa de introducción a las tecnologías de la Web Semántica. Su propósito es facilitar una interfaz de uso sencilla para que usuarios profanos puedan introducirse en este tipo de tecnologías a un nivel básico, permitiendo la realización de actividades académicas sobre manejo sencillo de grafos así como sobre consultas SPARQL al conjunto de datos de trabajo o a *endpoints* externos.

Si bien es cierto que la Web Semántica alcanza hoy en día un estado de madurez razonablemente avanzado, el ritmo de cambio de las tecnologías de *Frontend* en *Javascript* (en adelante, JS) es, cuanto menos, vertiginoso. Unido a ello, se tiene que la mayor parte de las implementaciones de componentes de la Web Semántica han sido desarrolladas en tecnologías de *backend* como *Java* (*Apache Jena*) o *Python* (*rdflib*), no existiendo aún estándares o implementaciones maduras puramente en JS.

Se abordan, por tanto, tres retos: la necesidad de conseguir un producto sencillo y fácilmente utilizable por usuarios no expertos, la dificultad para encontrar componentes maduros que aúnen Web Semántica con *frontend* y la conveniencia de apostar por un *framework* de desarrollo en JS estable y con una curva de aprendizaje suave.

Para dar solución a estos problemas, se ha optado por desarrollar una *Single Page Application* (SPA) con Vue.js (un *framework* JS que se jacta de aglutinar las mejores características de Angular y React, sus principales competidores) e integrarlo con las implementaciones recomendadas por el grupo de trabajo de bibliotecas JS *rdfjs* y con una plataforma de consultas para la web flexible y modular. El sistema está planteado para ejecutarse en un sencillo navegador contemporáneo, descargándose a través de un simple servidor web (siendo este la única infraestructura necesaria para su distribución).

Dado el alto nivel de incertidumbre existente en la implementación de las necesidades del proyecto, se ha optado por utilizar un enfoque metodológico incremental e iterativo basado en Extreme Programming y apoyado sobre tableros Kanban, lo que ha permitido una mejor organización y evolución del proyecto.

El resultado final ofrece un producto básico de introducción a las tecnologías de la Web Semántica y permite pensar en un desarrollo del mismo tan ambicioso como las propias necesidades de los equipos docentes, dado el estado de madurez actual del frontend web.

Abstract

<This project ...>

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

La Web Semántica es un término originalmente acuñado por Tim Berners-Lee en el año 2001. Hasta la fecha, la World Wide Web había sido concebida como una idea de colaboración abierta en la que múltiples contribuciones de varios autores podían tener cabida y ser compartidas universalmente. Dichas contribuciones, realizadas en forma de documentos, estaban dirigidas a personas y no a máquinas o computadores. Berners-Lee vio más allá y propuso su extensión para lograr su manipulación automática; en resumidas cuentas, permitir que agentes inteligentes (programas de ordenador) fueran capaces de encontrar datos y su significado a través de hiperenlaces a definiciones de términos clave y reglas de razonamiento e inferencia lógica.

Para lograr tan ambicioso objetivo, los agentes inteligentes necesitaban tener acceso a contenido y conocimiento estructurado, así como a las reglas de inferencia necesarias. En este contexto, la Web Semántica podía apoyarse en las siguientes tecnologías existentes:

- **RDF** (Resource Description Framework), que permite expresar conocimiento en forma de tripletas (a modo de sujeto, verbo y objeto en una oración) utilizando URIs (Uniform Resource Indicators).
- **XML** (eXtensible Markup Language), que permite la creación de documentos convenientemente etiquetados y con estructura arbitraria.

Sin embargo, un tercer pilar era necesario para resolver la problemática de la existencia de distintos identificadores en distintas bases de datos relacionadas con el mismo significado conceptual. Gracias a las **ontologías**, documentos que definen formalmente las relaciones entre distintos términos, estas

diferencias podían ser salvadas bien mediante el uso de términos estandarizados bien mediante la definición de relaciones conceptuales de igualdad o similitud entre dichos términos.

En palabras del propio Berners-Lee: «Con un diseño adecuado, la Web Semántica puede ayudar en la evolución del conocimiento humano como un todo.» ([1])

1.1. Motivación y objetivos

<Ejemplo referencia a sección 1.3 y a subsección de otro capítulo, que en este caso es un anexo A.1.1. **Para que funcionen correctamente las referencias a otros capítulos al exportar a pdf, debe estar abierto el archivo maestro, abrir/editar el archivo a exportar desde él y exportar desde el maestro**.>

<Ejemplo referencia a figura 1.1>

<Ejemplo referencia a tabla **??**>

<Ejemplo de nomenclatura (POO)>

<Ejemplo de listado de código, ver listado 1.1, lo mejor para utilizarlo es copiar el recuadro y cambiar el código contenido, así conservaremos las opciones que se han establecido, como: mostrar números de línea, quebrar líneas largas. etc. Para ver las opciones y poder modificarlas click derecho sobre el listado y elegir Configuración. Para cambiar el listado contenido copiar del IDE el trozo de código que se desee y pegarlo dentro del listado con Ctrl+May+v, en el menú de Lyx Editar->Pegado especial->Texto simple.>

1.2. Estado actual

El estado actual (o *state of the art*, del inglés) de las tecnologías de la Web Semántica alcanza distintos niveles de madurez en sus implementaciones.

Figura 1.1: Ejemplo de figura con un pie largo para mostrar el uso del ítem del menú de Lyx Insertar->Título breve. La utilidad del título breve se aprecia en la lista de figuras, es ahí donde aparece, si no, aparecería todo el pie en la lista, también es conveniente para títulos de capítulos secciones y demás largo que ocupen más de una línea en la lista de índice correspondiente.(Para insertar salto de línea en pie o en enumeración Ctrl-Enter)
La figura o tabla hay que insertarla dentro de un flotante.
Siempre hay que Insertar->Etiqueta de la figura (fig:Ejemplo-de-figura), tabla etc, para referenciarla desde el texto.

Listado de código 1.1: Código de edit

```
1     public final boolean edit() {
2
3         ArrayList<String> campos = cargaCampos();
4
5         Scanner in = new Scanner( System.in );
6
7     /* La variable resp sera true si el usuario acepta la edicion
8      */
9
10    boolean resp = false;
11        int numero_total_opciones = campos.size() +
              opciones_no_campos;
12        System.out.println("\n" + toString()); // muestra el
              registro
13
14        editLoop: while (true) {
15            editMensaje();
16            System.out.println("Seleccione un numero (1-" +
                  numero_total_opciones + "):  ");
17            int opcion;
18            if ( in.hasNextInt() ) {
19                opcion = in.nextInt();
20                in.nextLine();
21            } else {
22                System.out.println("\nOPCION NO VALIDA. POR FAVOR
                      , INTRODUZCA UN NUMERO.");
23                in.nextLine();
24                continue;
25            }
26            campos = procesaOpcion(campos, opcion, in);
27            if(salir) {
28                if(aceptar)
29                    resp = true;
30                salir = false;
31                aceptar = false;
32                break editLoop;
33            }
34
35        }
36        return resp;
37    }
```

1.2.1. Java

Así, para la plataforma Java se cuenta con Apache Jena[2], un marco de trabajo o *framework opensource* para construir aplicaciones para la Web Semántica o sobre datos enlazados. De entre sus componentes y funcionalidades, cabe destacar:

- TDB: una base de datos de tripletas nativa y de alto rendimiento, con excelente integración con el resto de APIs de Jena.
- ARQ: un motor SPARQL compatible con su versión 1.1 que soporta consultas federadas y búsqueda por texto libre.
- API RDF: una API principal que permite interactuar con RDF para crear y leer grafos y tripletas, así como serializarlas a los formatos más comunes (Turtle, XML, etc.).
- Fuseki: un *endpoint* SPARQL que permite exponer el grafo de trabajo y ofrece interacción tipo REST con tripletas RDF.
- Otras APIs: como las de ontologías e inferencias, que permiten añadir más semántica al modelo y razonar sobre reglas por defecto o personalizadas.

Apache Jena es una solución robusta y muy utilizada en entornos académicos y de producción empresarial con más de quince años de vida.

1.2.2. Python

RDFLib es una biblioteca *opensource* ligera pero funcionalmente completa para trabajar con RDF desde plataformas Python. Permite a las aplicaciones acceder a estructuras RDF a través de construcciones idiomáticas en Python, lo que permite acercar la tecnología al programador de Python experimentado; por ejemplo, un grafo no es más que una colección de tripletas *<sujeto, predicado, objeto>*. Entre el resto de sus características, destacan:

- Contiene procesadores y serializadores para XML, N3, Turtle, RDFa, etc.
- Presenta una interfaz para un grafo que puede soportarse sobre multitud de implementaciones de almacenes.

- Incluye una implementación de SPARQL v1.1.
- Presenta una arquitectura modular basada en *plugins* o complementos.

1.2.3. Javascript

1.3. Estructura de la memoria

La memoria de esta proyecto se estructura en los siguientes capítulos:

1. Introducción general y objetivos
2. <añadir los demás capítulos>
3. Conclusiones y trabajos futuros

<Comentar los capítulos>

Capítulo 2

Metodología

<....>

Capítulo 3

Planificación

<....>

Capítulo 4

Recursos

<....>

Capítulo 5

Análisis

5.1. Captura y documentación de requisitos

5.1.1. Captura de requisitos

La técnica de captura de requisitos utilizada para este proyecto ha sido, fundamentalmente, **la entrevista** con el tutor. Se eligió esta técnica por los siguientes motivos:

- Las entrevistas, bien a distancia o presenciales (y especialmente estas últimas), permiten una mayor implicación del usuario en la captura de requisitos.
- Combinada con una maqueta o prueba de concepto, una entrevista presencial puede dar lugar a la aparición de nuevos requisitos de producto, cambios en las especificaciones e incluso en el enfoque y objetivos del mismo.
- Permite la práctica de la escucha activa y la sugerencia de ideas por parte del analista, aportando un valor añadido que enriquece la simple captura de requisitos.
- Es claramente la técnica más obvia, directa y accesible en el contexto de la realización del proyecto.

Concretamente, se han llevado a cabo varias entrevistas utilizando la plataforma colaborativa Skype y una presencial, en la que el autor de este proyecto se ha desplazado a la sede del departamento

en Madrid con objeto de conseguir una comunicación más fluida y un mayor entendimiento a la hora de consensuar las necesidades y funcionalidades requeridas del producto.

La primera entrevista a distancia propició un intercambio de documentos e ideas que desembocó en la elaboración del documento del anteproyecto. El resto de entrevistas a través de Skype sirvieron para concretar en mayor medida las tareas a realizar y el enfoque del proyecto. Sin embargo, no fue hasta que no tuvo lugar la reunión presencial cuando realmente se le dio al proyecto su orientación final, con unos objetivos claramente definidos y una posibilidad para cumplir los hitos propuestos.

A continuación se resumen todas las sesiones de captura de requisitos:

id	Fecha	Resumen de la entrevista
1	18/10/17	Primer contacto y comunicación de ideas iniciales para la confección del anteproyecto
2	21/02/17	Consolidación de ideas y aportación de más documentación (vídeos sobre prototipos de cuaderno, documentos de texto con descripciones, etc.)
3	28/03/18	Primera demo a modo de POC con un entorno capaz de añadir tripletas.
4	10/07/18	Reunión presencial con demostración *in-situ* de los módulos de modelado e importación/exportación. Tiene lugar una tormenta de ideas y se enfoca el proyecto de otro modo, modificando sus objetivos hacia una herramienta formativa..
5	4/08/18	Revisión de los últimos avances con la integración de un endpoint SPARQL en el frontend y planificación del resto de funcionalidade requeridas.

Cuadro 5.1: Sesiones de entrevistas

5.1.2. Documentación

Para documentar la captura de requisitos, se utilizará la técnica de casos de uso. Se descarta la incorporación de diagramas UML de casos de uso, dado que dichos diagramas carecen de información esencial sobre los mismos (como qué actor lleva a cabo cada paso, o notas sobre el orden de ejecución

de los pasos). Si bien pueden ser útiles como resumen o índice de contenidos, se decide prescindir de ellos dado que el número de casos de uso contemplados en el proyecto es manejable.

Se utilizará una plantilla propuesta por Cockburn[3]: el estilo RUP (*Rational Unified Process*)[4], atractivo y fácil de seguir pese al elevado número de apartados, modificado para plasmar los aspectos más relevantes del proyecto (por ejemplo, no se incluirá un campo *ámbito* porque siempre va a estar referido al mismo sistema o aplicación). El motivo de no utilizar una tabla es meramente subjetivo, ya que el autor de esta memoria opina que puede oscurecer el contenido.

La plantilla sigue la siguiente estructura:

1. Nombre del caso de uso
    a) Descripción breve
    b) Actores, entre los que estará el actor principal. Presentan comportamiento.
    c) Disparadores: acciones sobre el sistema que inician los casos de uso.
2. Flujo de eventos
    a) Flujo básico: escenario principal de éxito.
    b) Flujos alternativos: qué puede pasar que no sea el flujo principal.
        1) Condicion 1
        2) Condición 2
        3) ...
3. Requisitos especiales (si se dieran): plataforma, etc.
4. Precondiciones: qué debe ser cierto antes de ejecutar el caso de uso.
5. Postcondiciones: qué debe ser cierto después de ejecutar el caso de uso.

Los requisitos fueron capturados inicialmente como notas manuscritas y convertidos en necesidades de alto nivel en la plataforma Trello. A partir de ahí, dichas necesidades se refinaron para dar lugar a la batería de casos de uso incluida en 5.3.

5.2. Necesidades

La reunión presencial marcó un punto de inflexión en cuanto a objetivos del proyecto, lo que se traduce en un cambio de necesidades. Para reflejar la evolución completa, se dividirá su captura en dos fases detalladas a continuación:

5.2.1. Captura inicial

A continuación se resumen, en lenguaje natural, las necesidades identificadas durante la primera fase de desarrollo del proyecto:

1. Permitir a usuarios de la UNED de distintos colectivos etiquetar y generar sus propios cuadernos con información y metainformación semántica.
2. Permitir a dichos usuarios realizar consultas sobre sus cuadernos.
3. Desarrollar una interfaz web que permita al usuario gestionar tripletas RDF.
4. Desarrollar un módulo de generación de consultas SPARQL a partir de consultas de lectura y escritura en formato JSON.
5. Desarrollar los correspondientes productos de interés para el usuario: consultas exportadas en forma diversa (CSV, JSON) o embebidas en una plantilla HTML significativa para el usuario.
6. Ofrecer la posibilidad de variar interfaces de entrada y exportadores en función de los distintos colectivos de usuario utilizando los metadatos sobre cuadernos RDF previamente almacenados en una base de datos relacional.
7. Permitir mostrar en pantalla una serie de términos como punto de partida que el usuario pueda utilizar para construir ternas RDF y relaciones entre ellas.
8. Permitir mostrar en pantalla una serie de términos como punto de partida que el usuario pueda utilizar para construir ternas RDF y relaciones entre ellas.
9. Ofrecer al usuario una visualización sencilla y correcta de su modelo que proporciona una perspectiva adecuada sobre la que trabajar.
10. Presentar una interfaz de mantenimiento del grafo: vocabulario e instancias (conceptualización y poblamiento de una ontología).

11. Importar y exportar información estructurada en formatos semánticos estándar.
12. Extender con vocabularios tales como SKOS y OWL.

5.2.2. Captura final

Una vez celebrada la reunión presencial, se decidió darle otro enfoque al proyecto. Si bien la idea inicial era desarrollar un sistema que permitiese generar cuadernos a través de la manipulación de grafos, una vez presentada una maqueta o prueba de concepto con funcionalidades básicas de modelado el tutor propuso convertir la herramienta en un *playground* o sistema de realización de actividades académicas con un enfoque docente orientado a facilitar el aprendizaje de las tecnologías de la Web Semántica (básicamente RDF y SPARQL) a personas con poco o ningún contacto con estas materias (por ejemplo, alumnos de los Grados de Ingeniería Informática o en Tecnologías de la Información de la UNED).

Este nuevo enfoque se tradujo en la siguiente instantánea de necesidades de alto nivel:

1. Permitir el modelado semántico con edición CRUD de clases, subclases y propiedades de clases (anotaciones básicas).
2. Permitir la edición CRUD de relaciones o propiedades, subpropiedades, etc.
3. Ofrecer mecanismos de poblamiento del grafo.
4. Permitir el lanzamiento de consultas SPARQL sobre el grafo local y visualización de resultados.
5. Permitir el lanzamiento de consultas SPARQL sobre endpoints remotos y visualización de resultados.
6. Ofrecer funcionalidad de carga de consultas SPARQL predefinidas desde archivo de texto.
7. Incorporar módulo para añadir un texto con la definición de la actividad a realizar en formato Markdown.
8. Permitir la importación de tripletas desde archivos o URL.
9. Permitir la incorporación (append) de tripletas al grafo local desde archivos o URL.
10. Ofrecer un panel para cargar en el grafo vocabularios comunes predefinidos.
11. Presentar una interfaz fácil de usar y responsiva para el usuario, con una gestión de errores adecuada y suficiente.

5.3. Casos de uso

5.3.1. Caso de uso 1

5.3.1.1. Contexto

5.3.1.2. Ámbito

5.3.1.3. Nivel

5.3.1.4. Actor principal

5.3.1.5. Participantes e interesados

5.3.1.6. Precondiciones

5.3.1.7. Garantías mínimas

5.3.1.8. Garantías de éxito

5.3.1.9. Disparador

5.3.1.10. Descripción

5.3.1.11. Extensiones

Capítulo 6

Implementación

<....>

Capítulo 7

Pruebas

<....>

Capítulo 8

Resultados

<....>

Capítulo 9

Conclusiones y trabajos futuros

9.1. Conclusiones

<....>

9.2. Trabajos futuros

<....>

Bibliografía

[1] Tim Berners-Lee. The semantic web. *Scientific American, Inc.*, 2001.

[2] Apache Software Foundation. Apache jena.

[3] Alistair Cockburn. *Writing Effective Use Cases*. 2001.

[4] Raional. Rational unified process: Best practices for software development teams. 1998.

Anexo A

<Título Anexo A>

<...>

A.1. <Primera sección anexo>

A.1.1. <Primera subsección anexo>