Universidad de las Ciencias Informáticas

Facultad 6

Extensión para semantizar el navegador Web Mozilla Firefox

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autores:

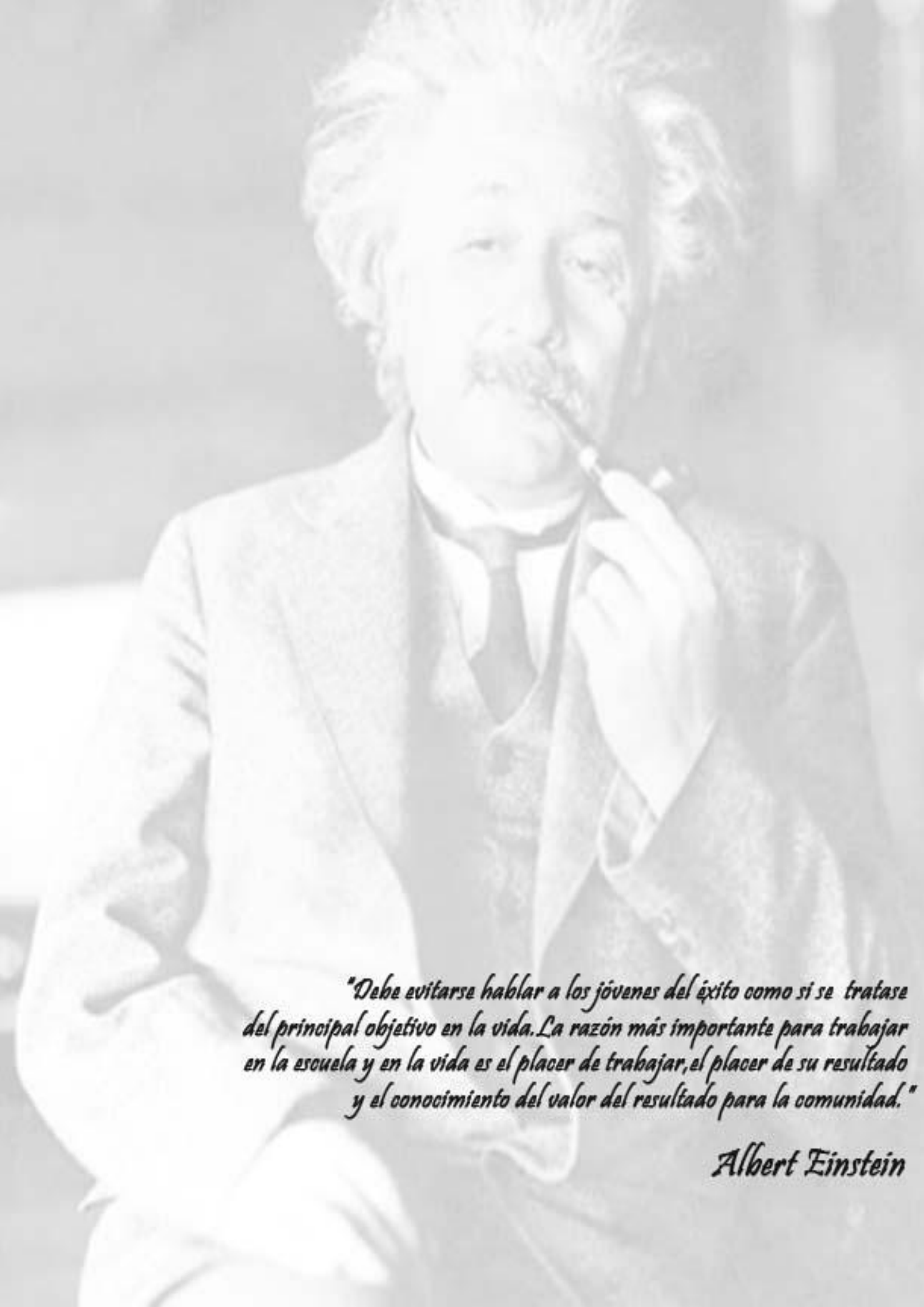
Gendrys Espinosa Vega.

Luilly Díaz Montero.

Tutora:

Ing. Aida Portelles Valdes.

Ciudad de la Habana, Junio 2011
"Año 53 de la Revolución"



"Debe evitarse hablar a los jóvenes del éxito como si se tratase del principal objetivo en la vida. La razón más importante para trabajar en la escuela y en la vida es el placer de trabajar, el placer de su resultado y el conocimiento del valor del resultado para la comunidad."

Albert Einstein

Declaración de autoría

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Gendrys Espinosa Vega

_____

Firma del Autor

Luilly Díaz Montero

_____

Firma del Autor

Ing. Aida Portelles Valdes

_____

Firma de la Tutora

Datos de contacto

Tutora: Ing. Aida Portelles Valdes
Especialidad de graduación: Ingeniería en Ciencias Informáticas
Categoría docente: Instructor
Categoría Científica: no
Años de experiencia en el tema: 0
Años de graduado: 3

Agradecimientos

A mis padres por darme la vida, por su sacrificio, por apoyarme en cada decisión que tomo, por creer en mí. A mis abuelos por la educación, la compresión y el amor que siempre me brindaron. A mi familia, porque estoy aquí gracias a su esfuerzo y porque me siento orgulloso al formar parte de ella.

A mi novia por su cariño, dedicación y apoyo, por todos los momentos de felicidad. A mi dúo de tesis Luilly por nuestra amistad, por estar presente en los buenos y malos momentos.

A nuestra tutora Aida por estar presente en todos los momentos, por la energía positiva que transmite, por su entrega incondicional a la realización de este trabajo.

A mis amigos por los momentos que compartimos juntos. A las chicas del apto 105207 por soportarme durante este tiempo. A Rosi por ser una extraordinaria persona, por su amor y confianza.

A la profe Yanelis y a todos los profesores que han contribuido a mi educación y a mi formación como profesional. A la Revolución y a Fidel por darme la oportunidad de estudiar en la UCI.

A todas las personas que de una forma u otra nos apoyaron en la realización de este trabajo.

Gendrys Espinosa Vega

A mi tutora le agradezco la cooperación, la constancia y todo al aporte que hizo para la realización de este trabajo, así como la fe y esfuerzo de mi compañero de tesis para que todo saliera bien.

A mis profesores que no solo me aportaron conocimiento como estudiante sino que colaboraron con su ejemplo a que sea una mejor persona. En especial a Leslie, llevándome a la práctica algo que me enseñaron mis padres la amistad y los deberes van separados, el profe Rolando ejemplo de preparación y amor a la profesión.

Mientras escribía los agradecimientos descubrí que tengo pocos amigos, pero son muy buenos, o los mejores que puedo tener. Desde que entre en la escuela estuvieron presentes en todo lo que sucedió en mi vida universitaria. Llegaron a ser una familia sustituta y cumplieron con la difícil tarea, ya que mi familia es especial. A ellos les dedico esta frase:

"Cuando se viaja en pos de un objetivo, es muy importante prestar atención al Camino. El Camino es el que nos enseña la mejor forma de llegar y nos enriquece mientras lo estamos cruzando."

Paulo Coelho

Yanelis Benítez eres muy especial no tengo palabras para agradecerte, entre menos digo mas siento.

A mi familia pilar donde crecí como persona, mis tías psicólogas, mis tías locas, mis tíos, mis primos y en especial a mis abuelas consentidas.

A mi mamá y mi papá, MIL GRACIAS.

A mi abuelo.

Luilly Díaz Montero

Dedicatoria

Dedico este trabajo a mis padres por el amor y cariño que me han proporcionado, por su apoyo incondicional, por el sacrificio y esfuerzo a lo largo de estos años. A mi familia, porque estoy aquí gracias a su esfuerzo y porque me siento orgulloso al formar parte de ella. A mi novia por su amor y amistad, A todos mis amigos.

Gendrys Espinosa Vega

Quiero dedicar esta tesis a mi abuelo que aunque no está, con su carácter, marcó un antes y un después en mi vida personal como profesional. A mis padres que nunca les digo cuan impórtate son en mi vida, pero espero que sepan que mucho. A toda mi familia que proporciono los valores y el apoyo para ser quien soy.

Luilly Díaz Montero

Resumen

La Web ha transformado la forma de comunicarse, realizar negocios, compartir información y conocimiento. Si bien establece un gran avance tecnológico, adolece de varias carencias. En la actualidad se hace muy difícil representar, organizar y gestionar la información presente en Internet. Una de las posibles alternativas de solución a estos problemas es la Web Semántica. Los Datos Enlazados es la forma que tiene la Web Semántica de vincular los distintos datos que están distribuidos en la Web, su utilización se está extendiendo hasta los dominios de la Biología computacional como una tecnología para conectar los conjuntos de datos diferentes que son utilizados por los investigadores en este campo. El presente trabajo tiene como objetivo desarrollar una extensión para semantizar el navegador Web Mozilla Firefox que facilite reutilizar recursos Web a partir de Datos Enlazados sobre repositorios RDF (*Resource Description Framework*). Para guiar el proceso de desarrollo de la extensión se definió la metodología XP. Se utilizaron los lenguajes de programación JavaScript y XUL, con NetBeans 6.9 como IDE .Se empleó el servicio SPARQL endpoint del Proyecto LinkedLifeData para consultar base de datos biológicas a través del lenguaje SPARQL. Además, se identificaron las funcionalidades y se realizó el diseño, implementación y prueba de la extensión. Como resultado se desarrolló una extensión para semantizar el navegador Web Mozilla Firefox, que ofrece características importantes a usar en el dominio de ciencias de la vida, y permite realizar consultas SPARQL de manera sencilla y visualizar los resultados obtenidos de las mismas.

Palabras Claves: Datos Enlazados, RDF, Web Semántica.

Índice

Índice de Figuras

Índice de Tablas

Introducción

El acelerado desarrollo de las tecnologías de la información y las comunicaciones ha posibilitado que la información y el conocimiento se conviertan en recursos indispensables para el desarrollo de cualquier institución en aras de lograr una mayor competitividad y el perfeccionamiento en los servicios. A dos décadas del surgimiento de la Web, ya se ha convertido en un medio de uso cotidiano en la sociedad, brindando innumerables posibilidades en las comunicaciones, el comercio, la difusión de conocimientos, el acceso a servicios e información. Permite comprar todo tipo de productos y servicios, gestionar una cuenta bancaria, buscar un restaurante, consultar la cartelera, leer la prensa, localizar a una persona, matricular en la universidad o trabajar.

El incremento de las herramientas que facilitan la construcción de páginas Web, unido a la estructura descentralizada de Internet, ha provocado un aumento exponencial del conocimiento contenido en Internet, posibilitando que circule a través de ella millones de documentos cada segundo. La Web está cerca de convertirse en una enciclopedia universal del conocimiento humano, pero a pesar de ello no está carente de problemas. Se hace engorroso el proceso de representación, organización y búsqueda de la información, debido al gran volumen que esta representa, es casi imposible automatizar cualquier proceso.

Esta problemática se traduce en las actuales carencias que poseen los buscadores Web en términos de análisis de relaciones más complejas entre dominios del conocimiento, de forma que la información pueda ser reutilizada por personas, organizaciones y computadoras de manera óptima. Para afrontar esta situación, la comunidad investigadora ha propuesto la inclusión de un significado semántico (metadatos[1]) en los recursos Web[2]. A esta solución Tim Berners-Lee, creador de la World Wide Web[3], la ha denominado Web Semántica (1).

Las tecnologías de la Web Semántica buscan desarrollar una Web más cohesionada, donde sea aún más fácil localizar, compartir e integrar información y servicios, para sacar un partido mayor de los recursos disponibles en la Web.

Para obtener una adecuada definición de los datos, la Web Semántica utiliza esencialmente RDF (Resource Descripción Framework), SPARQL (Simple Protocol and RDF Query Language), y OWL

[1]Metadato: Son etiquetas que describen el significado de cada uno de los datos.
[2]Recurso Web: páginas web, imágenes, videos, computadores, impresoras, etc.
[3]World Wide Web: es un sistema de distribución de información basado en hipertextos enlazados y accesibles a través de Internet.

(Ontoloy Web Language). Estos mecanismos ayudan a convertir la Web en una infraestructura global en la que es posible compartir y reutilizar recursos entre diferentes tipos de usuarios.

La Web Semántica no se trata únicamente de la publicación de datos en la Web, sino que éstos se puedan enlazar a otros. De forma tal que las personas y las máquinas puedan explorar la Web, pudiendo llegar a información relacionada que se hace referencia desde otros datos iniciales. Los Datos Enlazados o Linked Data es la manera que tiene la Web Semántica de vincular los distintos datos que están distribuidos en la Web, referenciándose del mismo modo que lo hacen los enlaces de las páginas Web (2).

La cantidad de Datos Enlazados que se encuentran publicados en la Web de Datos han experimentado un enorme crecimiento en los últimos años. La lista de recursos ya disponibles en Datos Enlazados crece día a día. El mayor auge hasta ahora se ha producido en el contexto de la publicación de datos del sector público. Sin embargo, su utilización se está extendiendo a otros sectores entre los que destacan los medios de comunicación, infraestructuras y logística, el de los datos geográficos y el ámbito universitario y científico. En este último los Datos Enlazados son casi la mitad, en volumen de los disponibles globalmente, con especial atención a dominios como la Biología computacional (3). Como el conocimiento biológico se encuentra distribuido entre múltiples bases de datos esto hace que sea difícil garantizar la coherencia de la información.

A la Web Semántica se puede acceder utilizando los navegadores de Datos Enlazados, al igual que la Web tradicional se accede mediante navegadores HTML. Sin embargo, en lugar de seguir los enlaces entre las páginas HTML, los navegadores de Datos Enlazados permiten a los usuarios navegar entre diferentes fuentes de datos siguiendo los enlaces RDF.

El desarrollo de navegadores especializados en Datos Enlazados está motivado por la dificultad de navegar a través de repositorios RDF distribuidos pero enlazados. Estos navegadores ofrecen funcionalidades sencillas que permiten navegar por el árbol RDF, pero no ofrecen características importantes a usar en el dominio de las ciencias de la vida.

Actualmente el usuario debe conocer el esquema RDF de los repositorios por los que navega y la sintaxis SPARQL. Aunque los navegadores ofrecen enlaces entre los datos, la navegación no es intuitiva, ni se tiene conocimiento de los repositorios por los que se está navegando. Estos Datos Enlazados no son mostrados de forma integrada al usuario, por lo que no se puede analizar de manera directa los datos encontrados, que potencien el conocimiento en la comunidad científica, ya que se encuentran dispersos en diferentes pasos realizados durante la navegación.

Por todo lo antes expuesto se propone como problema de la investigación:

¿Cómo facilitar el acceso a la información biológica publicada como Datos Enlazados, para su uso por usuarios no expertos en tecnologías semánticas?

Se propone como objeto de estudio: el proceso de desarrollo de las plataformas de Datos Enlazados y enmarcando como campo de acción: el proceso de desarrollo de extensiones para semantizar navegadores Web tradicionales.

Se plantea como objetivo general de este trabajo, desarrollar una extensión para semantizar el navegador Web Mozilla Firefox que facilite reutilizar recursos Web a partir de Datos Enlazados sobre repositorios RDF.

Se definen como objetivos específicos:

- Determinar las características de los navegadores para Datos Enlazados.
- Realizar el diseño de la extensión para navegador Web Mozilla Firefox.
- Implementar la extensión para navegador Web Mozilla Firefox.
- Validar la extensión para navegador Web Mozilla Firefox.

Para dar cumplimiento a los objetivos específicos, se trazaron las siguientes tareas de investigación:

- Revisión bibliográfica de las diferentes herramientas desarrolladas para acceder a la información publicada en repositorios RDF.
- Revisión bibliográfica de las tendencias y tecnologías para desarrollar una herramienta que facilite la navegabilidad sobre repositorios RDF.
- Elaboración de las historias de usuario.
- Elaboración de las tarjetas CRC (Clase, Responsabilidad, Colaboración).
- Implementación la extensión para navegador Web Mozilla Firefox.
- Validación de la extensión para el navegador Web Mozilla Firefox mediante pruebas de aceptación.

Esta investigación posee la siguiente estructura:

Capítulo 1: “Fundamentación Teórica”: En este capítulo se describe los elementos fundamentales relacionados con la Web Semántica y los Datos Enlazados. Ofrece un análisis de los navegadores de

Datos Enlazados y los complementos para semantizar el navegador Web Mozilla Firefox. Se detallan las principales característica de la metodología que guiará el proceso de desarrollo de la extensión. Se definen las herramientas y tecnologías que son utilizadas en la implementación.

Capítulo 2:"Planificación y diseño del sistema": En este capítulo se describe el sistema a desarrollar, con el objetivo de asegurar que los usuarios finales y desarrolladores tengan un entendimiento común del proyecto. Abarca las fases de planificación y diseño de la extensión. Se generan las historias de usuario, se estima el esfuerzo asociado a su implementación y se agrupan por iteraciones teniendo en cuenta al nivel de prioridad asociado. Además se estructura todo el sistema preparándolo para la implementación y prueba. Entre los contenidos a los que se hace referencia se encuentran los patrones de diseño seleccionados y la confección de las tarjetas CRC (Clases, Responsabilidad, Colaboración).

Capítulo 3: "Implementación y prueba": En este capítulo se describen las fases de desarrollo y prueba propias de la metodología de desarrollo XP. Se exponen detalladamente las iteraciones generadas por la planificación, así como las tareas que se plantearon para la realización de cada una de las historias de usuario. En cada iteración se desarrollan las historias de usuario ordenadas por prioridad en el sistema con el objetivo de obtener una primera versión del producto con las principales características o funcionalidades para ser mostrado al cliente. También se detallan las pruebas de aceptación para confirmar que las historias de usuario han sido implementadas correctamente al final de cada iteración, garantizando que los requerimientos han sido cumplidos y que la aplicación es aceptable.

Capítulo 1. Fundamentación teórica

En este capítulo se describe los elementos fundamentales relacionados con la Web Semántica y los Datos Enlazados. Ofrece un análisis de los navegadores de Datos Enlazados y los complementos para semantizar el navegador Web Mozilla Firefox. Se detallan las principales característica de la metodología que guiará el proceso de desarrollo de la extensión. Se definen las herramientas y tecnologías que son utilizadas en la implementación.

1.1 La Web Semántica

A partir de la integración de toda una infraestructura tecnológica que permita el intercambio global de conocimiento asistido por máquina y la codificación del significado de la información mediante lenguajes de marcado, toma forma el concepto de la Web Semántica como "*una extensión de la Web actual en la que el significado de la información esté bien definido, permitiendo al hombre y las máquinas trabajar en estrecha cooperación*" (4). La Web Semántica es un área prolífera en la confluencia de la Inteligencia Artificial y las tecnologías Web, que propone nuevas técnicas y paradigmas para la representación de la información y el conocimiento que faciliten la localización, compartición, integración y recuperación de recursos a través de la Web. Este enfoque propone enriquecer la estructuración de la información y la agregación de componentes semánticos que puedan ser procesados de forma automática. La nueva generación de formatos está encabezada por XML (Extensible Markup Language) y RDF, los cuales incluirán ontologías[4] que especificarán las reglas lógicas para que los agentes de software reconozcan y clasifiquen cada concepto.

Principales Componentes:

Entre los principales componentes de la Web Semántica se pueden encontrar XML, XML Schema, RDF, RDF Schema y OWL.

- **XML**: Provee una sintaxis elemental para las estructuras de contenidos dentro de documentos.
- **XMLSchema**: Es un lenguaje para proporcionar y restringir la estructura y el contenido de los elementos contenidos dentro de documentos XML.
- **RDF**: Es un lenguaje simple para expresar modelos de los datos, que refieren a los objetos, recursos y a sus relaciones. Un modelo de RDF-based se puede representar en sintaxis de XML.

[4]Ontología: taxonomía relacional de conceptos con atributos y relaciones, que proporcionan un vocabulario consensuado para definir redes semánticas de unidades de información interrelacionadas.

- **RDF Schema**: Es un vocabulario para describir propiedades y clases de recursos RDF-based, con semántica para generalizar jerarquías de las propiedades y clases.
- **OWL**: Es un mecanismo para desarrollar temas o vocabularios específicos en los que se puedan asociar esos recursos.

1.1.1 Capas de la Web Semántica

La infraestructura de tecnologías y lenguajes necesaria para la implementación de la Web Semántica se puede esquematizar en varias capas o niveles que se encuentran en fase de continuo desarrollo (5). Las capas superiores son las que dan el significado a los contenidos en tanto que las inferiores dan al soporte tecnológico y son la parte que el usuario final ve en la pantalla. En la Figura 1 se muestra el modelo de capas propuesto por Berners-Lee para la Web Semántica.

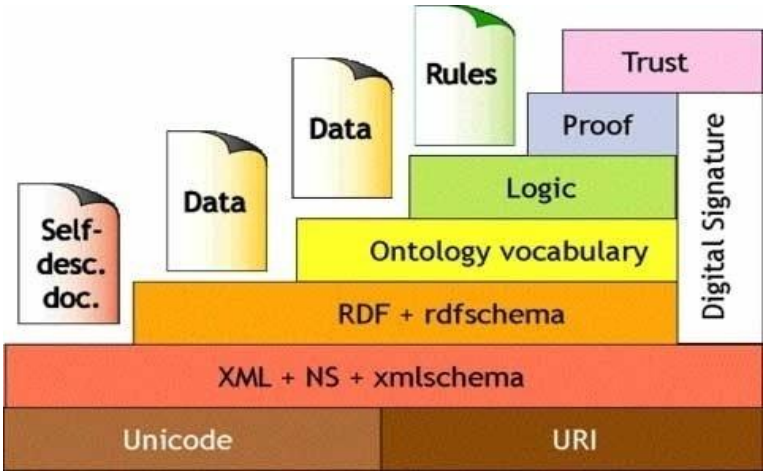


Figura 1 Modelo de capas propuesto por Berners-Lee para la Web Semántica

Unicode:

Es un estándar cuyo objetivo es proporcionar el medio por el cual un texto en cualquier forma e idioma pueda ser codificado para el uso informático. El mismo permite mostrar información en cualquier idioma y con la certeza de que no aparezcan símbolos extraños.

URI:

Son cadenas que permiten acceder a cualquier recurso de la Web. En la Web Semántica las URIs son las encargadas de identificar objetos. Si dos objetos cuentan con la misma URI pueden existir colisiones. El grupo de trabajo del W3C está intentando resolver este problema.

XML+NS+xmlschema:

Esta es la capa más técnica de la Web Semántica. En ella se encuentran agrupadas las diferentes tecnologías que posibilitan la comunicación entre agentes[5].

El XML ofrece un formato común para el intercambio de documentos. Namespaces (NS) proporciona un método para cualificar elementos y atributos de nombres usados en documentos XML asociándolos con espacios de nombre identificados por referencias URIs. XML Schema es un lenguaje que permite describir la estructura y restringir el contenido de documentos XML.

RDF+rdfschema:

Está basada en la capa anterior, define el lenguaje universal para expresar diferentes ideas en la Web Semántica. RDF es un lenguaje que define un modelo de datos para describir recursos mediante tripletas sujeto-predicado-objeto.

Los dos primeros serán URIs y el tercero puede ser URI o un valor literal. RDF Schema es un vocabulario RDF que permite describir recursos mediante una orientación a objetos. Esta capa no sólo ofrece una descripción de los datos, sino también cierta información semántica.

Ontology (Ontologías):

Posibilita clasificar la información. Esta capa permite extender la funcionalidad de la Web Semántica agregando nuevas clases y propiedades para describir los recursos.

Logic (Lógica):

Además de ontologías se precisan reglas de inferencia[6].

Proof (Pruebas):

Se intercambiarán "pruebas" escritas en el lenguaje unificador de la Web Semántica. Este lenguaje posibilita las inferencias lógicas realizadas a través del uso de reglas de inferencia.

Trust (Confianza):

Hasta que no se haya comprobado de forma exhaustiva las fuentes de información, los agentes deberían ser muy escépticos acerca de lo que leen en la Web Semántica.

[5] Agente inteligente: es una entidad capaz de percibir su entorno, procesar tales percepciones y responder o actuar en su entorno de manera racional, es decir, de manera correcta y tendiendo a maximizar un resultado esperado.
[6] Reglas de inferencia: La forma de representación del conocimiento más usada, también llamada reglas de producción. Casi todos los sistemas expertos están basados en este tipo de representación.

Digital Signature (Firma digital):

Utilizada por los ordenadores y agentes para verificar que la información ha sido ofrecida por una fuente de confianza. (6)

1.2 Lenguajes que sustentan la Web Semántica

La Web Semántica dispone de lenguajes para aportar descripciones explícitas de los recursos de la Web. Estos lenguajes permitirán a analizar la información y extraer de ella el significado semántico que representa.

1.2.1 Lenguaje RDF (Resource Description Framework)

RDF es un lenguaje para la representación de la información sobre los recursos en la Web, particularmente dirigido para la representación de los metadatos. Es una base para procesar metadatos; proporciona interoperabilidad entre aplicaciones que intercambian información legible por máquina en la Web. RDF destaca por la facilidad para habilitar el procesamiento automatizado de los recursos Web. Puede utilizarse en distintas áreas de aplicación; por ejemplo:

- En la recuperación de recursos para proporcionar mejores prestaciones a los motores de búsqueda.
- En la catalogación para describir el contenido y las relaciones de contenido disponibles en un sitio Web, una página Web, o una biblioteca digital particular, por los agentes de software inteligentes para facilitar el intercambio y para compartir conocimiento.
- En la calificación de contenido.
- En la descripción de colecciones de páginas que representan un "documento" lógico individual.
- Para describir los derechos de propiedad intelectual de las páginas Web.

El objetivo general de RDF es definir un mecanismo para describir recursos que no cree ninguna asunción sobre un dominio de aplicación particular, ni defina (a priori) la semántica de algún dominio de aplicación. La definición del mecanismo debe ser neutral con respecto al dominio, sin embargo el mecanismo debe ser adecuado para describir información sobre cualquier dominio. (7)

La esencia de RDF es pues, un modelo formal para la representación de las propiedades y los valores de esas propiedades de los recursos de información.

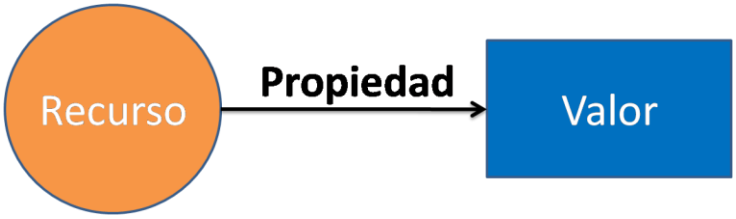


Figura 2 Modelo formal para la representación de las propiedades y valores

Las propiedades de RDF se pueden entender como atributos de los recursos y en este sentido corresponden a los pares tradicionales de atributo-valor. Se puede relacionar también con el diseño orientado a objetos donde los recursos corresponden a objetos y las propiedades corresponden a ejemplos de variables.

El modelo de datos que propone RDF consiste en tres tipos de objetos:

Recursos: cualquier objeto Web identificable unívocamente por un URI, es decir, un identificador uniforme de recursos como un URL. Un recurso puede ser un documento HTML; una parte de una página Web como por ejemplo un elemento HTML o XML dentro de un documento fuente, una colección de páginas, un sitio Web completo; y en síntesis, cualquier recurso entendido como objeto de información.

Propiedades: son aspectos específicos, características, atributos o relaciones utilizadas para describir recursos. Cada tipo de propiedad tiene sus valores específicos, define los valores permitidos, los tipos de recursos que puede describir y las relaciones que existen entre las distintas propiedades.

Declaraciones (sentencias o enunciados): una declaración RDF es una propiedad más el valor de dicha propiedad para un recurso específico. Está compuesta por 3 partes individuales:

- sujeto: recurso.
- predicado: propiedad.
- objeto: valor de la propiedad. Puede ser otro recurso o puede ser un literal, es decir, un recurso (especificado por un URI) o una cadena simple o primitiva otro tipo de datos definidos por XML. El contenido de un literal no es interpretado por RDF en sí mismo y puede contener marcado XML adicional. Los literales se distinguen de los recursos en que el modelo RDF no permite que los literales sean sujeto de una declaración.

Además se puede distinguir dos tipos de construcciones sintácticas para codificar RDF: por un lado la serializada que expresa, de una forma muy regular, todas las capacidades de un modelo de datos RDF; y por otro la sintaxis abreviada que incluye construcciones adicionales.

RDF es el modelo más promisorio para asociar información sobre el contenido de los recursos Web, y no es arriesgado decir que promete ser el modelo de descripción de la información para las bibliotecas digitales del siglo XXI, así como para optimizar, de forma generalizada, la búsqueda y recuperación en la Web. (8)

1.2.2 Lenguaje RDF Schema (RDFS)

RDF provee una forma para expresar enunciados simples acerca de los recursos usando propiedades y valores. Sin embargo, la comunidad de usuarios de RDF se percató que se necesitaba indicar algunas veces que lo que estaban describiendo eran tipos o clases específicas de recursos. RDF por sí sólo no proporciona tal vocabulario, por lo que las clases y propiedades se describen en RDF Vocabulary (también conocido como RDF schema).

RDF schema no proporciona un vocabulario sobre aplicaciones orientadas a clases, sino que provee de mecanismos para especificar que tales clases y propiedades son parte de un vocabulario y de cómo se espera su relación. También define a los recursos como instancias de una o más clases, las cuales pueden ser organizadas en forma jerárquica. Por lo tanto, RDF schema extiende a RDF para incluir un amplio vocabulario con un significado adicional. (9)

1.2.3 Lenguaje SPARQL (Simple Protocol and RDF Query Language)

Considerando que cada recurso estaría enlazado, se tendría una gran base de datos formada por ficheros RDF, viendo esta situación, se hace necesaria una herramienta que permita realizar consultas a dicha base de datos, y optimizar así las búsquedas de información, pero basada en metadatos. Ante esta necesidad el W3C ha propuesto como iniciativa a SPARQL, como un lenguaje de consulta RDF.

Los desarrolladores y usuarios finales se encontrarán con una gran cantidad de recursos Web teniendo la facilidad para representar y utilizar los resultados obtenidos de manera eficiente, sustentándose su utilidad en la necesidad de recuperar y organizar la información de diferentes fuentes.

Las consultas SPARQL cubren tres objetivos:

- Extraer información en forma de URIs y literales.
- Extraer sub-estructuras RDF.

- Construir nuevas estructuras RDF partiendo de resultados de consultas.

SPARQL tiene especificaciones que explican diferentes partes de su funcionalidad; consiste en un lenguaje de consulta, un formato para las respuestas, y un medio para el transporte de consultas y respuestas:

- SPARQL Query Language: Componente principal de SPARQL. Explica la sintaxis para la composición de sentencias y su concordancia.
- SPARQL Protocol for RDF: Formato utilizado para la devolución de los resultados de las búsquedas (queries SELECT o ASK), a partir de un esquema XML.
- SPARUL (SPARQL Update): Hace actualizaciones del contenido RDF. La especificación ha sido desarrollada por Hewlett-Packard y actualmente no es un standard reconocido.
- SPARQL Query Results XML Format: Describe el acceso remoto de datos y la transmisión de consultas de los clientes a los procesadores. Utiliza WSDL para definir protocolos remotos para la consulta de bases de datos basadas en RDF. (10)

Existe poca información sobre este lenguaje ya que es reciente en W3C, pero muchas plataformas de RDF ya lo implementan entre las que se destacan el editor de ontologías Protegé y la API [7] de Java para RDF Jena.

1.3 Datos Enlazados

La Web Semántica no se trata únicamente de la publicación de datos en la Web, sino que éstos se pueden vincular a otros, de forma que las personas y las máquinas puedan explorar la Web de los datos, pudiendo llegar a información relacionada que se hace referencia desde otros datos iniciales (10). Los Datos Enlazados es la forma que tiene la Web Semántica de vincular los distintos datos que están distribuidos en la Web.

De la misma forma que la Web del hipertexto, la Web de los datos se construye mediante documentos en la Web. Sin embargo, y a diferencia de la Web del hipertexto, donde los enlaces son relaciones entre puntos de los documentos escritos en HTML, los datos enlazan elementos arbitrarios que se describen en RDF. El objetivo de los Datos Enlazados es permitir a las personas compartir datos estructurados en la Web con la misma facilidad que en la Web tradicional.

[7] API (Interfaz de programación de aplicaciones): es el conjunto de funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción. Son usados generalmente en las bibliotecas

El término Datos Enlazados se refiere a un estilo de publicación y el mecanismo de interconexión de datos estructurados en la Web. El argumento detrás de Datos Enlazados es que el valor y la utilidad de los datos aumentan cuanto más se está interrelacionado con otros datos (11).

Los datos se vinculan y se exploran de una forma similar a la utilizada para vincular los documentos HTML. Se basa en la aplicación de ciertos principios básicos y necesarios, que fomentarán el crecimiento de la Web, tanto a nivel de los documentos HTML (vista clásica de la Web), como a nivel de los datos expresados en RDF (vista de la Web Semántica):

- Usar URIs para identificar las cosas.
- Usar URIs HTTP.
- Ofrecer información sobre los recursos usando RDF.
- Incluir enlaces a otros URIs.

La aplicación de estos principios da lugar a la creación de un patrimonio común de datos en la Web, un espacio donde las personas y organizaciones pueden publicar y consumir datos sobre cualquier cosa. Este patrimonio común de datos a menudo se llama Web Semántica.

Para lograr tener los datos interconectados, como si la Web fuese una gran base de datos, se deben respetar los cuatro pasos anteriores. Gracias a esta interconexión, se permite reutilizar la información de cualquier manera, lo que ofrece un valor añadido a la Web.

Así como los navegadores Web tradicionales permiten a los usuarios navegar entre las páginas HTML siguiendo enlaces de hipertexto, los navegadores de Datos Enlazados permiten a los usuarios navegar entre las fuentes de datos siguiendo los enlaces RDF. El resultado es que un usuario puede iniciar la navegación en una fuente de datos y recorrer progresivamente la Web siguiendo los enlaces RDF en lugar de enlaces HTML.

1.4 Navegadores de Datos Enlazados

Para el desarrollo de navegadores semánticos existen dos métodos diferentes. Uno de ellos es implementar navegadores semánticos, donde se pretende hacer navegadores pensados especialmente para este caso. El otro es, semantizar el navegador, el cual consiste en añadir determinados elementos a los navegadores actuales para extender sus posibilidades de navegabilidad aprovechando determinadas características semánticas adicionadas a las páginas Web.

1.4.1 Navegadores Semánticos

El Tabulador

El Tabulador es un navegador y editor de datos genéricos. Proporciona una manera de ver los datos RDF en la Web. Permite la construcción visual de consultas SPARQL. La idea es navegar por recursos de forma escalable, lo que supone que no necesita cargar en memoria toda la información contenida dentro del fichero que está visualizando (generalmente en RDF, pero no de forma excluyente). La información se va mostrando conforme el usuario la va necesitando. La extensión de tabulador es un programa en código abierto, basado en Ajax y funciona dentro de Firefox o como un widget de Opera permitiendo manejar datos y documentos.

La versión en línea es un conjunto de scripts de Javascript de código abierto, por lo que tienden a ser lentos con archivos de gran tamaño. Si usted recibe un mensaje de que una secuencia de comandos está tomando un tiempo, tiene la opción de dejar que se ejecute o abortarlo. Esta es una característica de Firefox, se puede controlar pero muchos elementos de interfaz de usuario se pierden, por ejemplo: la vista de tabla. Otra de las deficiencias encontradas cuando se instaló, es que para ver los sitios fuera del dominio local en Firefox, debe cambiar la configuración de Firefox (12).

Disco

Permite navegar por la Web Semántica como un conjunto de fuentes de datos no consolidados. El navegador hace que toda la información que se puede encontrar en la Web Semántica sobre un recurso específico, se muestre como una página HTML. Esta descripción de recursos contiene hipervínculos que le permiten navegar entre los recursos. El navegador Disco se implementa como una delgada capa de presentación en la parte superior de la Web Semántica Client Library. Este se encuentra distribuido bajo los términos de la licencia de la Berkeley Software Distribution (BSD). Disco está disponible en la siguiente dirección http://www4.wiwiss.fu-berlin.de/rdf_browse solo como una aplicación de servidor, por lo que su selección dificultaría la realización de extensiones (14).

RKBExplorer

RKBExplorer es un navegador para investigar en el dominio de las Ciencias de la Computación. Combina la información de múltiples fuentes heterogéneas, como fuentes RDF públicas, páginas Web personales y bases de datos con el fin de proporcionar una visión integral de este espacio multidimensional (15). Es una aplicación de servidor que utiliza las bibliotecas de java para su ejecución, razón por la cual el trabajo con el mismo se hace muy lento.

Longwell

Longwell es un navegador Web para RDF. Mezcla la flexibilidad del modelo de datos RDF con la eficiencia de la navegación facetada, permitiendo visualizar y navegar por un conjunto de datos RDF arbitrariamente complejos. El aspecto más importante de Longwell es la navegación facetada paradigma de interfaz de usuario. Una faceta es un campo de metadatos particular, que se considera importante para el conjunto de datos en el que se está navegando.

Los requisitos para el funcionamiento de Longwell son:

- Una Máquina Virtual de Java (JVM) versión 1.4 o posterior.
- Una instalación de Apache Maven 2.0 o posterior.

Longwell es un software de código abierto y está disponible bajo una licencia tipo BSD (16). Su principal problema radica en que depende de varias bibliotecas externas para su funcionamiento, las cuales no se distribuye con los paquetes de él, sino que todas las dependencias son descargadas por Maven y el repositorio presenta problemas, lo que trae como consecuencia que la instalación no se complete con éxito.

Haystack

Es un navegador Web Semántica que fue diseñado para ser extensible, basado en la plataforma RCP (Rich Client Platform, por su siglas en inglés) de Eclipse, Haystack permite a los desarrolladores añadir nuevas funcionalidades en forma de plug-ins de Eclipse. Por otra parte, los usuarios finales pueden refinar sus prorrogados por la creación y selección de componentes de interfaz de usuario llamado lentes de semántica. Una lente de semántica es una descripción legible por máquina (como RDF) de cómo extraer información relevante para un conjunto específico de circunstancias. Haystack es un software de código abierto, bajo la licencia BSD (17). Este navegador no se pudo probar debido a que las direcciones para las descargas del mismo se encuentran obsoletas.

Rhizomer

Rhizomik es una plataforma que permite crear portales Web semánticos. Usa la infraestructura Rhizomer que permite presentar datos semánticos, navegar por ellos, y la edición de los mismos. Tanto la presentación como la edición se realizan de forma automática. La presentación de datos mejora otras presentaciones automáticas de grafos RDF gracias al paradigma Rhizomic, mediante el cual se puede presentar información semántica “cercana” (en el grafo) aunque se interpongan nodos anónimos, de

forma que se muestra más información. El código fuente de Rhizomik o de Rhizomer no es público aún, por lo que la evaluación no puede ser exhaustiva (18).

OpenLink Data Explorer

El OpenLink Data Explorer(ODE), anteriormente conocido como el navegador de OpenLink RDF, es un navegador de Datos RDF implementado por OpenLink Software con código abierto OpenLink AJAX y AJAR Toolkit (OAT). Ofrece una interfaz de navegador Web impulsado para interactuar con RDF basado en Datos Enlazados (19). También está disponible como una extensión de Firefox al probarla se pudo observar que el funcionamiento del mismo se basa en el redireccionamiento hacia la aplicación de servidor. Su selección no sería la más conveniente por lo anteriormente explicado.

1.4.2 Complementos para semantizar Mozilla Firefox

Teniendo en cuenta que para los usuarios finales los navegadores tradicionales son herramientas con las que están familiarizadas, el método de semantizar el navegador está siendo muy utilizado por los desarrolladores para crear extensiones que permitan ampliar las posibilidades de los navegadores al interactuar con la Web Semántica. El mayor número de extensiones encontradas durante la búsqueda corresponden al navegador Web Mozilla Firefox. Este navegador es uno de los más utilizados en la actualidad, y se encuentra respaldado por una comunidad que constantemente desarrollan complementos y versiones del mismo. En este momento el estudio está dirigido hacia los complementos desarrollados para Firefox, los seleccionados se describen a continuación.

Piggy Bank 3.1.0

Se trata de una extensión desarrollada en Java para el navegador Firefox que permite extraer determinados elementos clave de una página Web y almacenarlos en RDF. Dependiendo de la información que se encuentre en una página Web, Piggy Bank actuará de dos maneras diferentes, si el sitio tiene un fichero RDF o meta tags del HTML, el programa capturará esa información y la integrará en un repositorio, a modo de base de datos local, organizada en función de la estructura descrita. Si por el contrario, el sitio no dispone de información de este tipo, el software invocará a un scraper para que extraiga esta información y la estructure. Los scrapers son programas capaces de trabajar con cualquier texto para procesarlo y estructurarlo. El Piggy Bank usa tres scrapers diferentes escritos en Java Script.

Además trabaja con un banco semántico, que es un repositorio comunitario de descripciones realizadas en RDF que permite a sus usuarios compartir la información que han recogido. Actualmente existen dos bancos semánticos: uno genérico que su uso trae muchas dificultades y otro que es

específico para un dominio. El Piggy Bank solo es compatible con la versión 2.0 de Firefox. Para su instalación requiere de un plug-in de java que dificulta la configuración del mismo. Es un software de código abierto distribuido bajo la licencia BSD, sin embargo las bibliotecas para las búsquedas no son liberadas bajo la misma licencia (13).

VisBrowser 1.0

Es un complemento de Firefox, compatible con las versiones 3.0 o superior. Este es un navegador visual de la Web Semántica desarrollado para navegar por espacios de Datos Enlazados. Utilizando el código JavaScript InfoVis Toolkit y la Biblioteca SPARQL JavaScript, que ofrece una vista de gráfico visual radial del recurso que se busca. Desarrollado por la Universidad de Queensland (20).

RDFa Developer 1.0b1

Es un complemento de Firefox, compatible con las versiones 3.5 o superior. Permite visualizar todas las tripletas RDFa en una página Web, muestra una lista con los errores y advertencias que se encuentran al analizar el documento y con ello es posible ejecutar consultas SPARQL sobre el contenido RDFa. Está desarrollado en JavaScript y se distribuye baja la licencia GNU General Public License versión 3.0 (GPLv3) (20).

Semantic Radar 1.1

Es un complemento de Firefox, compatible con las versiones 1.5 o superior. Inspecciona las páginas Web de enlaces a los metadatos de la Web Semántica y los datos en presencia de la misma, mostrando un icono en la barra de estado del navegador. Actualmente soporta auto-descubrimiento de RDF (SIOC, FOAF, DOAP y cualquier tipo) y la detección de RDFa metadatos (21).

Luego de realizar el estudio de las herramientas que han sido desarrolladas para visualizar el contenido semántico de los recursos que se encuentran disponibles en la Web; la dirección que seguirá esta investigación será la de semantizar el navegador Web Mozilla Firefox, para extender sus funcionalidades de forma tal que permita navegar y visualizar información con contenido semántico.

1.5 Metodología de desarrollo de software

"Las metodologías de desarrollo de software son un conjunto de procedimientos, técnicas y ayudas a la documentación para el desarrollo de productos software" (22).Estas metodologías pretenden guiar a los desarrolladores al crear un nuevo software. Debido a que los requisitos de un software a otro son tan variados y cambiantes, ha dado lugar a que exista una gran variedad de metodologías para el desarrollo del software, las cuales se pueden clasificar en dos grandes grupos:

Metodologías Pesadas: Están orientadas al control de los procesos, estableciendo rigurosamente las actividades a desarrollar, herramientas a utilizar y notaciones que se usarán.

Metodologías ligeras/ágiles: Son aquellas metodologías orientadas a la interactuación con el cliente y el desarrollo incremental del software, mostrando versiones parcialmente funcionales del software al cliente en intervalos cortos de tiempo, para que pueda evaluar y sugerir cambios en el producto según se va desarrollando (23). En la tabla 1, se muestran características que identifican a estos dos tipos de metodologías.

Tabla 1 Diferencias entre metodologías ágiles y no ágiles

Metodología Ágil	Metodología Tradicional
Pocos Roles, más genéricos y flexibles.	Más Roles, más específicos.
Cliente es parte del equipo de desarrollo (además in-situ).	El cliente interactúa con el equipo de desarrollo mediante reuniones.
Orientada a proyectos pequeños. Corta duración (o entregas frecuentes), equipos pequeños (< 10 integrantes) y trabajando en el mismo sitio.	Aplicables a proyectos de cualquier tamaño, pero suelen ser especialmente efectivas/usadas en proyectos grandes y con equipos posiblemente dispersos.
La arquitectura se va definiendo y mejorando a lo largo del proyecto.	Se promueve que la arquitectura se defina tempranamente en el proyecto.
Se esperan cambios durante el proyecto.	Se espera que no ocurran cambios de gran impacto durante el proyecto.

Luego de analizar las características que presentan las metodologías tradicionales y las ágiles, se llega a la conclusión que la metodología que más se adapta al contexto del proyecto por sus recursos técnico y humanos, el tiempo de desarrollo y el tipo de sistema es una metodología ágil. Una de las cualidades más destacables en una metodología ágil es su sencillez, tanto en su aprendizaje como en su aplicación, lo que implica que se reduzca los costos de implantación en un equipo de desarrollo.

Una de las metodologías ágiles más usadas y documentadas en la actualidad es Extreme Programming (XP), la cual es la metodología seleccionada para que guíe todo el proceso de desarrollo del software.

1.5.1 Extreme Programming

Extreme Programming (XP) es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico.

XP define cuatro variables para proyectos de software: coste, tiempo, calidad y ámbito. Además de estas cuatro variables, Beck propone que sólo tres puedan ser establecidas por las fuerzas externas (jefes de proyecto y clientes), mientras que el valor de la cuarta variable debe ser establecido por los programadores en función de las otras tres.

Roles de XP

En esta metodología se utiliza el concepto de roles para organizar quienes se encargan de cada una de la actividades que deben realizarse en el transcurso del proyecto. Cada uno de estos papeles son desempeñados por unos o varios integrantes del grupo, sin descartar la posibilidad de rotar los roles entre el equipo durante la realización del sistema. A continuación se detallan los roles que se desempeñarán en el proyecto:

Programador: escribe las pruebas unitarias y produce el código del sistema. Debe existir una comunicación y coordinación adecuada entre los programadores y otros miembros del equipo.

Encargado de pruebas (Tester): ayuda al cliente a escribir las pruebas funcionales. Ejecuta las pruebas regularmente, difunde los resultados en el equipo y es responsable de las herramientas de soporte para pruebas (24).

1.6 Herramientas y tecnologías actuales

1.6.1 Herramienta CASE (*Computer Aided Software Engineering*)

Se puede definir a las herramientas CASE como un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo de un Software (25).

Visual Paradigm 6.4

Visual Paradigm es una herramienta CASE que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Este software de modelado UML ayuda a una rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Soporta todos los tipos de diagramas de clases y genera documentación. La herramienta también proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML. Presenta licencia gratuita y comercial. Es multiplataforma, fácil de instalar y actualizar, compatible entre ediciones.

1.6.2 Lenguajes de Programación

JavaScript

JavaScript es un lenguaje de programación que se utiliza principalmente para crear páginas Web dinámicas. Es un lenguaje con muchas posibilidades, permite la programación de pequeños scripts, pero también de programas más grandes, orientados a objetos, con funciones y estructuras de datos complejas.

Técnicamente, JavaScript es un lenguaje de programación interpretado, por lo que no necesita una fase de compilación como Java o C y el explorador no ha de cargar ninguna máquina virtual para ejecutar el código. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios. A pesar de su nombre, JavaScript no guarda ninguna relación directa con el lenguaje de programación Java (26).

La inigualable popularidad de JavaScript como lenguaje de programación de aplicaciones Web se ha extendido a otras aplicaciones y otros entornos no relacionados con la Web. Herramientas como Adobe Acrobat permiten incluir código JavaScript en archivos PDF. Otras herramientas de Adobe como Flash y Flex utilizan ActionScript, un dialecto del mismo estándar de JavaScript, Photoshop permite realizar pequeños scripts mediante JavaScript y la versión 6 de Java incluye un nuevo paquete (denominado JavaScript) que permite integrar ambos lenguajes.

Los navegadores modernos disponibles actualmente incluyen soporte de JavaScript hasta la versión correspondiente a la tercera edición del estándar ECMA-262.Las acciones del usuario en el navegador Mozilla Firefox se controlan mediante JavaScript .A través de este lenguaje se puede ampliar el navegador Firefox, se pueden modificar o añadir elementos de la interfaz de usuario, y realizar cambios de apariencia y funcionalidades.

1.6.3 AJAX (Asynchronous JavaScript and XML)

El término AJAX se presentó por primera vez en el artículo "Ajax: A New Approach to Web Applications" publicado por Jesse James Garrett el 18 de Febrero de 2005. Hasta ese momento, no existía un término normalizado que hiciera referencia a un nuevo tipo de aplicación Web que estaba apareciendo.

En realidad, el término AJAX es un acrónimo de *Asynchronous JavaScript + XML*, que se puede traducir como "JavaScript asíncrono + XML".

El artículo define AJAX de la siguiente forma: "*Ajax no es una tecnología en sí mismo. En realidad, se trata de varias tecnologías independientes que se unen de formas nuevas y sorprendentes*".

Las tecnologías que forman AJAX son:

- XHTML y CSS, para crear una presentación basada en estándares.
- DOM, para la interacción y manipulación dinámica de la presentación.
- XML, XSLT y JSON, para el intercambio y la manipulación de información.
- XMLHttpRequest, para el intercambio asíncrono de información.
- JavaScript, para unir todas las demás tecnologías.

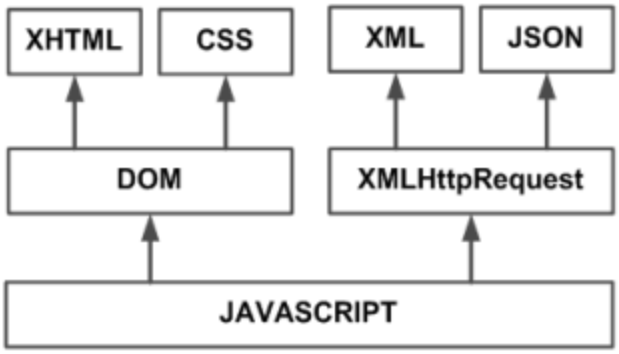


Figura 3 Tecnologías agrupadas bajo el concepto de AJAX

AJAX permite mejorar completamente la interacción del usuario con la aplicación, evitando las recargas constantes de la página, ya que el intercambio de información con el servidor se produce en un segundo plano.

Las aplicaciones construidas con AJAX eliminan la recarga constante de páginas mediante la creación de un elemento intermedio entre el usuario y el servidor. La nueva capa intermedia de AJAX mejora la respuesta de la aplicación, evitando que el usuario nunca se encuentre con una ventana del navegador vacía esperando la respuesta del servidor.

Desarrollar aplicaciones AJAX requiere un conocimiento avanzado de todas y cada una de las tecnologías anteriores. Desde su aparición, se han creado cientos de aplicaciones Web basadas en AJAX. En la mayoría de casos, AJAX puede sustituir completamente a otras técnicas como Flash. Además, en el caso de las aplicaciones Web más avanzadas, pueden llegar a sustituir a las aplicaciones de escritorio. (27)

XUL

XUL (Lenguaje de interfaz de usuario) es un lenguaje multiplataforma para describir la interfaz de usuario de aplicaciones de Mozilla, basado en XML. Fue creado para facilitar y acelerar el desarrollo del navegador Mozilla. Permite construir aplicaciones ricas en funcionalidades y plataforma que se pueden ejecutar conectado o desconectado de Internet.

Presenta todas las ventajas de otros lenguajes XML. Por ejemplo, XHTML[8] u otros lenguajes XML como MathML[9] o SVG[10] se pueden usar junto a él. Además, el texto que se muestra con XUL es fácilmente localizable, lo que significa que puede ser traducido a otros idiomas con poco esfuerzo. Permite aplicar hojas de estilo para modificar la apariencia de la interfaz de usuario (28).

Proporciona la habilidad de crear la mayoría de los elementos encontrados en las interfaces gráficas modernas. Es tan general que este puede ser aplicado a las necesidades específicas de ciertos dispositivos y tan poderoso que los desarrolladores pueden crear sofisticadas interfaces.

Algunos elementos que pueden ser creados son:

- Controles de entrada tales como cuadros de texto y cajas de chequeo.

[8] XHTML: Lenguaje extensible de marcado de hipertexto, es el lenguaje de marcado pensado para sustituir a HTML como estándar para las páginas web.
[9] MathML: Mathematical Markup Language es un lenguaje de marcado basado en XML, cuyo objetivo es expresar notación matemática de forma que distintas máquinas puedan entenderla.
[10] SVG: Scalable Vector Graphics es una especificación para describir gráficos vectoriales bidimensionales, tanto estáticos como animados, en formato XML.

- Barra de herramientas con botones u otros contenidos.
- Menús en barras de menú o menú emergente.
- Pestañas de diálogo.
- Árbol de información jerárquica o tabulada.
- Teclas de accesos directo (28).

Estas aplicaciones son fácilmente personalizables con texto alternativo, gráficos y el diseño para que puedan ser fácilmente localizados por el usuario. Con XUL, una interfaz puede ser implementada y modificada de forma fácil y rápida.

1.6.4 Framework de desarrollo

En el contexto del desarrollo de software, un framework es una estructura de archivos y utilidades que aceleran la programación de una aplicación informática, proporcionando una metodología de trabajo que sistematiza y facilita la generación de formularios, funciones y módulos de uso común, permitiendo al desarrollador dedicar su atención hacia los aspectos específicos de cada aplicación.

JQuery

Es una biblioteca o framework de JavaScript creada inicialmente por John Resig. Posee un doble licenciamiento bajo la Licencia MIT y la Licencia Pública General de GNU v2, permitiendo su uso en proyectos libres y privativos. JQuery consiste en un único fichero JavaScript que contiene las funcionalidades comunes de DOM, eventos, efectos y AJAX.
Permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la tecnología AJAX a páginas Web.
JQuery, al igual que otras bibliotecas, ofrece una serie de funcionalidades basadas en JavaScript que de otra manera requerirían de mucho más código, es decir, con las funciones propias de esta biblioteca se logran grandes resultados en menos tiempo y espacio.
Características:

- Selección de elementos DOM.
- Eventos.
- Manipulación de la hoja de estilos CSS.

- AJAX.
- Soporta extensiones.
- Compatible con los navegadores Mozilla Firefox 2.0+, Internet Explorer 6+, Safari 3+, Opera 9+ y Google Chrome. (29)

1.6.5 Arbor.js

Arbor.js es una librería de visualización gráfica construida utilizando JQuery para realizar diagramas animados de forma muy simple y sin la necesidad de prácticamente escribir código, simplemente es necesario pasarle al plugin por parámetros los datos necesarios para crear el diagrama de relaciones. En lugar de tratar de ser un marco que todo lo abarca, Arbor proporciona un algoritmo eficiente para la organización gráfica.

Este plugin permite personalizar aspectos importantes del diagrama, como los colores de los nodos, el tipo de relación entre los mismos y la clase de animación a realizar al abrirse el nodo y mostrar sus hijos. Además posibilita que cada nodo realice una acción determinada al hacerse click sobre él, por ejemplo la página Web oficial de este plugin utiliza un diagrama para mostrar las opciones del sitio. Arbor.js permite la incorporación de datos mediante JSON, algo imprescindible para que el plugin sea completo y fácil de actualizar mediante otros lenguajes como PHP en comunión con MySQL. Arbor se publica bajo la licencia MIT.

1.6.6 Entorno de desarrollo integrado

Un entorno de desarrollo integrado (en inglés Integrated Development Environment o IDE) es un programa compuesto por una serie de herramientas que utilizan los programadores para desarrollar código. Esta herramienta puede estar pensada para su utilización con un único lenguaje de programación o bien puede dar cabida a varios de estos. (30)

NetBeans IDE 6.9

El IDE NetBeans es un reconocido entorno de desarrollo integrado disponible para Windows, Mac, Linux y Solaris. El proyecto NetBeans está formado por un IDE de código abierto y una plataforma de aplicación que permite a los desarrolladores crear con rapidez aplicaciones Web, empresariales, de escritorio y móviles utilizando la plataforma Java, así como JavaFX, PHP, JavaScript y Ajax, Ruby y Ruby on Rails, Groovy and Grails y C/C++.

El proyecto de NetBeans está apoyado por una comunidad de desarrolladores dinámica y ofrece documentación y recursos de formación exhaustivos, así como una amplia selección de complementos de terceros.

Algunas funciones importantes para su uso en los Lenguajes Web: HTML, CSS, JavaScript:

- Reestructuración y búsqueda de usos para CSS y lenguajes parecidos a HTML.
- Autocompletado y enlace para atributos id y class.
- Reestructuración de estilos en línea de CSS. (31)

1.6.7 SPARQL endpoint

Un SPARQL endpoint es un protocolo de servicio conforme SPARQL. Permite a los usuarios consultar una base de conocimientos a través del lenguaje SPARQL. Los resultados suelen ser devuelto en uno o más formatos procesables por máquina. Un SPARQL endpoint fue concebido como una interfaz hombre-máquina amigable hacia una base de conocimientos. Tanto en la formulación de las preguntas y la presentación legible de los resultados normalmente serán ejecutadas por el programa de llamada, y no se realiza de forma manual por los usuarios humanos. (32)

1.7 Conclusiones parciales del capítulo

- Se describieron las principales características de los lenguajes que sustentan la Web Semántica y los mecanismos para estructurar y acceder a la información. A partir de la investigación realizada sobre los navegadores de Datos Enlazados y los complementos para semantizar el navegador Mozilla Firefox, no se encontró un navegador que permita una navegación intuitiva y sencilla por un árbol RDF y que ofrezca características importantes a usar en el dominio de ciencias de la vida. Por lo que la investigación esta dirigida a semantizar el navegador Web Mozilla Firefox.
- Del estudio de las herramientas y tecnologías actuales se definió para el proceso de desarrollo de la extensión la metodología XP, Visual Paradigm 6.4 como herramienta CASE y UML como lenguaje de modelado. Los lenguajes de programación a utilizar serán JavaScript y XUL, con NetBeans 6.9 como IDE .Se empleará el protocolo de servicios SPARQL endpoint para consultar base de datos biológicas a través del lenguaje SPARQL.

Capítulo 2. Planificación y diseño del sistema

En este capítulo se describe el sistema a desarrollar, con el objetivo de asegurar que los usuarios finales y desarrolladores tengan un entendimiento común del proyecto. Abarca las fases de planificación y diseño de la extensión. Se generan las historias de usuario, se estima el esfuerzo asociado a su implementación y se agrupan por iteraciones teniendo en cuenta al nivel de prioridad asociado. Además se estructura todo el sistema preparándolo para la implementación y prueba. Entre los contenidos a los que se hace referencia se encuentran los patrones de diseño seleccionados y la confección de las tarjetas CRC (Clases, Responsabilidad, Colaboración).

2.1 Objeto de estudio

2.1.1 Problema y situación problemática

La utilización de los Datos Enlazados se esta extendiendo a otros sectores entre los que se destaca la Biología. La información biológica se encuentra distribuida en diferentes repositorios, por lo que se hace difícil garantizar la coherencia de la información. El desarrollo de navegadores para Datos Enlazados esta influenciado por la complejidad a que se someten los usuarios al intentar navegar por repositorios RDF distribuidos pero enlazados. Estos navegadores poseen las características para navegar por un árbol RDF, pero no están preparados para su utilización en el ámbito de la biología. La utilización de estos navegadores por usuarios sin conocimientos profundos sobre las tecnologías semánticas se ve afectada, impidiendo la utilización de los recursos Web que pueden obtener de los repositorios. Actualmente el usuario debe conocer el esquema RDF de los repositorios por los que navega y la sintaxis SPARQL. La navegación que ofrecen los navegadores de Datos Enlazados actuales no es intuitiva, y los usuarios no tienen conocimiento sobre los repositorios por los que navega. No se potencia el conocimiento de la comunidad científica ya que los datos no son mostrados de forma integrada a los usuarios, por lo que no se pueden analizar de manera directa dicha información.

Como se observa se hace necesario un navegador o incorporar nuevas funcionalidades a los ya existentes que le permita a los usuarios no expertos en tecnologías semánticas navegar por los repositorios RDF, de manera sencilla.

2.1.2 Propuesta de sistema

Se propone desarrollar una extensión para semantizar el navegador Web Mozilla Firefox que facilite reutilizar recursos a partir de Datos Enlazados sobre repositorios RDF. Para ello se tuvo en cuenta algunas de las características más significativas de navegadores estudiados.

La nueva extensión tendrá las siguientes características:

- Diseño de consultas SPARQL mediante formularios.
- Creación de consultas SPARQL.
- Enviar la consulta al repositorio RDF.
- Visualizar resultados de consultas.
- Salvar consultas.
- Cargar consulta.
- Visualización de un subgrafo RDF en formato HTML.

La extensión utilizará el servicio SPARQL endpoints del Proyecto LinkedLifeData. LinkedLifeData es una plataforma para la integración de datos semánticos a través de almacenamiento RDF y razonamiento eficaz que ayuda a resolver los conflictos en los datos. La plataforma utiliza una extensión del modelo RDF que es capaz de rastrear la procedencia de cada hecho individual en el repositorio y, por tanto actualizar la información.

El modelo de RDF es un modelo abstracto de datos. Es compatible con la expresión eficaz de reglas declarativas que se derivan de nueva información implícita que esta materializada y catalogada. Un escenario típico de la empresa es vincular los datos internos de la empresa con la información pública de una manera significativa. LinkedLifeData hace posible la traducción de toda la información de la empresa en un modelo común de datos altamente abstracto.

Esta plataforma busca y explora más de 6 billones de declaraciones RDF desde varias fuentes, incluyendo UniProt, PubMed, Disease Ontology, LinkedCT, Reactome, Human Phenotype Ontology, HPRD, CellMap, NCBI Entrez-Gene, BioGRID, IMID, UMLS, IntAct, Symptom Ontology, DailyMed, Diseasome, ChEBI, MINT, SIDER, DrugBank, LHGDN, HumanCYC, NCI Nature.

La utilidad de esta plataforma para la presente investigación, es que se puede consultar toda la información que la misma integra haciendo uso de un SPARQL endpoint que se encuentra disponible en la siguiente URL: http://linkedlifedata.com/sparql.

2.2 Requisitos no funcionales

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener. En muchos casos los requisitos no funcionales son fundamentales en el éxito del producto. Existen múltiples categorías para clasificar a los requisitos no funcionales. A continuación se muestran las propiedades y

cualidades que permitirán satisfacer al cliente y lograr una buena calidad:

Usabilidad

- La aplicación podrá ser utilizada por personal vinculado a la biotecnología, que tengan conocimientos básicos de computación y de aplicaciones Web.

Disponibilidad

- La extensión necesita de un servicio para su funcionamiento y además de conexión a internet.

Soporte

- Se debe entregar junto a la extensión un manual de usuario donde se especifiquen como trabajar con la misma.

Interfaces de usuario

- El sistema debe tener una interfaz de usuario sencilla y fácil de usar.
- Debe garantizar claridad y correcta organización de la información para lograr una mejor comprensión de la misma.

Requisitos de Software

- La extensión podrá ser utilizada en el navegador Mozilla Firefox en su versión 2.0 o superior.

Restricciones del diseño y la implementación

- Para guiar el proceso de desarrollo de la extensión se utilizará la metodología XP, usando el lenguaje de modelación UML y como herramienta CASE Visual Paradigm 6.4.
- Los lenguajes de programación que será usado para la implementación son Javascript y XUL, con NetBeans 6.9 como IDE.

2.3 Planificación

En esta fase, los clientes plantean a grandes rasgos las historias de usuario que son de interés para la primera entrega del producto. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto.

XP plantea la planificación como un permanente diálogo entre la parte empresarial y técnica del proyecto, en la que el cliente establece la prioridad de cada historia de usuario, y correspondientemente, los programadores realizan una estimación del esfuerzo necesario de cada una

de ellas. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente. Una entrega debería obtenerse en no más de tres meses. Esta fase dura unos pocos días.

Las estimaciones de esfuerzo asociado a la implementación de las historias la establecen los programadores utilizando como medida el punto. Un punto, equivale a una semana ideal de programación. Las historias generalmente valen de 1 a 3 puntos. Por otra parte, el equipo de desarrollo mantiene un registro de la "velocidad" de desarrollo, establecida en puntos por iteración, basándose principalmente en la suma de puntos correspondientes a las historias de usuario que fueron terminadas en la última iteración.

La planificación se puede realizar basándose en el tiempo o el alcance. La velocidad del proyecto es utilizada para establecer cuántas historias se pueden implementar antes de una fecha determinada o cuánto tiempo tomará implementar un conjunto de historias. Al planificar por tiempo, se multiplica el número de iteraciones por la velocidad del proyecto, determinándose cuántos puntos se pueden completar. Al planificar según el alcance del sistema, se divide la suma de puntos de las historias de usuario seleccionadas entre la velocidad del proyecto, obteniendo el número de iteraciones necesarias para su implementación (24).

2.3.1 Historias de Usuario

Las historias de usuario (HU) representan una breve descripción del comportamiento del sistema empleando una terminología del cliente sin lenguaje técnico. Estas se realizan una por cada característica principal del sistema y se emplean para hacer estimaciones de tiempo y para el plan de lanzamientos. Las HU deben proporcionar sólo el detalle suficiente como para poder hacer razonable la estimación de cuánto tiempo requiere la implementación de la historia. Además deben impulsar la creación de las pruebas de aceptación. Durante la fase de planificación se identificaron ocho HU que a continuación se describen:

Tabla 2 Historia de usuario creación de consultas SPARQL

Historia de usuario	
Número: 1	**Nombre:** Creación de consultas SPARQL
Usuario: Desarrollador	
Prioridad en negocio: Alta	**Riesgo de desarrollo:** Medio

Puntos Estimados: 1	**Iteración asignada:** 1
Descripción: Realiza consultas a las base de datos de forma tradicional. Los usuarios con un poco más de conocimiento de SPARQL realizarán las consultas a través de un cuadro de diálogo, permitiendo que se realicen consultas a los repositorios RDF con un nivel de complejidad mayor. Los usuarios contarán con las URL de los grafos RDF, para facilitar la consulta de los mismos.	
Observaciones: Su uso esta ligado a un servicio SPARQL endpoints.	

Tabla 3 Historia de usuario enviar la consulta al repositorio RDF

Historia de usuario	
Número: 2	**Nombre:** Enviar la consulta al repositorio RDF
Usuario: Desarrollador	
Prioridad en negocio: Alta	**Riesgo de desarrollo:** Medio
Puntos Estimados: 1	**Iteración asignada:** 1
Descripción: Se obtendrá una consulta a través de formularios o de la forma tradicional y se enviará al servicio SPARQL endpoints del proyecto Linked Life Data.	
Observaciones: Esta consulta debe estar codificada en formato de URL.	

Tabla 4 Historia de usuario visualizar resultados de consultas

Historia de usuario	
Número: 3	**Nombre:** Visualizar resultados de consultas
Usuario: Desarrollador	
Prioridad en negocio: Alta	**Riesgo de desarrollo:** Medio
Puntos Estimados: 1	**Iteración asignada:** 1
Descripción: Luego de obtener el resultado de la consulta es necesario mostrar los resultados de forma integrada de manera que sea más fácil su uso por los biólogos.	

Observaciones:

Tabla 5 Historia de usuario diseño de consultas SPARQL mediante formularios

Historia de usuario	
Número: 4	**Nombre:** Diseño de consultas SPARQL mediante formularios
Usuario: Desarrollador	
Prioridad en negocio: Medio	**Riesgo de desarrollo:** Medio
Puntos Estimados: 2	**Iteración asignada:** 2
Descripción: El diseño de consultas mediante formularios les permitirá a los biólogos abstraerse de la complejidad de la sintaxis del lenguaje SPARQL, para obtener resultados de los repositorios RDF. Se inicia cuando el biólogo desea crear consultas mediante formularios. Se seleccionan los resultados que se desean obtener y las relaciones con otras variables. En dependencia de las variables escogidas se cargarán los atributos correspondientes.	
Observaciones: Su uso está ligado a un servicio SPARQL endpoints.	

Tabla 6 Historia de usuario visualizar resultados de consultas en forma de grafo

Historia de usuario	
Número: 5	**Nombre:** Representación de bases de datos en forma grafo.
Usuario: Desarrollador	
Prioridad en negocio: Baja	**Riesgo de desarrollo:** Medio
Puntos Estimados: 2	**Iteración asignada:** 2
Descripción: La visualización de las relaciones de las bases de datos biológicas en forma de grafo le permitirá a los biólogos una mejor compresión de los atributos que relacionan cada base dato. Las bases de datos serán representadas mediante nodos de diferentes formas y colores para facilitar su reconocimiento. Las relaciones que unen a cada nodo serán representadas mediante aristas.	

Observaciones:

Tabla 7 Historia de usuario salvar consulta

Historia de usuario	
Número: 6	**Nombre:** Salvar consultas
Usuario: Desarrollador	
Prioridad en negocio: Baja	**Riesgo de desarrollo:** Medio
Puntos Estimados: 1	**Iteración asignada:** 3
Descripción: Esta opción le permitirá al usuario almacenar las consultas para su próxima utilización.	
Observaciones:	

Tabla 8 Historia de usuario cargar consulta

Historia de usuario	
Número: 7	**Nombre:** Cargar consulta
Usuario: Desarrollador	
Prioridad en negocio: Baja	**Riesgo de desarrollo:** Medio
Puntos Estimados: 1	**Iteración asignada:** 3
Descripción: Esta opción le permitirá al usuario reutilizar una consulta creada con anterioridad.	
Observaciones:	

Tabla 9 Historia de usuario visualización de un subgrafo RDF en formato HTML

Historia de usuario	
Número: 8	**Nombre:** Visualización de un subgrafo RDF en formato HTML.

Usuario: Desarrollador	
Prioridad en negocio: Media	**Riesgo de desarrollo:** Medio
Puntos Estimados: 2	**Iteración asignada:** 1
Descripción: Se realizarán transformaciones de RDF a HTML, permitiendo la visualización de los subgrafos RDF en formato HTML. De esta manera se mostrarán las descripciones que se realicen sobre algún recurso en específico en un formato compresible para un usuario básico. El resultado ofrecido al usuario final puede ser totalmente personalizado mostrando los datos semánticos de forma amigable al usuario.	
Observaciones:	

2.3.2 Estimación de esfuerzo

La estimación de esfuerzo que se decidió para el desarrollo de cada una de las historias de usuario de la investigación se representa en la siguiente tabla.

Tabla 10 Plan de estimación de esfuerzo por historias de usuario

Historias de Usuario	Puntos Estimados
Diseño de consultas SPARQL mediante formularios	2
Creación de consultas SPARQL	1
Enviar la consulta al repositorio RDF	1
Visualizar resultados de consultas	1
Representación de bases de datos en forma grafo	2
Salvar consultas	1
Cargar consultas	1
Visualización de un subgrafo RDF en formato HTML.	2

2.3.3 Iteraciones

La metodología XP contiene la filosofía de las metodologías ágiles, las cuales dan mayor valor al individuo, a la colaboración con el cliente y al desarrollo incremental del software con iteraciones muy cortas (24). La etapa de implementación se ha planificado realizarla en tres iteraciones atendiendo a la cantidad de HU junto a sus prioridades en el negocio y los puntos estimados de cada una. La idea es producir rápidamente versiones del sistema que sean operativas, aunque obviamente no cuenten con

toda la funcionalidad pretendida para el sistema pero si que constituyan un resultado de valor para el negocio.

Iteración 1:

Esta iteración tiene como objetivo las HU de una prioridad alta y establecer una arquitectura del sistema que pueda ser utilizada durante el resto del proyecto. Al finalizar la misma estarán listos los elementos fundamentales de la extensión como son la conexión con el repositorio RDF, la obtención de resultados después de realizar la consulta SPARQL y la visualización de dichos resultados.

Iteración 2:

En esta iteración se implementarán las HU de una prioridad en negocio media y se corregirán errores o disconformidades del usuario con los componentes implementados en la iteración anterior. Como resultado se obtendrá una versión más completa sobre las funcionalidades de la extensión.

Iteración 3:

El objetivo fundamental de esta iteración es la implementación de las HU de baja prioridad en el negocio y corregir errores de iteraciones anteriores. Al final de esta iteración se obtendrá la versión 1.0 final del producto.

2.3.4 Duración de las Iteraciones

Tabla 11 Plan de duración de iteraciones para equipo de trabajo

Orden de la HU a implementar		**Duración total de la iteración**
Iteración #1	1- Creación de consultas SPARQL 2- Enviar la consulta al repositorio RDF 3- Visualizar resultados de consultas.	3 semanas
Iteración #2	1- Diseño de consultas SPARQL mediante formularios 2- Visualización de un subgrafo RDF en formato HTML.	4 semanas
Iteración #3	1-Salvar consultas 2-Cargar consultas 3- Representación de bases de datos en forma grafo	4 semanas

2.3.5 Plan de entregas

En el plan de entrega que se plantea a continuación se hace una propuesta de la fecha aproximada en que se harán versiones (releases) del sistema al finalizar cada iteración en la fase de implementación. El objetivo de este plan es producir rápidamente versiones de la aplicación que sean operativas, aunque estas no cuenten con toda la funcionalidad pretendida.

Tabla 12 Plan de entregas

Módulo	Final 1 iteración 3ra semana Marzo	Final 2 iteración 4ta semana Abril	Final 3 iteración 4ta semana Mayo
Extensión	0.1	0.2	1.0

2.4 Diseño

A diferencia de las metodologías pesadas, el diseño se realiza durante todo el tiempo de vida del proyecto, siendo constantemente revisado y muy probablemente modificado debido a cambios presentados durante el desarrollo. Entre los elementos más importantes que menciona XP referentes al diseño está la simplicidad, las tarjetas CRC y el *refactoring*.

Simplicidad:

Siempre cuesta menos tiempo de implementar un diseño sencillo que uno complejo. Por lo que, se debe tratar siempre de realizar las cosas de la manera más sencilla posible. Si alguna parte de la implementación resulta especialmente compleja, debe replantearse (divide y vencerás). Así, cualquier cambio y modificación será mucho más sencillo. En ocasiones, realizar un diseño sencillo puede resultar una tarea especialmente difícil. (33)

Tarjetas CRC (Cargo o clase, Responsabilidad y Colaboración):

Para poder diseñar el sistema como un equipo se debe cumplir con tres principios: Cargo o Clase, Responsabilidad y Colaboración (CRC). Las tarjetas CRC permiten desprenderse del método de trabajo basado en procedimientos y trabajar con una metodología basada en objetos. Las tarjetas CRC permiten que el equipo completo contribuya en la tarea del diseño.

Una tarjeta CRC representa un objeto. El nombre de la clase se coloca a modo de título en la tarjeta, las responsabilidades se colocan a la izquierda, y las clases que se implican en cada responsabilidad a la derecha, en la misma línea que su requerimiento correspondiente. (33)

Refactoring:

Cuando se eliminan redundancia, se elimina la funcionalidad inútil, y se rejuvenece antiguos diseños, se está reciclando código. El reciclaje, dentro del ciclo de vida de un proyecto, ahorra tiempo e incrementa la calidad. El reciclaje implicará mantener el código limpio y fácil de comprender, modificar y ampliar. Esto puede resultar un poco costoso al principio, pero resulta fundamental a la hora de realizar diseños futuros. (33)

2.4.1 Tarjetas CRC (Cargo o clase, Responsabilidad y Colaboración)

Las tarjetas determinan el comportamiento de cada actividad. De esta manera nuestra extensión estará conformada por las siguientes clases:

- Query
- GuidedQuery
- Result
- Graph
- Node
- Edge
- Presentation

Tabla 13 Descripción de la tarjeta CRC Query

Clase: Query

Responsabilidades	Clases relacionadas
Obtener la consulta.	Presentation
Transformar la consulta SPARQL en una URL válida.	Presentation
Enviar la consulta al SPARQL endpoints	Presentation
Obtener el resultado de la consulta.	Presentation
Salvar consultas en formato XML	Presentation, Trasformation
Mostrar las consultas guardadas.	Presentation

Visualizar la consulta.	Presentation

Tabla 14 Descripción de la tarjeta CRC GuidedQuery

Clase: GuidedQuery

Responsabilidades	Clases relacionadas
Mostrar las base de datos disponibles para el SPARQL endpoints.	Presentation
Cargar atributos de una base de datos.	Presentation
Insertar relación	Presentation
Eliminar relación	Presentation
Insertar filtro	Presentation
Eliminar filtro	Presentation
Insertar variable	Presentation
Eliminar variable	Presentation
Establecer orden de los resultados	Presentation
Establecer cantidad de resultados	Presentation

Tabla 15 Descripción de la tarjeta CRC Result

Clase: Result

Responsabilidades	Clases relacionadas
Crear la tabla de resultados	Presentation,Query
Cambiar los resultados	Presentation,Query
Cambiar las cabeceras de la tabla de resultados	Presentation,Query

Tabla 16 Descripción de la tarjeta CRC Node

Clase: Node

Responsabilidades	Clases relacionadas
Crear nodo.	Graph

Tabla 17 Descripción de la tarjeta CRC Edge

Clase: Edge

Responsabilidades	Clases relacionadas
Crear arista.	Graph

Tabla 18 Descripción de la tarjeta CRC Graph

Clase: Graph

Responsabilidades	Clases relacionadas
Visualizar grafo	Presentation
Crear grafo	Node, Edge

Tabla 19 Descripción de la tarjeta CRC Presentation

Clase: Presentation

Responsabilidades	Clases relacionadas
Crear la consola de consulta.	
Mostrar los namespace de las base datos biológicas	
Mostrar la tabla de resultados	Result
Mostrar y ocultar la aplicación.	

2.4.2 Decisiones de diseño

“Los patrones de diseño son el esqueleto de las soluciones a problemas comunes en el desarrollo de software” (34). Estos brindan una solución ya probada y documentada a problemas de desarrollo de software que están sujetos a contextos similares.

En el diseño de la extensión que se propone en este trabajo se utilizan algunos de los patrones GRASP (del inglés General Responsibility Assignment Software Patterns) para la asignación de responsabilidades en el diseño de objetos, a continuación se mencionan los de mayor peso.

Bajo Acoplamiento: El sistema debe contener pocas dependencias entre clases (principio básico para la creación un buen diseño). En la extensión las clases solo acceden a las necesarias para obtener la información que requieren, de tal forma que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de clases, potenciando la reutilización, y disminuyendo la dependencia entre las clases (Ver Anexo 1).

Alta Cohesión: Cada elemento del diseño debe realizar una labor única dentro del sistema, no desempeñada por el resto de los elementos y auto-identificable. En la extensión una clase tiene responsabilidades moderadas en un área funcional y colabora con las otras para llevar a cabo las tareas. La clase Graph es la responsable de crear el grafo, de esta forma esta clase almacena información coherente y que esta relacionada con la misma, evitando la realización de trabajos enormes (Ver Anexo 1).

Experto: La responsabilidad de realizar una labor es de la clase que tiene o puede tener los datos involucrados (atributos). Una clase, contiene toda la información necesaria para realizar la labor que tiene encomendada. En la extensión la clase Query cuenta con la información necesaria para cumplir con la responsabilidad de enviar una consulta SPARQL a un SPARQL endpoints (Ver Anexo 1).

Creador: El sistema debe definir un principio general para la asignación de las responsabilidades de creación entre clases. El empleo de este patrón provee al diseño la capacidad de soportar un bajo acoplamiento, mayor claridad, encapsulación y reutilización. En la extensión la clase Presentation agrega un objeto Query por ello, el patrón Creador sugiere que Presentation es idónea para asumir la responsabilidad de crear las instancias Query (Ver Anexo 1).

2.5 Conclusiones parciales del capítulo

- Se definieron 8 HU agrupadas en tres iteraciones para dar a conocer los requerimientos del sistema al equipo de desarrollo y conducir el proceso de creación de los pruebas de aceptación.
- Mediante la aplicación de los patrones de diseño se confeccionaron 6 tarjetas CRC para representar clases en la programación orientada a objetos y definir sus responsabilidades y las colaboraciones con las otras clases.

Capítulo 3. Implementación y prueba

En este capítulo se describen las fases de desarrollo y prueba propias de la metodología de desarrollo XP. Se exponen detalladamente las iteraciones generadas por la planificación, así como las tareas que se plantearon para la realización de cada una de las historias de usuario. En cada iteración se desarrollan las historias de usuario ordenadas por prioridad en el sistema con el objetivo de obtener una primera versión del producto con las principales características o funcionalidades para ser mostrado al cliente. También se detallan las pruebas de aceptación para confirmar que las historias de usuario han sido implementadas correctamente al final de cada iteración, garantizando que los requerimientos han sido cumplidos y que la aplicación es aceptable.

3.1 Desarrollo

El desarrollo es la parte más importante en el proceso de la programación extrema. Prácticamente desde un principio se inicia con la codificación, favoreciendo el logro del objetivo de estar haciendo entregas frecuentemente al cliente. Todos los trabajos tienen como objetivo que se programen lo más rápidamente posible, sin interrupciones y en dirección correcta.

Para tener éxito en la fase de desarrollo XP plantea un conjunto de prácticas que se deben de seguir al pie de la letra. A continuación se detallan las que fueron fundamentales en el desarrollo de la investigación.

Disponibilidad del cliente

Una de las pocas condiciones que impone la metodología XP es tener al usuario siempre disponible. No sólo para ayudar al equipo de desarrollo, sino formando parte de él. Todas las fases que se realizan en un proyecto XP requieren de comunicación con el usuario, preferiblemente cara a cara, en persona, sin intermediarios. Durante la reunión del plan de entregas, el usuario propondrá qué historia de usuario se incluye en cada plan. También se negociarán los plazos de entrega. El usuario o cliente tomará las decisiones que le afecten para alcanzar los objetivos de su negocio.

También es necesario que el cliente colabore en la realización de los test. Estos test comprobarán que el sistema está listo para pasar a la fase de producción. El usuario comprobará los resultados obtenidos y tomará decisiones en cuanto a la utilización o no del sistema realizado.

Estándares de implementación

El código a de ser desarrollado siguiendo los estándares de desarrollo para facilitar su lectura y modificación por cualquier miembro del equipo de desarrollo. Es decisiva, para poder plantear con éxito

la propiedad colectiva del código. Ésta sería impensable sin una codificación basada en estándares que haga que todo el mundo se sienta cómodo con el código escrito por cualquier otro miembro del equipo.

Programación en Parejas

Todo el código que forma parte del plan, será desarrollado por dos personas que trabajarán de forma conjunta en un ordenador. De esta manera, se incrementa la calidad del software desarrollado sin afectar al tiempo de entrega. Se parte de la idea de que este equipo de dos personas posee unos conocimientos similares en cuanto a la tarea que van a realizar, es decir, están aproximadamente al mismo nivel.

Código Colectivo

Esta filosofía permite que cualquiera contribuya al desarrollo de cualquier parte del proyecto. Cualquier programador podrá cambiar una línea de código para añadir funcionalidad o eliminar algún fallo. Cualquiera puede realizar un cambio en el mismo siempre y cuando beneficie a la arquitectura del mismo. La responsabilidad del funcionamiento recaerá sobre el equipo completo. (33)

3.1.1 Tareas generadas por cada historia de usuario

Todo el trabajo de cada iteración es expresado en tareas de programación. Cada una de las historias de usuario se transformará en tareas que serán desarrolladas por programadores, dentro del equipo de desarrollo, aplicando la práctica de la programación en parejas. Cada tarea de desarrollo corresponderá a un período de uno a tres días de desarrollo. Las ocho HU fueron trasformadas en veintisiete tareas, en la tabla 20 se muestra la descripción de una de ellas.

Tabla 20 Tarea #7 Mostrar los resultados en forma de tabla

Tarea	
Número de tarea:7	**Número de historia:** 3
Nombre de la tarea: Mostrar los resultados en forma de tabla.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 2
Fecha inicio: 16/03/2011	**Fecha fin:** 19/03/2011
Programador responsable: Gendrys Espinosa Vega	
Descripción: Se mostrarán los resultados en forma de tabla, especificando en la cabecera de la tabla	

las variables introducidas por el usuario.

Las tareas que se plantearon para la realización de cada una de las historia de usuarios se encuentran en la Plantilla Tarea de Ingeniería del expediente de proyecto.

3.2 Estándar de código

Formato de Líneas

Se utilizaron líneas en blanco para la separación entre bloques de código conceptualmente diferentes. Lo cantidad de caracteres por línea no excede de 80 caracteres. En caso de exceder de este tamaño se dividió la línea en varias partes, siempre luego de cerrar un paréntesis o declarar un operador lógico.

Sentencia por línea de código

Cada sentencia ocupa un solo renglón o línea:

```
variable=this.sparqlVar[i];
asignada=false;
```

Figura 4 Sentencia por línea de código

Identación

Se utilizaron de 4 a 8 espacios para la estructura del código.

```
indexVar:function(varName){
    index=-1;
    for (i = 0; i < this.sparqlVar.length; i++) {
        if(varName==this.sparqlVar[i])
        {
            index=i;
            break;
        }
    }
    return index;
},
```

Figura 5 Espacios para la estructura del código

Sentencias de Control

Se colocó la apertura del bloque ({) al final de la línea inicial de la sentencia. El fin de bloque (}) se colocó en una línea aparte y alienado con el principio de la sentencia. Cada nuevo bloque de sentencias se insertó entre llaves aunque conste de una sola sentencia.

```
returnSparqlVar:function(){
    var sparqlResultVar= new Array();
    rowVar=document.getElementById('variables').getElementsByTagName('row');

    for (i = 1; i < rowVar.length; i++) {
  return sparqlResultVar;
}
```

Figura 6 Sentencias de Control

Ficheros

Se incluyó una sola clase o interfaz por fichero. Si hay comentarios globales para los contenidos del fichero se colocaron en primer lugar.

Nombres de los identificadores

El estilo de código para la declaración de las variables es el dromedaryCase. Los nombres de los objetos que se crean deben ser claros y sugerentes en la programación, reflejando claramente su significado en el negocio.

- No se utilizaron abreviaturas, a menos que sea una abreviatura clara y ampliamente usada en el área de aplicación del software.
- No se utilizaron nombres excesivamente largos
- Se evitó la redundancia.
- Se usó el idioma inglés para la declaración de variables y funciones.

Variables utilizadas dentro de ciclos

Se utilizó la variable ***i*** dentro de un ciclo ***for.*** En caso de ser un ciclo anidado se utilizó ***j*** y ***k*** en el orden correspondiente y de acuerdo a las necesidades.

Uso de comentarios

La inclusión de comentarios dentro de los códigos facilita la comprensión del significado y el objetivo de las funciones, secuencias de control, sentencias en general. Se comentó el código que sea fundamental en el programa. Se utilizaron las siguientes categorías para escribir los comentarios:

- Comentarios en los archivos: la explicación del contenido del archivo, resumiendo lo que se "encierra" dentro del mismo.
- Comentarios en el código: de tipo aclaratorios, solo cuando algo necesita o es digno de una explicación o justificación.
- Se utilizó // para comentarios de una línea y /*…*/ para más de una línea.

```
//comentario de una solo línea.
/*
este comentario es
para varias líneas.
*/
```

Figura 7 Uso de comentarios

3.3 Código fuente

JavaScript no permite la creación de clases del mismo tipo que otros lenguajes como Java o C++. De hecho, ni siquiera tiene definida la palabra reservada class para crear clases. No obstante, JavaScript si que tiene reservada la palabra class para su uso futuro, en el que también está prevista la inclusión del concepto tradicional de clase. A pesar de estas limitaciones, es posible crear en JavaScript algunos elementos parecidos a las clases, lo que se suele denominar pseudoclase. Los conceptos que se utilizan para simular las clases son las funciones constructoras y el prototype de los objetos.

Para la definición de objetos se utilizó la notación de objetos mediante JSON, esta una de las características principales de JavaScript y un mecanismo definido en los fundamentos básicos del lenguaje. JSON permite definir arrays y objetos de una manera concisa, lo que supone una gran ventaja respecto de la notación tradicional de los objetos y los arrays.

Entre las funciones desarrolladas está la función **selectDatabase()** , esta es la encarga de cargar todos los atributos de la base de datos seleccionada , primeramente se eliminan todos los campos existentes de la base de datos anterior y después se muestran los de la base de datos seleccionada.

Para ello se realiza la lectura de un documento XML que contiene la información correspondiente a cada base de datos.

```
selectDatabase:function(menuListBd){

    var element = menuListBd.parentNode.getElementsByTagName("menulist")[2].getElementsByTagName("me

    while (element.firstChild) {
        element.removeChild(element.firstChild);
    }
    function documentLoaded (e) {
        var labels = xmlDoc.getElementsByTagName(menuListBd.label);
        for (i=0; i < labels.length; i++)
        {
            for(j=0; j < labels[i].getElementsByTagName('uri').length; j++) {

                menuitem = document.createElement('menuitem');
                str=labels[i].getElementsByTagName('uri')[j].childNodes[0].nodeValue;
                atributo=str.split('/');
                menuitem.setAttribute("label",labels[i].getAttribute("name")+":"+atributo[atributo.l
                element.appendChild(menuitem);
            }
        }

    }

    xmlDoc.addEventListener("load",documentLoaded, false);
    xmlDoc.load('chrome://AlasRDF/content/pubmed.xml');
}
```

Figura 8 Ejemplo de la funcionalidad seleccionar base de datos

3.4 Pruebas

Uno de los pilares de la Extreme Programming es el proceso de pruebas. XP anima a probar constantemente tanto como sea posible. Esto permite aumentar la calidad de los sistemas reduciendo el número de errores no detectados y disminuyendo el tiempo transcurrido entre la aparición de un error y su detección. También permite aumentar la seguridad de evitar efectos colaterales no deseados a la hora de realizar modificaciones y refactorizaciones.

XP divide las pruebas del sistema en dos grupos: pruebas unitarias, encargadas de verificar el código y diseñada por los programadores, y pruebas de aceptación o pruebas funcionales destinadas a evaluar si al final de una iteración se consiguió la funcionalidad requerida diseñadas por el cliente final.

Las pruebas del sistema tienen como objetivo verificar la funcionalidad del sistema a través de sus interfaces externas comprobando que dicha funcionalidad sea la esperada en función de los requisitos

del sistema. Generalmente las pruebas del sistema son desarrolladas por los programadores para verificar que su sistema se comporta de la manera esperada, por lo que podrían encajar dentro de la definición de pruebas unitarias que propone XP.

Sin embargo, las pruebas del sistema tienen como objetivo verificar que el sistema cumple los requisitos establecidos por el usuario por lo que también pueden encajar dentro de la categoría de pruebas de aceptación.

Las pruebas de aceptación son más importantes que las pruebas unitarias dado que significan la satisfacción del cliente con el producto desarrollado y el final de una iteración y el comienzo de la siguiente por esto, el cliente es la persona adecuada para diseñar las pruebas de aceptación.

Las pruebas de aceptación

El objetivo de estas pruebas es verificar los requisitos, por este motivo, los propios requisitos del sistema son la principal fuente de información a la hora de construir las pruebas de aceptación.

Las pruebas de aceptación son creadas a partir de las historias de usuario. Durante una iteración la historia de usuario seleccionada en la planificación de iteraciones se convertirá en una prueba de aceptación. El cliente o usuario especifica los aspectos a testear cuando una historia de usuario ha sido correctamente implementada.

Una historia de usuario puede tener más de una prueba de aceptación, tantas como sean necesarias para garantizar su correcto funcionamiento y no se considera completa hasta que no supera sus pruebas de aceptación. Esto significa que debe desarrollarse un nuevo test de aceptación para cada iteración o se considerará que el equipo de desarrollo no realiza ningún progreso.

Una prueba de aceptación es como una prueba de caja negra. Cada una de ellas representa una salida esperada del sistema. Es responsabilidad del cliente verificar la corrección de las pruebas de aceptación y tomar decisiones a cerca de las mismas.

La garantía de calidad es una parte esencial en el proceso de XP. La realización de este tipo de pruebas y la publicación de los resultados debe ser los más rápido posibles, para que los desarrolladores puedan realizar con la mayor rapidez los cambios que sean necesarios. (33)

Durante el proceso de pruebas el cliente diseñó 14 casos de pruebas agrupados por historias de usuarios, los encargados de pruebas o *tester*, dentro del equipo de trabajo, ayudaron al cliente a escribir dichas pruebas. A continuación se describen las pruebas de aceptación realizadas a la extensión:

Tabla 21 Caso de Prueba de Aceptación insertar namespace en la consola de consulta

Caso de Prueba de Aceptación	
Código Caso de Prueba: AR11	**Nombre Historia de Usuario:** Creación de consultas SPARQL.
Nombre: Insertar namespace en la consola de consulta.	
Descripción de la Prueba: Prueba la funcionalidad de mostrar en un árbol los namespace almacenados que contienen información de las base datos biológicas, estos namespace serán utilizados posteriormente en la creación de consultas avanzadas.	
Condiciones de Ejecución: El fichero que contendrán los namespace debe encontrarse en la carpeta content dentro de la extensión con el nombre namesapce.xml.	
Entrada / Pasos de ejecución: El sistema muestra automáticamente los namespace al iniciar al navegador.	
Resultado Esperado: Los namespace son mostrados sin error.	
Evaluación de la Prueba: Satisfactoria	

Tabla 22 Caso de Prueba de Aceptación enviar la consulta al repositorio RDF

Caso de Prueba de Aceptación	
Código Caso de Prueba: AR21	**Nombre Historia de Usuario:** Ejecutar consulta.
Nombre: Enviar la consulta al repositorio RDF	
Descripción de la Prueba: Prueba la funcionalidad de ejecutar la consulta previamente creada por el usuario, la consulta se enviará al servicio SPARQL endpoints del proyecto Linkend Life Data(LLD).	
Condiciones de Ejecución: La consulta debe ser previamente realizada a través de formularios o de la forma tradicional.	
Entrada / Pasos de ejecución: Crear la consulta adecuadamente	
Resultado Esperado: La consulta es enviada sin error.	
Evaluación de la Prueba: Satisfactoria	

Tabla 23 Caso de Prueba de Aceptación visualizar resultados de consultas

Caso de Prueba de Aceptación	
Código Caso de Prueba: AR31	**Nombre Historia de Usuario:** Visualizar resultados de consultas
Nombre: Visualizar resultados de consultas.	
Descripción de la Prueba: Prueba la funcionalidad de visualizar resultados de consultas obtenidos del servicio SPARQL endpoints referentes a una consulta específica.	
Condiciones de Ejecución: La consulta debe estar elaborada previamente. La consulta elaborada debe devolver resultados.	
Entrada / Pasos de ejecución: Tomar los resultados de la consulta	
Resultado Esperado: Los resultados son mostrados sin error	
Evaluación de la Prueba: Satisfactoria	

Tabla 24 Caso de Prueba de Aceptación cargar atributos de una base de datos

Caso de Prueba de Aceptación	
Código Caso de Prueba: AR41	**Nombre Historia de Usuario:** Diseño de consultas SPARQL mediante formularios
Nombre: Cargar atributos de una base de datos.	
Descripción de la Prueba: Prueba la funcionalidad de cargar atributos de una base de datos, en la creación de consultas mediante formularios.	
Condiciones de Ejecución: Se deben tener almacenados previamente los atributos de la base de datos seleccionada.	
Entrada / Pasos de ejecución: Seleccionar una base de datos	
Resultado Esperado: Los atributos de la base de datos seleccionada son mostrados sin error.	
Evaluación de la Prueba: Satisfactoria	

Tabla 25 Caso de Prueba de Aceptación insertar relación

Caso de Prueba de Aceptación	
Código Caso de Prueba: AR42	**Nombre Historia de Usuario:** Diseño de consultas SPARQL mediante formularios
Nombre: Insertar relación.	
Descripción de la Prueba: Prueba la funcionalidad de insertar relaciones entre distintos elementos de las bases de datos en las consultas realizadas a través de formulario.	
Condiciones de Ejecución: Se deben seleccionar la opción tipo de búsqueda avanzada.	
Entrada / Pasos de ejecución: Se selecciona la opción de insertar relación	
Resultado Esperado: La relación es insertada sin error.	
Evaluación de la Prueba: Satisfactoria	

Tabla 26 Caso de Prueba de Aceptación eliminar relación

Caso de Prueba de Aceptación	
Código Caso de Prueba: AR43	**Nombre Historia de Usuario:** Diseño de consultas SPARQL mediante formularios
Nombre: Eliminar relación.	
Descripción de la Prueba: Prueba la funcionalidad de eliminar relaciones entre distintos elementos de las bases de datos en las consultas realizadas a través de formulario.	
Condiciones de Ejecución: Deben existir más de una relación	
Entrada / Pasos de ejecución:	
Resultado Esperado: Se elimina una relación sin error.	
Evaluación de la Prueba: Satisfactoria	

Tabla 27 Caso de Prueba de Aceptación insertar filtro

Caso de Prueba de Aceptación	
Código Caso de Prueba: AR44	**Nombre Historia de Usuario:** Diseño de consultas SPARQL

	mediante formularios
Nombre: Insertar filtro.	
Descripción de la Prueba: Prueba la funcionalidad de insertar un filtro a una consulta, con el objetivo de obtener solo determinados resultados.	
Condiciones de Ejecución: Deben existir en la consulta los valores a filtrar	
Entrada / Pasos de ejecución: Creación de la consulta con posibles resultados	
Resultado Esperado: Se inserta un filtro sin error.	
Evaluación de la Prueba: Satisfactoria	

Tabla 28 Caso de Prueba de Aceptación eliminar filtro

Caso de Prueba de Aceptación	
Código Caso de Prueba: AR45	**Nombre Historia de Usuario:** Diseño de consultas SPARQL mediante formularios
Nombre: Eliminar filtro	
Descripción de la Prueba: Prueba la funcionalidad eliminar un filtro de la consulta SPARQL.	
Condiciones de Ejecución: El filtro debe estar creado previamente.	
Entrada / Pasos de ejecución: Se selecciona el botón para generar el evento encargado de eliminar el filtro.	
Resultado Esperado: Se elimina el filtro sin error.	
Evaluación de la Prueba: Satisfactoria	

Tabla 29 Caso de Prueba de Aceptación insertar variable

Caso de Prueba de Aceptación	
Código Caso de Prueba: AR46	**Nombre Historia de Usuario:** Diseño de consultas SPARQL mediante formularios.
Nombre: Insertar variable	

Descripción de la Prueba: Prueba la funcionalidad de insertar variable para almacenar los datos o recursos obtenidos de las consultas.
Condiciones de Ejecución: Debe de estar seleccionada la opción de consultas avanzadas.
Entrada / Pasos de ejecución: Se selecciona la opción de insertar variable. Se inserta el nombre de la variable.
Resultado Esperado: La variable es creada sin error.
Evaluación de la Prueba: Satisfactoria

Tabla 30 Caso de Prueba de Aceptación eliminar variable

Caso de Prueba de Aceptación	
Código Caso de Prueba: AR47	**Nombre Historia de Usuario:** Diseño de consultas SPARQL mediante formularios.
Nombre: Eliminar variable	
Descripción de la Prueba: Prueba la funcionalidad de eliminar variable.	
Condiciones de Ejecución: Debe de existir más de dos variables como mínimo. La variable debe estar creada	
Entrada / Pasos de ejecución: Se selecciona el botón para generar el evento encargado de eliminar la variable.	
Resultado Esperado: La variable es eliminada sin error.	
Evaluación de la Prueba: Satisfactoria	

Tabla 31 Caso de Prueba de Aceptación establecer orden de los resultados

Caso de Prueba de Aceptación	
Código Caso de Prueba: AR48	**Nombre Historia de Usuario:** Diseño de consultas SPARQL mediante formularios.

Nombre: Establecer orden de los resultados
Descripción de la Prueba: Prueba la funcionalidad establecer orden de los resultados mediante la cual el usuario puede obtener los resultados ordenados ascendente y descendentemente por alguna variable.
Condiciones de Ejecución: La opción ordenar debe estar habilitada
Entrada / Pasos de ejecución: Se selecciona el tipo de ordenamiento que puede ser ascendente o descendente. Se selecciona la variable por la cual se quiere ordenar.
Resultado Esperado: Se establece orden de los resultados sin error.
Evaluación de la Prueba: Satisfactoria

Tabla 32 Caso de Prueba de Aceptación establecer cantidad de resultados

Caso de Prueba de Aceptación	
Código Caso de Prueba: AR49	**Nombre Historia de Usuario:** Diseño de consultas SPARQL mediante formularios.
Nombre: Establecer cantidad de resultados	
Descripción de la Prueba: Prueba la funcionalidad establecer cantidad de resultados mediante la cual el usuario puede obtener la cantidad de resultados que desee.	
Condiciones de Ejecución:	
Entrada / Pasos de ejecución: Se introduce en el campo cantidad de resultados el valor deseado	
Resultado Esperado: Se establece la cantidad de resultados sin error.	
Evaluación de la Prueba: Satisfactoria	

Tabla 33 Caso de Prueba de Aceptación salvar consultas en formato XML

Caso de Prueba de Aceptación	
Código Caso de Prueba: AR61	**Nombre Historia de Usuario:** Salvar consulta
Nombre: Salvar consultas en formato XML	
Descripción de la Prueba: Prueba la funcionalidad salvar consultas en formato XML para guardar las consultas SPARQL realizadas.	
Condiciones de Ejecución: La consulta SPARQL debe estar creada	
Entrada / Pasos de ejecución: Se acciona sobre el elemento guardar consulta. Se introduce el nombre de la consulta. Seleccionamos el directorio deseado.	
Resultado Esperado: Se guarda una consulta SPARQL en formato XML sin error.	
Evaluación de la Prueba: Satisfactoria	

Tabla 34 Caso de Prueba de Aceptación visualizar la consulta

Caso de Prueba de Aceptación	
Código Caso de Prueba: AR71	**Nombre Historia de Usuario:** Cargar consulta
Nombre: Visualizar la consulta	
Descripción de la Prueba: Prueba la funcionalidad visualizar la consulta guardada en formato XML.	
Condiciones de Ejecución: La consulta SPARQL guardada en formato XML debe estar creada	
Entrada / Pasos de ejecución: Se acciona sobre el elemento cargar consulta. Se selecciona la consulta que se desea cargar.	
Resultado Esperado: Se carga una consulta SPARQL en formato XML sin error.	
Evaluación de la Prueba: Satisfactoria	

3.5 Pantallas principales de la aplicación

Área para la realización de consultas SPARQL de forma visual

La aplicación cuenta con una interfaz para la realización de consultas SPARQL de forma visual. En la parte superior de la misma se encuentra una barra de menú que brinda las opciones de realizar una nueva consulta, cargar consulta , salvar consulta , ejecutar y abortar la consulta, insertar relaciones, filtros y variables. En la parte izquierda se muestra el tipo de búsqueda a realizar y las relaciones que han sido insertadas y configuradas. En la parte derecha se visualizan las variables creadas, los filtros y las opciones para el control de resultados. En la figura 13 se observa una consulta SPARQL realizada a través de formularios (Ver Anexo 2).

Área de visualización de los resultados de la consulta SPARQL

La aplicación consta con una interfaz para visualizar los resultados obtenidos de las consultas SPARQL diseñadas. Estos resultados se muestran en una tabla, la misma está integrada por diferentes columnas que corresponden a las variables utilizadas en las columnas y las filas son los resultados que corresponden a cada variable. Los resultados pueden ser literales o URL, estas últimas pueden ser visualizadas en una pestaña del navegador a través de un clic. En la figura 14 se muestran los resultados de una consulta (Ver Anexo 2).

3.6 Conclusiones parciales del capítulo

- Se describieron las 27 tareas trazadas para la realización del proyecto y se implementó la aplicación haciendo uso de los lenguajes JavaScript y XUL, y de NetBeans 6.9 como IDE.
- Se estableció el estilo de código a seguir, para facilitar su lectura y modificación por cualquier integrante del equipo de trabajo.
- Se realizaron de pruebas de aceptación a la aplicación a través del diseño de 14 casos de prueba, arrojaron nueve no conformidades no significativas que fueron solucionadas por el equipo de trabajo en un corto plazo de tiempo.

Conclusiones

- Semantizar un navegador fue la técnica escogida para el desarrollo de la extensión, seleccionando para ello el navegador Web Mozilla Firefox.
- Haciendo uso de XP como metodología de desarrollo, se definieron 8 HU agrupadas en tres iteraciones y se confeccionaron las tarjetas CRC (Clases, Responsabilidad, Colaboración).
- Se realizó la implementación de la extensión en su versión 1.0 para semantizar el navegador Web Mozilla Firefox orientada a facilitar la utilización de los Datos Enlazados por los biólogos.
- Se realizaron pruebas de aceptación a la extensión a partir de las historias de usuario con el objetivo de verificar los requisitos, arrojando resultados satisfactorios.

Recomendaciones

Por las experiencias alcanzadas durante la realización de este trabajo y con vista a enriquecer la solución, los autores sugieren:

- Implementar una funcionalidad que permita exportar los resultados obtenidos de una consulta SPARQL en diferentes formatos.
- Incorporar nuevos elementos que faciliten la realización de las consultas SPARQL de forma visual.
- Realizar un estudio detallado de los atributos de las bases de datos y las posibles relaciones de algunos de los atributos con las diferentes base de datos.

Referencias Bibliográficas

1. **Valencia Castillo, Edwin.** Recuperación y organización de la información a través de RDF usando SPARQL. [En línea] http://ggomez.files.wordpress.com/2008/09/informe-sparql.doc.

2. W3C Oficina Española. *Guía Breve de Web Semántica.* [En línea] http://www.w3c.es/divulgacion/guiasbreves/websemantica.

3. **Gómez Pérez, Asunción.** *Apúntate al reto de los Datos Enlazados en la Web.* [En línea] http://www.madrimasd.org/informacionIdi/analisis/analisis/analisis.asp?id=44078.

4. *The Semantic Web.* **Berners-Lee, Tim, Hendler, James y Lassila, Ora.** 2001, Scientific American.

5. **Berners-Lee, Tim.** Semantic Web Road map. [En línea] Septiembre de 1998. [Citado el: 18 de noviembre de 2010.] http://www.w3.org/DesignIssues/Semantic.html.

6. **Pérez Valdés, Damián.** Web Semántica y sus principales características. *Maestros del Web.* [En línea] [Citado el: 20 de Octubre de 2010.] http://www.maestrosdelweb.com/editorial/web-semantica-y-sus-principales-caracteristicas/.

7. **Lassila, Ora y Swick, Ralph R.** Resource Description Framework(RDF) Especificación del Modelo y la Sitaxis. *W3C.* [En línea] 22 de febrero de 1999. [Citado el: 3 de noviembre de 2010.] http://www.w3.org/TR/1999/REC-rdf-syntax-19990222.

8. **Méndez Rodríguez, Eva.** RDF : Un modelo de metadatos flexible para las bibliotecas digitales del próximo milenio. [En línea] 5 de Noviembre de 1999. [Citado el: 23 de Octubre de 2010.] http://eprints.rclis.org/15492/.

9. RDF y RDF Schema. [En línea] 21 de Noviembre de 2010. http://www.matem.unam.mx/~grecia/semantic_web/rdf.html.

10. **Berners-Lee, Tim.** [En línea] 27 de junio de 2006. [Citado el: 17 de noviembre de 2010.] http://www.w3.org/DesignIssues/LinkedData.html.

11. **Bizer, Chris, Cyganiak, Richard y Heath, Tom.** How to Publish Linked Data on the Web. [En línea] 2007. [Citado el: 22 de noviembre de 2010.]

12. How to use the Tabulator. [En línea] http://dig.csail.mit.edu/2005/ajar/ajaw/Help.html.

13. SIMILE Project. *Semantic Interoperability of Metadata and Information in unLike Environments.* [En línea] http://simile.mit.edu/wiki/Piggy_Bank.

14. Disco - Hyperdata Browser . *Disco - Hyperdata Browser.* [En línea] http://www4.wiwiss.fu-berlin.de/bizer/ng4j/disco/.

15. RKBExplorer. [En línea] http://semanticweb.org/wiki/RKBExplorer.

16. SIMILE Project. *Semantic Interoperability of Metadata and Information in unLike Environments.* [En línea] http://simile.mit.edu/wiki/Longwell_User_Guide.

17. *Haystack project home page.* [En línea] [Citado el: 20 de 10 de 2010.] http://haystack.lcs.mit.edu/..

18. Rhizomik. [En línea] http://rhizomik.net/html/rhizomer/.

19. The OpenLink Data Explorer Extension. *OpenLink Data Explorer Extension.* [En línea] http://ode.openlinksw.com.

20. Mozilla. *Complementos para Firefox.* [En línea] https://addons.mozilla.org/es-ES/firefox/tag/linked%20data.

21. Mozilla. *Complementos para Firefox.* [En línea] https://addons.mozilla.org/es-ES/firefox/addon/3886/.

22. Página profesor Rafael Barzanallana. *Universidad de Murcia.* [En línea] [Citado el: 19 de Noviembre de 2010.] http://www.um.es/docencia/barzana/IAGP/Iagp2.html.

23. **Carrillo Pérez, Isaías , Pérez González, Rodrigo y Rodríguez Martín, Aureliano David .** Metodologías de desarrollo. [En línea] 18 de Noviembre de 2010. [Citado el: 15 de Octubre de 2008.] http://www.solusoftg11.googlecode.com/files/Metodologias%20de%20desarrollo.pdf.

24. **Letelier, Patricio y Penadés, Carmen.** Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP). *Cyta.* [En línea] junio de 2006. http://www.cyta.com.ar.

25. Capitulo-I-HERRAMIENTAS-CASE. [En línea] [Citado el: 18 de noviembre de 2010.] http://www.scribd.com/doc/3062020/Capitulo-I-HERRAMIENTAS-CASE.

26. **Eguíluz Pérez, Javier.** Introducción a JavaScript. *Libros Web.* [En línea] [Citado el: 20 de noviembre de 2010.] url: http://www.librosweb.es/javascript/.

27. Introducción a AJAX. *Libros Web.* [En línea] [Citado el: 24 de Febrero de 2011.] http://www.librosweb.es/ajax/index.html.

28. XUL Tutorial. *MDN.* [En línea] [Citado el: 20 de noviembre de 2010.] https://developer.mozilla.org/en/XUL_Tutorial.

29. jQuery. *Ecured.* [En línea] [Citado el: 15 de Febrero de 2011.] http://www.ecured.cu/index.php/JQuery.

30. Entornos de Desarrollo Integrado. [En línea] [Citado el: 21 de noviembre de 2010.] http://petra.euitio.uniovi.es%2F~i1667065%2FHD%2Fdocumentos%2FEntornos%2520de%2520Desarrollo%2520Integrado.pdf.

31. Información del lanzamiento del IDE NetBeans 6.9.1. *Netbeans.* [En línea] [Citado el: 30 de Noviembre de 2010.] http://netbeans.org/community/releases/69/index_es.html.

32. SPARQL endpoint. *Semantic web.* [En línea] [Citado el: 20 de Noviembre de 2010.] http://semanticweb.org/wiki/SPARQL_endpoint.

33. **Gerardo, Fernández Escribano.** Introducción a Extreme Programing. 2002.

34. **Tedeschi, Nicolás.** ¿Qué es un Patrón de Diseño? MSDN. [En línea] [Citado el: 25 de Febrero de 2011.] http://msdn.microsoft.com/es-es/library/bb972240.aspx.

Bibliografía

- **Berners-Lee, Tim.** [En línea] 27 de junio de 2006. [Citado el: 17 de noviembre de 2010.] http://www.w3.org/DesignIssues/LinkedData.html.
- **Berners-Lee, Tim.** Semantic Web Road map. [En línea] Septiembre de 1998. [Citado el: 18 de noviembre de 2010.] http://www.w3.org/DesignIssues/Semantic.html.
- **Bizer, Chris, Cyganiak, Richard y Heath, Tom.** How to Publish Linked Data on the Web. [En línea] 2007. [Citado el: 22 de noviembre de 2010.]
- Capitulo-I-HERRAMIENTAS-CASE. [En línea] [Citado el: 18 de noviembre de 2010.] http://www.scribd.com/doc/3062020/Capitulo-I-HERRAMIENTAS-CASE.
- **Carrillo Pérez, Isaías , Pérez González, Rodrigo y Rodríguez Martín, Aureliano David .** Metodologías de desarrollo. [En línea] 18 de Noviembre de 2010. [Citado el: 15 de Octubre de 2008.] http://www.solusoftg11.googlecode.com/files/Metodologias%20de%20desarrollo.pdf.
- Disco - Hyperdata Browser . *Disco - Hyperdata Browser.* [En línea] http://www4.wiwiss.fu-berlin.de/bizer/ng4j/disco/.
- **Eguíluz Pérez, Javier.** Introducción a JavaScript. *Libros Web.* [En línea] [Citado el: 20 de noviembre de 2010.] url: http://www.librosweb.es/javascript/.
- Entornos de Desarrollo Integrado. [En línea] [Citado el: 21 de noviembre de 2010.] http://petra.euitio.uniovi.es%2F~i1667065%2FHD%2Fdocumentos%2FEntornos%2520de%2520Desarrollo%2520Integrado.pdf.
- **Gerardo, Fernández Escribano.** *Introducción a Extreme Programing.* 2002.
- **Gómez Pérez, Asunción.** *Apúntate al reto de los Datos Enlazados en la Web.* [En línea] http://www.madrimasd.org/informacionIdi/analisis/analisis/analisis.asp?id=44078.
- *Haystack project home page.* [En línea] [Citado el: 20 de 10 de 2010.] http://haystack.lcs.mit.edu/.
- Herramientas Case. [En línea] [Citado el: 24 de Noviembre de 2010.] http://www1.inei.gob.pe/biblioineipub/bancopub/Inf/Lib5103/Libro.pdf.
- How to use the Tabulator. [En línea] http://dig.csail.mit.edu/2005/ajar/ajaw/Help.html.
- Información del lanzamiento del IDE NetBeans 6.9.1. *Netbeans.* [En línea] [Citado el: 30 de Noviembre de 2010.] http://netbeans.org/community/releases/69/index_es.html.

- Introducción a AJAX. *Libros Web.* [En línea] [Citado el: 24 de Febrero de 2011.] http://www.librosweb.es/ajax/index.html.
- jQuery. *Ecured.* [En línea] [Citado el: 15 de Febrero de 2011.] http://www.ecured.cu/index.php/JQuery.
- **Lassila, Ora y Swick, Ralph R.** Resource Description Framework(RDF) Especificación del Modelo y la Sitaxis. *W3C.* [En línea] 22 de febrero de 1999. [Citado el: 3 de noviembre de 2010.] http://www.w3.org/TR/1999/REC-rdf-syntax-19990222.
- **Letelier, Patricio y Penadés, Carmen.** Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP). *Cyta.* [En línea] junio de 2006. http://www.cyta.com.ar.
- **Lic. Julia, Ramírez Céspedes.** Web semántica: una alternativa para poner orden al caos. [En línea] 2004.
- **Mariano, Rico Almodóvar.** Una aproximación para la simplificación del desarrollo de aplicaciones web semánticas y el uso de datos semánticos. [En línea] Mayo de 2009. [Citado el: 15 de Octubre de 2010.]
- **Méndez Rodríguez, Eva.** RDF : Un modelo de metadatos flexible para las bibliotecas digitales del próximo milenio. [En línea] 5 de Noviembre de 1999. [Citado el: 23 de Octubre de 2010.] http://eprints.rclis.org/15492/.
- Mozilla. *Complementos para Firefox.* [En línea] https://addons.mozilla.org/es-ES/firefox/addon/3886/.
- Mozilla. *Complementos para Firefox.* [En línea] https://addons.mozilla.org/es-ES/firefox/tag/linked%20data.
- **Muzás, Antonio López.** Sistema modular de presentación de información para la plataforma Rhizomer,. [En línea] julio de 2009.
- **Pablo, Castells.** La web semántica. [En línea] [Citado el: 12 de 10 de 2010.]
- *Patricio Orlando Letelier Torres, Mª Carmen Penadés.* **Programming, Métodologías ágiles para el desarrollo de software: eXtreme.** Técnica administrativa, ISSN 1666-1680, Vol. 5, Nº. 26, 2006.

- **Pérez Valdés, Damián.** Web Semántica y sus principales características. *Maestros del Web.* [En línea] [Citado el: 20 de Octubre de 2010.] http://www.maestrosdelweb.com/editorial/web-semantica-y-sus-principales-caracteristicas/.
- **Perojo, Lic. Keilyn Rodríguez.** Web Semántica: un nuevo enfoque hacia la Organización de Información en los Sistemas de Gestión de Contenidos. [En línea]
- RDF y RDF Schema. [En línea] 21 de Noviembre de 2010. http://www.matem.unam.mx/~grecia/semantic_web/rdf.html.
- Rhizomik. [En línea] http://rhizomik.net/html/rhizomer/.
- RKBExplorer. [En línea] http://semanticweb.org/wiki/RKBExplorer.
- **Senso, Jose A.** Navegadores semánticos o semantizar navegadores. [En línea] http://dialnet.unirioja.es/servlet/fichero_articulo?codigo=3190854&orden=0.
- SIMILE Project. *Semantic Interoperability of Metadata and Information in unLike Environments.* [En línea] http://simile.mit.edu/wiki/Longwell_User_Guide.
- SIMILE Project. *Semantic Interoperability of Metadata and Information in unLike Environments.* [En línea] http://simile.mit.edu/wiki/Piggy_Bank.
- SPARQL endpoint. *Semantic web.* [En línea] [Citado el: 20 de Noviembre de 2010.] http://semanticweb.org/wiki/SPARQL_endpoint.
- **Tedeschi, Nicolás.** ¿Qué es un Patrón de Diseño? MSDN. [En línea] [Citado el: 25 de Febrero de 2011.] http://msdn.microsoft.com/es-es/library/bb972240.aspx.
- The OpenLink Data Explorer Extension. *OpenLink Data Explorer Extension.* [En línea] http://ode.openlinksw.com.
- *The Semantic Web.* **Berners-Lee, Tim, Hendler, James y Lassila, Ora.** 2001, Scientific American.
- **Toledo, Lic. Wilberto Ramos.** La Web Semántica en el contexto de las comunidades científicas cubanas. Caso red de la ciencia cubana. [En línea]
- **Valencia Castillo, Edwin.** Recuperación y organización de la información a través de RDF usando SPARQL. [En línea] http://ggomez.files.wordpress.com/2008/09/informe-sparql.doc.

- W3C Oficina Española. *Guía Breve de Linked Data.* [En línea] http://www.w3c.es/divulgacion/guiasbreves/LinkedData.
- W3C Oficina Española. *Guía Breve de Web Semántica.* [En línea] http://www.w3c.es/divulgacion/guiasbreves/websemantica.
- W3C. [En línea] http://www.w3.org/2005/04/swls/BioDash/Demo/What%20is%20Haystack.html.
- XUL Tutorial. *MDN.* [En línea] [Citado el: 20 de noviembre de 2010.] https://developer.mozilla.org/.

Anexos

Anexo 1: Patrones GRASP

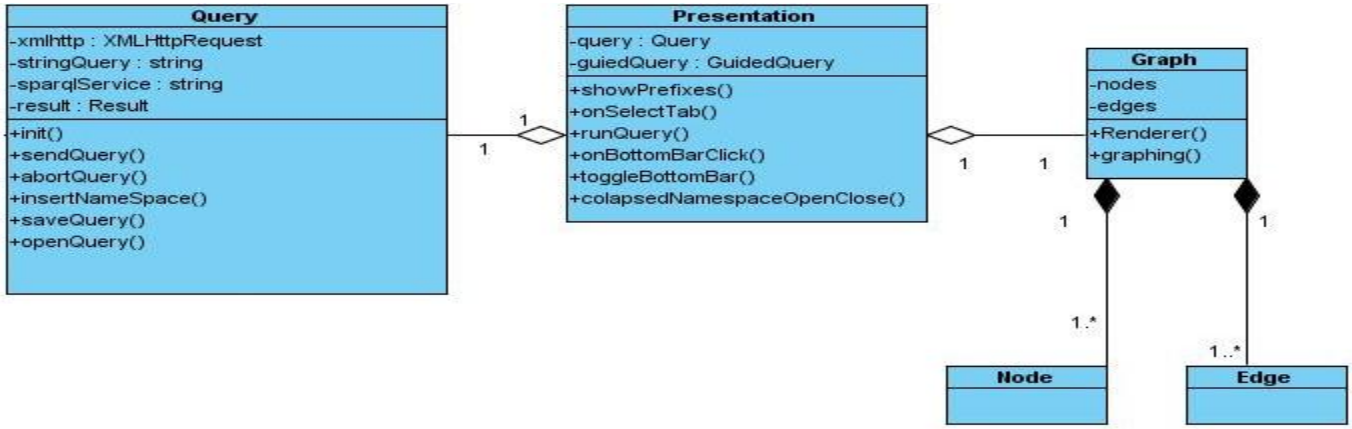


Figura 9 Ejemplo de la aplicación del patrón alta cohesión

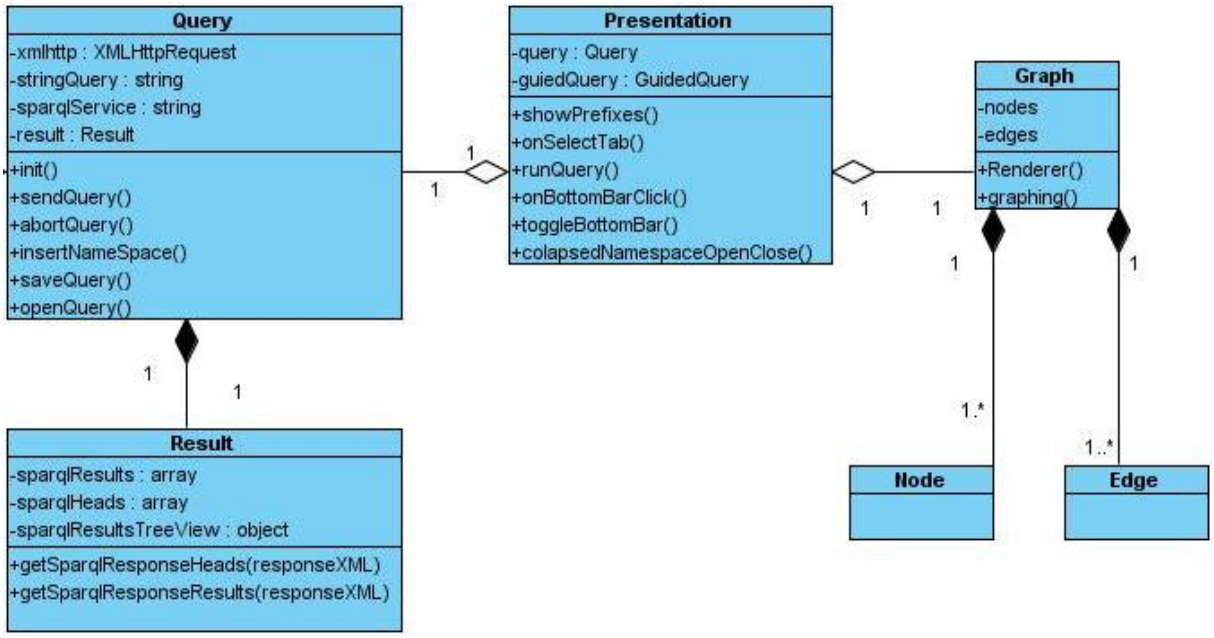


Figura 10 Ejemplo de la aplicación del patrón bajo acoplamiento

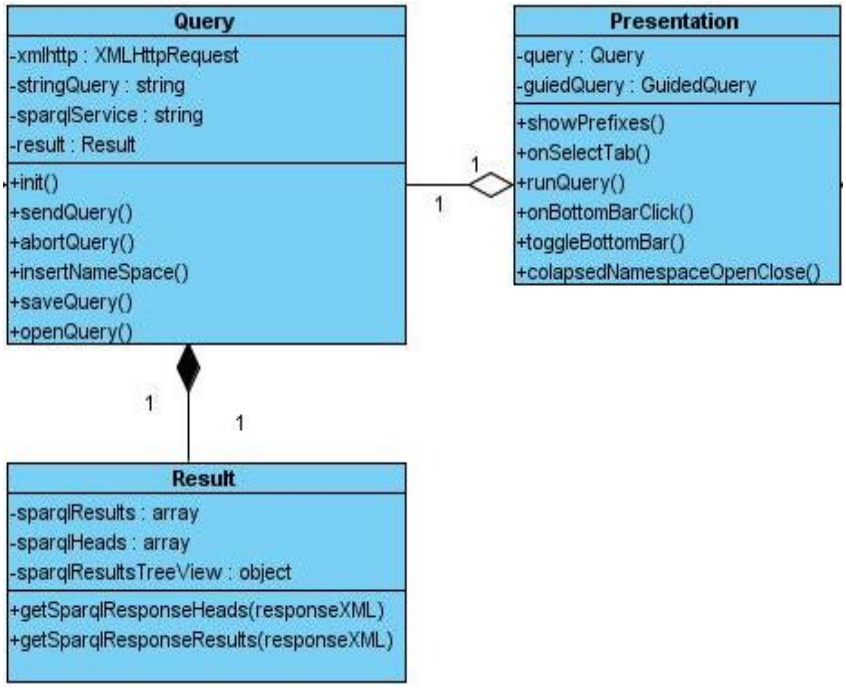


Figura 11 Ejemplo de la aplicación del patrón experto

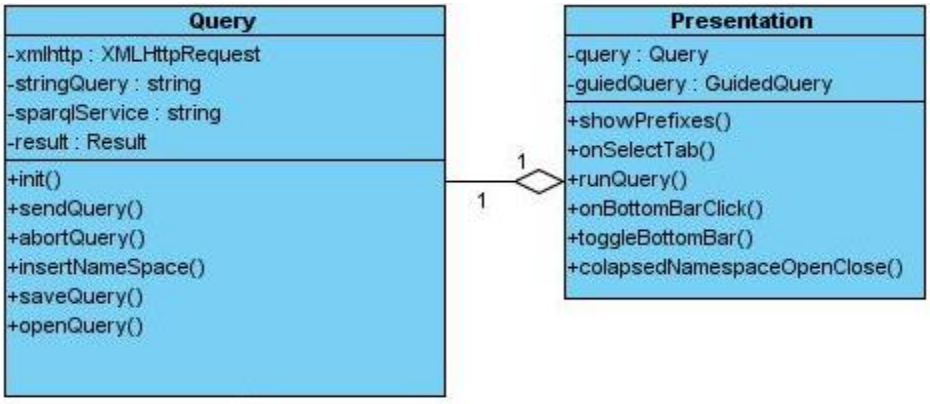


Figura 12 Ejemplo de la aplicación del patrón creador

Anexo 2: Pantallas de la aplicación

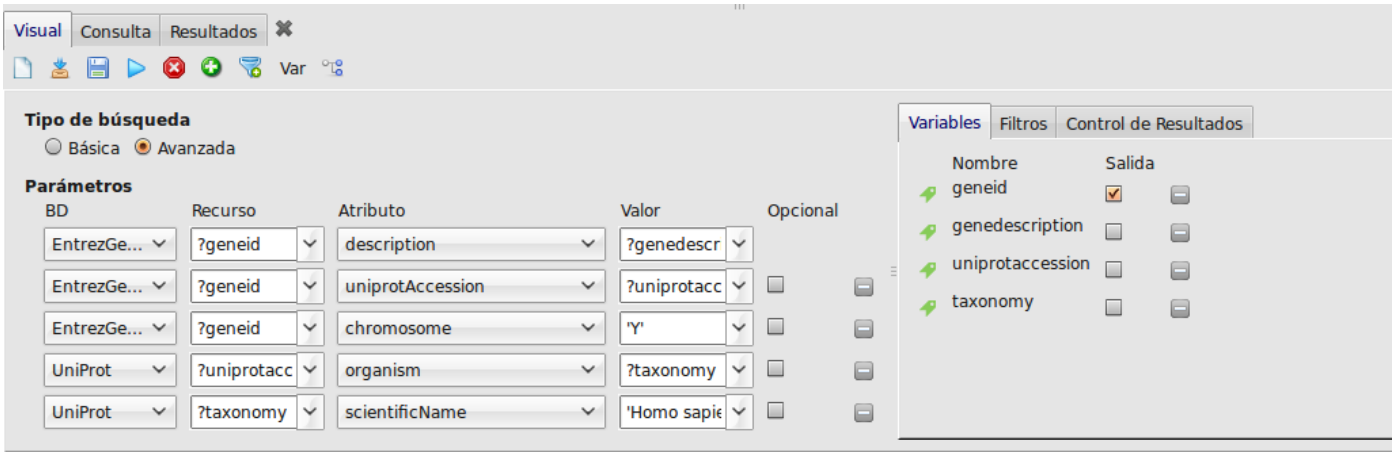


Figura 13 Interfaz para la realización de consultas SPARQL mediante formularios

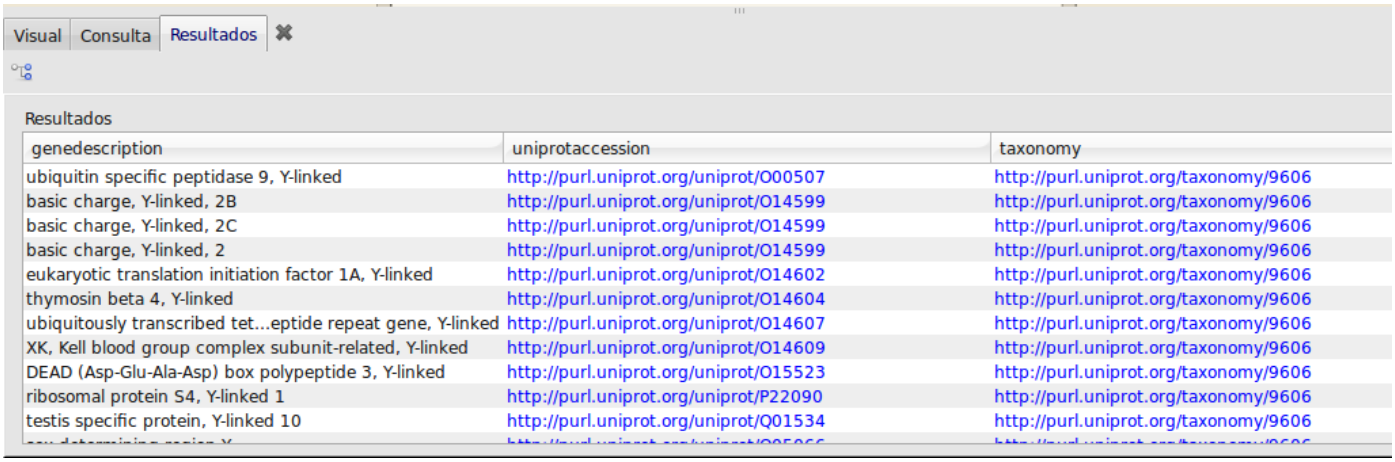


Figura 14 Interfaz para visualizar resultados

Glosario de términos

CSS: Hojas de Estilo en Cascada (CSS por sus siglas en inglés). Es una tecnología que permite crear páginas Web de una manera más exacta. Proporciona hacer cosas como incluir márgenes, tipos de letra, fondos, colores.

DOAP *(Description Of A Project*): es un vocabulario dentro de la Web semántica desarrollado por Edd Dumbill para describir proyectos de software libre mediante RDF, y que así puedan ser procesados fácilmente de una manera automática.

DOM (*Document Object Model* o Modelo de objetos en documentos): Es una interfaz independiente de la plataforma y del lenguaje que permite que los programas y scripts tengan acceso dinámicamente y actualicen el contenido, la estructura y estilo de los documentos.

FOAF (*Friend Of A Friend*): Proyecto dentro de la Web Semántica para describir relaciones mediante RDF que puedan ser procesadas fácilmente por máquinas.

JSON (*JavaScript Object Notation*): Formato ligero para el intercambio de datos. JSON es un subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de XML.

Script: Pequeño programa para realizar efectos especiales en las páginas.

Schema (esquema): Define un conjunto de obligaciones estructurales y de valores aplicables a documentos XML. Los esquemas pueden ser expresados en lenguajes de esquema, como XML.

SIOC (*Semantically-Interlinked Online Communities*): Provee métodos para interconectar diferentes sitios de discusión, desde blogs, foros y listas de correo. Consiste en una ontología, definida en RDFS y OWL-DL, que, reutilizando ontologías ya existentes como Dublin Core o FOAF, permite expresar de una manera procesable (no sólo por personas) información de estos tipos de sitios en Internet.

URIs (*Uniform Resource Identifier*): Cadena corta de caracteres que identifica inequívocamente un recurso (servicio, página, documento, dirección de correo electrónico, enciclopedia, etc.). Normalmente estos recursos son accesibles en una red o sistema.

URL (*Uniform Resource Locator*): Es una secuencia de caracteres, de acuerdo a un formato modélico y estándar, que se usa para nombrar recursos en Internet para su localización o identificación, como por ejemplo documentos textuales, imágenes, vídeos, presentaciones digitales, etc.

XHTML (*Extensible Hypertext Markup Language*): Lenguaje de marcado pensado para sustituir a HTML como estándar para las páginas Web.

XML: Lenguaje de Marcas Extensible (XML por sus siglas en inglés). Es un metalenguaje, es decir, un lenguaje que se usa para hablar acerca de otro lenguaje, pues no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades.

XMLHttpRequest: Es una interfaz para realizar llamadas mediante http.

XPath (*XML Path Language*): Lenguaje que permite construir expresiones que recorren y procesan un documento XML.

XSLT: Lenguaje de Transformación de Etiquetas Extensible de hojas de estilo, presenta una forma de transformar documentos XML en otros, incluso a formatos que no son XML.