UNIVERSIDAD NACIONAL DE LA PLATA

FACULTAD DE INFORMÁTICA

TESINA DE LICENCIATURA

TÍTULO: Extensión P2P para enriquecer la Web con soporte semántico para la toma de decisiones.

AUTORES: Iván Exequiel Colman

DIRECTOR: Dr. Alejandro Fernández

CODIRECTOR: Dr. Diego Torres

ASESOR PROFESIONAL:

CARRERA: Licenciatura en Sistemas - Plan 2015

Resumen

Esta tesina propone una estrategia para extraer información de la Web, crear un modelo semántico siguiendo las pautas de la Web Semántica y compartir el modelo entre usuarios mediante la arquitectura Peer-to-Peer (P2P) logrando así: (1) que los usuarios enriquezcan la Web; (2) integrar y agregar información a la Web Semántica; (3) mejorar la toma de decisiones basada en el modelo semántico que un usuario genera y comparte; (4) evitar la dependencia de un nodo central para compartir los modelos semánticos generados.

Palabras Clave

Web Semántica - Peer to Peer - Extensión Web - Toma de decisiones

Conclusiones

El presente trabajo propone mejorar la etapa de recolección de información en la toma de decisiones de los usuarios en la Web. Para el mismo se presenta una extensión Web Cross-Browser, el cual permite a los usuarios enriquecer la Web, crear plantillas e instancias mediante las técnicas soportadas por la Web Semántica y luego se comparten mediante una Arquitectura P2P. Todo nuevo recurso generado por los usuarios como así también la semánticas son nombradas mediante una URI logrando así respetar la filosofía de Linked Data, permitiendo modelar los mismo con RDF.

Trabajos Realizados

Se realizó una extensión Web quee permite agregar funcionalidad a la Web que los usuarios visitan, es por eso que un usuario experimentado puede generar una plantilla remarcando según su criterio, los más importante del producto, asigna un nombre, una semántica y sus propiedades. Luego, otro usuario que necesita tomar una decisión utiliza la extensión Web para comunicarse con otros pares y poder obtener las plantillas remotas correspondientes al sitio Web que visitó. El mismo examina la plantilla, observa cuales son las características más relevantes que señaló otro usuario experimentado, y si entiende que se aplica a su búsqueda crea la instancia.

Trabajos Futuros

Definir o encontrar una estrategia que permita identificar los Spam y Fake Data en la Web semántica. Definir o encontrar una estrategia que permita evitar la generación de Spam o Fake Data y su replicaciones en la red P2P. Definir una ontología que permite contar con nuevos predicados para representar al Spam o Fake Data en la Web semántica.

Fecha de la presentación: Febrero 2021

Extensión P2P para enriquecer la Web con soporte semántico para la toma de decisiones

Iván Colman

2021

Agradecimientos

Agradezco a mi madre, a mi padre que hicieron que esto sea posible.

También agradezco al Dr. Alejandro Fernandez y a todos los miembros del LIFIA.

A la Universidad pública, en especial a la Universidad Nacional de La Plata y a la Facultad de Informática por darme la oportunidad de realizar una carrera universitaria.

Resumen

La Web es una gran fuente de información que crece día a día, donde los usuarios y sistemas informáticos (agentes) consumen y producen información todo el tiempo. Esto la vuelve una importante ayuda para la toma de decisiones, ya sea por ejemplo para planificar un viaje o comprar un producto en base a sus características inherentes que permiten valorarlo con respecto a otras alternativas de su misma especie en base a las necesidades del usuario. Esta tesina propone una estrategia para extraer información de la Web, crear un modelo semántico siguiendo las pautas de la Web Semántica y compartir el modelo entre usuarios mediante la arquitectura Peer-to-Peer (P2P) logrando así: (1) que los usuarios enriquezcan la Web; (2) integrar y agregar información a la Web Semántica; (3) mejorar la toma de decisiones basada en el modelo semántico que un usuario genera y comparte; (4) evitar la dependencia de un nodo central para compartir los modelos semánticos generados.

Capítulo 1

1 Introducción

Internet es una gran red de redes de computadoras y una enorme cantidad de dispositivos inteligentes, todos conectados compartiendo información a través de diferentes protocolos. Día a día el contenido de la Web crece en tamaño y diversidad, muchas veces convirtiéndose en soporte para la toma de decisiones, por ejemplo: comparar un producto en base a las características propias, las opiniones y recomendaciones, planificar actividades o viajes, etc.

Comparar alternativas en base a múltiples criterios puede resultar complejo a los usuarios que no son especialistas en el tema. Generalmente, marcan las páginas que consideran relevantes a favoritos, arman una hoja de cálculo para comparar, les dan prioridades, toman nota, marcan con color lo más interesante o destacado, etc., un sinfín de tareas que cada usuario hace al momento de tomar una decisión para sí mismo.

Estas tareas traen diferentes desafíos: (1) la informaciòn colectada sólo tienen sentido hasta el momento de tomar la decisión final, luego el usuario la descarta; (2) no es compartida con otros usuarios que potencialmente pueden estar en la misma toma de decisión; (3) está armada en base a la percepción del usuario, lo cual lo hace difícil de entender por otro usuario si es que lo comparte de alguna forma; (4) la diversidad de tareas que realiza un usuario intentando extraer la información que considera importante para luego tomar una decisión son guardadas de forma local (por ejemplo: marcar sitios como favoritos, crear notas, crear planillas excel, etc.), no se comparten y por eso no se pueden reutilizar.

1.1 Motivación

Cada uno de nosotros puede crear y publicar sus documentos Web para compartirlos con el resto de las personas en Internet y referenciar desde el mismo a otros documentos Web, creando así una gran fuente de información para los usuarios. Es así como crece exponencialmente la cantidad y diversidad de información en la Web (Industria, gastronomía, viajes, portales de venta de productos, etc.) y la cantidad de usuarios que tienen acceso al mismo **[1]**.

Como resultado, la búsqueda de información en la Web y toma de decisiones a partir de la misma se convierte en uno de los principales usos de internet. De manera puntual podemos observar que el 81% de los adultos busca en la Web productos de su interés, el 68% genera las reservas de sus vacaciones, 32% califica a un producto y el 71% compra algo online **[2]**.

Las tareas que realiza un usuario para tomar una decisión en base a la búsqueda en la Web son complejas, marcan las páginas que consideran relevantes a favoritos, arman una hoja de cálculo para comparar, les dan prioridades, toman nota, marcan con color lo más interesante o destacado, etc. Una vez finalizadas estas tareas no pueden compartirse con otros usuarios porque son almacenadas de forma local en el usuario y dependen de la percepción del usuario. Es por eso que nace la necesidad de agilizar la toma de decisión

(más específicamente, agilizar la obtención de datos para la toma de decisiones), permitiendo crear un modelo semántico con la información extraída de la Web y compartirla con todos los demás usuarios y al mismo tiempo enriquecer la Web.

Se debe mencionar también que hoy la Web está pensada para ser interpretada por los humanos, es decir, un documento Web está diseñado para presentar la información de forma legible y de fácil lectura al ojo humano (lenguaje natural, imágenes, colores, tipos de fuentes y tamaño de letra, etc.) dando lugar a la libre interpretación y percepción de los usuarios. Si bien es posible crear aplicaciones (agentes) que extraen información de las páginas Web (scraping) para procesarla, el constante cambio de la estructura de los sitios obliga actualizar regularmente los algoritmos de extracción de dichos sistemas. Nace así la necesidad de representar todos los datos de un recurso en la Web de forma tal que también pueda ser interpretado por los sistemas informáticos (agentes) para automatizar tareas en las búsqueda, extracción y toma de decisiones. Esto se logra mediante la Web Semántica **[3]** (más adelante se profundiza el tema), los pasos a seguir son: (1) se crea un vocabulario para un dominio específico; (2) a partir del vocabulario se crea la metadata que le da significado al contenido de la Web y se lo asocia al mismo; (3) el modelo semántico se integra con otros modelos generados del mismo modo que los pasos anteriores; (4) ahora el contenido con significado semántico se publica para extender la Web Semántica.

Publicar datos para que estén disponibles en la Web Semántica requiere, por lo general, herramientas y técnicas que no están al alcance de los usuarios. Nos interesa simplificar esta actividad para que esté al alcance de más usuarios, generando más modelos semánticos compartidos e integrados en la Web Semántica. Es por eso que se pensó en una arquitectura P2P, sin un servidor que actúe como intermediario, desacoplando así a los usuarios de un servidor central, ya que en la Web Semántica es muy común compartir las fuentes de conocimiento y modelos RDF por cualquier usuario el cual colabora autónomamente con los demás **[4]**.

1.2 Objetivos

Esta tesina tiene como objetivo proponer y evaluar una estrategia y herramientas que permita a usuarios finales anotar sitios Web con información semántica, y compartir dichas anotaciones con una filosofía P2P. Para cumplir dicho objetivo esta tesina propone:

- Desarrollar una extensión que permita modelar, extraer y almacenar objetos (abstracciones) presentes en páginas Web siguiendo los últimos avances de la Web semántica.
- Integrar la extensión en una red P2P para que se constituya en una estrategia adicional de publicación de datos semánticos (complementaria de los mecanismos actuales de publicación de datos semánticos).
- Evaluar la extensión para el caso de información sobre teléfonos celulares en 4 portales diferentes.

Capítulo 2

2 Fundamentos

El objetivo principal de esta sección es explicar brevemente los diferentes conceptos, técnicas y estrategias que se necesitan para poder comprender lo desarrollado en esta tesina. Comenzaremos principalmente con la Web Semántica. Daremos una definición de la misma, los componentes que la forman y cómo facilita la integración de contenido que existe en la Web para su extracción y generar nueva información a partir de la misma. Posteriormente se explica que es una extensión Web junto con sus beneficios. Finalmente una breve descripción de las Arquitecturas P2P, sus características más importantes y sus ventajas frente a una arquitectura Cliente-Servidor.

2.1 Web Semántica

Como se mencionó anteriormente, la Web es una gran fuente de información, es un gran conjunto de documentos Webs interconectados y disponibles en Internet. Un usuario que cuente con un servidor Web puede publicar su documento HTML, luego referenciar a otros documentos y dejarlo disponible para el resto de los usuarios los cuales pueden interactuar (Figura 1).

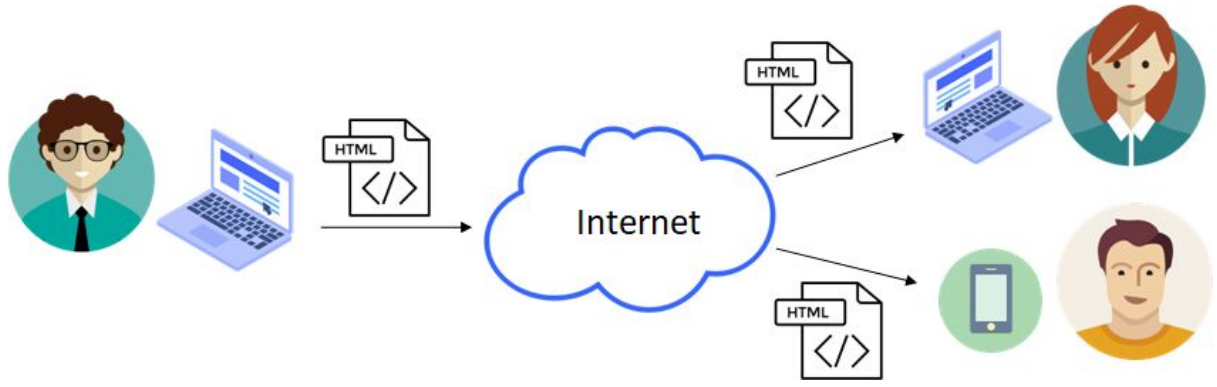


Figura 1

En simultáneo y construida sobre la Web anteriormente mencionada, existe otro tipo de Web denominada “Web Semántica”. Pero para poder explicar qué es la Web Semántica comenzamos explicando brevemente qué es la integración de datos, luego se desarrolla un ejemplo remarcando la complejidad de su proceso y finalmente se presentan dos escenarios donde los agentes informáticos intentan recolectar toda la información posible de un recurso en la Web.

La integración de datos hace referencia al proceso de buscar y combinar información de diferentes recursos que existen en la Web de manera tal que toda esa información relacionada tiene un significado importante o valioso para el usuario la cual le permite tomar una decisión. Supongamos, por ejemplo, que un usuario desea ir a cenar a un restaurante oriental, el proceso comienza cuando el usuario busca en la Web mediante su motor de

búsqueda favorito restaurantes dedicados a comida oriental mediante el lenguaje natural (aquí ya encontramos el problema de la ambigüedad de los resultados, esto está fuera del alcance de este tesina), elige uno y toma nota de la dirección, luego abre una nueva pestaña en su explorador Web y busca en su mapa favorito la ubicación del mismo y obtiene la ruta necesaria para llegar al lugar desde su casa. Si el usuario decide avanzar con ese restaurante lo siguiente que hace es buscar comentarios de otras personas en las redes sociales sobre la atención, precios y calificaciones del menú. En este punto el usuario está listo para tomar una decisión en cuanto a un restaurante específico entre todos los que encontró en la búsqueda inicial.

El ejemplo anterior intenta remarcar lo tedioso que puede llegar a ser el proceso de integración de datos en la Web para una posterior toma de decisión por parte del usuario. Una posible solución a este problema es que un agente (sistema informático) sea capaz de recolectar toda esta información desde la Web de forma automática y luego otro usuario (desarrollador de software) sea capaz de hacer uso de este agente y automatizar las búsquedas como la del ejemplo anterior para todos los usuarios que desean cenar comida oriental en un restaurante.

En cuanto al proceso que realizan los agentes para obtener la mayor cantidad de información de un recurso en la Web existen dos casos: 1) documento Web tradicional; 2) documento Web no tradicional **[3 p. 2]**. Si el sitio Web es un documento Web tradicional el agente tiene que ser capaz de recolectar toda la información posible mediante los elementos DOM del sitio, es decir acceder a los nodos del DOM (span, div, p, h1) los cuales son utilizados para presentar la información y no para asignarle significado, por cual el resultado final del agente sigue siendo cadenas de texto sin significado. En capítulos posteriores se detalla los problemas que tiene navegar sobre los elementos DOM de un sitio Web para extraer información.

Por otra parte existen los sitios Web no tradicionales, es decir además de las sentencias HTML contiene un tipo de sentencia especial que puede ser recolectada por los agentes con menor esfuerzo que el caso anterior. Analizemos por ejemplo como se extrae toda la información posible por parte de un agente en este caso, dadas las siguientes sentencias especiales:

ns0:JapanFood ns0:name “Japan Food”
ns0:JapanFood ns0:a ns0:Restaurante
ns0:JapanFood ns0:address ”Av. Corrientes 123”
ns0:JapanFood ns0:owner <ns0:_b2>
ns0:_b2 ns0:name “Lee”
ns0:_b2 ns0:wasBorn <http://www.china.org.cn>

“ns0” representa el namespace, es el prefijo del Sitio Web del restaurante.
ns0:JapanFood representa el recurso descrito por el namespace ns0.

La expresión (tripleta) "ns0:JapanFood ns0:a ns0:Restaurante" se lee así: El recurso ns0:JapanFood tiene ns0:name cuyo valor es “Japan Food”. O también de la siguiente forma: el recurso ns0:JapanFood tiene una propiedad ns0:name, cuyo valor es “Japan Food”.

"ns0:JapanFood ns0:a ns0:Restaurante" es diferente a la sentencia anterior, el valor (ns0:Restaurant) no es una cadena de texto, sino otro recurso en la Web. "ns0:JapanFood ns0:owner <ns0:_b2>" también es una sentencia diferente a las dos anteriores, <ns0:_b2> es un recurso anónimo, el cual en las sentencias siguientes tiene la propiedad ns0:name con valor “Lee” y la propiedad ns0:wasBorn con valor <http://www.china.org.cn>, el cual representa un recurso en la Web. Esta nomenclatura se explica con detalle en la sección RDF.

El agente ahora es capaz de organizar toda esta información leída en un grafo como el siguiente (Figura 2), luego el proceso se repite por cada recurso que encuentre en dichas sentencias, por ejemplo al encontrar el recurso <http://www.china.org.cn> arma otro grafo (Figura 2) logrando así la integración de los datos en la Web.

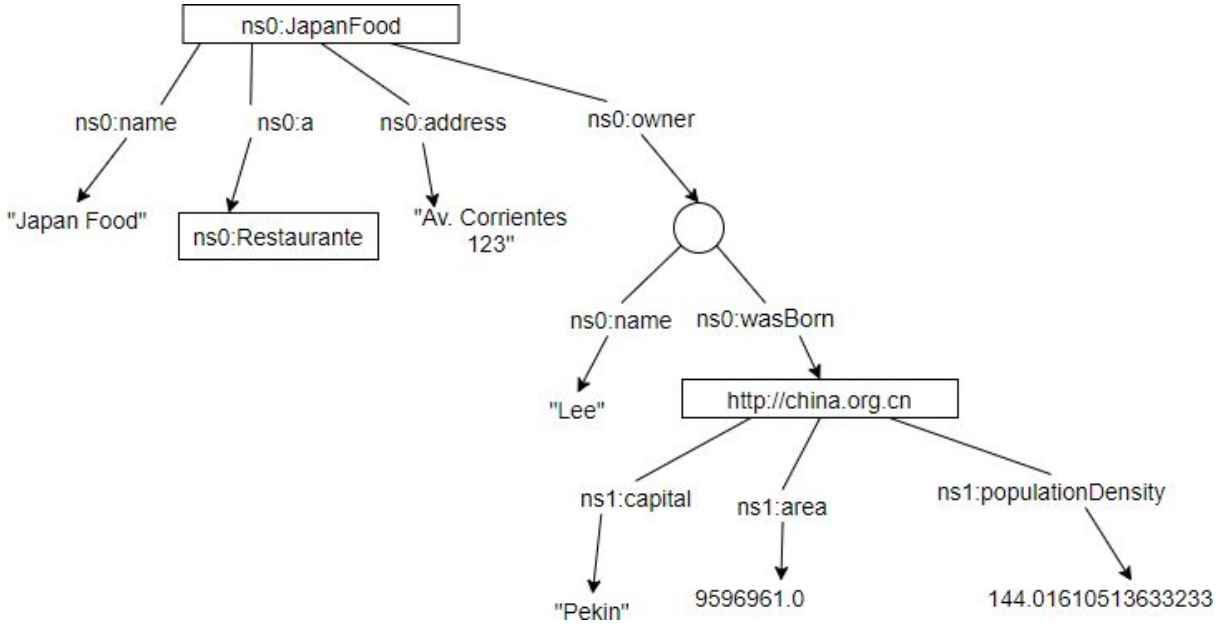


Figura 2

En la mayoría de los lenguajes de programación la palabra “semántica” se relaciona fuertemente con la palabra “sintaxis”. Sintaxis indica “cómo” se deben escribir las sentencias, y Semántica hace referencia a “que” significa cada expresión, por lo tanto la “Web Semántica” puede ser entendida como “La Web de los significados” **[3]** el cual se relaciona fuertemente con el ejemplo anterior de las búsquedas en la Web. Con todo lo expuesto hasta aquí estamos en condiciones de dar una primera definición formal de Web Semántica.

Si bien hay muchas ideas acerca de que es la Web Semántica, su creador Berners-Lee la define como:

“The Semantic Web is an extension of the current Web in which information is given well-defined meaning, better enabling computers and people to work in cooperation.” **[7]**

Liyang Yu, autor del Libro “A Developer`s Guide to the Semantic Web”, define la Web Semántica como:

“The Semantic Web is a collection of technologies and standards that allow machines to understand the meaning (semantics) of information on the Web.” **[3].**

Contar con una versión semántica de la información contenida en la Web puede reducir en gran medida el esfuerzo de los usuarios que colectan información para tomar decisiones, permitiendo compartir la información y mejorar los resultados de las búsquedas en Internet. La idea es que la Web no sirva solamente para presentar la información sino también para que los sistemas informáticos (agentes) sean capaces de encontrar, procesar, transformar y hacer un buen uso de ellos (Figura 3).

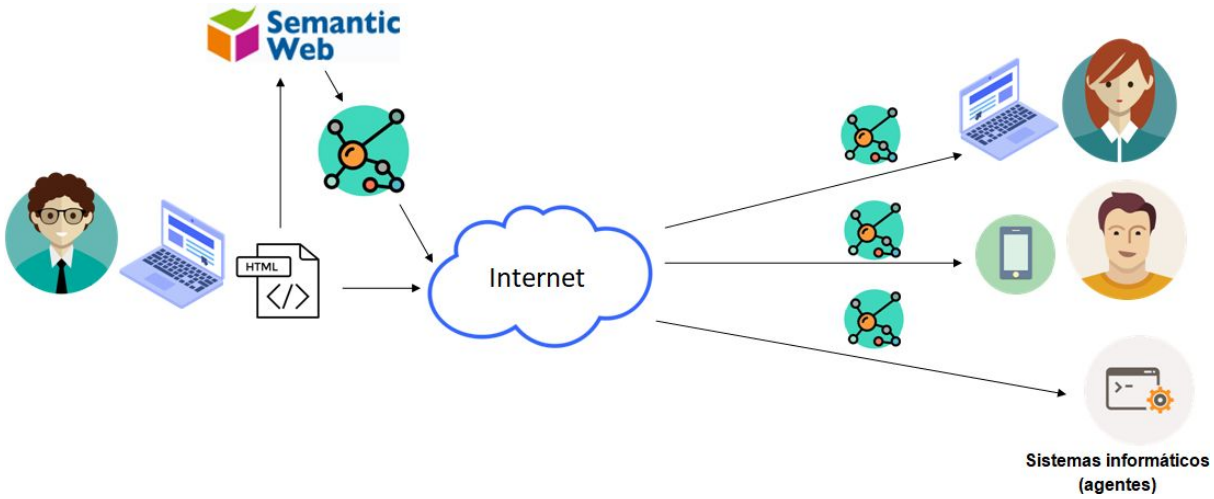


Figura 3

Avanzando en el concepto de Semántica, necesitamos una técnica para representar la información en algún formato para que tanto los desarrolladores de aplicaciones como los agentes informáticos puedan entenderla. A continuación se describe RDF, una de las técnicas que define la base de la Web Semántica, se enumeran sus características más importantes, como aporta a la Web Semántica y luego se ilustran algunos ejemplos para terminar de comprender.

2.1.1 RDF

Como hemos dicho, la Web fue pensada para ser legible por los humanos pero no así para los agentes informáticos, proponer este cambio manualmente es casi imposible debido al gran volumen de información contenida en la Web, es por eso que en febrero del año 1999 W3C desarrolló el estándar para modelar los datos contenidos en la Web, RDF (Resource Description Framework)**[8]**. RDF enfatiza la interoperabilidad de un modelo de propiedades y valores entre aplicaciones que intercambian información comprensible por las máquinas (agentes informáticos) en la Web. Para comprender mejor RDF comencemos con su estructura abstracta básica del modelo de RDF el cual consta de tres componentes:

El primer componente es el **Recurso**. Todas las cosas que se representan mediante RDF se denominan “recurso”. Un recurso puede ser: una página Web completa (documento HTML), un fragmento del Sitio Web, también puede ser una colección de páginas o un objeto que no puede ser accedido a través de la Web (libro impreso).

Un punto importante es que los recursos siempre se nombran mediante una URI porque identifica unívocamente al recurso en la Web. En la sección de Linked Data se desarrolla este punto con respecto a las URI y su importancia en la Web Semántica.

Como segundo componente se encuentra la **Propiedad** el cual representa una característica, atributo, relación que se utilizan para describir un recurso. Cada propiedad tiene restricciones sobre sus posibles valores, el tipo al cual puede describir, es decir al "recurso" al cual puede describir y también su relación con otras propiedades, así como el recurso, la propiedad también se nombra con una URI.

Como último componente se encuentra la **Sentencia**. Una sentencia RDF es la combinación de un recurso específico junto con una propiedad nombrada más el valor de dicha propiedad. Estas tres partes que conforman la sentencia se denominan respectivamente: Sujeto, Predicado y Objeto, donde el objeto puede ser el valor literal de la propiedad (string, number) u otro recurso (especificado mediante una URI).

<Sujeto><predicado><Objeto>

Hay que mencionar que el modelo RDF propuesto por W3C utiliza XML (Figura 4.c) para la creación e intercambio del modelo RDF, pero también puede ser representado en más de un formato para lograr ser más entendible por el humano (pensando en los desarrolladores de aplicaciones, que a veces tienen que inspeccionar documentos RDF). Algunos de los más usados son: Turttle (Figura 4.a), grafo (Figura 4.b). En las secciones posteriores se continúa con este tema, y se explica que formato utiliza esta tesina con el fin de cumplir su propósito general. De forma simplificada a continuación veremos un ejemplo de cada uno.

```
<www.ivancolman.com.ar><http://www.mydomain.org/site-owner><"Iván Colman">
```

Figura 4(a): Terna - turtle .ttl

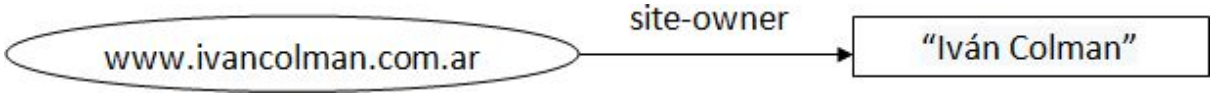


Figura 4(b): grafo

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:mydomain="http://www.mydomain.org/my-rdf-ns">
        <rdf:Description
          rdf:about="www.ivancolman.com.ar">
           <mydomain:site-owner>Iván Colman</mydomain:site-owner>
        </rdf:Description>
</rdf:RDF>
```

Figura 4(c): XML

Prosigamos con el siguiente ejemplo para asentar los conceptos de RDF y resaltar su potencial. En la figura 5.a existen dos grafos, ambos son resultado de modelar datos de la Web siguiendo la filosofía de RDF, el primer grafo se extrajo los datos en inglés, y el segundo contiene los mismo datos en francés agregando más información.

Ahora bien, el libro tiene la misma URI, estamos frente al mismo recurso en ambos grafos, por lo cual podemos hacer un “merge” de ambos grafos, es decir unir la información de ambos, logrando así hacer crecer la información y dejarla disponible al mundo (figura 5.b).

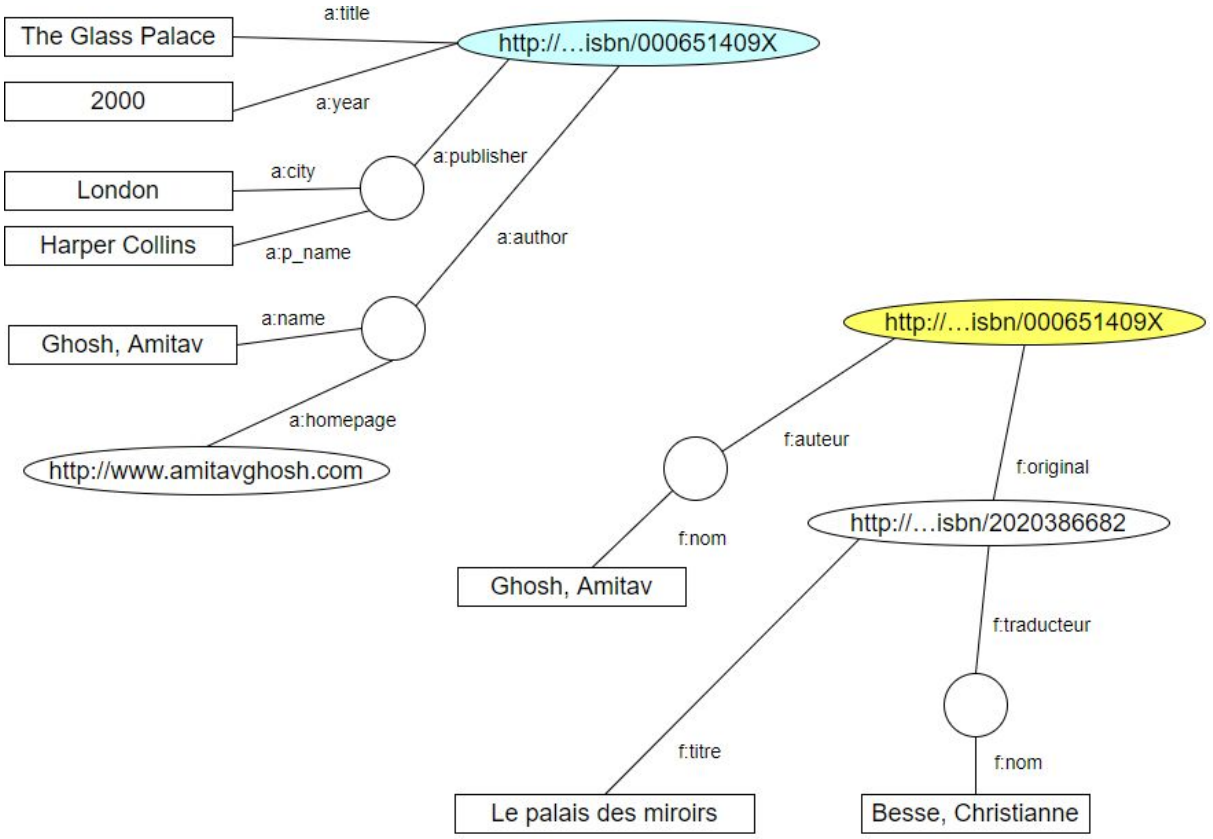


Figura 5.a

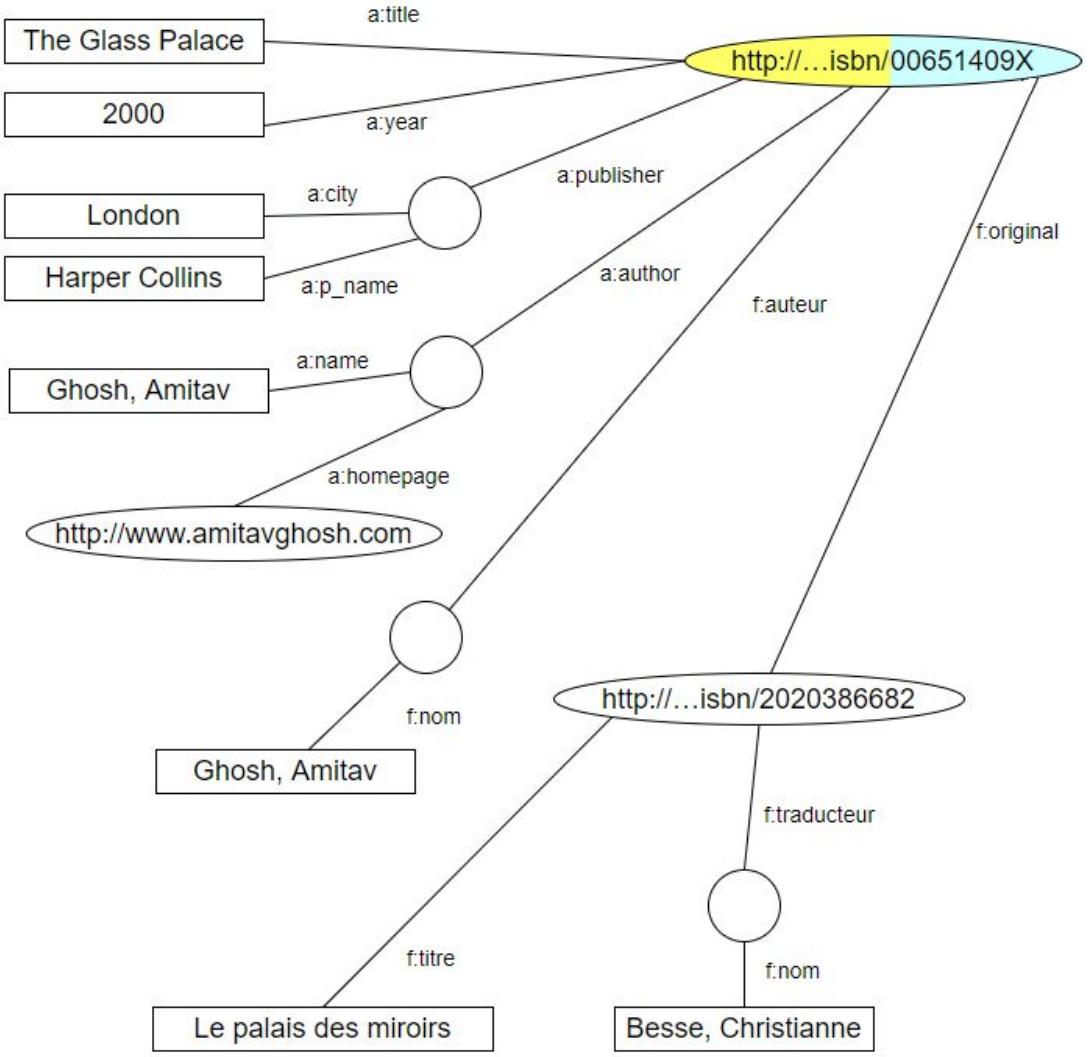


Figura 5.b

Con lo expuesto hasta aquí podemos ver que RDF es una técnica para modelar los datos en la Web permitiendo la operabilidad entre máquinas (agentes informáticos) y para los desarrolladores, es decir es la base para la Web Semántica.

Ahora bien, el paso siguiente es agregar semántica a los datos, darles un contexto y permitir el razonamiento y la inferencia mediante agentes de software automatizados. Esta técnica tiene que ser la base para generar la semántica de la información. Dicha técnica es conocida con el nombre de Ontología y forma parte de los fundamentos más importantes de la Web Semántica. El propósito de la siguiente sección es definir que es una Ontología, analizar su estructura y finalmente proponer unos ejemplos de la misma.

2.1.2 Ontologías

Los últimos acontecimientos informáticos han dado lugar a que vuelva a manifestarse el campo filosófico de la ontología. Podría decirse que la ontología filosófica comienza con los filósofos griegos, hace más de 2400 años.

Los Ontólogos buscan crear una teoría a nivel general, de todos los diferentes tipos y cosas que existen en el mundo incluyendo las relaciones que existen entre ellos. De manera

semejante, desde el punto de vista Informático, la ontología computacional comenzó hace unos 30 años. La W3C describe una ontología como:

"Una ontología define formalmente un conjunto común de términos que se utilizan para describir y representar un dominio... Una ontología define los términos utilizados para describir y representar un área de conocimiento." **[9].**

Consideremos ahora los aspectos más importantes de una ontología, a continuación se enumeran y se explican brevemente cada una de ellas:

1. Pertenecen a un dominio específico, representan un área de conocimiento. Un dominio es una temática de un área de conocimiento específico por ejemplo: tecnología, fotografía, medicina, arte.
2. Contienen "Clases" el cual hace referencia a cosas o conceptos del mundo real. Las relaciones entre clases pueden representarse mediante herencia.
3. Además de las relaciones entre clases, existe otro tipo de relación, las "propiedades". Estas propiedades describen las características y atributos de los conceptos o cosas y también pueden ser usados para asociar diferentes clases entre sí.
4. También define las reglas y restricciones sobre las propiedades.
5. Existe el concepto de instancias.

Esto permite crear un modelo abstracto de un dominio del mundo real en el cual mediante el concepto de clases, propiedades y la relación entre ellos deja de existir la ambigüedad y permite que el mismo pueda ser procesado por agentes informáticos y entendido al mismo tiempo por los humanos.

Una ontología con el conjunto de instancias de las clases constituyen la base de conocimiento. Así por ejemplo en la Figura 6 marca en color blanco la ontología y en rojo las instancias.

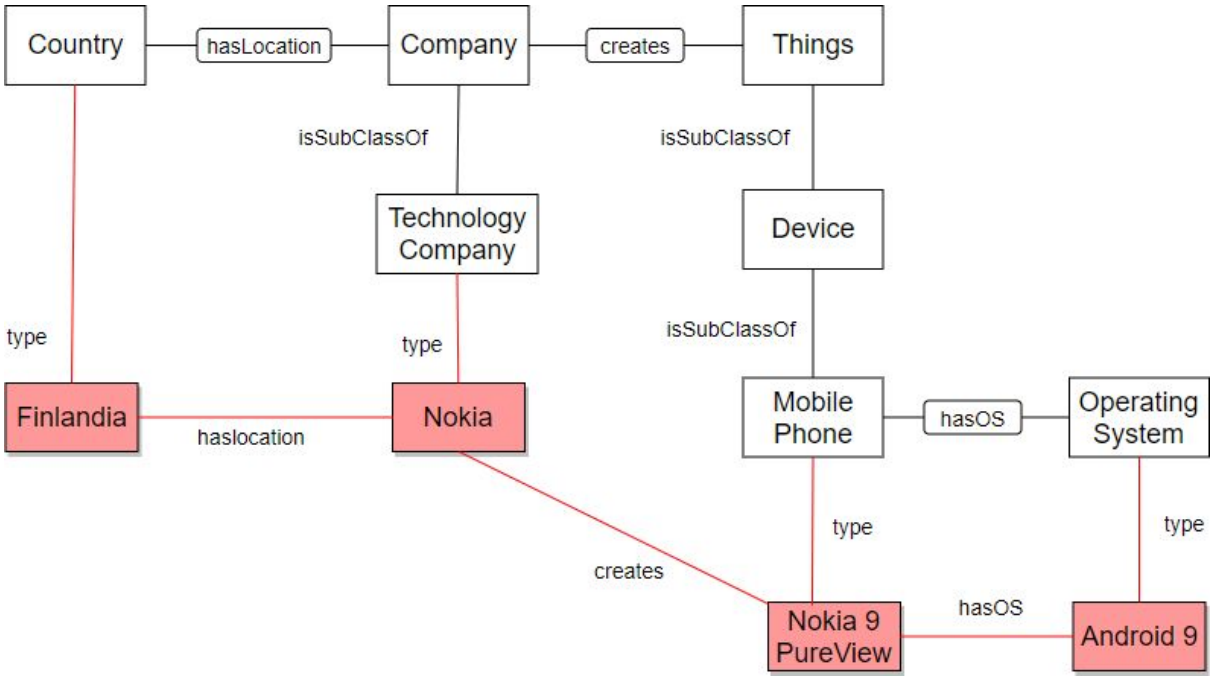


Figura 6: Ontología (blanco), instancia (rojo)

2.1.3 RDF Schema

En el año 2004 el grupo W3C definió a RDFS (RDF Schema) como una extensión semántica de RDF **[10]**, es decir provee los mecanismos para describir a un individuo, un grupo de individuos y las relaciones entre ellos mediante reglas ontologicas.

Todos los términos definidos en RDFS son identificados mediante una URI y todos sus terminos comienzan con la siguiente URI http://www.w3.org/2000/01/rdf-schema#. Por convención esta URI está asociada al namespace “rdfs:”.

Es importante remarcar que RDFS no hace referencia a un dominio específico, lo que propone es definir un mecanismo para definir clases, propiedades basado en la filosofía RDF.

Sigamos el análisis de RDFS con sus principios técnicos y luego enumeramos algunos de sus términos principales:

El concepto de clases y propiedades en RDFS es similar a la programación orientada a objetos (POO) como así también el concepto de instancias y herencia entre clases. Pero a diferencia de POO donde define las clases en término de sus propiedades, RDFS describe las propiedades en función de las clases de recursos a los que se aplica, es decir se limitan a un dominio y posible rango de valores.

Otra diferencia con POO es que permite la herencia de propiedades.

Consideremos ahora algunos de sus términos más importantes:

Clases:

- rdfs:Class permite definir nuevas clases a partir de un nuevo recurso.
- rdfs:Resource todo las cosas para RDF son un recurso, es la clase base para todas las cosas. (rdfs:Resource es una instancia de rdfs:Class)
- rdfs:Literal es la clase para representar a las cadenas de texto o números enteros.
- rdfs: DataType corresponde a todos los tipos de datos del modelo RDF.
- rdfs:Property es la clase para representar a todas las propiedades de RDF y definir nuevas propiedades.

Propiedades:

Todas las siguientes propiedades son instancias rdfs:Property:

- rdfs:type esta propiedad es usada para indicar que un recurso es instancia de una clase.
- rdfs:subClassOf indica que la instancia de una clase es instancia de otra clase.
- rdfs:subPropertyOf esta propiedad es usada para indicar que una instancia de una propiedad es instancia de otra propiedad.
- rdfs:label es usada para proveer una versión legible para el humano sobre el nombre del recurso.
- rdfs:domain esta propiedad indica que el recurso que tiene dicha propiedad es instancia de al menos la clase. Ejemplo: P rdfs:domain C, donde “P” es una propiedad y “C” es una clase, entonces la propiedad “P” pertenece a instancias de la clase “C”.
- rdfs:range permite restringir los posibles valores de una propiedad. Ejemplo: "P rdfs:range C, donde “P” es una propiedad y “C” es una clase, entonces la propiedad “P” admite como valor instancias de la clase “C”.

Si bien el propósito de esta tesina no es crear una ontología, la próxima sección corresponde a un lenguaje para crear ontologías para la Web: OWL

2.1.4 OWL

Examinaremos ahora el lenguaje de Ontologías Web más usado actualmente, Web Ontology Language (OWL).

OWL fue publicado por W3C Web Ontology Working Group en el año 2004, fue diseñado para que las computadoras puedan procesar el contenido de la información en lugar de ser solo legible para el humano. Actualmente se recomienda usar la versión 2 de OWL.

A diferencia de RDFS, OWL permite crear agentes con mayor capacidad de razonamiento porque agrega semántica y quita la ambigüedad a los datos, logrando así mejorar la toma de decisiones de los usuarios al hacer uso de la gran fuente de información disponible en la Web.

Podemos indicar que OWL contiene todas las funcionalidades de RDF Schema pero agrega mejores estructuras para mejorar la expresividad de los datos. OWL agrega más vocabulario para describir clases, propiedades, relaciones, cardinalidad, igualdad, más tipos y características de propiedades y clases enumeradas. **[11].**

OWL ofrece tres sublenguajes, cada uno con mayor expresividad que el anterior, OWL Lite, OWL DL, OWL Full.

$$\text{OWL Lite} \subseteq \text{OWL DL} \subseteq \text{OWL Full}$$

Continuemos con un simple ejemplo que define una pequeña ontología (notación turtle) con OWL2 para aplicar lo visto hasta el momento, recordar que todo lo visto de RDFS sigue siendo válido en OWL (figura 7):

```
# Prefix
@prefix owl: <http://w3.org/2002/07/owl#>
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
@prefix xsd: <http://www.w3.org/2001/XMLSchema>

# Classes
:Book a owl:Class .
:Writer a owl:Class .
:Novel a owl:Class ;
        rdfs:subClassOf :Book .
:Poet a owl:Class ;
        rdfs:subClassOf :Writer .
:Book owl:disjointWith :Writer .

# Object properties
:author a owl:ObjectProperty ;
        rdfs:domain :Book ;
        rdfs:range :Writer .

# Data properties
:publicationYear a owl:DataTypeProperty ;
        rdfs:domain :Book ;
        rdfs:range xsd:integer .

# Individuals
:GeorgeOrwell a :Writer .
:NineteenEightyFour a :Book ;
        :author :GeorgeOrwell ;
        :publicationYear 1948 .
```

Figura 7

Analizando el ejemplo lo primero que define son los prefijos a ser usados, los necesarios para el ejemplo.

Luego se definen dos clases, “Book” y “Writer” mediante la sentencia “owl:Class” y además dos clases que heredan de “Book” (owl:Novel) y de “Writer” (owl:Poet). Inmediatamente a las definición de las clases agrega una regla semántica: un individuo no puede ser instancia de “Book” e instancia de “Writer” al mismo tiempo (:Book owl:disjointWith :Writer).

Una vez definidas las clases estamos en condiciones de definir propiedades de objetos (owl:ObjectProperty) y propiedades de tipo de datos (owl:DataTypeProperty), cada uno restringiendo los posibles valores de dominio (owl:domain) y rango (owl:range).

Una vez finalizada la ontología, se crean las instancias (:publicationYear y :NineteenEightyFour) y se asignan valores a las propiedades definidas en la clase “Book”.

Deseo subrayar que el predicado ‘a’ representa la IRI: http://www.w3.org/1999/02/22-rdf-syntax-ns#type el cual permite crear una instancia de una clase en particular.

En el caso del ejemplo anterior se decidió usar la notación turtle (.ttl), ya que es compatible con la sintaxis de RDF, permite escribir en este formato a los grafos RDF que se analizó en las secciones anteriores y además es más legible para el humano.

Es por eso que la próxima sección analiza brevemente la notación Turtle nombrando sus características y ventajas.

2.1.5 Tripletas

A pesar que RDF es un framework para representar la información de la Web, se necesita un modelo de datos para serializar todas las características que presenta RDF (grafo, IRIs, Literales, Blank Node) **[12]**. Es por eso que en esta sección se analiza el formato Turtle.

El formato Turtle define la sintaxis para representar y serializar todas las características de RDF definida por W3C **[12]**. Además su formato es compacto, legible al humano (en caso de que deba hacerlo, por ejemplo durante el desarrollo de aplicaciones) y representa de forma textual un grafo RDF.

Un grafo RDF puede ser serializado en forma de tripletas, las tripletas (o “triples”, en inglés) consta de tres componentes denominados de la siguiente forma: Sujeto, Predicado y Objeto.

Esta tripleta es escrita por convención en el siguiente orden:

<Sujeto><Predicado><Objeto>

Siguiendo con la estructura anterior, a continuación se analiza los posibles valores de la tripleta.

Sujeto puede ser cualquiera de los siguientes elementos: (1) IRI, (2) Blank Node.

El segundo término es Predicado, el cual debe ser una IRI.

Como tercer elemento de la tripleta encontramos al Objeto, el cual puede ser algunos de los siguientes elementos: (1) IRI; (2) Blank Node; (3) Literal.

Antes de comenzar con la sintaxis de Turtle, continuaremos analizando brevemente algunos de los términos mencionados anteriormente:

- IRI: (Internationalized Resource Identifier) es una secuencia de caracteres UTF-8 limitada delimitada entre los caracteres ‘<’ y ‘>’. Toda URI o URL es un IRI pero no todo IRI es una URI **[13]**.
- Blank Node: Es aquel componente de RDF que no tiene identificación en la Web, es decir no tiene una IRI, no se lo puede representar unívocamente en la Web. Todas las características de RDF se pueden aplicar al Blank Node, es decir, se puede pensar a Blank Node como un elemento anónimo que tiene propiedades, es instancia de una clase, se relaciona con instancias de otras clases o de si mismo, etc. Ejemplo:

```
_:bn1 <http://an.example/predicate1> "value1" .
_:bn2 <http://an.example/predicate2> <http://an.example/resource1> .
```

- Literal: Es una cadena de texto UNICODE limitado entre comillas dobles.

Opcionalmente seguido al objeto del tipo Literal puede definirse el tipo (^^IRI), es decir si el valor entre comillas corresponde a una cadena de texto, entero o decimal. Ejemplo:

<object><predicate>"value 1"^^ <http://www.w3.org/2001/XMLSchema#string> .

Volviendo a la sintaxis de Turtle, toda notación Turtle comienza definiendo los prefijos “@prefix”, los mismos permiten definir un nombre de espacio de nombre (namespace) y asociarla a una URL la cual forma la la primera parte de las IRI a usar con ese namespace. También es posible definir el prefijo por defecto que se usa en la notación “@base”. Por ejemplo:

@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix rel: <http://www.perceive.net/schemas/relationship/> .

Continuando con las sentencias, cada tripleta finaliza con un espacio y con “.”, pero se permite escribir varios predicados para el mismo objeto separado por un espacio y “;”, por ejemplo:

<http://an.example/resource1> <http://an.example/predicate1> "value1" **;**
<http://an.example/predicate2> "value2" **;**
<http://an.example/predicate3> "value3" **.**

Podemos considerar con lo dicho hasta aquí que Turtle permite modelar de forma compacta el modelo abstracto que propone RDF, de manera legible y totalmente compatible con la filosofía de RDF.

2.2 Extensiones Web

Las extensiones Web son programas que se ejecutan en el contexto de los exploradores Web. Son un pequeño software desarrollado en Javascript, HTML, CSS que se integra al navegador Web como un simple icono, un menú desplegable, en la barra de exploración o también puede presentarse como una página Web entera. Los exploradores Web agregaron esta funcionalidad progresivamente: 1) Internet Explorer, versión 5, año 1999; 2) Firefox, desde su lanzamiento año 2004; 3) Opera, versión 10, año 2009; 4) Google Chrome, versión 4, año 2010; 5) Safari, versión 5, año 2010; 6) Microsoft Edge, marzo 2016;

En la actualidad los exploradores Web permiten agregar o modificar funcionalidad a sí mismos o a los sitios Web, logrando así enriquecer la experiencia de usuario mediante: (1) seguridad Web; (2) administración de cookies; (3) notificaciones de redes sociales; (4) administración de tabs; (5) accesos directos de portales de compras; etc. Un fin de

funcionalidades que día a día se desarrollan con el fin de mejorar la experiencia de usuario final.

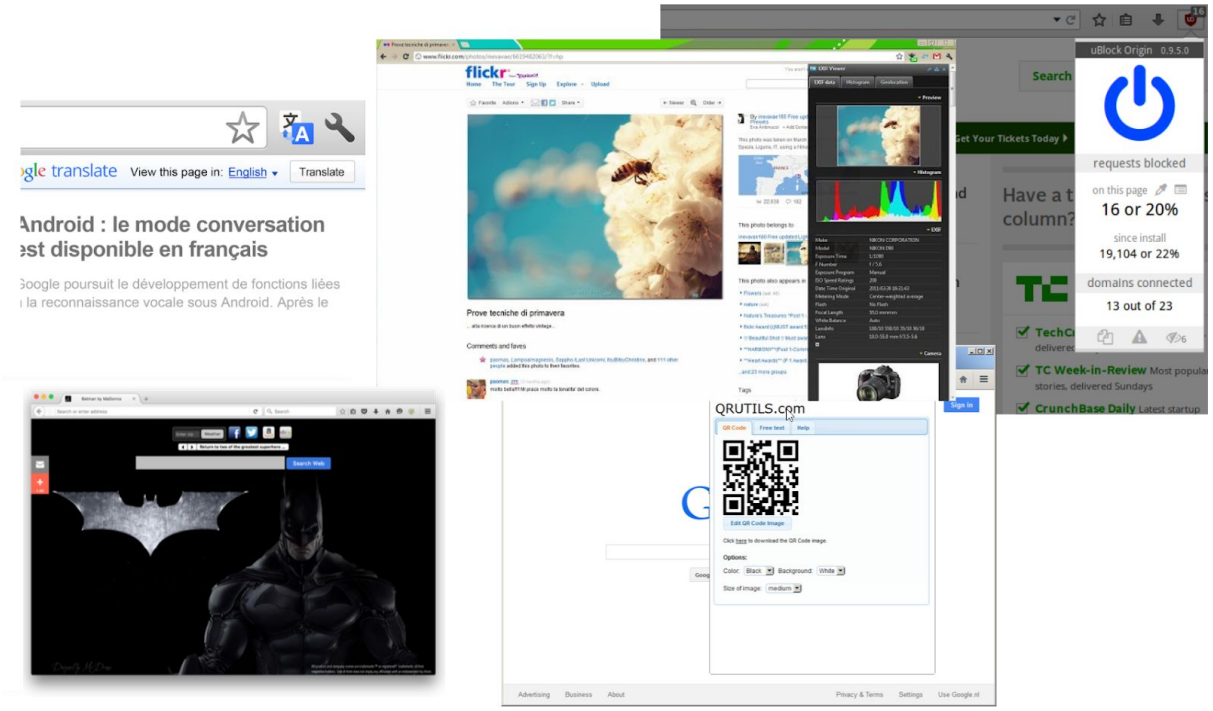


Figura 8: Ejemplos de extensiones Web

En cuanto al trabajo de esta tesina se utilizó la librería javascript “Webextension-polyfill” la cual permite que la extensión Web desarrollada sea compatible en los exploradores Web como: Chrome, Opera, Mozilla Firefox, Microsoft Edge.

Con respecto a la arquitectura de las extensiones Web, se destacan los siguientes componentes: 1) manifest; 2) background script; 3) elementos UI; 4) content script; 5) opciones de página.

El archivo manifest es un archivo json que contiene principalmente: nombre, versión, los permisos, archivos utilizados, etc.

Background Script se encarga de manejar todos los eventos generados por la extensión, existe solo uno a nivel global para todos los tabs abiertos del explorador. Su función principal es administrar los datos almacenados localmente, realizar llamadas asíncronas y administrar los tabs.

Los elementos UI son las interfaces que interactúan con el usuario, botones, campos de texto, combos, popup, etc.

Content script contiene código javascript el cual tiene acceso a todos los elementos DOM del sitio Web el cual se cargó en el explorador Web.

Las opciones de página brindan versatilidad a los usuarios para configurar la extensión, activar a desactivar funcionalidad de la extensión dentro del explorador Web.

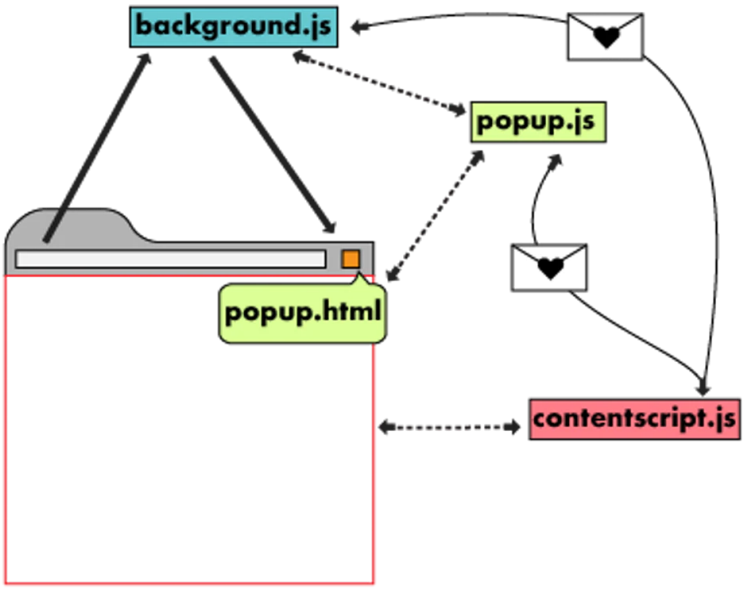


Figura 9: Arquitectura de las extensiones Web.

2.3 Arquitectura P2P

En esta tesina se adopta el estilo de arquitectura P2P; a continuación se analiza brevemente esta arquitectura y se enumeran las ventajas de la misma.

Las arquitecturas peer-to-peer se denominan así porque elimina al servidor que actúa como intermediario, permitiendo que los sistemas finales en los que se encuentran las aplicaciones interactúen entre ellos compartiendo información. Así por ejemplo el primer sistema de intercambio de archivos ampliamente utilizado con arquitectura P2P fue Napster, donde los pares transferían directamente los archivos entre ellos. Luego surgieron una gran variedad de sistemas de intercambios de archivos, tales como eDonkey, eMule, BitTorrent, etc.

Continuando con la arquitectura P2P, a continuación se enumeran las principales características de la misma **[14]**:

- Recursos compartidos: Cada par aporta recursos a la red P2P, es decir lo ideal es que sea proporcional los recursos que comparte un usuario con los que obtiene. A veces esto no se cumple cuando se suman usuarios “gratuitos”, aquellos que solo obtienen recursos sin compartir alguno.
- Red: todos los nodos están conectados con otros nodos en el sistema P2P formando así un grafo conectado.
- Descentralización: No existe un control central, los nodos pares son quienes determinan el comportamiento de la arquitectura P2P. En algunos casos se protege a la red mediante un servidor de inicio de sesión central.
- Simetría: todos los nodos cumplen el mismo rol, en un momento determinado actúan como servidor y en otro momento dado como cliente. Pero existe la posibilidad de asignar roles especiales (super pares o “relay” pares).

- Autonomía: Cada par determina su participación en la red P2P, es decir cada nodo decide cuándo conectarse o desconectarse. No existe un sistema de administración de pares.
- Auto-organización: la organización del sistema P2P aumenta con el tiempo utilizando el conocimiento local y las operaciones locales en cada par.
- Escalable: Más que una característica, es una condición para la arquitectura.
- Estabilidad: Más allá de la tasa máxima de abandonos, el sistema P2P debe ser estable, debe mantener al grafo conectado y poder enrutar deterministicamente dentro de los límites prácticos de conteo de saltos.

Dicho lo anterior encontramos una definición formal de una arquitectura P2P:

"Podemos definir Peer-to-Peer de la siguiente manera: El término "Peer-to-Peer" describe los sistemas que utilizan una arquitectura descentralizada que permite a los pares individuales proporcionar y consumir recursos sin un control centralizado." **[4 (p. 4)].**

En el modelo cliente/servidor los roles y responsabilidades están bien definidas (Figura 10).

En cambio en el modelo P2P (figura 11) los pares pueden actuar como servidor en un momento dado y como cliente en otro, dando así más flexibilidad y escalabilidad.

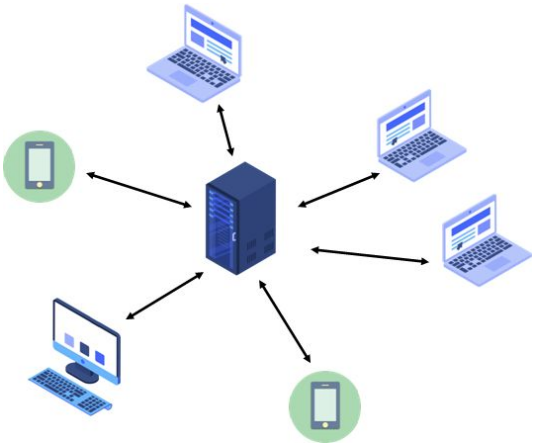


Figura 10: Cliente/Servidor

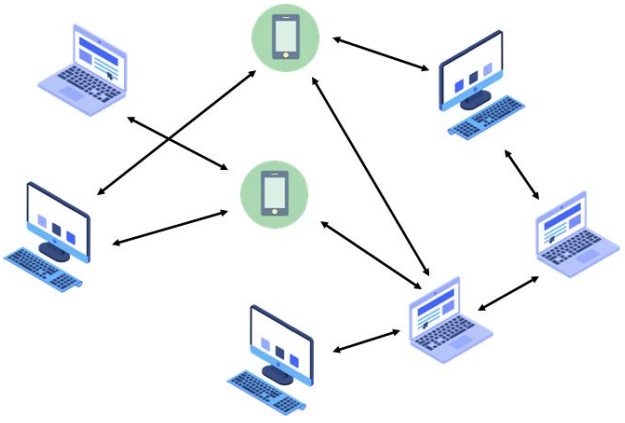


Figura 11: Peer to Peer

Capítulo 3

3 Antecedentes y trabajos relacionados

En esta sección se presentan algunos antecedentes remarcando las características y ventajas que se consideran más importantes y que fueron tomadas como punto de partida para poder cumplir el objetivo de esta tesina. Dichos antecedentes son: “WOA” **[15],** “A Browser-Based P2P Architecture for Collaborative End-User Artifacts in the Edge” **[16]**, “P2PSW” **[17]**, y “Robula+” **[18]**.

Examinaremos brevemente cada uno destacando su aporte a esta tesina.

3.1 WOA: Web Object Ambient

WOA propone una plataforma para recopilar contenido de la Web mediante objetos a partir de diferentes estrategias, a los cuales se les agrega comportamiento para que que los usuarios puedan interactuar con los mismos, bajo un ambiente en común, mejorando así la experiencia del usuario.

WOA propone dos roles para los usuarios: 1) usuario final y 2) desarrollador.

Como usuario final puede recopilar información existente en la Web asignarles un comportamiento e interactuar con los mismos. En cuanto al modo desarrollador, permite crear artefactos de comportamiento a través de decoradores y aplicaciones concretas del lado del cliente.

Conforme a este punto, el concepto de materialización de objetos a partir del contenido extraído en la Web es una característica importante que se adoptó en esta tesina junto con: 1) los pasos de identificación de los objetos de interés del usuario; 2) la abstracción de los mismo mediante semántica; 3) la creación de plantillas y finalmente; 4) las instancias de la misma. Esta tesina extiende lo ofrecido por WOA en dos líneas: a) incorpora un formato de tripletas, acorde a los lineamientos de la Web semántica; b) se basa de una arquitectura P2P para aportar robustez y flexibilidad.

3.2 A Browser-Based P2P Architecture for Collaborative End-User Artifacts in the Edge

Este trabajo propone una Arquitectura P2P con el fin de almacenar y compartir entre usuarios finales todo artefacto desarrollo por los mismos (End-user development - EUD) como por ejemplo extensiones Web, sin la necesidad de un servidor centralizado que actúe como intermediario, mejorando de esta manera la experiencia de usuario y la escalabilidad.

De otra manera todo artefacto generado tiene que ser sincronizado a través de un servidor para publicarlo en la nube o intranet que actúa como repositorio público. Esto genera un fuerte acoplamiento y dependencia con el servidor “back-end” y genera un cuello de botella para el avance y crecimiento de EUD, debido a la fuerte dependencia técnica y habilidades que tienen que tener los usuarios para compartir sus desarrollos.

Una arquitectura P2P permite la comunicación directa entre usuarios finales, permitiendo que cada usuario actúe como cliente o servidor en diferentes instantes de tiempo.

De manera semejante, esta tesina propone compartir las plantillas y las instancias creadas por los usuarios finales siguiendo la filosofía de la Web Semántica sin la necesidad ni intervención de un servidor central mediante una arquitectura P2P.

3.3 P2PSW: P2P Semantic Web

P2PSW (P2P Semantic Web) es una Wiki Semántica P2P, es decir una herramienta de edición colaborativa para crear, administrar y compartir anotaciones semánticas y ontologías. Asegura una colaboración masiva de manera distribuida replicando la semántica de las páginas Wiki y las anotaciones semánticas en todos los nodos. El incremento de los pares en la arquitectura P2P puede crecer sin afectar la escalabilidad, funcionamiento o tolerancia ante fallos, un nodo puede dinámicamente sumarse o abandonar la red P2P.

En cuanto a la persistencia cada nodo tiene una copia de la semántica de las páginas Wiki y existen dos formas de persistir la información: 1) el texto puede ser almacenado en archivos o base de datos; 2) las anotaciones semánticas son mapeadas a sentencias RDF en forma de ternas (sujeto, predicado, objeto) donde el sujeto siempre es el nombre de la página.

Las ternas almacenadas en RDF son almacenadas en un grafo el cual permite generar consultas del tipo SPARQL.

De manera semejante con lo anterior, esta tesina propone: 1) los usuarios pueden visualizar las plantillas propias y las creadas por todos los demás usuarios pares para poder generar una instancia; 2) cada usuario antes de crear una instancia a partir de una plantilla creada por un nodo par, se genera una copia de la plantilla asignando un nuevo ID, se copia localmente y luego si se genera la instancia referenciando al nuevo ID de plantilla; 3) las plantillas y las instancias se guardan como ternas con el formato n3 permitiendo de este modo la serialización (para el fin de esta tesina este formato es visible al usuario solo para ejemplificar el modelo RDF generado) 4) generar una copia de la plantilla creada por un nodo para asegurar la consistencia entre plantilla-instancia ante el caso por ejemplo que un par se desconecte o elimine la plantilla, dejando así instancias de otros pares con referencias a plantillas que ya se encuentran disponibles en la red P2P.

3.4 Robula+

Las técnicas de extracción de información Web basadas en plantillas como las que se proponen en esta tesis dependen de DOM locators. Los DOM locators son estrategias que facilitan y agilizan el acceso a todos los elementos DOM existentes en el sitio Web. Los DOM Locator son utilizados en el contexto de Javascript. Las técnicas más frecuentes para implementar DOM locators con el XPATH y JQuery Selectors.

Ahora veamos dos antecedentes referido a algoritmos y técnicas para generar un XPath robusto, es decir, poder identificar unívocamente a un elemento del DOM de un sitio Web mediante sus atributos, id, clase, posición dentro del árbol DOM, contenido, o combinación de los mismos, de manera tal que cuando el sitio Web modifique su estructura (DOM), el XPath identifica unívocamente al mismo elemento correctamente sin tener que modificarlo o reescribir.

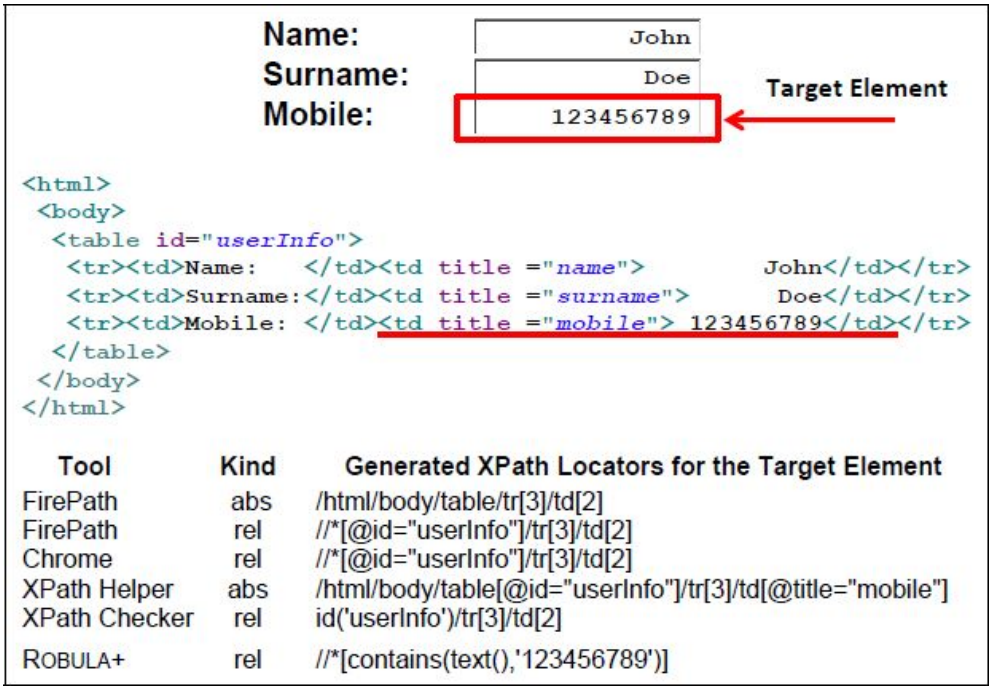
Es muy común que un sitio Web actualice cada cierto tiempo la estructura del DOM, ya sea para agregar funcionalidad, nuevo estilo, nueva presentación de la información, etc.

Es por eso que los siguientes antecedentes hacen referencia a la forma de generar un XPath robusto para los Test GUIs (Graphical User Interface), ya que estos son test automáticos para asegurar que los botones, iconos, animaciones, menús, funcionen correctamente ante posibles eventos generados por el usuario en el sitio Web.

Con el propósito de poder extraer el contenido de la Web para luego generar un modelo semántico, los XPath tienen un rol importante en esta tesina, es por eso que a continuación se analizan brevemente algunas técnicas y algoritmos.

Robula+ propone un algoritmo para generar DOM locators robustos. El algoritmo como primera instancia genera un XPath genérico, luego aplica siete transformaciones de refinamiento para lograr un mejor XPath. El algoritmo también clasifica de acuerdo a su solidez o a su fragilidad (lista negra) para identificar unívocamente al elemento DOM. Otra característica es que utiliza el texto contenido de los nodos, identifica múltiples atributos para generar los XPath.

Veamos ahora un ejemplo que propone Robula+ con respecto a otras herramientas para generar un XPath robusto. En la figura 12 podemos ver los XPath generados a partir de un formulario Web, luego en la figura 13 el DOM del documento HTML contiene modificaciones y evalúa cual de los algoritmos sigue funcionando correctamente.




```
<html>
 <body>
  <table id="userInfo">
   <tr><td>Name:    </td><td title ="name">         John</td></tr>
   <tr><td>Surname:</td><td title ="surname">       Doe</td></tr>
   <tr><td>Mobile: </td><td title ="mobile"> 123456789</td></tr>
  </table>
 </body>
</html>
```

Tool	Kind	Generated XPath Locators for the Target Element
FirePath	abs	/html/body/table/tr[3]/td[2]
FirePath	rel	//*[@id="userInfo"]/tr[3]/td[2]
Chrome	rel	//*[@id="userInfo"]/tr[3]/td[2]
XPath Helper	abs	/html/body/table[@id="userInfo"]/tr[3]/td[@title="mobile"]
XPath Checker	rel	id('userInfo')/tr[3]/td[2]
ROBULA+	rel	//*[contains(text(),'123456789')]

Figura 12

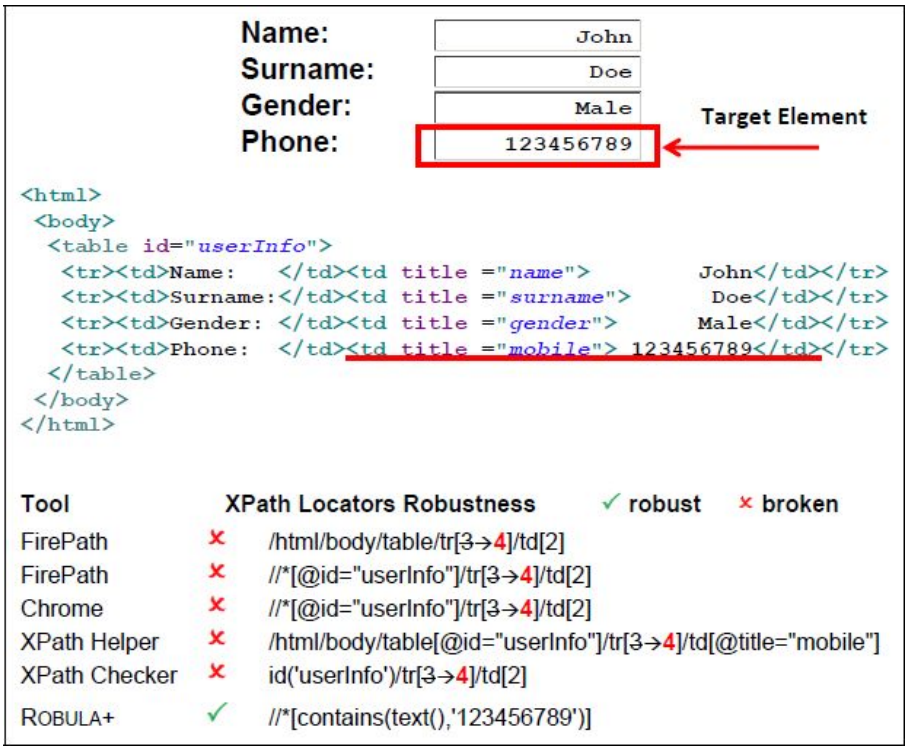


Figura 13

Hasta aquí Robula+ se presenta como una buena opción para generar un XPath robusto, la estrategia de transformaciones es una técnica que inspira a esta tesina para poder obtener un XPath y a partir de él extraer contenido de la Web.

Sin embargo, entre todas sus transformaciones existen algunas que no se pueden aplicar al momento de generar una Plantilla, el cual forma parte de la estrategia general de esta tesina. Algunas de ellas son:

Robula+ toma en cuenta el texto contenido en los elementos DOM, sin embargo esta característica no se aplica correctamente para el objetivo de esta tesina en el cual uno de sus pasos consiste en crear una Plantilla a partir de un sitio Web.

Es necesario mencionar que las plantillas representan una estructura DOM para de un dominio específico en el cual la estructura DOM no varía entre mismas plantillas del sitio Web, pero no así su contenido, el cual cambia completamente de acuerdo al recurso visitado.

Otro punto que debemos tener en cuenta es cuando los sitios Web generar el ID de un elemento del DOM concatenando el ID del recurso visitado, por ejemplo si el producto tiene el ID 80951 (Figura 14):

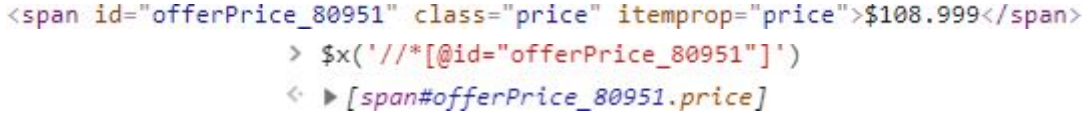


Figura 14: Encuentra el elemento DOM a partir del XPath

Si en la plantilla que estamos generando, agregamos una propiedad del tipo “price” y como XPath guardamos //*[@id="offerPrice_80951"], esta plantilla deja de funcionar para

otro recurso del mismo dominio Web pero con diferente ID, debido a que no podrá encontrar el elemento DOM (Figura 15).

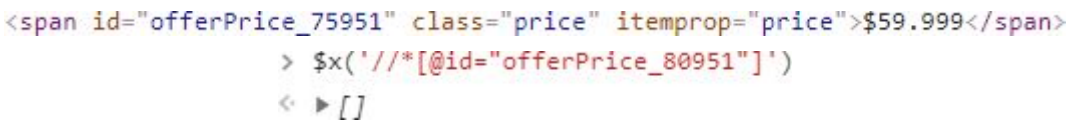


Figura 15: No encuentra el elemento DOM a partir del XPath

Capítulo 4

4 Estrategia General

Consideremos el caso donde un usuario experimentado y con conocimiento técnico extrae la información que considera relevante de un recurso en la Web, por ejemplo las características más importantes de diferentes dispositivos móviles a partir de un portal de ventas online. Una vez que extrae los datos del Sitio Web, genera un modelo semántico y los guarda localmente, este usuario se convierte en una fuente de información (nodo productor). A su vez existe otro usuario que necesita información relevante para decidir qué dispositivo móvil se adapta mejor a sus necesidades (nodo consumidor). En un escenario multi-usuario ambos usuarios pueden llegar a tener el mismo rol (productor y consumidor) cooperan y participan en una gran fuente de información, ninguno tiene más privilegio que otro, la información está distribuida y tiene que ser escalable debido a que los nuevos usuarios pueden sumarse cuando lo deseen. En una arquitectura Cliente-Servidor esto genera un gran cuello de botella en el servidor centralizado, obteniendo problemas de rendimiento en la red y de capacidad del servidor.

Como indican Steffen Staab y Heiner Stuckenschmidt **[4 (p. 6)]**, existe una fuerte asimetría entre los nodos de este espacio de direcciones que actúan como servidores de contenido y los nodos que actúan como clientes. Estimaciones recientes indican la presencia de 50 millones de servidores Web, pero hasta 150 millones de clientes. En la escala de la World Wide Web, cualquier forma de centralización crearía cuellos de botella inmediatos, en términos de rendimiento de la red y capacidad del servidor. Los autores explican de manera conceptual y técnica porque una arquitectura P2P se aplica de manera perfecta a la Web Semántica:

Por un lado la idea conceptual es:

"La necesidad de manejar múltiples fuentes de conocimiento e información es bastante obvia en el contexto de las aplicaciones de Web Semántica. En primer lugar, tenemos la dualidad de esquema y contenido de información donde múltiples fuentes de información pueden adherirse al mismo esquema. Además, la reutilización, extensión y combinación de varios archivos de esquema se considera una práctica común en la Web Semántica." **[4 (p. 3)].**

Igualmente pero de manera técnica,los autores Steffen Staab y Heiner Stuckenschmidt definen una series de razones por cual la arquitectura P2P se adapta muy bien a las necesidades de la Web Semántica:

"El enfoque comúnmente utilizado de utilizar una copia local de una fuente remota adolece del problema de cambiar la información. El uso directo de la fuente remota nos libera de la necesidad de gestionar el cambio, ya que siempre estamos trabajando con el original."**[4 (p. 3)].**

“Mantener diferentes fuentes separadas unas de otras nos proporciona una mayor flexibilidad en cuanto a la adición y eliminación de fuentes. En la configuración distribuida, solo tenemos que ajustar los parámetros del sistema correspondientes.”**[4 (p. 4)].**

Con el fin lograr el objetivo mencionado, en esta tesina se optó entonces por desarrollar una extensión Web que permite agregar funcionalidad a los Exploradores Web y enriquecer la experiencia del usuario. Los usuarios podrán definir plantillas de extracción, también podrán crear instancias a partir de las mismas extrayendo contenido de la Web, actualizar las instancias y finalmente compartirlas con otros navegadores Web mediante la arquitectura P2P.

La importancia de una arquitectura P2P en esta tesina consiste en que hace posible la comunicación de navegador Web a navegador Web para compartir datos, evitando así un cuello de botella en un servidor intermedio. Los usuarios pueden comunicarse directamente entre sí, esto implica que los usuarios pueden actuar como servidores o clientes en diferentes momentos, de acuerdo con las colaboraciones realizadas en un momento dado.

Una decisión importante para esta tesina es el formato semántico que se utiliza para persistir las plantillas y los objetos extraídos. A continuación se analiza algunos de los formatos existentes y se menciona cual es el elegido en esta tesina:

RDF es un modelo de dato abstracto, el estándar RDF en sí mismo no especifica su representación. La representación más popular y conocida es XML, pero su diseño no es amigable para los humanos (RDF/XML). Notation-3 es otro modelo más amigable que RDF/XML para el ojo del desarrollador pero tiene muchas características que no son necesarias para serializar un modelo RFD.

Otro formato es Turtle, el cual es mucho más legible para el desarrollador, permite el uso de prefijos y es compacto. Este formato es muy utilizado para generar grafos RDF siendo este un punto muy importante ya que permite generar consultas SPARQL (Estandarizado por la W3C en Enero del 2008) siendo este el componente más importante de la Web Semántica por que permite tratar a la Web como una gran base de datos semántica. Es por eso que el formato turtle es el elegido para persistir y compartir a las plantillas e instancias.

Capítulo 5

5 Modelo Semántico de plantillas e ítems extraídos

Con el fin de integrar nuestros datos extraídos a la Web Semántica necesitamos un modelo semántico el cual tiene que ser capaz de representar las plantillas generadas (a partir de ahora llamado scraper) como también los objetos extraídos a partir de una plantilla dada (a partir de ahora llamado instancia).

El modelo para el scraper tiene que ser capaz de asociar cada elemento DOM de la Web (del cual se van a extraer los datos para formar parte de una futura instancia) con su correspondiente semántica, además también tiene que ser capaz de modelar fecha de creación, nombre y la semántica del mismo, estos dos últimos definidos por el usuario.

Con respecto a la instancia, el modelo elegido tiene que tener la capacidad de representar toda la información extraída de la Web, su semántica y la referencia con un recurso en la Web para así enriquecer la Web Semántica.

Esta tesina propone entre otras cosas generar un modelo que luego será serializado a N-Triples y compartido mediantes diferentes usuarios en una arquitectura P2P. Los nuevos recursos generados van a respetar los estándar de la Web Semántica, por lo que será necesario generar URIs únicas para cada uno como se explicará más adelante.

5.1 Linked Data

Para poder extender y hacer crecer la Web Semántica, es necesario que el nuevo recurso se pueda identificar, esté disponible y accesible en la Web. Es aquí donde se presenta un problema, diferentes personas pueden hacer referencia al mismo recurso pero al momento de publicarlo en la Web cada uno asigna un nombre diferente. Esta pequeña diferencia se convierte en un gran problema porque puede ocurrir que un mismo nombre haga referencia a diferentes recursos. Esto genera que un nombre tenga significados distintos **[3]**.

En el año 2006 Berners-Lee conocido como padre de World Wide Web se refirió al término Linked Data como al conjunto de buenas prácticas para publicar e interconectar datos estructurados en la Web. **[19].**

El primer principio de Linked Data es que las URI's deben ser usadas como nombre de los recursos para que otros pueden referenciar, ya sean objetos del mundo real o conceptos abstractos.

Su segundo principio es utilizar el protocolo HTTP para que cualquier usuario pueda acceder al recurso mediante el Explorador Web (http://URI) igual que cualquier otro recurso Web. **[20]**.

Por todos estos puntos mencionados en esta tesina, los nuevos recursos que se generen para compartir mediante una arquitectura P2P tendrán como nombre un HASH y su forma de encontrarlos y compartirlos en la WEB será:

http://ivancolman.com.ar/resource/HASH_DEL_RECURSO

Se utiliza el dominio nombrado anteriormente a modo de ejemplo.

5.2 DOM Locators

Esta sección tiene como fin explicar brevemente como se implementó un XPath locator con el fin de poder extraer datos desde la Web y así también cubrir los puntos a mejorar que se consideraron al hablar de Robula+.

La implementación de una estrategia para generar un XPath es un punto importante porque al momento de generar una plantilla se agregan propiedades, cada propiedad tiene una semántica y un XPath para poder extraer su valor del elemento DOM; ahora bien, esta plantilla tiene que ser capaz de extraer el valor de las propiedades para diferentes URL del mismo dominio (por ejemplo extraer información de diferentes productos del mismo portal Web de ventas).

En este trabajo se presentan varias estrategias las cuales tienen prioridad para generar un XPath robusto, siempre tratando de referencias directamente al atributo “id” o partir desde un “id” para evitar que ante una actualización del sitio Web el DOM cambie y por consecuencia el XPath deje de funcionar.

Como estrategia de mayor prioridad es la de referenciar directamente al atributo “id” del elemento si es que lo tiene. De esta forma se presenta dos posibles escenarios, “id” estático (Figura 16) o “id” dinámico (Figura 17)

```
<span id="offerPrice" class="price" itemprop="price">$59.999</span>
```

Figura 16: atributo id estático

```
<span id="offerPrice_80951" class="price" itemprop="price">$108.999</span>
```

Figura 17: atributo id dinamico

El atributo con “id” estático significa que su valor varía dentro del Sitio Web entre diferentes recursos. Con lo cual una plantilla puede extraer el valor del elemento DOM sin complicaciones entre diferentes recursos del mismo dominio (Figura 18).

```
> $x('(//*[@id,"offerPrice"])')
< ▶ [span#offerPrice.price]
```

Figura 18

Ahora bien, una dificultad que se presentó es la de atributo con “id” dinámico, donde el valor del atributo “id” varía según el recurso.

En este trabajo lo que se hizo fue identificar los atributos con “id” dinámicos y generar los XPath de forma diferente a los “id” estáticos, es decir se toma el valor del atributo “id” sin el valor numérico que representa al ID del recurso.

También debemos mencionar que esta técnica no es suficiente, ya que encontramos en un mismo sitios referencias a diferentes recursos, por cual esta valor parcial del atributo “id” no identifica unívocamente a un elemento, es por eso que se debe agregar al XPath a cual de todos los elementos hace referencia. Por ejemplo en la Figura 19 se identifica a los

elementos con id parcial “offerPrice_” y que sea el primero “[1]” dentro del árbol DOM del documento HTML.

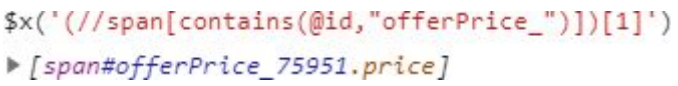


Figura 19

En este punto tal vez el lector se pregunte porque se busca crear como primera opción un XPath en relación al id del elemento DOM. Esto se debe a que es una manera de indexar a un elemento dentro del árbol DOM de un sitio, evitando armar todo el path completo el cual puede verse afectado a futuro por modificaciones de diseño, funcionalidad o estética del sitio Web. Todo el DOM previo al elemento puede ser modificado y nuestro XPath seguirá funcionando correctamente.

Consideremos ahora el caso donde el elemento DOM no contiene un atributo “id”, en este caso la estrategia es buscar el primer nodo antecesor que si contenga un atributo “id” y a partir de este generar el XPath hasta el elemento seleccionado a extraer información Figura 20.

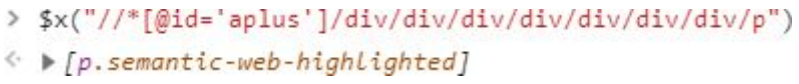


Figura 20

Ahora veamos la estrategia con menor prioridad, la cual genera un XPath completo desde el nodo root del documento DOM hasta el nodo seleccionado. Esta estrategia sirve para identificar al elemento y poder extraer información a partir de él, pero es probablemente el que deje de funcionar ante un cambio en el documento DOM. Figura 21.

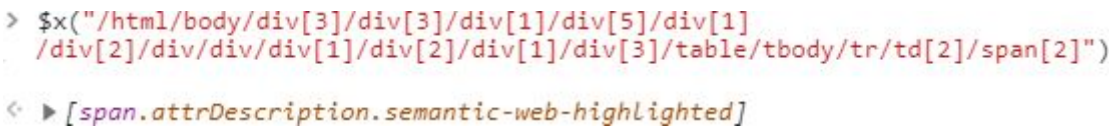


Figura 21

5.3 Scraping Ontology Specification (Scrappy)

Teniendo en cuenta la necesidad del modelo Semántico para esta tesina, la ontología encontrada que más se acerca a lo mencionado es Scraping Ontology Specification **[21]**.

Semantic Scraping define el mapeo entre los datos de la Web y los recursos de la Web Semántica, es decir un mapeo entre la información contenida en los documentos HTML y un modelo Semántico basado en RDF.

La figura 22 muestra la Ontología de Scrappy, si bien esta ontología cubre en gran parte la necesidad de nuestro modelo, en color amarillo (Figura 22) se marca la extensión que se hizo sobre Scrappy para cubrir totalmente la necesidad de esta tesina. En la clase Scraper se agrega nombre y una fecha de creación ambas representadas por las ontologias dcterms:http://purl.org/dc/terms/ y por dbo:http://dbpedia.org/ontology/.

A continuación se describe brevemente Scrappy junto con los elementos más importantes que serán usados en esta tesina.

Dentro de la Figura 22 encontramos las **clases** (Círculos), las **propiedades** (flecha) y su **valores literales** (rectángulo). Dentro de las clases más importantes se encuentran:

- **Scraper**: Es el agente capaz de extraer datos desde los documentos HTML.
- **Fragment**: Hace referencia a cualquier elemento DOM o inclusive el sitio Web completo.
- **Selector**: Se utilizan para identificar un fragment del documento Web. Las clases más importantes que extienden de Selector son: **UriSelector** que asocia un recurso en la Web. **XPathSelector** hace referencia a una expresión xpath para un Fragment específico.
- **Mapping**: Es el mapeo entre un fragment y un recurso RDF o un blanck node.

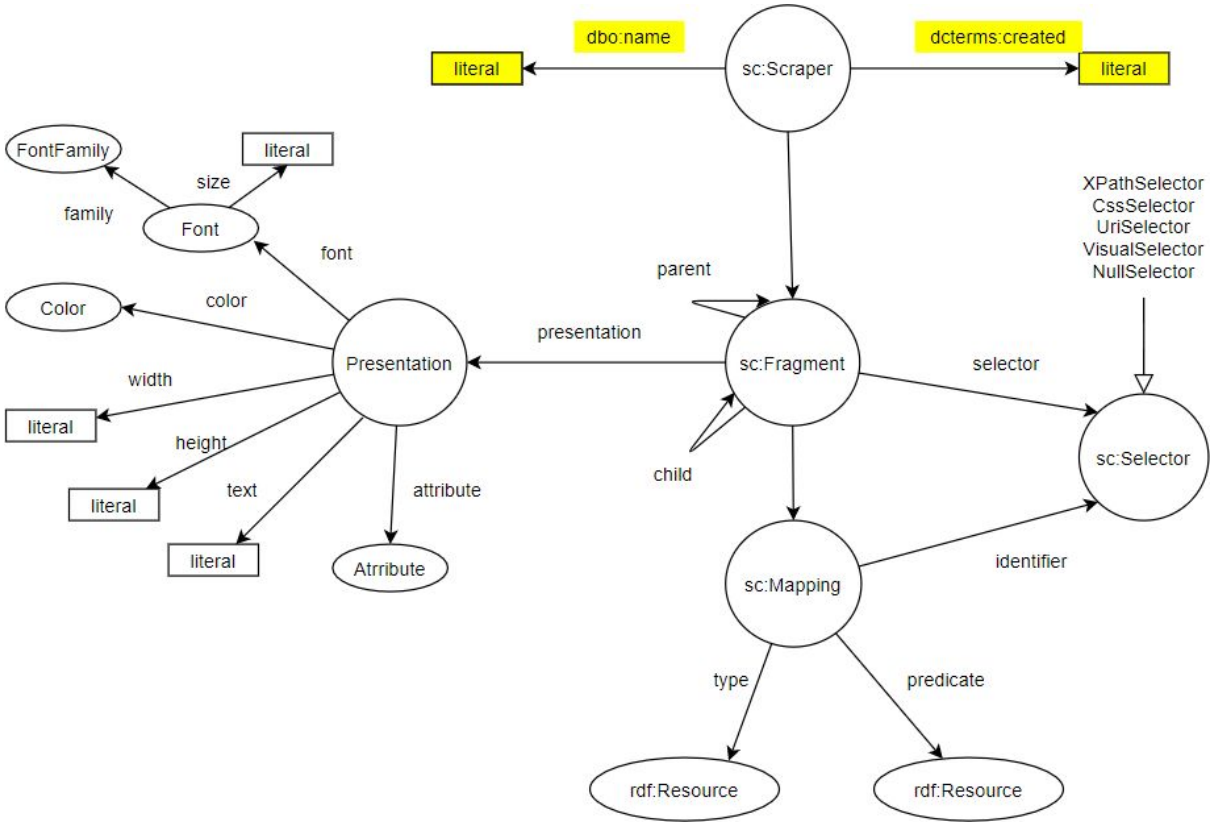


Figura 22: Ontología de Scrappy y su extensión en amarillo

Llegados a este punto, estamos en condiciones de mostrar un ejemplo sencillo de una instancia de scraper. El contexto del ejemplo es una tienda Web de celulares del cual se extraen datos para un celular en particular. Figura 23.

Los prefijos de las ontologías en los ejemplos utilizados son:

- @prefix dbo: <About: http://dbpedia.org/ontology/>.
- @prefix rdf: <https://www.w3.org/1999/02/22-rdf-syntax-ns#>.
- @prefix dcterms: <http://purl.org/dc/terms/>.
- @prefix sc: <http://gsi.dit.upm.es/ontologies/scraping/#>.
- @prefix ic: <https://www.ivancolman.com/resource/>.

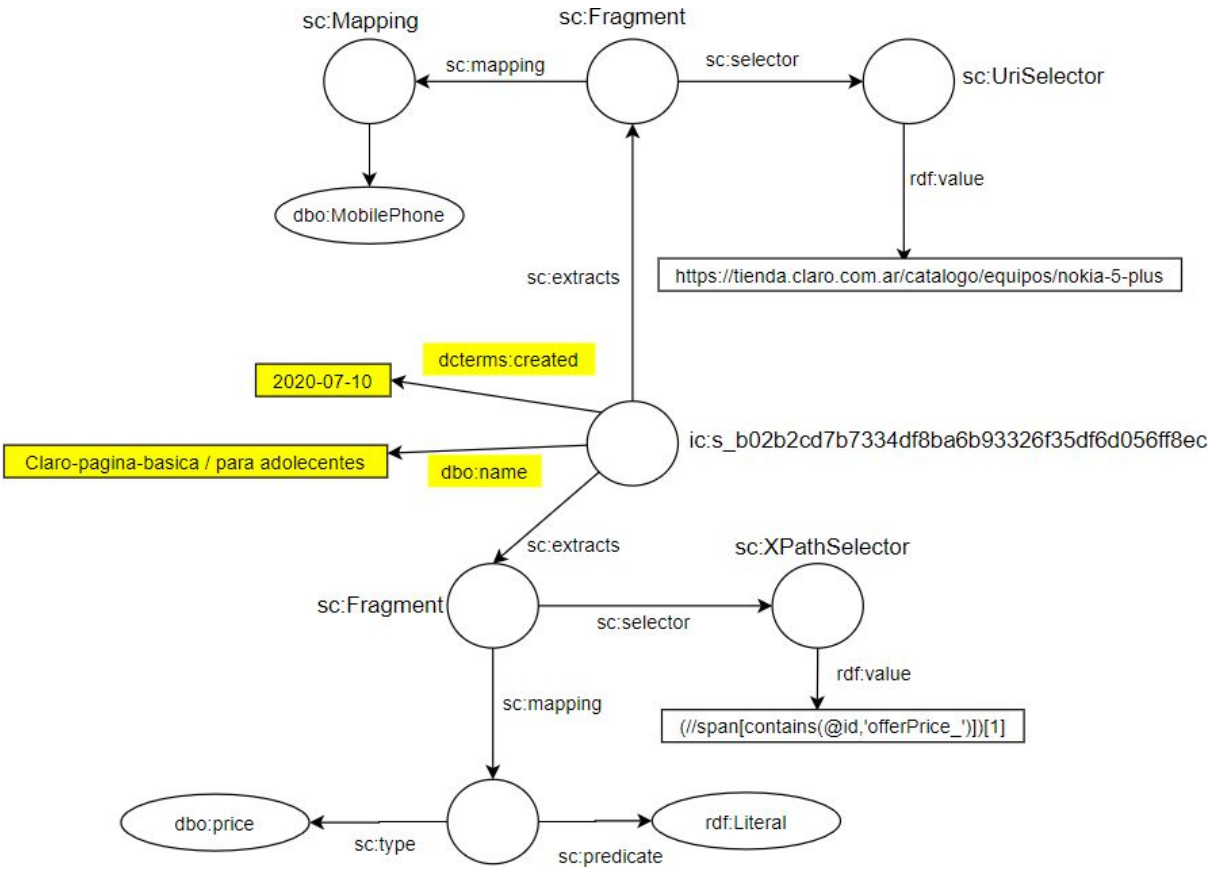


Figura 23: instancia del modelo de scraper

Desarrollando el ejemplo, encontramos: un scraper y dos fragment's.

- El scraper tiene su propio ID, en este caso es un hash (ic:s_b02b2cd7b7334df8ba6b93326f35df6d056ff8ec). El mismo se traduce a http://ivancolman.com.ar/resource/s_b02b2cd7b7334df8ba6b93326f35df6d056ff8ec lo cual lo hace un recurso unico para ser compartido en la arquitectura P2P que propone este tesina.
- El fragment superior hace referencia al sitio Web completo, el mismo mapea con un recurso RDF (dbo:MobilePhone) y tiene un selector que identifica a un recurso en la Web (https://.../nokia-5-plus).
- El Fragment inferior hace referencia a un elemento del DOM (para el ejemplo solo se extrae el precio). El mismo mapea con un modelo RDF (dbo:price) e identifica al elemento DOM mediante una expresión xpath mediante la clase XPathSelector.

Continuando con los ejemplos, en la figura 24 encontramos ejemplos de instancias del modelo extraído para diferentes sitios Web de tienda de celulares.

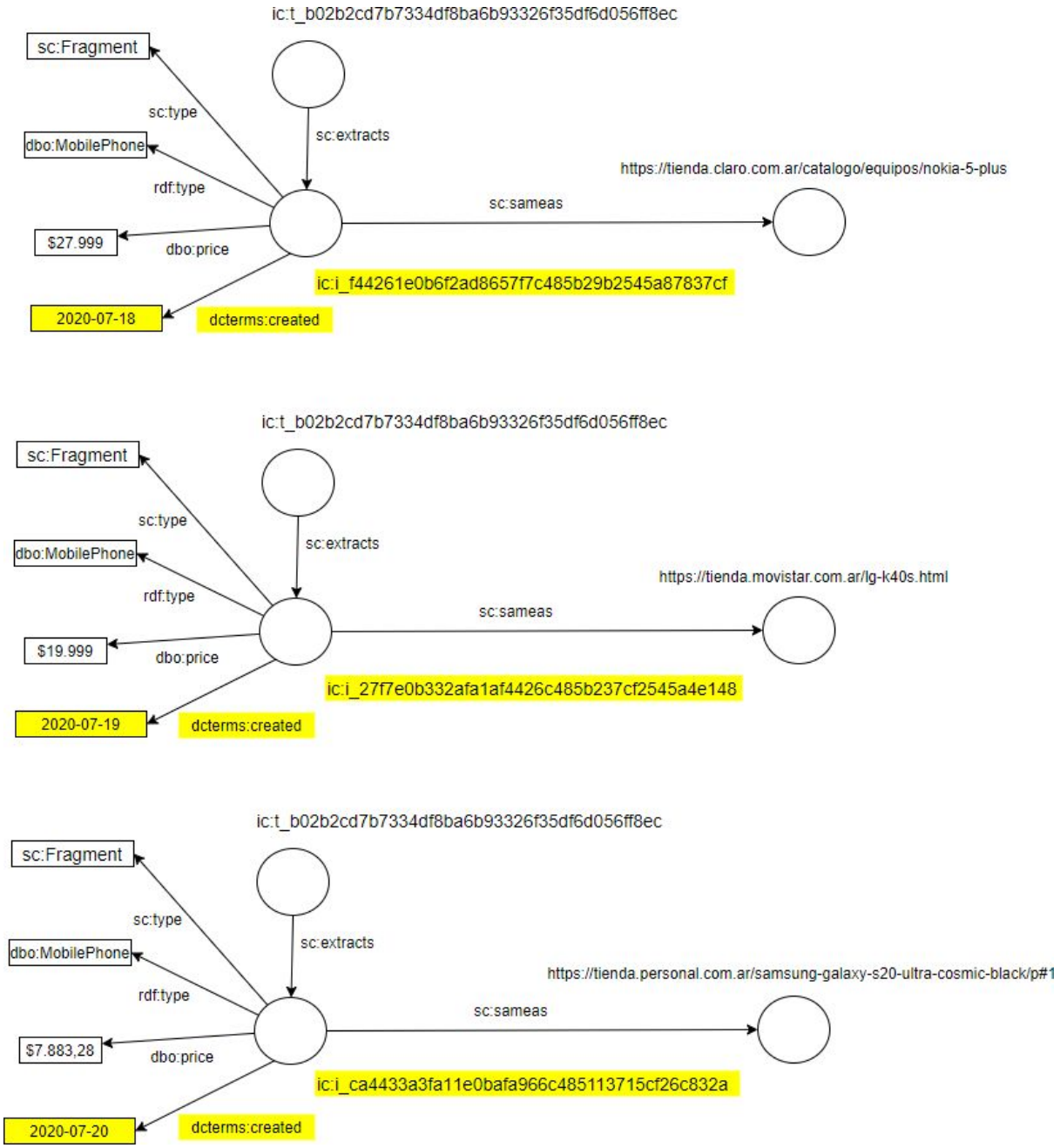


Figura 24: Ejemplo de una instancia extraída

Capítulo 6

6 Interacción P2P

A continuación, se describe el flujo de información entre pares en cada uno de los casos de uso de la extensión Web propuesta por esta tesina: (1) consulta y creación de una plantilla; (2) creación de una instancia; (3) consulta de instancias. Comenzamos con la creación de una plantilla. El capítulo 7, Caso de estudio, complementa al presente con mayores detalles y ejemplos concretos.

6.1 Consulta y creación de plantillas

Inicialmente el usuario accede a la extensión para ver el listado de plantillas existentes en la red P2P correspondientes al sitio Web al cual accedió (Figura 25), en ese momento el peer inicial solicita a todos los nodos de la red P2P las plantillas guardadas localmente correspondientes al sitio Web.

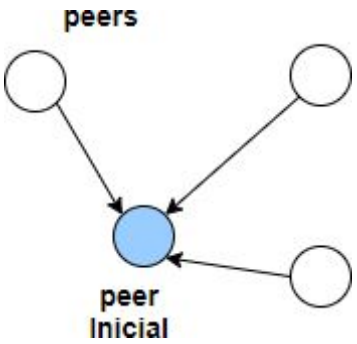


Figura 25: Solicitud de plantillas existentes en la red P2P correspondientes a un sitio Web.

Si el usuario decide crear una nueva plantilla, la misma será almacenada localmente. La misma será distribuida a los pares, solo cuando ellos lo soliciten (Figura 26).

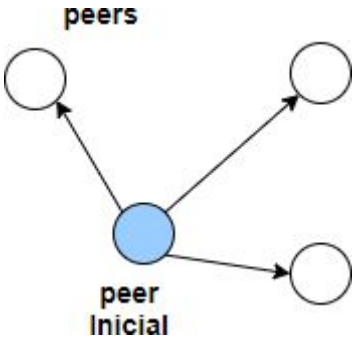


Figura 26: La nueva plantilla está disponible en la red P2P.

6.2 Creación y consulta de instancias

Continuando con el ejemplo anterior, analizamos ahora la creación de una instancia. Ahora otro peer obtiene todas las plantillas existentes en la red P2P, incluyendo la creada por el peer inicial. Visualiza por cada una la información que extrae cada plantilla y crea una instancia a partir de alguna (Figura 27).

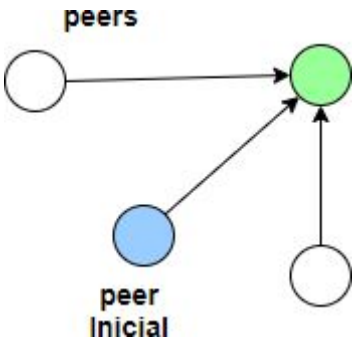


Figura 27

Con respecto a las consultas de las instancias, existe un criterio de búsqueda llamado “Ubicación”, el mismo puede tomar los valores “local” (Figura 28.a), “remoto” (Figura 28.b), “todos” (Figura 28.c). Es decir: “local” visualiza las instancias guardadas localmente en el peer, “remoto” busca y visualiza sólo las instancias generadas por otros pares, “todos” es la suma de los dos criterios anteriores.

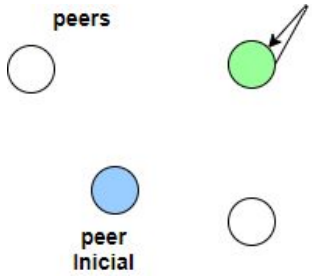


Figura 28.a: Local

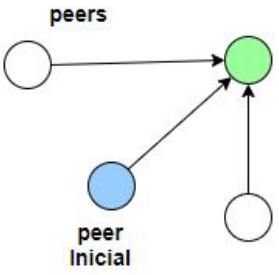


Figura 28.b: Remoto

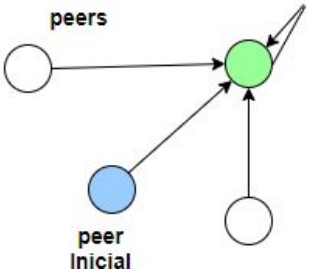


Figura 28.c: Todos

6.3 Comunicación P2P

Para implementar la funcionalidad de comunicación P2P se hace uso de la extensión Web desarrollada por Rodolfo Atilio Gonzales en su tesina **[22]**, la cual funciona como middleware y hace posible la conexión P2P entre diferentes usuarios, generando así una red de pares en la Web, para poder compartir el modelo semántico generado en esta tesina.

La extensión se agrega al explorador Web aumentando así su funcionalidad (Figura 29).

Figura 29: Extensión P2P

Para poder hacer uso de la misma, la extensión Web que propone esta tesina tiene que hacer uso del archivo “conectos.js”, heredar de la clase propuesta por el middleware e implementar dos métodos necesarios para la comunicación entre pares (Figura 30).

```
...
import AbstractP2PExtensionBackground from './conector'
...
export default class P2PExtension extends  AbstractP2PExtensionBackground{
    constructor(){
        super()
        this.setExtensionName("semantic_web");
        this.setExtensionId(browser.runtime.id);
    }
    // OVERRIDE
    automaticProcessing(msg , peer){
        //..
    }

    receiveResponse(msg, peer){
        //..
    }
}
```

Figura 30: configuraciones necesarias e implementaciones de métodos necesarios.

En este caso “semantic_Web” hace referencia al valor del campo “name” en el archivo manifest.json del proyecto práctico de esta tesina (Figura 31).

Figura 31: parte del archivo manifest.json

Capítulo 7

7 Evaluación - Caso de estudio

En esta sección se describen las experiencias realizadas con las extensiones Web las cuales permiten: (1) generar una plantilla; (2) compartirla entre los pares mediante una arquitectura P2P; (3) instanciar la plantilla logrando así extraer los datos de la Web; (4) compartir la instancia entre los pares para mejorar la toma de decisiones.

La experiencia se realizó entre dos exploradores Web: Chrome y Firefox, logrando así que la experiencia sea Cross-Browser, de la misma forma podría realizarse con Opera o Microsoft Edge.

En relación con los pasos pautados para realización de la evaluación se encuentran los siguientes:

1. El usuario “A” comienza creando una plantilla a partir de un portal online de ventas de dispositivos móviles.
2. Luego, el usuario “B” desde una terminal y explorador distinto al usuario “A” obtiene todas las plantillas existentes en la red P2P.
3. El usuario “B” explora las plantillas obtenidas.
4. Una vez conforme con una plantilla, el usuario “B” genera una instancia de la misma.
5. Ambos usuarios están en condiciones de obtener todas las instancias existentes, ya sea propias o las que pertenecen a la red P2P siendo las mismas presentadas en formato turtle.

A continuación se analiza la evaluación y se ejecuta los pasos planteados anteriormente.

7.1 Ejecución de la Evaluación

Como primer paso para realizar la evaluación es instalar el extención del middleware P2P (MDP2P) en el Explorador Web el cual funciona como una capa de abstracción para todas aquellas extensiones Web que quieran comunicarse y hacer uso de una arquitectura P2P, luego se necesita instalar la extensión que propone esta tesina (Figura 32).

Figura 32: Extensiones Web instaladas, middleware P2P y Web Semántica en Google Chrome.

Consideremos ahora el menú principal de la extensión, el cual inicialmente consta de 3 opciones: (1) plantillas: Lista todas las plantillas de la red P2P diferenciando las propias y las del resto de la red, también permite crear una nueva plantilla; (2) Extractor: Lista en forma de “carrusel” todas plantillas existentes, marcando todos los elementos DOM del cual

se van a extraer los datos en tal caso de instanciar la plantilla; (3) Repositorio: Permite buscar instancias en toda la red P2P, contiene diferentes criterios de búsqueda. (Figura 33)

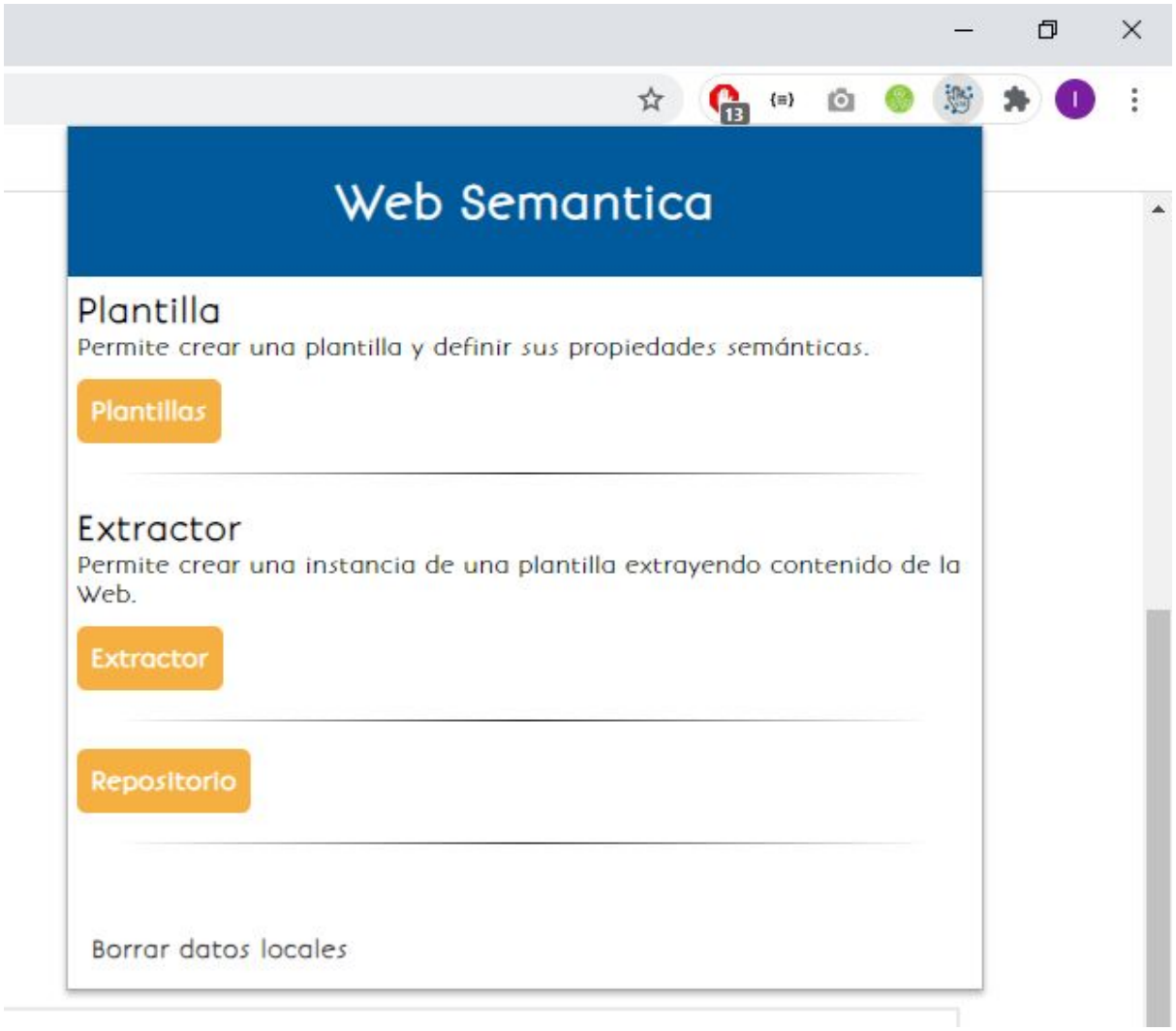


Figura 33: menú principal de la extensión.

7.1.2 Plantillas

Inicialmente no existen plantillas en la red P2P es por eso que el usuario A crea una plantilla asignando un nombre, una semántica, y sus propiedades (Figura 34).

La semántica es un control HTML del tipo autocomplete el cual al tipear genera los parámetros para realizar un GET al servicio de dbpedia (http://dbpedia.org/sparql), el cual permite buscar clases (owl:Class) o propiedades (rdf:Property), en el ejemplo de la Figura 34 la solicitud es la siguiente:

http://dbpedia.org/sparql?default-graph-uri=http://dbpedia.org&query=SELECT%20*%20WHERE%20%7B%20%3Fresource%20a%20owl%3AClass.%20%3Fresource%20rdfs%3Alabel%20%3Flabel.%20FILTER%20(%20contains(%3Flabel%2C%20%22phone%22)%20%26%26%20contains(str(%3Fresource)%2C%20%22http%3A%2F%2Fdbpedia.org%2Fontolo

gy%22)).%20%7D%20LIMIT%2050&format=application%2Fsparql-results%2Bjson&CXML_redir_for_subjs=121&CXML_redir_for_hrefs=&timeout=30000&debug=on

Analizando la URL encontramos los siguientes parámetros:

- default-graph-uri: http://dbpedia.org
- query:
  SELECT *
  WHERE {
  ?resource a '**owl:Class**'.
  ?resource rdfs:label ?label.
  FILTER (contains(?label, "**phone**") && contains(str(?resource), "http://dbpedia.org/ontology")).
  }
  LIMIT 50
- format: application%2Fsparql-results%2Bjson
- CXML_redir_for_subjs: 121
- timeout: 3000
- debug: on

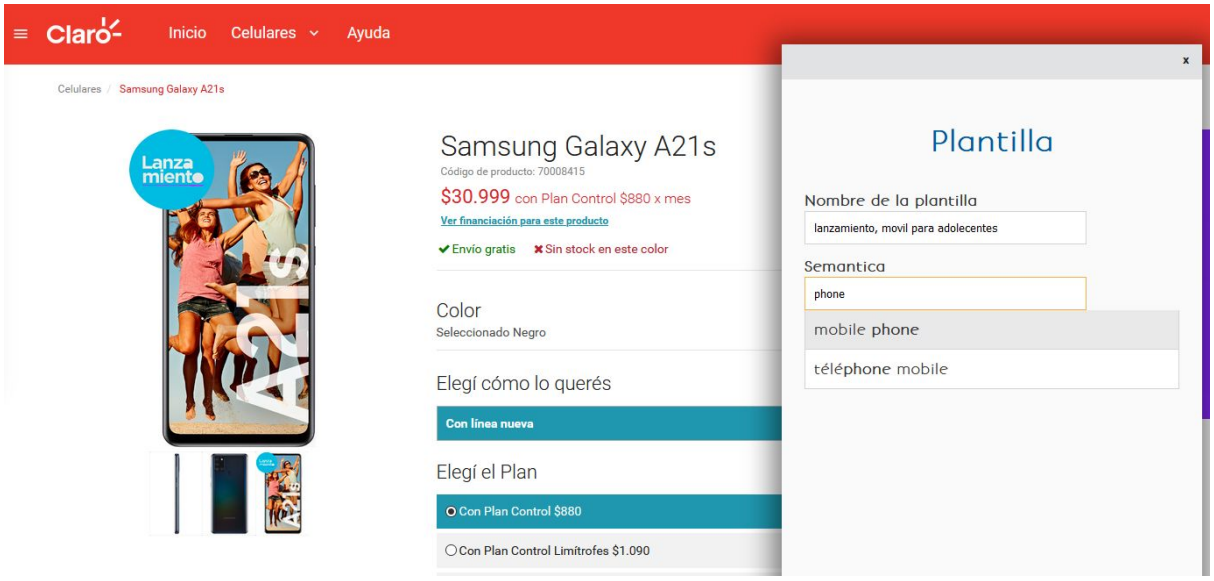


Figura 34

Una vez definida la semántica se definen las propiedades que forman parte de la plantilla, es decir el usuario selecciona una semántica del autocomplete y hace click sobre el elemento DOM del cual se extrae el dato y se asigna como valor a esa propiedad para las futuras instancias de la plantilla (Figura 35).

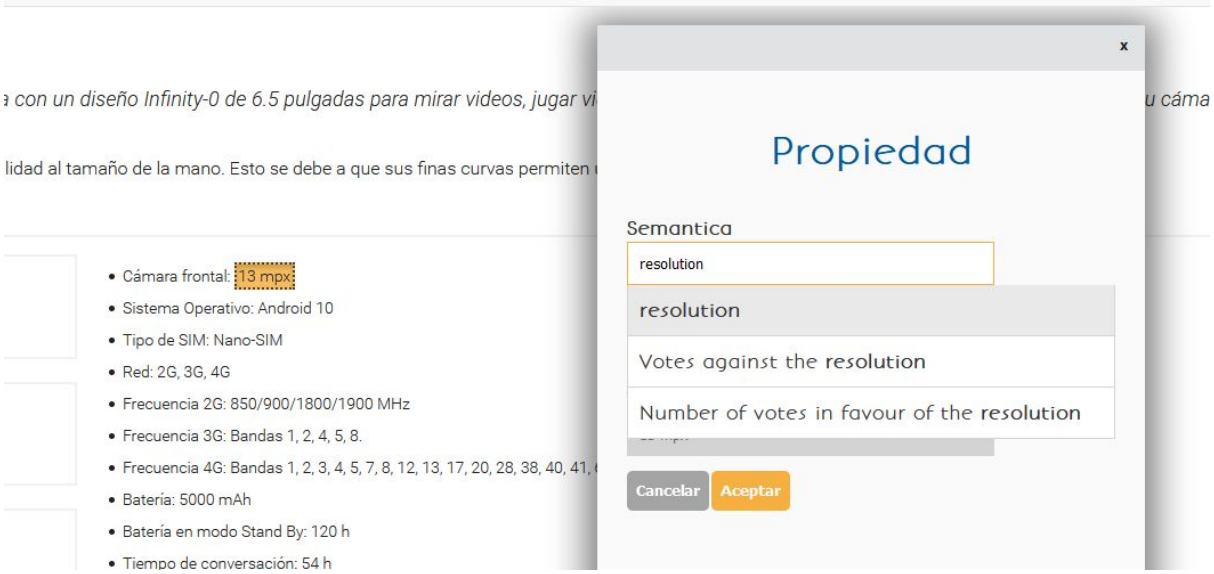


Figura 35

Antes de avanzar en el Wizard, se listan las propiedades existentes en forma de listado (Figura 36).

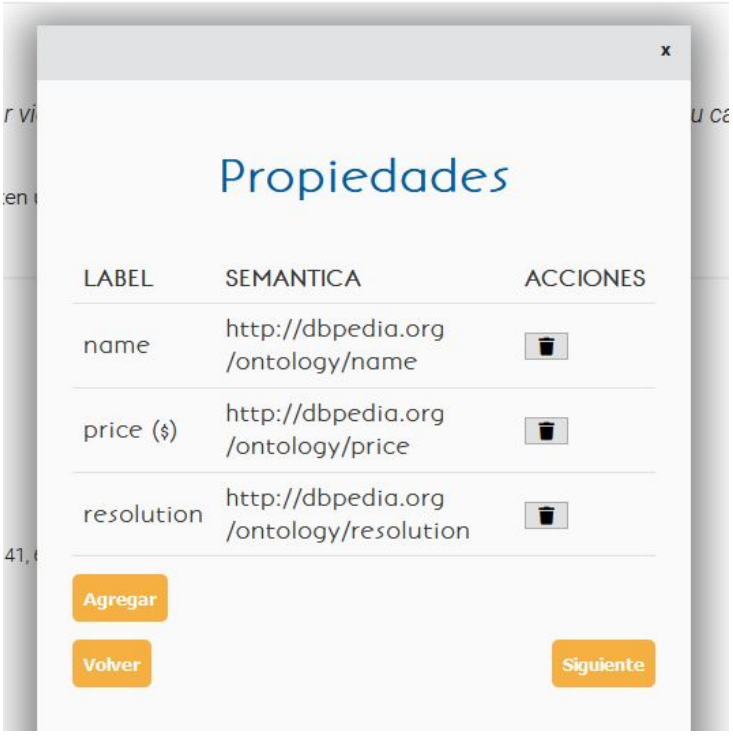


Figura 36

El último paso consiste en un breve resumen con toda la información de la plantilla, es decir: nombre, semántica y el listado de propiedades la cual consiste en: (1) label: Label es el nombre legible para el humano; (2) URL representa a la propiedad en la Web.

Además se resaltan los elementos DOM contenedores de los valores de las propiedades creadas (Figura 37).

Como último paso resta guardar la plantilla y la misma estará disponible para todos los pares de la red P2P.

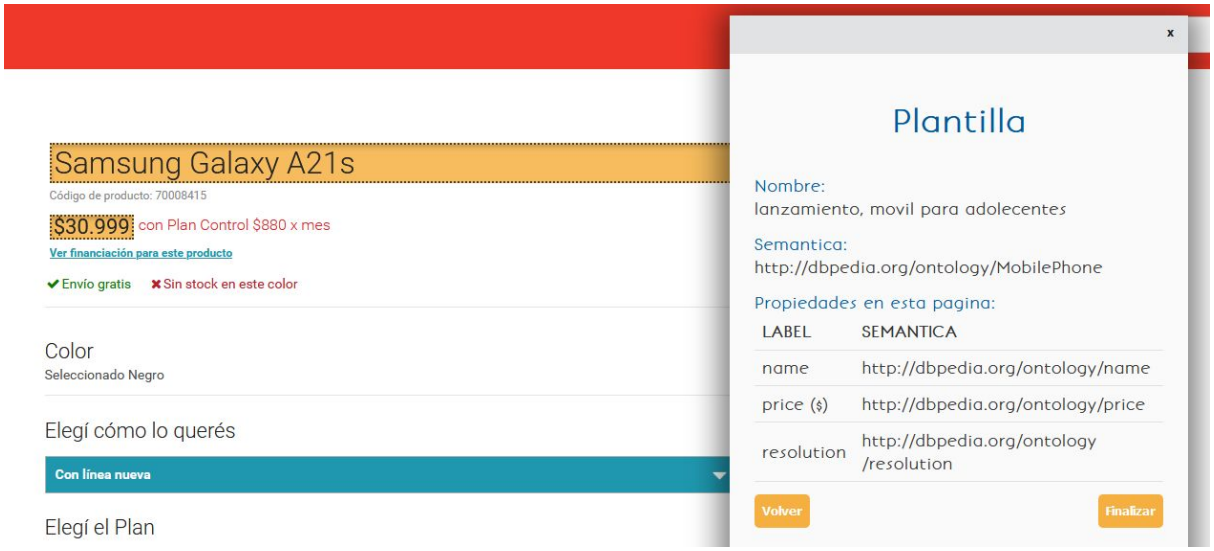


Figura 37

Al Finalizar se listan todas las plantillas existentes en la red P2P, las propias (local) o las que se encuentran localmente en otros pares (remotas). Figura 38.

Figura 38

7.1.3 Instancias

El siguiente escenario ocurre cuando el usuario B, el cual es un par en la red P2P distinto al par del usuario A, crea una instancia a partir de la plantilla creada por A.

Como primer paso, el usuario B obtiene todas las plantillas de los demás pares (Figura 39). La plantilla generadas por los demás pares se visualizan en forma de “carrusel”, es decir, permite avanzar o retroceder para ver los detalles de cada una de las plantillas. Como detalle de cada una encontramos: semántica, nombre, fecha de creación, una ubicación (propia o remoto), también se indica si ya existe una instancia local de la plantilla (Guardado), fecha de extracción si es que ya existe la instancia localmente y luego en forma de tabla se listan todas las propiedades junto a su valor, el cual corresponde a los datos que se van a extraer del sitio visitado.

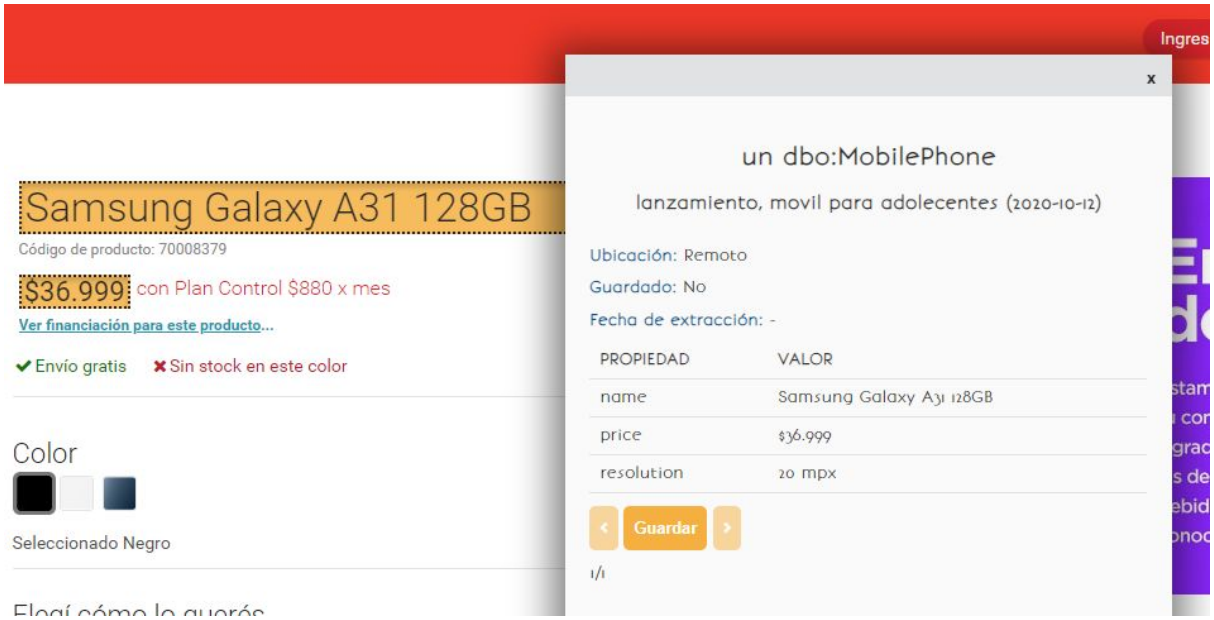


Figura 39

Por cada plantilla el usuario tiene la posibilidad de crear una instancia mediante el botón “Guardar” (figura 39). Una vez creada la instancia, la misma es guardada localmente y está disponible para todos los pares de la red P2P.

7.1.4 Repositorio

La última evaluación es la búsqueda de instancias. La vista de repositorio realizar una búsqueda de todas las instancias que cumplen con los criterios aplicados con los filtros de búsqueda. Figura 40.

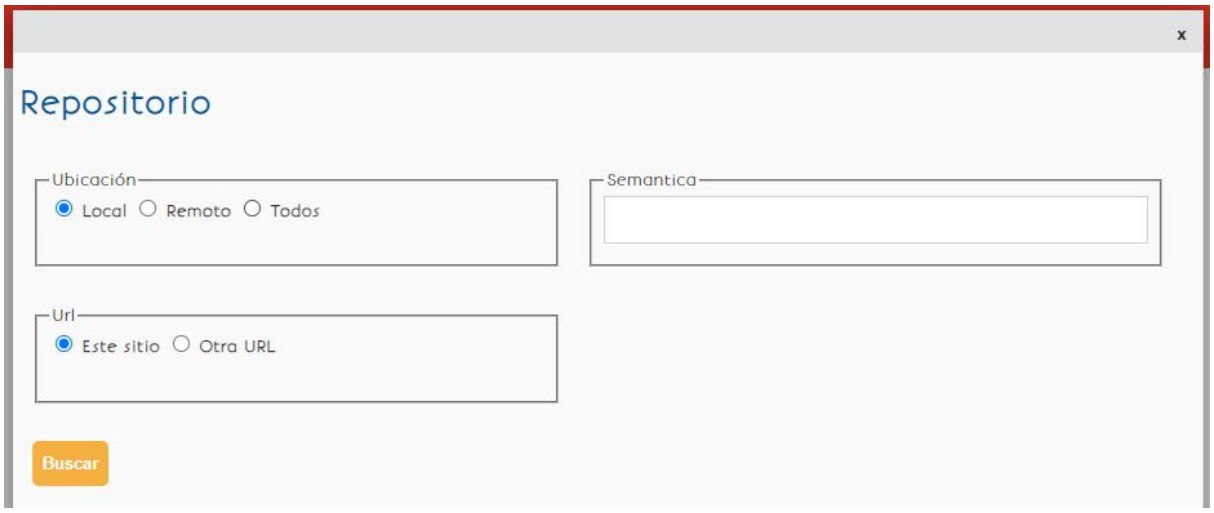


Figura 40

Los criterios de búsqueda pueden ser: instancias locales al usuario (Local) o las existentes en los demás pares de la red (Remoto) o ambas (Todos). Otro criterio de búsqueda es una semántica específica de la plantilla y finalmente buscar las instancias para un dominio específico o para el dominio actual que visita el usuario.

El resultado de la búsqueda anterior es el carrusel con todas las instancias que fueron encontradas según los filtros de búsqueda aplicados. También indica la cantidad de instancias que cumplen con los filtros aplicados. Para que la instancia sea legible al desarrollador de aplicaciones se utiliza el formato turtle para su visualización. Figura 41.

Figura 41

Capítulo 8

8 Conclusiones

El presente trabajo propone mejorar la etapa de recolección de información en la toma de decisiones de los usuarios en la Web, como por ejemplo la compra de algún dispositivo móvil en algún portal online. Para el mismo se presenta una extensión Web Cross-Browser, el cual permite a los usuarios enriquecer la Web, crear plantillas e instancias mediante las técnicas soportadas por la Web Semántica y luego se comparten mediante una Arquitectura P2P. Todo nuevo recurso generado por los usuarios como así también la semánticas son nombradas mediante una URI logrando así respetar la filosofía de Linked Data, permitiendo modelar los mismo con RDF.

La extensión Web permite agregar funcionalidad a la Web que los usuarios visitan, es por eso que un usuario experimentado puede generar una plantilla remarcando según su criterio, los más importante del producto, asigna un nombre, una semántica y sus propiedades. Las propiedades de la plantilla están conformadas por una URI semántica y un XPath el cual sirve para extraer los datos de la Web.

Luego, otro usuario que necesita tomar una decisión utiliza la extensión Web para comunicarse con otros pares y poder obtener las plantillas remotas correspondientes al sitio Web que visitó. El mismo examina la plantilla, observa cuales son las características más relevantes que señaló otro usuario experimentado, y si entiende que se aplica a su búsqueda crea la instancia.

Como hemos visto los grafos de RDF permiten modelar los datos y sus relaciones por cual esta es una forma de compartir la información que generan los usuarios mediante la extracción de contenido en la Web a la gran base de conocimiento de la Web Semántica.

Continuemos el análisis remarcando los principales desafíos que se presentaron en el desarrollo de este trabajo.

En primer lugar se considera la comunicación en la arquitectura P2P. El presente trabajo hace uso de una extensión Web que actúa como middleware para la comunicación P2P, el cual facilitó la comunicación y tiempo de trabajo. Con solo importar una clase y sobreescribir algunos métodos el trabajo de la comunicación P2P fue resuelto para este trabajo.

Como segundo desafío se nombra la capacidad de las extensiones de ser Cross-Browser, es decir que sean compatibles en los exploradores Web como Firefox, Chrome, Opera, Edge. Esto se logró mediante el uso de la librería JS “Webextension-polyfill”.

En cuanto a las limitaciones, podemos enumerar las siguientes:

Al momento de crear plantillas e instancias se generan nuevos recursos, se necesita que estos recursos se puedan nombrar e identificar unívocamente en la Web para así cumplir la filosofía de Linked Data y poder aplicar las técnicas que propone RDF y OWL. Es decir, se necesitaba una nueva URI por cada nuevo recurso generado en este trabajo. Por este motivo se propone generar un Hash por cada nuevo recurso que forma parte de la URI. Este punto puede mejorarse utilizando alguna mejor técnica de Hash y permitiendo que los nuevos recursos si se encuentren disponibles en la Web.

Como segundo punto se considera la edición de las plantillas. Cuando un sitio Web actualiza su estructura DOM en el documento HTML ya sea por nuevas funcionalidades, presentación, las propiedades de las plantillas quedan desactualizadas. Si bien en este trabajo se marca en rojo estas inconsistencias, a futuro se podría pensar en una estrategia para actualizar la plantilla y replicar los datos en toda la red P2P.

Capítulo 9

9 Trabajo a futuro

En la sección de Evaluación se describieron los pasos a seguir para crear una plantilla mediante la extensión Web propuesta, en el cual se asigna un nombre, una semántica y sus propiedades. Luego otro usuario o el mismo usuario, puede generar una instancia de la misma y por consecuencia generar un grafo RDF y dejarlo disponible en la red para todos los demás usuarios. De esta forma se aportan los datos extraídos en la Web a la gran base de conocimiento que es la Web Semántica.

El potencial problema que puede ocurrir es la creación de Spam o Fake Data (a partir de ahora definimos con las siglas SFDG, Spam Fake Data Graph) que se genera de forma intencional con el fin de influenciar a los usuarios en la toma de decisiones.

Es decir se generan relaciones de datos falsos con datos existentes en la Web Semántica (Figura 42).

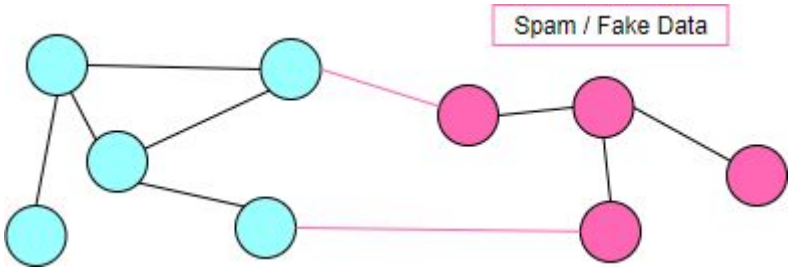


Figura 42

En particular este tipo de datos (SFDG) tienen las siguientes características: (1) son de bajo costo, es decir, solo hace falta armar las relaciones de la información del tipo SFDG con la información existente en la Web y generar el grafo RDF. (2) rápida difusión: una vez generada la instancia se replica de manera inmediata en toda la red P2P, es decir todos los pares ya acceden a ella. (3) atenta contra la credibilidad de la Web Semántica.

Si bien la evaluación de la tesina es aplicada sobre portales de venta online, el impacto que puede generar SFDG no solo es en la toma de decisiones de los usuarios en algún portal online de ventas, también puede impactar en la opinión pública, orden social, credibilidad, etc.

Con todo lo expresado anteriormente estamos en condiciones de proponer una extensión a esta tesina.

Como extensión a este trabajo se propone:

- Definir o encontrar una estrategia que permita identificar los Spam y Fake Data.
- Definir o encontrar una estrategia que permita evitar la generación de SFDG y su replicaciones en la red P2P.
- Definir una ontología que permite contar con nuevos predicados para representar al Spam o Fake Data.

Bibliografía

[1] Internet World Stats, "Internet Growth Statistics". https://www.internetworldstats.com/emarketing.htm, Recuperado en Febrero 2021.

[2] Pew Research Center, "Generational differences in online activities". https://www.pewresearch.org/internet/2009/01/28/generational-differences-in-online-activities , Recuperado en Febrero 2021.

[3] Liyang Yu (2011). A Developer's Guide to the Semantic Web.

[4] Steffen Staab, Heiner Stuckenschmidt (2006). Peer-to-Peer and Semantic Web.

[5] The PageRank Citation Ranking: Bringing Order to the Web (1998), page 5.

[6] Mathieu d'Aquin, Li Ding, Enrico Motta (2011). Semantic Web Search Engines.

[7] Berners-Lee (2001), Scientific American article "The Semantic Web".

[8] W3 ORG., "Resource Description Framework (RDF) Model and Syntax Specification". https://www.w3.org/TR/1999/REC-rdf-syntax-19990222 ,Recuperado en Febrero 2021.

[9] W3 ORG., "OWL Web Ontology Language Use Cases and Requirements". https://www.w3.org/TR/2004/REC-Webont-req-20040210/ , Recuperado en Febrero 2021

[10] W3 ORG., "RDF Schema 1.1". https://www.w3.org/TR/rdf-schema , Recuperado en Febrero 2021.

[11] W3 ORG., "OWL Web Ontology Language Overview". https://www.w3.org/TR/2004/REC-owl-features-20040210 , Recuperado en Febrero 2021.

[12] W3 ORG., "RDF 1.1 Concepts and Abstract Syntax". https://www.w3.org/TR/2014/REC-rdf11-concepts-20140225 ,Recuperado en Febrero 2021.

[13] IETF., "Internationalized Resource Identifiers (IRIs)". https://www.ietf.org/rfc/rfc3987.txt , Recuperado en Febrero 2021.

[14] Xuemin Shen, Heather Yu, John Buford (2010). Handbook of Peer-to-Peer Networking.

[15] Firmenich, S. D.; et. al. Web Objects Ambient: an integrated platform supporting new kinds of personal Web experiences.

[16] Rodolfo Gonzalez, Sergio Firmenich, Gustavo Rossi, A Browser-based P2P architecture for collaborative enduser artifacts in the edge.

[17] Hameurlain, A. (Ed), Küng, J. (Ed), Wagner, R. (Ed), Decker, H. (Ed) (2017). A Second Generation of Peer-to-Peer Semantic Wikis. In Transactions on Large-Scale Data- and Knowledge-Centered Systems XXXIV.

[18] Maurizio Leotta, Andrea Stocco, Filippo Ricca, Paolo Tonella. Robula+: An Algorithm for Generating Robust XPath Locators for Web Testing. 2017.

[19] W3 ORG., "Linked Data". https://www.w3.org/DesignIssues/LinkedData.html , Recuperado en Febrero 2021.

[20] Tom Heath, Christian Bizer. Linked Data Evolving the Web into a Global Data Space.

[21] GSI UPM, "Scraping Ontology Specification". http://gsi.dit.upm.es/ontologies/scraping/1.0/index.html , Recuperado en Febrero 2021.
[22] Rodolfo Atilio Gonzalez. Colaboración P2P para ambientes de desarrollo de usuario final basados en navegadores Web. 2020.