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

1.1. Web Semántica

La Web Semántica es un término originalmente acuñado por Tim Berners-Lee en el año 2001[1]. 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 necesitarían 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])

1.1.1. Linked Data

Estrechamente relacionado con la Web Semántica se encuentra el concepto de datos enlazados o *Linked Data*, también propuesto por Berners-Lee en 2006[**?**] en lo que se considera como su introducción oficial y formal. Se trata de un movimiento respaldado por el propio W3C que se centra en conectar conjuntos de datos a lo largo de la Web y que puede verse como un subconjunto de la Web Semántica.

Se basa, por tanto, en dos aspectos clave:

- La publicación de conjuntos datos estructurados en línea.
- El establecimiento de enlaces entre dichos conjuntos de datos.

La Web Semántica es el fin y los datos enlazados proporcionan el medio para alcanzar dicho fin.

1.1.2. RDF

Las tecnologías de la Web Semántica permiten, por tanto, almacenar conocimiento en forma de conceptos y relaciones a través de ternas o tripletas, modelar dominios de conocimiento con vocabularios estándar y ofrecer potentes facilidades de consulta sobre dichos modelos. Dentro de estas tecnologías, RDF es, sin duda, el bloque de construcción fundamental en la Web Semántica (como HTML lo ha sido para la Web).

RDF fue propuesto por el W3C en 1999[2] como un estándar para crear y procesar metadatos con el objetivo de describir recursos independientemente de su dominio de aplicación, ayudando por tanto a promover la interoperabilidad entre aplicaciones.

Sin embargo, con la llegada de la Web Semántica el alcance de RDF creció, y ya no sólo se usa para codificar metadatos sobre recursos web, sino también **para describir cualquier recurso así como sus relaciones** en el mundo real.

El nuevo alcance de RDF quedó reflejado en las especificaciones publicadas en 2004 por el *RDF Core Working Group*[1], ahora actualizadas a fecha de 2014[2], de las que se pueden extraer las siguientes definiciones de RDF:

Recomendación	Definición
RDF Primer	Un framework para expresar información sobre recursos.
RDF Concepts & Syntax	Un framework para representar información en la Web.

Cuadro 1.1: Definiciones de RDF en las recomendaciones del W3C

1.1.2.1. Modelo Abstracto de RDF

RDF ofrece un modelo abstracto que permite descomponer el conocimiento en pequeñas piezas llamadas sentencias (*statements*), tripletas o ternas y que toman la forma:

Sujeto - Predicado - Objeto

En donde *Sujeto* y *Objeto* representan dos conceptos o "cosas" en el mundo (también denominados **recursos**) y *Predicado* la relación que los conecta.

Los nombres de estos recursos (así como de los predicados) han de ser globales y deberían identificarse por un *Uniform Resource Idenfifier* (URI).

[1] Recomendaciones del W3C, https://www.w3.org/2001/sw/RDFCore/

[2] Versiones actualizadas de las recomendaciones del W3C, https://www.w3.org/2011/rdf-wg/wiki/Main_Page

Por tanto, un modelo RDF puede expresarse como una colección de tripletas o un grafo, dado que estas no son más que grafos dirigidos. Los sujetos y objetos serían los nodos del grafo y los predicados sus aristas.

Otra nomenclatura utilizada para representar una terna sería la siguiente:

Recurso - Propiedad - Valor de la propiedad

En donde el valor de la propiedad u Objeto también puede ser un literal, que consiste simplemente en un dato textual bruto.

Cabe destacar que una terna RDF sólo puede modelar relaciones binarias. Para modelar una relación n-aria, suelen utilizarse recursos intermedios como nodos en blanco, que no son más que sujetos u objetos que no tienen un URI como identificador (nodos sin nombre o anónimos).

En resumen:

Sujeto	Puede ser un URI o un nodo en blanco.
Predicado o Propiedad	Debe ser un URI.
Objeto o Valor de la Propiedad	Puede ser un URI, un literal o un nodo en blanco.

Cuadro 1.2: Resumen conceptual de ternas

A continuación se muestran dos ejemplos de tripletas:

Sujeto	Predicado	Objeto
http://www.uned.es/ia/example#Analista	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#Class
http://www.uned.es/ia/example#Analista	http://www.w3.org/2000/01/rdf-schema#label	"Analista Informático"

Cuadro 1.3: Ejemplos de tripletas reales

1.1.2.2. Aplicaciones prácticas de RDF

Gracias a RDF, es posible construir la Web Semántica y enlazar conjuntos de datos. A continuación se enumeran algunas de sus aplicaciones reales:

- Añadir información legible a los motores de búsqueda.
- Enriquecer un conjunto de datos enlazándolo con conjuntos de datos de terceros.
- Facilitar el descubrimiento de APIs[3] a través de su entrelazado.
- Construir agregaciones de datos sobre determinados temas.
- Proporcionar un estándar de intercambio de datos entre bases de datos.
- Enriquecer, describir y contextualizar los datos mediante un enfoque rico en metadatos.

Su principal ventaja frente a los modelos y bases de datos relacionales es una mayor facilidad para evolucionar los grafos, algo que en un sistema relacional es complejo (dado que requiere de modificaciones en las estructuras de almacenamiento de datos y sus consultas asociadas). En otras palabras, su **flexibilidad para modelar lo inesperado**.

Un ejemplo claro a nivel global es Wikidata[4], una base de datos de conocimiento abierta que puede ser leída y editada tanto por personas como por máquinas. Wikidata actúa como un almacén centralizado para datos estructurados de otros proyectos hermanos como Wikipedia, Wikisource, etc. En el momento de redacción de esta memoria, cuenta con 50,035,140 elementos de datos que cualquiera puede editar.

De una forma más concreta, es destacable el importante rol de RDF y las tecnologías de la Web Semántica en ámbitos de **búsqueda y clasificación bibliográfica o documental**, pudiendo citarse como ejemplo la publicación en RDF del *Diccionario de Lugares Geográficos* y el *Diccionario y Tesauro de Materias* del Patrimonio Cultural de España, gestionado en la actualidad por el Ministerio de Educación, Cultura y Deporte del Gobierno de España[5].

[3] *Aplication Programming Interface*, en este caso referido a puntos accesibles a través de la Web.
[4] https://www.wikidata.org/wiki/Wikidata:Main_Page
[5] https://www.mecd.gob.es/cultura-mecd/areas-cultura/museos/destacados/2015/tesauros.html

1.2. Motivación y objetivos

En el ámbito académico, la Web Semántica se contextualiza e integra en materias como la Gestión Avanzada de la Información y el conocimiento, donde las Tecnologías de la Información se insertan con la Inteligencia Artificial con el objeto de modelar conocimiento humano para un posterior procesamiento y aprendizaje automático.

El objetivo principal del proyecto es obtener una herramienta académica que permita a los Equipos Docentes de las asignaturas relacionadas con tecnologías de la Web Semántica ofrecer a los alumnos (o a cualquier no iniciado en estas tecnologías) un *playground* o entorno de pruebas que les permita consolidar su aprendizaje teórico con una base práctica adaptada al mundo real.

Las tecnologías de la Web Semántica se caracterizan por un cambio de paradigma bastante fuerte con respecto a otras tecnologías de manipulación de datos con las que el alumno pueda estar familiarizado a la hora de introducirse en este nuevo mundo. Es cierto que pueden establecerse ciertas comparativas conceptuales que pueden facilitar su comprensión; como ejemplo, el cuadro 1.4 equipara la Web Semántica con un sistema basado en bases de datos relacionales.

Mundo Relacional	Mundo Semántico
Registro de tabla	Nodo RDF
Columna de tabla	RDF propertyType
Celda de tabla	Valor
Consulta SQL	Consulta SPARQL
Modelo de datosL	Ontología

Cuadro 1.4: Comparativa paradigma relacional vs semántico

Sin embargo, determinados indicadores pueden dar lugar a entender que la interiorización de los conceptos asociados a la Web Semántica no es sencilla. Por ejemplo: su uso casi exclusivamente académico con poca penetración en el mundo empresarial, su lenta velocidad de propagación, su soporte limitado en determinadas plataformas tecnológicas o incluso su tímido interés en el ámbito público (excluyendo honrosas excepciones como el académico o universitario, las iniciativas de datos abiertos o lo relativo a la biblioteconomía). Un ejemplo muy concreto, real y tangible de esta escasa

penetración es el reducido número de artículos (cuatro) presentado en *workshops* como SemStats[6][3], de cuyo *Program Comittee* este autor forma parte:

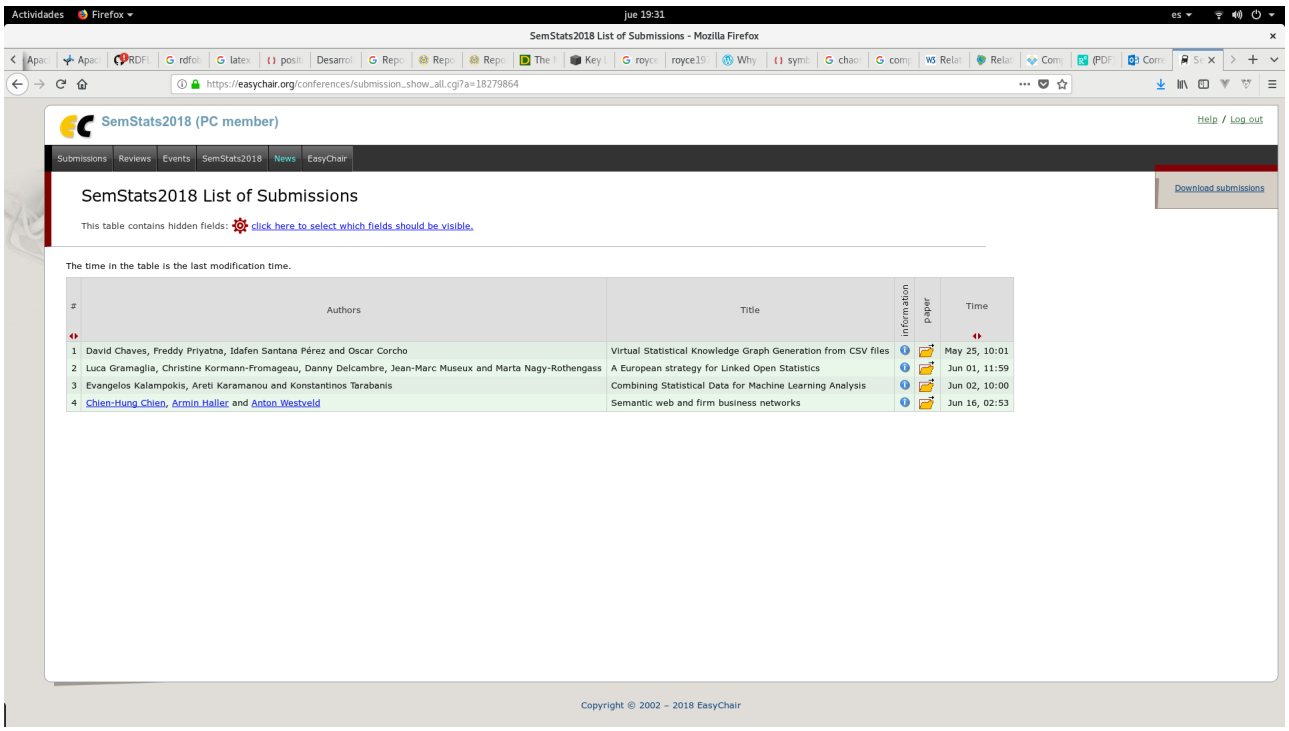

Figura 1.1: Trabajos enviados a Semstats 2018

Resulta cuando menos sorprendente que las Oficinas Estadísticas Públicas, encargadas de generar y publicar datos estadísticos sobre economía, población y sociedad en general, no estén mostrando mayor interés a la hora ya no de publicar sus datos en formatos enlazables, sino de construir aplicaciones semánticas para explotar y relacionar mejor sus datos.

Por tanto, parece interesante trabajar en la línea de conseguir una herramienta que acerque este tipo de tecnologías a quienes tengan interés u obligación de trabajar con ellas y ayude a su divulgación.

Si bien es cierto que existen, por una parte, varias aproximaciones gratuitas en la red a modo de *playground* que permiten el lanzamiento de consultas SPARQL a determinados *endpoints* y por otra, editores y herramientas de modelado del calibre de Protégé[7], no parece haber en el mercado referencias de aplicaciones que ofrezcan subconjuntos reducidos de ambas funcionalidades. Llenar este hueco y **conseguir una herramienta formativa única de modelado y consulta en tecnologías de la Web Semántica es la motivación** que da lugar al desarrollo de este proyecto.

[6]SemStats es un *workshop* de celebración anual centrado en el estado actual de las tecnologías de la Web Semántica aplicadas a la estadística pública

[7]https://protege.stanford.edu/

De forma más específica, es posible enumerar los siguientes objetivos:

- Facilitar el aprendizaje y la **familiarización con tecnologías de la Web Semántica** a usuarios inexpertos.
- Ofrecer a los Equipos Docentes la posibilidad de **distribuir actividades formativas** autocontenidas para que los alumnos trabajen sobre ellas en la herramienta.
- Agrupar en un solo producto funcionalidades básicas de edición y consulta sobre grafos, de cara a cubrir las expectativas educativas en este ámbito.
- Permitir al alumno comprobar *in situ* los efectos de sus acciones sobre un grafo y ofrecerle la **flexibilidad suficiente** como para permitirle dar rienda suelta a su creatividad e inquietudes.

1.3. Trabajos previos

Para enfocar el presente proyecto se llevó a cabo un estudio previo de los trabajos existentes que pudieran estar relacionado con los objetivos del mismo. En las siguientes subsecciones se analizan las familias de herramientas analizadas.

1.3.1. Herramientas de consulta

La herramienta de consulta web que se asemeja más a los resultados obtenidos con este proyecto es el SPARQL *playground* desarrollado por el *Swiss Institute of Bioinformatics*[8].

Se trata de una aplicación web *standalone* y multiplataforma cuyo propósito fundamental es el aprendizaje de SPARQL. Sus autores proporcionan una demo *online*[9] además de documentación y consultas de ejemplo para que cualquiera pueda familiarizarse con este lenguaje de consulta.

Si bien sus funcionalidades de consulta son muy completas, carece de facilidades de edición de grafos propios o de consulta a *endpoints* externos, por ejemplo.

[8] https://www.isb-sib.ch
[9] http://sparql-playground.sib.swiss/

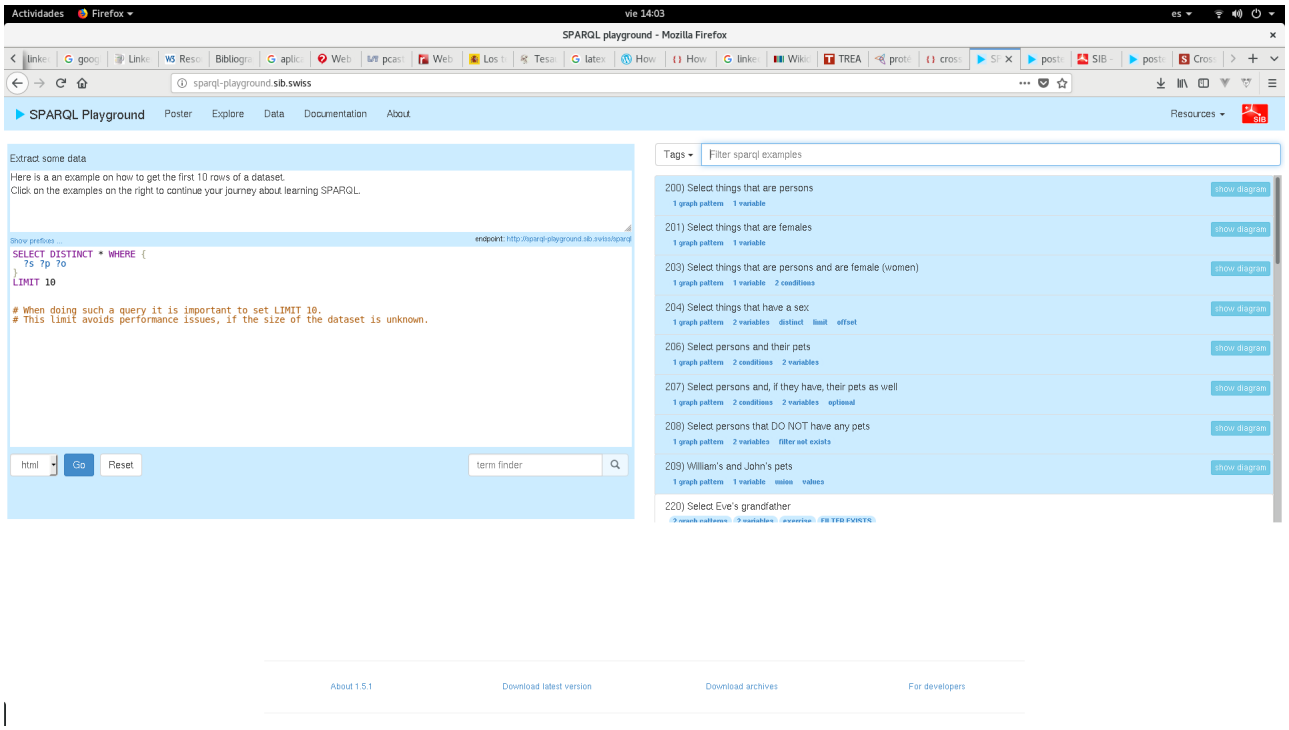

Figura 1.2: Interfaz de SPARQL *playground*

1.3.2. Herramientas de modelado

1.3.2.1. Web

Existen pocas herramientas con funcionalidades de edición de grafos RDF en la Web. Al margen de la versión web de Protégé, cuya versión de escritorio (más representativa) se comenta en el punto 1.3.2.2, el resto de trabajos encontrados son fundamentalmente bibliotecas o paquetes de edición de formularios.

1.3.2.2. De escritorio

La herramienta de modelado de ontologías en RDF por antonomasia es Protégé, un editor de ontologías opensource gratuito y un marco de trabajo para construir sistemas inteligentes soportados por una fuerte comunidad de usuarios académicos, gubernamentales y corporativos. Se utiliza en áreas tan diversas como la biomedicina, el comercio electrónico, la predicción metereológica o el modelado organizacional.

Protégé se instala como una aplicación de escritorio multiplataforma y tiene una arquitectura

basada en *plugins* o complementos que permite extender fácilmente su funcionalidad. Tiene soporte para razonadores sobre las distintas ontologías que se modelan y facilidades de representación y edición de grafos RDF.

De cara a un primer encuentro con las tecnologías de la Web Semántica, puede parecer abrumador por su gran flexibilidad y potencia. Además, no cuenta con facilidades de consulta sobre los grafos modelados.

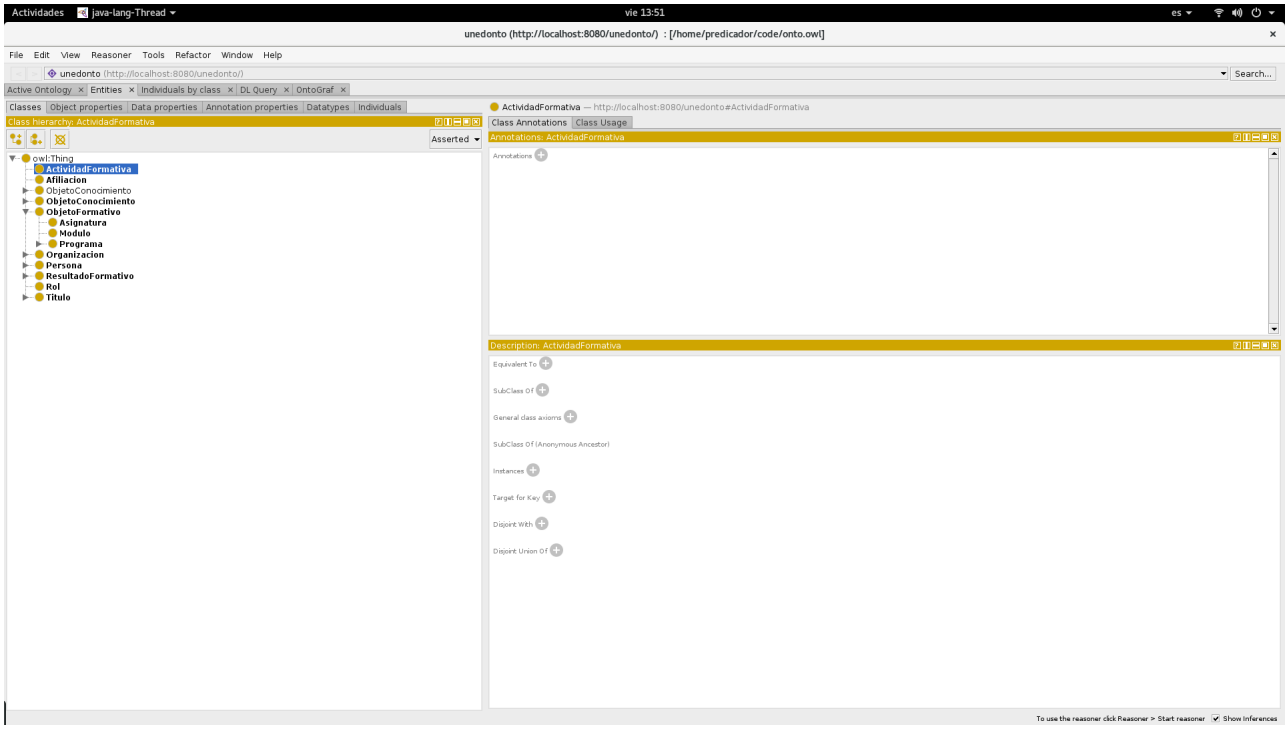

Figura 1.3: Interfaz de Protégé

1.4. 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.

1.4.1. Java

Así, para la plataforma Java se cuenta con Apache Jena[4], 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.

1.4.2. 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.

1.4.3. Javascript

1.5. 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

2.1. Elección de la metodología

Para acometer este proyecto, se han valorado dos familias de metodologías de desarrollo de *software*:

1. Metodologías en cascada (*Waterfall*)
2. Metodologías ágiles (incrementales e iterativas)

La metodología de desarrollo en cascada surgió como idea en un artículo de Winston W. Royce en 1970[5]. Históricamente, este modelo se ha extendido tanto en ámbitos académicos como profesionales, siendo estas sus principales características:

- Gestión predictiva de proyectos llevada al software.
- Toma como modelo la forma de proceder en el resto de ingenierías.
- Intenta llenar el vacío del *code & fix*.
- Cada fase se realiza, en principio, una única vez.
- Cada fase produce un entregable que será entrada de la siguiente.

- Los entregables no son, en principio, modificables.

Es decir, se basa en la separación entre diseño y construcción (o entre creatividad y repetición).

La propuesta de Royce,tal y como se desprende de la lectura del artículo original, describía el modelo en cascada como la «descripción más simple» ([5]) que solo funcionaría para los proyectos más sencillos. Irónicamente, este mensaje malentendido ha sido el origen de la popularidad de la metodología en cascada, que hoy en día se sigue promoviendo en muchos casos por inercia, desconocimiento, comodidad o ilusión de control sobre el proyecto.

Lamentablemente, en la *Ingeniería de Software* los pesos de diseño y construcción están invertidos con respecto a otras ingenierías, siendo el *software* un dominio de cambio y alta inestabilidad. El desarrollo de *software* es, intrínsecamente, una labor creativa; y la creatividad no es fácilmente predecible. Esto ha dado lugar a que los desarrollos tradicionales adolezcan de ciertos problemas[6]:

- Existencia de muchos requisitos vagos o especulativos y diseño detallado por adelantado.
- Están fuertemente asociados con las tasas de fallo más altas en proyectos.
- Se encuentran promovidos históricamente por creencia más que por evidencia estadística significativa.
- Su rigidez incrementa el riesgo de fracaso, pospuesto hasta las fases finales del proyecto.
- Asume que las especificaciones son predecibles, estables y completas.
- Pospone integración y pruebas hasta fases tardías.
- Se basa en estimaciones y planificación "fiables".

Entre los estudios que ratifican las afirmaciones anteriores, cabe citar los siguientes:

- Informe Chaos 2015[7].
- *Dr. Dobb's Journal article The Non'Existent Software Crisis: Debunking the Chaos Report*[8].
- Encuesta de Gartner[9].
- TODO

Por tanto, atendiendo a esta exposición y considerando que el proyecto en cuestión presentaba un alto nivel de incertidumbre debido a su alto componente en investigación del estado tecnológico actual, se ha optado por utilizar un enfoque metodológico incremental e iterativo, puesto que:

- Facilita llevar a cabo proyectos pequeños.
- Fomenta la interacción entre el desarrollador y el usuario.
- Fuerza a que los inevitables cambios en requisitos sucedan en fases tempranas del proyecto.

Entre sus características es posible citar:

- Se trabaja sobre subconjuntos de funcionalidad (*features*).
- Los incrementos permiten añadir funcionalidad al producto (mejora del proceso).
- Las iteraciones permiten rediseñar, revisar y refactorizar el producto (mejora del producto).
- Se basa en entregas frecuentes y ciclos prueba/error.
- Ofrece flexibilidad a la hora de gestionar el cambio.

La referencia más clara en este ámbito es *Extreme Programming*, de Kent Beck[10]. *Extreme Programming* es «un estilo de desarrollo de software centrado en la aplicación excelente de técnicas de programación, comunicación clara y trabajo en equipo que permite conseguir objetivos antes impensables» ([5]). Se trata de una metodología basada en valores como la comunicación, realimentación, simplicidad, valentía y respeto, soportada sobre un cuerpo de prácticas útiles y con un conjunto de principios complementarios, además de contar con una comunidad de usuarios que comparte todo lo anterior.

Su aplicación al desarrollo de este proyecto no ha sido estricta; por ejemplo, no se han definido ciclos estrictos por la propia naturaleza inestable de la dedicación al desarrollo del mismo, pero sí que ha realizado un diseño evolutivo soportado sobre un número suficiente de pruebas unitarias así como una planificación incremental y adaptativa a las problemáticas que iban surgiendo.

Prueba del acierto a la hora de elegir esta metodología es el cambio de necesidades y requisitos funcionales que tuvo lugar en la reunión presencial mantenida con el tutor, donde **la metodología aportó que no fuera necesario descartar ningún desarrollo o trabajo realizado hasta**

la fecha. Permitió realizar una gestión del cambio efectiva, re-orientando el trabajo a tiempo sin impactar en el diseño ni en la implementación existente. Finalmente, tanto la organización como la planificación de las tareas han sido lo suficientemente flexibles para conseguir un ritmo adecuado de desarrollo, reduciendo los puntos de bloqueo.

Capítulo 3

Planificación

3.1. Planificación global

Para realizar la planificación del proyecto, y dada la metodología de desarrollo incremental e iterativa elegida, no se ha seguido un modelo típico en fases de *Análisis, Diseño, Implementación, etc.* sino que se ha optado por un enfoque orientado a tareas relacionadas con la funcionalidad del producto final esperado.

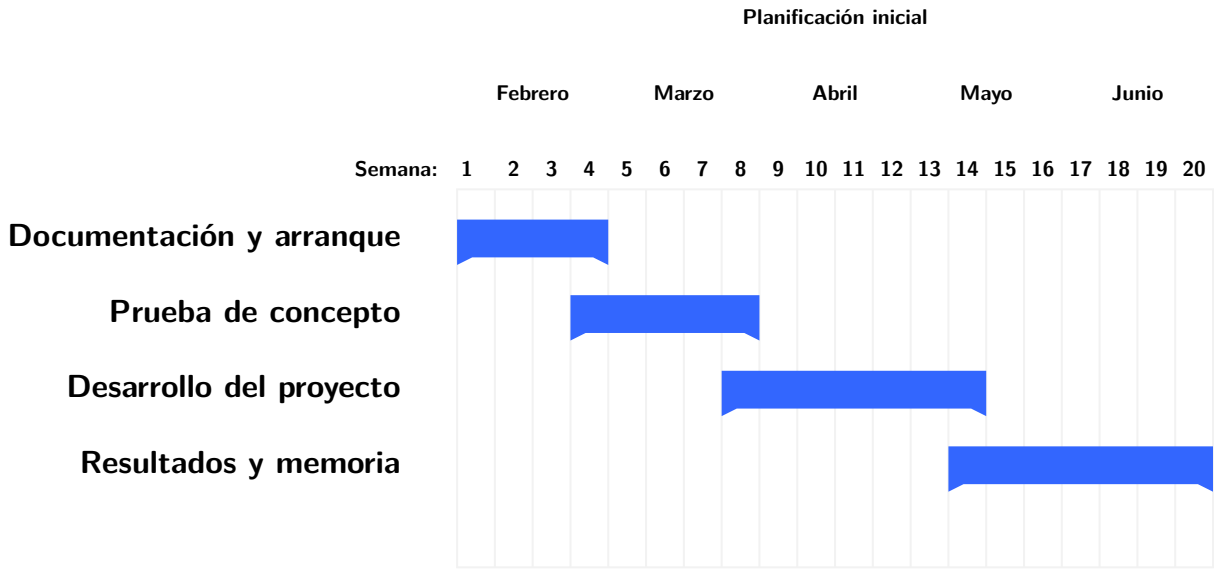

Figura 3.1: Planificación inicial

La figura anterior (3.1) muestra la planificación propuesta para la elaboración del anteproyecto.

En esta propuesta era clave la revisión de la prueba de concepto con el tutor para comprobar que efectivamente se habían comprendido las necesidades desprendidas del análisis y para poder continuar con la especificación funcional de forma más refinada y precisa.

En la práctica, la evolución del proyecto ha sido bien distinta. En la figura 3.2 se puede comprobar cuál ha sido la planificación efectiva, producto esta de cambios sobre la inicial para resolver los distintos imprevistos que han ido surgiendo durante la ejecución del mismo.

Con respecto a la planificación inicial, se tiene que:

- El proyecto se ha retrasado un total de ocho semanas.
- El período de formación llevó al menos dos semanas más de lo esperado (en realidad, el aprendizaje del *framework Vue* ha estado presente a lo largo de prácticamente todo el desarrollo del proyecto).
- El desarrollo del producto ha consumido seis semanas más de las previstas inicialmente.
- La memoria se ha redactado en dos semanas menos con respecto a la estimación inicial.

Las desviaciones surgidas con respecto a la planificación inicial tienen su explicación en los siguientes motivos:

- El alto grado de incertidumbre a la hora de estimar la planificación inicial, dado que se desconocía cuál era la situación actual del ecosistema tecnológico de *front-end web* y su integración con las tecnologías de la Web Semántica.
- La curva de aprendizaje de *Vue*, si bien es considerada más suave que la de sus competidores (*React, Angular*) fue mayor de lo esperado. El bajo grado de familiarización del autor del proyecto con las tecnologías de *front-end* y, especialmente, *Javascript ES6+*, no facilitó el aprendizaje.
- El **bajo nivel de madurez de las bibliotecas existentes** para la manipulación de RDF en Javascript, así como la heterogeneidad y poca estabilidad de estas implementaciones, ha suscitado muchas dudas sobre cuáles utilizar y cómo enfocar su integración con *frameworks* más maduros como *Vue*.

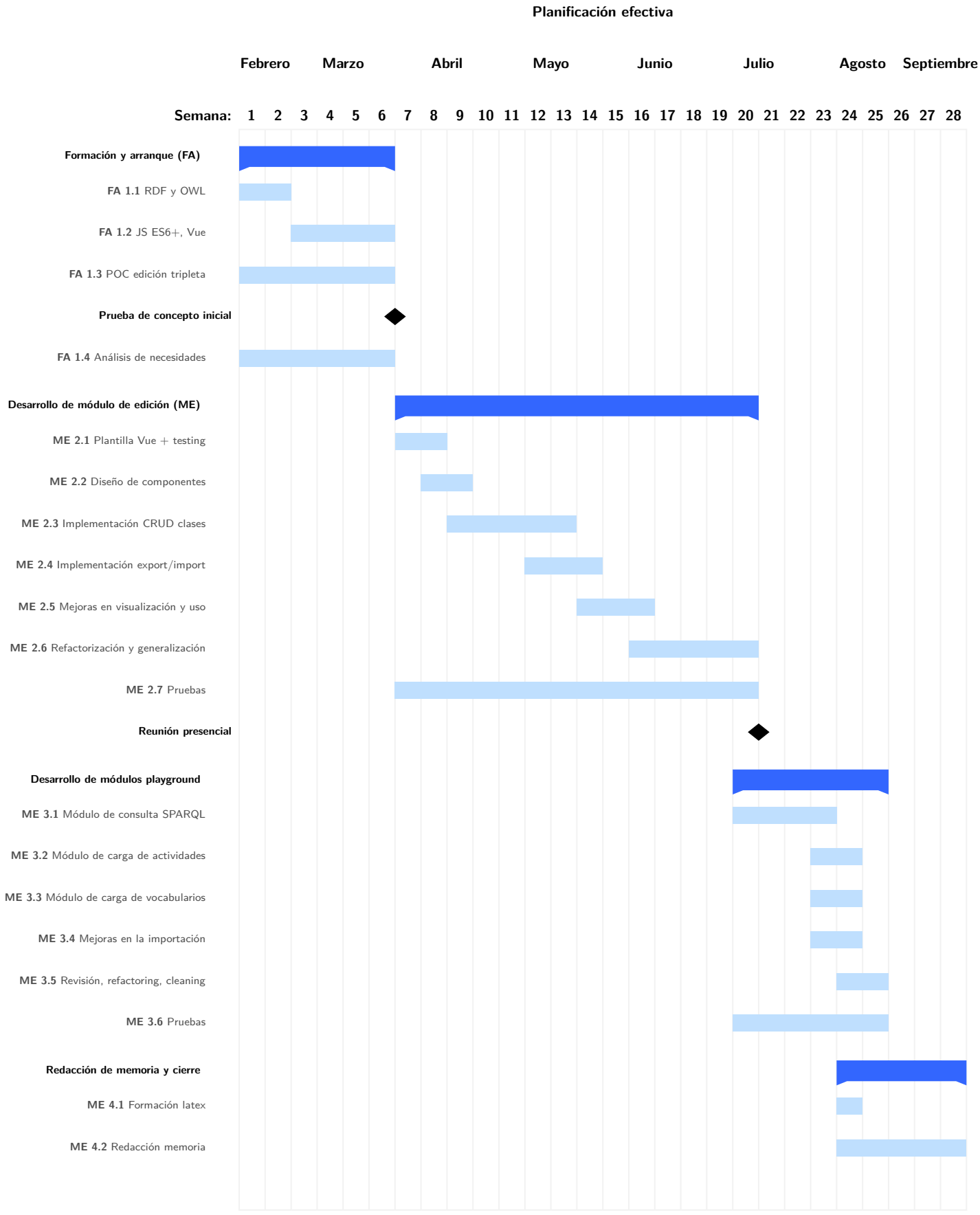

Figura 3.2: Planificación efectiva

- La **práctica ausencia de bibliotecas** o componentes de interacción con SPARQL **conformes a la especificación estándar de la interfaz** *rdf.js*[11] (que por otra parte, es un borrador de 2017) ha puesto en peligro la viabilidad del proyecto. La plataforma finalmente utilizada, *Comunica*[12], aún no tiene una versión estable muchos de sus módulos (concretamente, el endpoint SPARQL utilizado para el proyecto no la tiene), con lo que ha sido necesario estar en contacto directo con sus autores y colaborar con ellos en la revisión de defectos o *bugs* y dependiendo por tanto de sus tiempos de respuesta (hay que tener en cuenta que la plataforma es *opensource* y por tanto no existe acuerdo de nivel de servicio alguno.)
- La dificultad existente en llevar a cabo un análisis de requisitos a través de una plataforma online de colaboración como puede ser *Skype*: para constatar este hecho no hay más que verificar el cambio de rumbo del proyecto una vez mantenida la reunión presencial, que sirvió para definir objetivos más claros y comprender las necesidades del Departamento.

A pesar de todo ello, el autor de este proyecto está satisfecho con el nivel de conocimiento adquirido en el ámbito de todas las tecnologías empleadas y el tiempo consumido para obtener como resultado un producto desplegado en producción y listo para utilizar.

3.2. Planificación ágil

Si bien para la planificación global del proyecto se han utilizado herramientas tales como el *diagrama de Gantt*, para gestionar el trabajo del día a día otro enfoque ha sido necesario. En el contexto de metodologías ágiles de desarrollo de software de tipo incremental e iterativo como Extreme Programming, **las tareas de planificación adquieren un carácter adaptativo** muy distinto del que presenta una planificación tradicional con una metodología de desarrollo en cascada, por ejemplo.

La planificación ágil es un proceso en continua evolución, también iterativo e incremental como las metodologías a las que pertenece, basada en ciclos del tipo:

1. Añadir tareas
2. Estimar tareas
3. Priorizar tareas

En este caso, la estimación de tareas se llevó a cabo de forma heurística, utilizando la experiencia del autor y el conocimiento del contexto existente en el momento de la estimación. En base a ello, la priorización (ordenación) de las tareas tenía lugar inmediatamente, relegando a la estimación a un segundo plano.

Para gestionar estas tareas, se ha optado por utilizar como herramienta Kanban. Un tablero Kanban puede definirse como un dispositivo de señalización que introduce el flujo de trabajo de un proceso a un ritmo manejable. Presenta las siguientes características:

- Solo envía trabajo cuando lo ordena el cliente o usuario (en este caso, el propio autor del proyecto)
- Indica específicamente qué trabajo debe hacerse.
- Controla la cantidad de trabajo en progreso.
- Regula las interrupciones y orquesta el ritmo de trabajo.

Básicamente, consiste en utilizar una tabla con varias columnas para visualizar el estado de una tarea a lo largo de las distintas fases que se consideren. Para el caso de la realización de este proyecto, se crearon las siguientes columnas:

Columna	Descripción
To-Do	Tareas por realizar en orden de prioridad descendente
Work in progress	Tareas realizándose en un momento dado. No más de dos o tres.
Stand-by	Tareas a la espera por motivos ajenos al autor del proyecto.
Done	Tareas finalizadas.
Discarded	Tareas descartadas debido a cambios de diseño, requisitos, etc.
Doubts	Dudas planteadas a lo largo del desarrollo del proyecto.
Resources	Recursos documentales en la web útiles para del desarrollo del proyecto.

Cuadro 3.1: Diseño del tablero Kanban

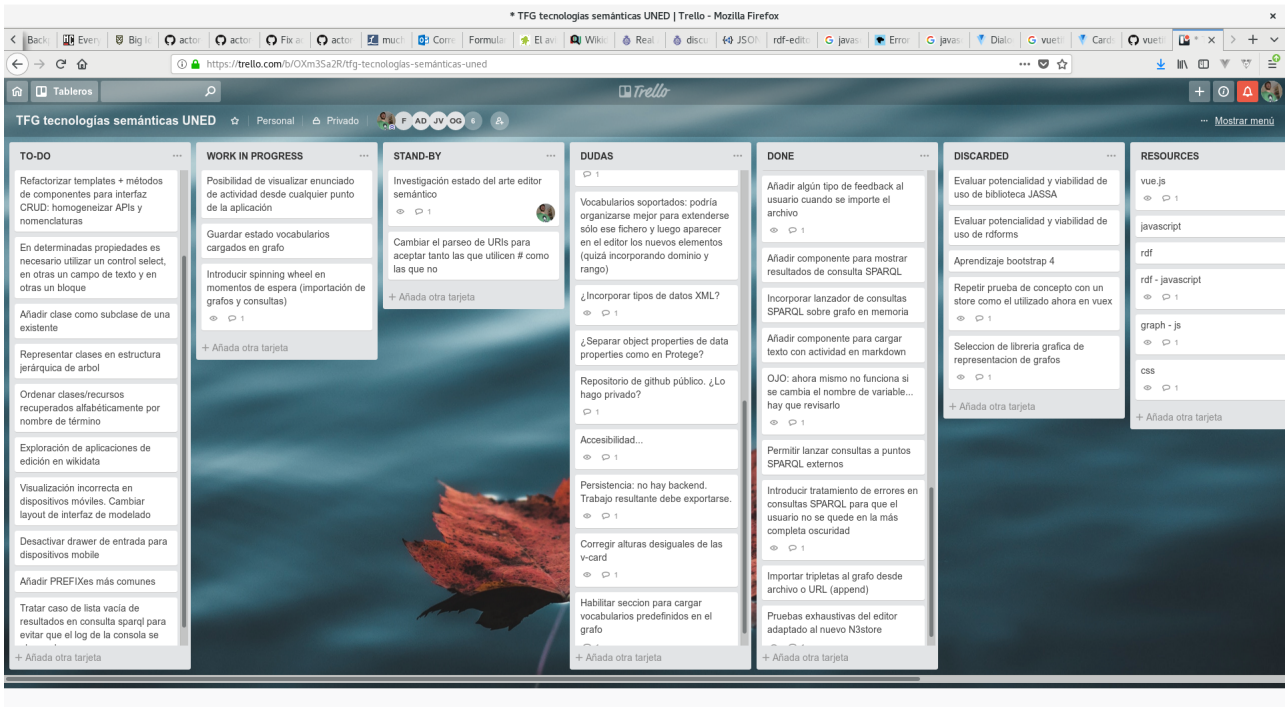

Figura 3.3: Ejemplo de tablero Trello

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[13]: el estilo RUP (*Rational Unified Process*)[14], 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] Ora Lassila. Resource description framework (RDF) model and syntax specification. W3C recommendation, W3C, February 1999. http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/.

[3] SemStats. 2018.

[4] Apache Software Foundation. Apache jena.

[5] Dr. Winston W. Royce. Managing the development of large software systems. *IEEE WESCON*, 1970.

[6] Craig Larman. *Applying UML and patterns*. 2005.

[7] The Standish Group. Informe chaos, 2015.

[8] Scott W. Ambler. The non-existent software crisis: Debunking the chaos report. *Dr. Dobb's*, 2014.

[9] Lars Mierizt. Gartnet survey shows why projects fail, 2012.

[10] Cynthia Andres Kent Beck. *Extreme Programming Explained: Embrace Change*. 2005.

[11] elf Pavlik Blake Regalia Piero Savastano Ruben Verborgh Thomas Bergwinkl, Michael Luggen. Interface specification: Rdf representation, 2017.

[12] Joachim Van Herwegen Ruben Taelman. Comunica query engine platform, 2018.

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

[14] Rational. 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>