Instituto Politécnico Nacional
Centro de Investigación en Computación

Integración semántica de datos geoespaciales utilizando una ontología de aplicación

T E S I S

QUE PARA OBTENER EL GRADO DE

MAESTRA EN CIENCIAS DE LA COMPUTACIÓN

P R E S E N T A

ING. MARISOL DE JESÚS PAREDES REYES

DIRECTOR DE TESIS

DR. MIGUEL JESÚS TORRES RUIZ

México, D.F., Febrero de 2014

INSTITUTO POLITÉCNICO NACIONAL

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

CARTA CESIÓN DE DERECHOS

En la Ciudad de México, D.F. el día 21 del mes de Marzo del año 2013, el (la) que suscribe Marisol de Jesús Paredes Reyes alumno(a) del Programa de Maestría en Ciencias de la Computación, con número de registro B020900, adscrito(a) al Centro de Investigación en Computación, manifiesto(a) que es el (la) autor(a) intelectual del presente trabajo de Tesis bajo la dirección del (de la, de los) Dr. Miguel Jesús Torres Ruiz y cede los derechos del trabajo titulado Integración Semántica de Datos Geoespaciales utilizando una ontología de aplicación, al Instituto Politécnico Nacional para su difusión, con fines académicos y de investigación.

Los usuarios de la información no deben reproducir el contenido textual, gráficas o datos del trabajo sin el permiso expreso del (de la) autor(a) y/o director(es) del trabajo. Este puede ser obtenido escribiendo a las siguientes direcciones marisol.paredes@gmail.com. Si el permiso se otorga, el usuario deberá dar el agradecimiento correspondiente y citar la fuente del mismo.

Marisol de Jesús Paredes Reyes

Nombre y firma del alumno(a)

Resumen

El presente trabajo de tesis se encuentra enfocado en el desarrollo de un sistema Web que hace uso de una ontología de aplicación para organizar e integrar información del dominio turístico a partir de fuentes de datos heterogéneas. Dicha ontología está escrita en OWL, basando su construcción en el diccionario de datos turísticos del INEGI para su diseño y jerarquización.

El caso de estudio para este sistema se encuentra orientado al sector hotelero de Campeche, con un modelo persistente de datos relativo a diversas entidades turísticas de la ciudad. Dicha información puede ser accesada para su consulta desde Internet, a través de una aplicación Web híbrida (mash-up), la cual combina la interfaz de búsqueda del usuario con los servicios de Google Maps para la visualización de los datos resultantes, tanto descriptivos como geoespaciales. El módulo de búsqueda propuesto para el sistema está basado en SPARQL, lenguaje de consultas para ontologías, de manera que permite resolver consultas avanzadas o complejas, tales como "ubicación de hoteles con restaurante y bar en Campeche", por ejemplo.

Para la construcción de los diferentes módulos componentes del sistema se hizo uso de tecnologías estándares y de libre distribución casi en su totalidad, exceptuando la parte tocante a uno de los formatos de la información de origen, los archivos tipo *shapefile,* ya que, a pesar de ser propietario de ESRI y de distribución comercial, es uno de los estándares más populares en el mercado dada su simplicidad tanto en su creación como en su administración.

Finalmente, es imprescindible destacar la importancia de las ontologías como línea de investigación, ya que encierran un gran potencial no sólo en lo que se refiere a interoperabilidad e integración, sino también en la generación de nuevo conocimiento a partir de los datos de origen, más allá de su sola descripción.

Abstract

The present thesis is focused on the development of a Web system, which uses an application ontology to organize and integrate information in the touristic domain from heterogeneous data sources. The ontology is written in OWL, and its construction is based upon the INEGI dictionary of touristic data for the ontology's hierarchy and design.

The case study for this application is oriented towards the hotel sector at Campeche, with a persistent data model relating to a diversity of the city's touristic entities. Such information can be queried from the Internet through a hybrid Web application (mash-up), which combines the user query interface with the Google Maps services for visualizing the descriptive and geospatial data results. The proposed search module for this system is based on SPARQL, which is a query language for ontologies that provides the facility to solve advanced queries such as “location of hotels with bar and restaurant in Campeche”, for example.

For the development of diferent modules that compose the system, standard and free distribution tecnologies have been used; the only exception is made with the *shapefile* format, as a data source for the system; despite the fact it is ESRI's proprietary, it was chosen because of its creation and management simplicity, and thus one of the most popular standards in the market nowadays.

Finally, it is important to remark the significance of ontologies as a line of investigation, because of the many potentialities, not only on semantic interoperability and data integration, but also in the generation of brand new knowledge derived from the original data provided, beyond information's sole description.

Índice

Índice de figuras

Índice de tablas

Capítulo 1. Introducción

Capítulo 1. Introducción

En este capítulo se presenta la introducción al tema y se definen los objetivos del presente trabajo de tesis, así como sus alcances, justificación y estructura del documento.

1.1 Antecedentes

Gracias a la generalización del uso de las computadoras personales y al crecimiento de las tecnologías Web, hoy en día contamos con acceso a enormes y variadas fuentes de información, lo cual se refleja en todos los ámbitos de la actividad humana.

Una de las herramientas más utilizadas actualmente vía Web son los Sistemas de Información Geográfica (SIGs) [1], dado que sus características les permiten abarcar un amplio espectro en diversas actividades de los sectores económicos y productivos de nuestro país.

Los SIGs constituyen una herramienta indispensable en campos que conjugan los datos de localizaciones geográficas con la consulta de información concerniente a dicha localización; tal es el caso de áreas de importancia como salud pública, turismo, ecología, mercadotecnia, planeación urbana, entre muchas más.

Actualmente podemos encontrar diversas aplicaciones Web que nos proporcionan herramientas de búsqueda tales como consultas de información combinadas con el manejo de localizaciones puntuales sobre mapas digitales, lo cual conlleva a un amplio rango de aplicaciones potenciales en este sentido. Algunos ejemplos de estos sistemas son guiaroji.com [2], geonames.org [3] y Google Maps [4]. En este sentido, los Mashups o aplicaciones Web híbridas [5] proporcionan la capacidad de combinar el contenido de más de una fuente de datos en un conjunto integrado, ya que el contenido de los Mashups proviene de una interfaz de programación o Servicio Web [6], de manera que se ha hecho posible el incluir servicios de Google Maps, Amazon, Youtube, Yahoo, etc., en aplicaciones creadas con propósitos específicos.

Sin embargo, un problema en cuanto al uso de sitios integrados de esta manera es la enorme cantidad de información que sus motores de búsqueda deben manejar, lo cual ha originado que cada vez sea más complejo el tratamiento de la misma y más difícil la obtención de resultados adecuados o manejables para el usuario final, ya que no sólo se trata de una cantidad importante de datos sino que también está la cuestión de la diversidad del origen de los mismos.

En este sentido, las investigaciones sobre *integración* de datos provenientes de fuentes heterogéneas para su almacenamiento y recuperación han cobrado gran relevancia durante los últimos años. Una forma de abordar este problema es mediante el uso de *ontologías*, las cuales modelan formalmente un dominio de la realidad, entendiéndose por dominio un conjunto de objetos que se relacionan entre sí [7]. Una ontología proporciona así mismo un vocabulario común entre dichos objetos, entendible y que puede ser leído por una máquina. De esta manera, es posible obtener un modelo de datos que abarca una amplia descripción semántica del dominio de la información que se desee, garantizando que el tratamiento de dicha información arrojará resultados confiables, eficientes y optimizados.

Una ontología puede ser visualizada como el sitio donde diversas fuentes de datos pueden converger para una organización completa, detallada, clasificada y jerarquizada; ésta es una de las muchas ventajas que observa la representación de datos, a través de ontologías y que pretende explotarse en el presente trabajo de tesis a través de una *aplicación Web híbrida orientada a servicios turísticos, con representación y recuperación de datos basados en una ontología*.

Actualmente, el turismo es una industria altamente competitiva a nivel nacional e internacional. De acuerdo con la Organización Mundial del Turismo, las llegadas de turistas a diferentes puntos del planeta aumentarán en un 200% para el año 2020 [8]. Se sabe que Internet es hoy la principal manera de localizar destinos turísticos [9], y en este sentido las tecnologías Web han facilitado el desarrollo de portales turísticos motivando el crecimiento de áreas de estudio como el e-Turismo (Turismo Electrónico), el cual consiste en aplicar las tecnologías de la información en la industria turística [10].

Cabe resaltar que en la elaboración de una aplicación orientada al e-Turismo confluye una cantidad considerable de información, proveniente no sólo de la infraestructura turística creada por el hombre, como son hoteles, museos, centros nocturnos, etc., sino además existen muchas otras áreas que pueden ser de interés para el turista, como gastronomía, tradiciones, cartelera cultural de la localidad, medios de transporte, actividades disponibles, en fin, podría nombrarse un área de interés por cada turista en potencia. La buena noticia es que actualmente cualquier número de ontologías pueden ser integradas para formar una base de conocimientos que represente adecuadamente la magnitud del sector turístico, no sólo en nuestro país sino también a nivel internacional. Hoy en día, existen investigaciones encaminadas a este respecto [11], de manera que el presente trabajo de tesis pretende representar un aporte en este sentido.

Ante la diversidad de procedimientos que explotan la utilización de diversas fuentes de datos, en los últimos años se han propuesto diversos mecanismos que permiten integrar la información de fuentes heterogéneas [12], de tal manera que sea posible resolver peticiones mediante un componente integral. Una ontología proporciona tanto las características como los estándares necesarios para lograr dicha integración, además de contar con la ventaja de poder aplicar inferencias en el conocimiento almacenado de esta manera a través de razonadores creados para este propósito; es decir, generar nuevo conocimiento a partir de la información depositada en la ontología.

Por tanto, ésta es precisamente una característica que se explota en la presente tesis, al trabajar sobre datos integrados a partir de diferentes orígenes, como sobre información generada a partir de los mismos; para la generación de dicha información se hace uso de un razonador, especificando reglas de inferencia para la organización de la ontología con enunciados descritos en lógica de primer orden para que puedan ser interpretados por el razonador.

Como puede observarse, éstos son los principios en los que se basa la *Web Semántica* [13] y, a pesar de que el ideal de una Web completamente integrada semánticamente se vislumbra aún lejano, es el sentido hacia el cual van encaminados los esfuerzos de muchas investigaciones actualmente alrededor del mundo.

1.2 Objetivos

En cuanto a los objetivos que delimitan el alcance del presente trabajo de tesis, encontramos los siguientes:

Objetivo general

- Proporcionar una herramienta accesible desde la Web que recupere información geográfica y descriptiva de diversas fuentes en un contexto turístico, integre y organice los datos a través de una ontología de aplicación, genere un modelo persistente de datos sobre el cual se realice la búsqueda solicitada y devuelva los resultados a través de la interfaz Web, incluyendo la localización geográfica por medio de un mash-up (Google Maps).

Objetivos particulares

- Desarrollar una aplicación híbrida accesible desde la Web para la recuperación y visualización de datos geográficos y descriptivos según los requerimientos del usuario final, utilizando una ontología de aplicación para la integración y organización de los mismos.

- Diseñar y construir una ontología de aplicación para la representación y descripción de los datos geográficos del problema planteado con el caso de estudio, de manera que se promueva la reutilización de las estructuras generadas para ampliaciones posteriores del mismo o de otros sistemas.

- Generar un modelo persistente de datos a través de la ontología para la organización, la integración, el manejo y la recuperación de los datos geográficos y descriptivos requeridos por el usuario, provenientes de fuentes heterogéneas (instancias ontológicas y shapefiles).

- El sistema será implementado en un contexto turístico, tomando como ejemplo funcional la Ciudad y Puerto de San Francisco de Campeche; cabe señalar que, en esta ciudad, el turismo representa la principal fuente de ingresos para sus habitantes.

1.3 Descripción del problema

Como se ha mencionado anteriormente, el Internet es actualmente el medio principal por el cual se localiza información turística alrededor del mundo. Existen diferentes sitios en la Web ampliamente utilizados para este fin; sin embargo, se trata de sistemas comerciales independientes entre sí, cuyas consultas se basan en diferentes fuentes de datos, proporcionando al usuario resultados más o menos satisfactorios pero sin una visión unificada del servicio turístico.

Para muchos usuarios del turismo electrónico, existe el ideal de contar con sistemas que combinen en tiempo real diferentes fuentes de información para realizar planeación de viajes, de tal manera que sea posible resolver consultas de manera óptima, completa y que realmente responda a las necesidades planteadas en la consulta; es aquí donde las ontologías juegan un papel preponderante, al modelar una representación y clasificación de datos que abarque una amplia descripción semántica de la información turística, para un área geográfica determinada, valiéndose de muchas fuentes de información complementarias a la vez.

El presente trabajo está enfocado en el desarrollo de un sistema Web orientado al sector hotelero. En esta propuesta se hace uso de una ontología de aplicación que describe el dominio turístico acorde a la ciudad de Campeche, basada en el diccionario de datos turísticos del INEGI [14]. Se plantea igualmente el uso de una aplicación Web híbrida para explotar la información de dicha ontología, trabajando con los servicios de Google Maps para la visualización de los resultados. Como característica adicional, el modelo persistente de datos construido a través de la ontología toma como fuentes tanto las instancias existentes en la ontología como registros provenientes de archivos tipo shapefile, enfocando el trabajo desarrollado hacia la integración y representación semántica de datos en la Web.

1.4 Justificación

El nacimiento de la *Red de Redes* como tal fue el resultado de una evolución que desde entonces no se ha detenido un solo momento. Dicha evolución corresponde a la búsqueda de los individuos por plasmar de la mejor manera posible el rasgo más característico de la humanidad en esta forma de comunicación que es la Web: el lenguaje.

Sabemos que el objetivo principal de una red es la de compartir recursos, y el recurso más importante en Internet es la información. En la búsqueda de lograr un lenguaje compartido individuo-máquina-individuo, que fuera comprendido, analizado y extendido por nuestro indispensable intermediario, hemos visto surgir fenómenos conceptualizados como la Web 1.0 [15], que nos maravilla con la capacidad de empoderamiento que proporciona al individuo común a través de la información al alcance de la mano; posteriormente, la Web 2.0 [15], con enfoques colaborativos, redes sociales y representaciones más detalladas de la información gracias al XML [16] y todos los estándares derivados. Asimismo, en este momento como el principal objetivo de muchas líneas de investigación, la Web Semántica, que pretende describir, organizar e integrar todos los dominios de interés humano, de manera que puedan ser compartidos, ampliados y reutilizados, con la ventaja añadida de que la representación semántica de los datos no sólo pueda ser comprendida por una máquina, sino que nueva información puede ser inferida a partir de la existente sin necesidad de la intervención humana.

Al analizar las implicaciones de este paradigma es posible apreciar las ventajas que representan las ontologías como herramientas de organización del conocimiento en todos los ámbitos del quehacer humano. Actualmente existen iniciativas en varios campos importantes tales como la medicina, finanzas, matemáticas, entre otros, para la estandarización de los conceptos descritos en cada ontología, así como de su utilidad práctica y la colaboración entre las mismas [17].

Añadido a todo lo anterior, encontramos que una de las características más importantes de toda ontología es la organización de la información, de tal manera que funcione como una

herramienta para la integración de fuentes de datos heterogéneas para búsquedas o consultas por campo de conocimiento.

Considerando lo anterior, en la presente tesis se propone el uso de un sistema Web capaz de recuperar datos geográficos y descriptivos de un sitio de interés en el contexto turístico, utilizando una ontología de aplicación para llevar a cabo el proceso de integración, organización y recuperación de la información a través de fuentes de datos heterogéneas, obteniendo de esta manera el máximo beneficio posible de las principales características de una ontología.

1.5 Aportaciones

Dentro de las aportaciones que el presente trabajo de tesis ofrece a la línea de investigación seleccionada se encuentran:

- Construcción de una ontología de aplicación en el dominio turístico.
- Integración de fuentes de datos heterogéneas a través de una ontología.
- Generación de información nueva a partir de conocimiento *a priori,* mediante el uso de un razonador y reglas de inferencia.
- Una metodología de integración semántica del dominio turístico, conducida por una ontología de aplicación e inferencia.
- Aplicación de la metodología en un caso de estudio basado en un sistema *web-mapping* del orden turístico.

Cada una de las aportaciones consideradas en este apartado representa una oportunidad para el establecimiento de líneas de investigación afines a lo realizado hasta el momento en el presente trabajo.

1.6 Organización de la tesis

La presente tesis se encuentra organizada de la siguiente manera:

Capítulo 1. Introducción: antecedentes, objetivos, descripción del problema, justificación y organización del presente trabajo.

Capítulo 2. Estado del arte: sistemas Web de contexto turístico y/o basados en ontologías; trabajos relacionados.

Capítulo 3. Metodología propuesta: Análisis y Diseño de un Sistema de Representación de Información Geoespacial en la Web basado en ontologías; selección de un caso de estudio del mundo real: localización de sitios turísticos en la ciudad de San Francisco de Campeche.

Capítulo 4. Resultados experimentales: Implementación del Sistema Web híbrido, ontología de aplicación y modelo persistente de datos, así como módulo para la recuperación de instancias en shapefiles y módulo mediador para la integración de los mismos como individuos de la ontología.

Capítulo 5. Conclusiones y trabajos a futuro.

Referencias. Presenta las fuentes bibliográficas utilizadas en este trabajo.

Anexos. Engloban la representación gráfica de la ontología de aplicación, así como el código fuente del módulo mediador.

Capítulo 2. Estado del Arte

Capítulo 2. Estado del arte

En este capítulo se presentan los antecedentes y definiciones necesarios para la comprensión de este trabajo de tesis, así como algunos trabajos relacionados con la temática de la misma.

Como se mencionó en el capítulo anterior, el objetivo principal de este trabajo de tesis es proporcionar un sistema accesible desde la Web que recupere información geográfica y descriptiva de fuentes heterogéneas en un contexto turístico, que integre y organice la información a través de una ontología de aplicación, genere un modelo persistente de datos sobre el cual se realicen las consultas solicitadas y devuelva los resultados a través de la interfaz Web, incluyendo la localización geográfica por medio de un mash-up. Es a partir de esta descripción que podemos enumerar las herramientas, recursos y procesos a utilizar a lo largo del presente trabajo.

2.1 Evolución de la Web

En sus inicios, la Web fue diseñada como una forma simple de acceder a documentos situados en diversos lugares sin tener que preocuparse por permisos de acceso o formatos. El HTML permitía esta independencia de formatos, y la Web se construyó a partir de documentos estáticos que contenían información mezclada con formatos de estilo.

Escribir y preparar páginas estáticas puede ser costoso, mucho más si el origen de esas páginas es dinámico y cambia constantemente. Para evitar ese trabajo de constante actualización comenzaron a surgir aplicaciones que generaban los HTML de forma dinámica. De este modo, surgieron los CGI y posteriormente los lenguajes basados en *script*, que permitían generar dinámicamente las páginas Web a partir de las bases de datos, y a la inversa, introduciendo información en bases de datos a partir de formularios llenados por el usuario en una página Web.

El auge de las Webs dinámicas expandió la utilidad de la red; las empresas y organizaciones podían mostrar productos y servicios a sus clientes, pero los usuarios eran espectadores mudos de toda esta información. En consecuencia, comenzaron a surgir sitios que permitían al usuario expresarse, cerrando el círculo de comunicación entre los proveedores y los consumidores y diluyendo las diferencias entre éstos. De este modo los usuarios, antes mudos, pueden votar, comentar, proponer y crear. Esta evolución ha venido a llamarse Web 2.0.

Sin embargo, debido a que el contenido Web está pensado para ser entendido por humanos, no es fácil hacer una clasificación automática de los contenidos que se encuentran en la red. Las búsquedas de información se hacen difíciles y normalmente un usuario tiene que mirar y descartar varios documentos hasta que encuentra lo que estaba buscando. A veces la información permanece oculta tras una multitud de resultados de búsqueda sólo parcialmente correctos.

La Web Semántica [13] surge como un intento de organización de la información de manera que ésta pueda ser comprendida por una computadora, basándose en metadatos semánticos y ontológicos, es decir, la descripción formal de cada uno de los dominios de interés existentes en la Web. Actualmente, la World Wide Web está basada principalmente en documentos escritos en HTML; en este sentido, esta tesis pretende representar una contribución a los trabajos de investigación que de manera lenta pero consistente se van proporcionando las bases para llegar a una verdadera Web Semántica.

2.2 Ontologías

El término *Ontología* es un campo de la filosofía que estudia la naturaleza de la existencia, identificando los tipos de cosas que existen y cómo se describen. En computación, una ontología describe formalmente un dominio de interés, proporcionando una lista finita de términos y las relaciones entre ellos. Estos términos muestran conceptos importantes (clases de objetos) del dominio. Por ejemplo, en el marco de la literatura se describen libros, autores, géneros literarios y otros conceptos; se puede decir entonces que una ontología utiliza un cierto vocabulario para

definir términos específicos con definiciones formales, es decir, comprensibles para una máquina.

En el contexto de las Ciencias de la Información Geográfica (GIScience), otro propósito de las ontologías es la definición de un vocabulario común que permita la interoperabilidad y minimice problemas relacionados con la integración de datos, tanto entre sistemas como entre usuarios y sistemas [64].

En 1993, T. R. Gruber [18] definió una ontología como "la especificación explícita de una conceptualización"; más tarde, Borst [40] modificó ligeramente dicha definición añadiendo: "Las ontologías están definidas como especificaciones formales de conceptualizaciones compartidas". Ambas definiciones son clarificadas por Studer [41] al explicar los siguientes conceptos:

- **Conceptualización:** Modelo abstracto de algún fenómeno en el mundo habiendo identificado los conceptos relevantes de dicho fenómeno.
- **Explícito:** Tanto los tipos de conceptos usados como las restricciones en su utilización están explícitamente definidos.
- **Formal:** Se refiere al hecho de que la ontología debe poder ser interpretada por una máquina.
- **Compartido:** Una ontología debe capturar conocimiento consensuado, no privativo de un solo individuo, sino aceptado por un grupo.

De acuerdo con Gruber [18], una ontología debe de estar formada por los siguientes elementos:

- **Conceptos:** Son las ideas básicas reflejadas en términos que se intentan formalizar. Pueden ser objetos del mundo real, métodos, planes, etc.
- **Relaciones:** Éstas representan la interacción entre los conceptos de la ontología, definidos en función de un contexto en particular. Tenemos como ejemplos las relaciones de sinonimia, antonimia, hiponimia (es-un), y relaciones de composición y agregación.
- **Funciones:** Son un tipo concreto de relación en donde se identifica un elemento a través

del cálculo de una función, la cual toma en cuenta varios de dichos elementos. Por ejemplo, la función *armar_carro*, *clasificar_objetos*, etc.

- **Instancias:** Son usadas para representar objetos determinados de una clase, también conocidos como individuos. Por ejemplo: Manuel, Factura_no_123, Hotel_Lopez.
- **Reglas de restricción o axiomas:** Son teoremas que se declaran sobre relaciones que deben cumplir los elementos de la ontología.

Las relaciones normalmente muestran jerarquías entre los términos, indicando, por ejemplo, que la fantasía épica es un tipo de novela. A partir de estas relaciones entre los términos, una ontología también muestra las propiedades que definen los términos. Por ejemplo, todo libro tiene un título, un autor tiene nombre, un libro ha sido escrito por uno o más autores, etc. Esta capacidad de organización y jerarquización de la información pone en evidencia el uso potencial de las ontologías como integradoras de datos a partir de fuentes heterogéneas.

Las ontologías proporcionan un conocimiento formal acerca de un dominio, y en Internet, ese conocimiento se vuelve compartido, permitiendo describir los elementos de un dominio de una forma común. En la figura siguiente podemos ver un ejemplo simple de ontología referida a autores y libros. La parte superior de la figura muestra la ontología, donde se definen los tipos de elementos que tenemos en nuestro dominio (autores, personas y libros) y cómo se relacionan entre ellos (un autor es un tipo de persona, un autor escribe libros). La parte inferior de la figura muestra un uso de esta ontología definiendo un autor y tres de sus libros escritos.

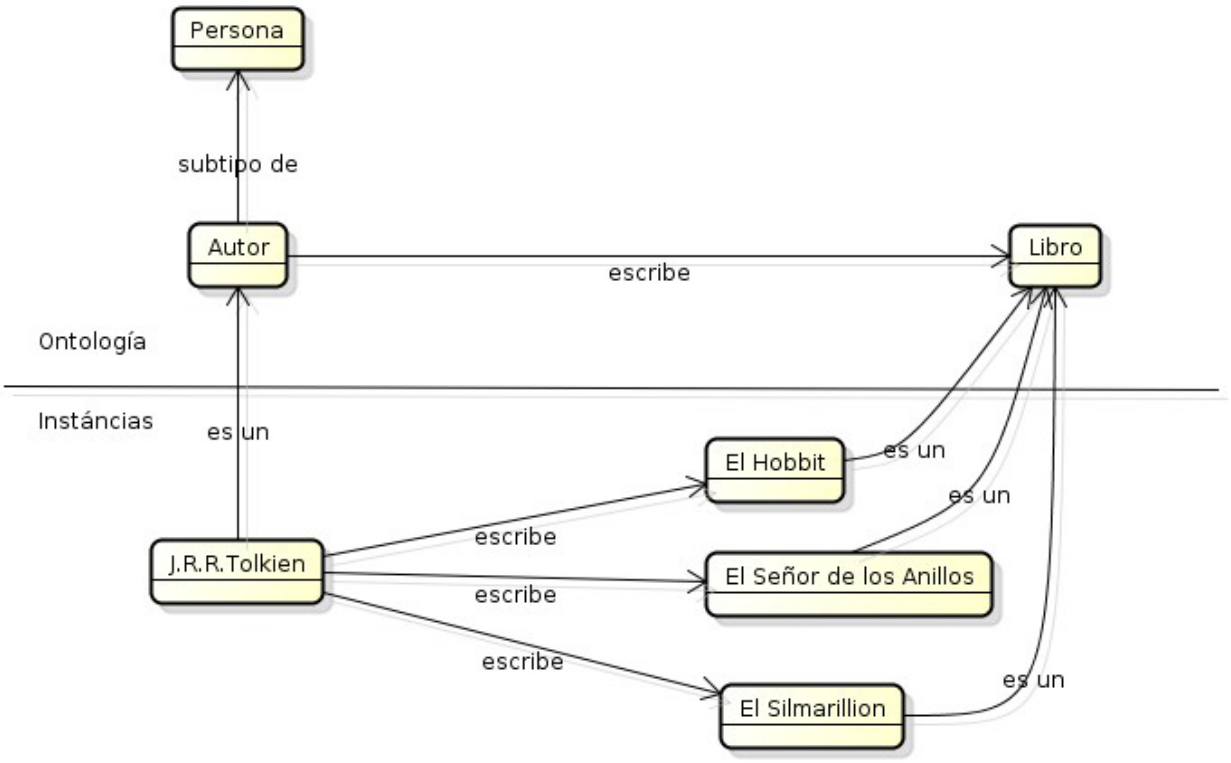

Fig. 2.1

Ontología de dominio literario

Una clasificación para las ontologías de acuerdo con su nivel de precisión con respecto a la conceptualización, es decir, su nivel de generalidad, fue propuesta por Nicola Guarino [19]:

- **Ontologías de nivel superior:** Describen conceptos muy generales, tales como tiempo, objeto, acción, etc., los cuales son independientes de un problema o dominio definido.

- **Ontologías de dominio y de tarea:** Describen respectivamente, el vocabulario relacionado con un dominio general, como medicina o automóviles, o bien una tarea o actividad, como diagnosticar o vender, por medio de la especialización de los conceptos introducidos en las ontologías de nivel superior.

- **Ontologías de Aplicación:** Describen conceptos dependiendo de un dominio particular y de una tarea específica; estos conceptos generalmente corresponden a roles desempeñados por entidades de dominio mientras realizan una actividad determinada.

En el presente trabajo de tesis se hará uso de una ontología de aplicación, correspondiente al dominio turístico y con tareas específicas de recuperación e integración de datos.

2.3. Lenguajes de ontologías

Evolución de los lenguajes: de HTML a OWL.

El primer paso para que la información sea entendible por una computadora es estructurarla; en **HTML,** la información se encuentra plasmada predominantemente en lenguaje natural. Una página sobre un libro contendrá la información sobre quién es su autor mezclada con otra información redactada (por ejemplo "En <El Hobbit> J.R.R.Tolkien nos cuenta..."). Aunque a una persona le pueda resultar fácil intuir y extraer el nombre del autor a partir del contexto, a una máquina le resulta muy difícil [42].

XML es un lenguaje de marcado que permite la estructuración de la información con base en etiquetas (conocidas como tags) que indican cuál es la información que contiene. La información anterior en XML podría ser:

```
<libro>
  <titulo>El Hobbit</titulo>
  <autor>J.R.R.Tolkien</autor>
</libro>
```

XML-Schema es un lenguaje que permite definir cuál es la estructura que debe tener un determinado documento XML [43]. De este modo, pueden definirse esquemas comunes para facilitar la interoperabilidad entre distintos documentos. Podría pensarse que con XML se

favorece la comprensión del documento por parte de las máquinas a costa de empeorar la legibilidad. XSLT son documentos que pueden adjuntarse a documentos XML y procesarlos, transformando su contenido a otro más fácilmente entendible por las personas. Esto tiene la ventaja adicional que de un mismo XML pueden surgir documentos diferentes (HTML, PDF) simplemente cambiando el documento de transformación XSLT que lo procesa.

De esta manera, XML es un metalenguaje universal para definir marcas de información. Provee un entorno uniforme y una gama de herramientas como procesadores (parsers), para permitir el intercambio de datos entre aplicaciones, al mismo tiempo que provee de legibilidad dicha información para los humanos.

RDF y RDF-Schema

Las sentencias en RDF

Hemos visto que XML permite el intercambio de información, pero no transmite la *semántica* de esta información. Es responsabilidad de cada aplicación el interpretar los tags de un documento XML. Por ejemplo, supongamos que tenemos un XML que muestra una lista de personas, convenientemente etiquetadas como <persona>. Prestando atención al XML de libros que se mostraron antes, ¿es J.R.R.Tolkien una persona?

RDF es un modelo de datos que permite definir información de manera semántica [44]. Esta información se organiza en forma de tripletas del tipo *sujeto-predicado-objeto*, llamados sentencias (statements). Utilizando el ejemplo anterior, en RDF tendríamos dos sentencias:

1. “El hobbit” es un libro
2. “El hobbit” tiene como autor a J.R.R.Tolkien

Si se compara esta forma de definir la información con el modo en XML, puede observarse que, mientras que en XML se tiene un dato “título” con valor “El hobbit” y un dato “autor” con valor

"J.R.R.Tolkien", en RDF es posible relacionar la información añadiendo semántica.
Ampliando el ejemplo mediante un nuevo enunciado:
"El señor de los anillos" tiene como autor a J.R.R.Tolkien

es posible plantear la pregunta "¿Cuáles son los libros que tienen como autor a J.R.R.Tolkien?". A partir de las sentencias anteriores, se puede ver que esos libros son "El hobbit" y "El señor de los anillos".

A grandes rasgos, esta es la manera como RDF permite definir información, de manera que es posible localizar información en esta estructura, pero ¿es J.R.R.Tolkien una persona?

Hasta este punto, contamos con la información:
"J.R.R. Tolkien es un autor"

Para llegar al enunciado
"J.R.R. Tolkien es una persona"

requerimos saber si un autor puede considerarse una persona. Antes de responder a esta pregunta es necesario entender el funcionamiento de la siguiente capa, RDF-Schema.

RDF-Schema como herramienta para la creación de ontologías

RDF-Schema (**RDFS**) es un lenguaje que permite definir la terminología que va a ser empleada en un determinado contexto. RDF se encarga de utilizar dicha terminología para establecer relaciones, creando así una ontología. RDFS permite añadir información semántica sobre cómo se relacionan entre sí las clases de una ontología, estableciendo una jerarquía entre dichas clases [45].

Clases, propiedades e instancias

La base para la definición de ontologías mediante RDF se encuentra en la definición de clases, propiedades e instancias. Una clase indica una categoría característica dentro de una ontología. En una ontología literaria, "Libro" sería una clase, así como también "Autor". Una clase contendrá instancias, que, al igual que en el paradigma de orientación a objetos, constituirán los elementos de esa clase. La clase se relaciona con la instancia respondiendo a la pregunta "¿qué es esta instancia?". De este modo "J.R.R.Tolkien" es un "Autor", mientras que "El Hobbit" es un "Libro".

En RDFS, las clases permiten crear jerarquías entre ellas (taxonomías), respondiendo a la sentencia "es un subtipo de". Siguiendo con el ejemplo "Autor" es un tipo de "Persona". Esto indica que todo autor es también una persona y comparte todos los atributos que tienen las personas.

Las propiedades permiten definir los atributos de una clase, estableciendo relaciones entre clases o entre clases y valores. En este caso, un "Libro" "tiene un" "Autor" (o más de uno), o un "Libro" "tiene como nombre" "un texto". De esta manera, es posible ampliar el ejemplo con la siguiente sentencia, indicando la relación comentada: **Autor es un subtipo de Persona.**

Dominios y rangos

En una sentencia RDF, formada por *sujeto-predicado-objeto* y cuyo objeto sea una propiedad, no se determina información sobre qué debe ser el sujeto o el objeto. RDFS permite definir esta información. El sujeto es el dominio de la propiedad, mientras que el objeto es el rango. Ampliando el ejemplo anterior con las siguientes sentencias en RDFS:

- la propiedad "*tiene como autor*" tiene como dominio un *Libro*
- la propiedad *"tiene como autor"* tiene como rango un *Autor*

Así, de la sentencia del ejemplo: **"El hobbit" tiene como autor a J.R.R.Tolkien** es posible saber que "El hobbit" es un *Libro* (por el dominio de la propiedad) y que J.R.R.Tolkien es un *Autor*. Este proceso de descubrir información a partir de las definiciones y sentencias disponibles es lo que se entiende como ***inferencia***, es decir, obtención de nueva información a partir de información semántica existente. Desde el punto de vista de la información, es la creación de nuevos conocimientos a partir de la información ya conocida, y es donde reside gran parte de la potencia de la tecnología de Web Semántica.

A partir de la aplicación de reglas simples en las sentencias que ya se tienen, es posible inferir nuevas sentencias. En este punto se puede utilizar la inferencia para responder a la pregunta que se hacía al comenzar el ejemplo: ¿es J.R.R.Tolkien una persona?

Partiendo de las sentencias:
"El hobbit" tiene como autor a J.R.R.Tolkien
La propiedad *"tiene como autor"* tiene como rango *Autor*

se puede inferir la sentencia:
J.R.R.Tolkien es un Autor

y dado que se sabe que
Autor es un subtipo de Persona

es posible inferir que
J.R.R.Tolkien es una Persona

La inferencia es una herramienta poderosa al permitir la creación de nuevo conocimiento a partir del conocimiento existente y las reglas definidas, pero debe ser observado con cuidado, ya que una regla mal utilizada puede dar lugar a sentencias incorrectas.

OWL

La expresividad de RDF/RDF-Schema es limitada. Hasta este punto se ha visto cómo RDF permite definir predicados binarios ("J.R.R.Tokien" es un "Autor"), mientras que RDF-Schema amplía la expresividad de RDF, permitiendo la definición de jerarquías de clases de elementos y propiedades y sus relaciones.

El grupo de trabajo de ontologías Web del W3C [20], identificó un número de casos de uso para la Web Semántica que requieren mayor expresividad de la que provee RDF/RDF-Schema. El trabajo de varios grupos de investigación en Europa y Estados Unidos, que habían identificado estas necesidades y trabajado en una solución, se unió como punto de partida para desarrollar OWL (Ontology Web Language) [23], el cual está enfocado en ser el lenguaje estándar para la Web Semántica y el desarrollo de ontologías.

OWL ofrece mayor expresividad de la que tiene RDF/RDF-Schema, permitiendo crear ontologías más ricas. Entre las características que aporta OWL se encuentra la posibilidad de definir restricciones de cardinalidad en propiedades ("una persona tiene exactamente dos padres") o características especiales de propiedades como la propiedad transitiva o la inversa.

Para entender mejor la expresividad de OWL, replanteamos la pregunta "¿Cuáles son los libros que tienen como autor a J.R.R.Tolkien?" de la forma: "¿Qué libros ha escrito J.R.R.Tolkien?" De esta manera, se puede observar que falta definir la propiedad "ha escrito" para poder responder la pregunta.

Aunque es posible definir esta propiedad y ampliar el ejemplo con las sentencias:
J.R.R.Tolkien ha escrito "El hobbit"
J.R.R.Tolkien ha escrito "El señor de los anillos"

es algo engorroso teniendo en cuenta que ya se tenía que:
"El hobbit" tiene como autor a J.R.R.Tolkien.

La expresividad de OWL permite superar esta dificultad, mediante la definición de la propiedad *"ha escrito"* como inversa de la propiedad *"tiene como autor"*, *"ha escrito"* es una propiedad inversa de *"tiene como autor"*. Entonces, es posible inferir que las sentencias anteriores y también la sentencia *"tiene como autor"* es una propiedad inversa de *"ha escrito"* que permitiría que al ampliar el ejemplo con:

J.R.R.Tolkien ha escrito "El Silmarillion".

da lugar por la inferencia a la sentencia:
"El Silmarillion" tiene como autor a J.R.R.Tolkien.

Como puede observarse, OWL posee mayor capacidad para expresar significado y semántica que XML, RDF, y RDF-S, de manera que es capaz de manejar contenido interpretable por una máquina en la Web. OWL proporciona a su vez tres lenguajes, cada uno con un nivel de expresividad mayor que el anterior, diseñados para ser usados por comunidades específicas de desarrolladores y usuarios.

OWL LITE

Es utilizado para la migración desde otras taxonomías. Está orientado a la clasificación de jerarquías y restricciones simples, de manera que puede facilitar el desarrollo, a la vez que admite restricciones de cardinalidad. OWL Lite proporciona una ruta rápida de migración para tesauros y otras taxonomías. Posee una menor complejidad formal que OWL DL. En OWL Lite cada clase contiene: subclases, condiciones necesarias y suficientes para la clase, y expresiones de clase que incluyan el operador de intersección [46].

OWL DL (Description Logic - Descripción Lógica)

Contiene los constructores del lenguaje con restricciones jerárquicas y resolubilidad (todos los cálculos se resolverán en un tiempo finito). OWL DL incluye todas las construcciones del lenguaje de OWL, que pueden ser utilizados bajo ciertas restricciones (por ejemplo, mientras una clase puede ser una subclase de otras muchas clases, una clase no puede ser una instancia de otra). Se extienden las funciones de OWL LITE para incluir: aserciones que las instancias de clase no pueden compartir con la expresión de clase, definiciones de clase en forma extensiva y expresiones de clase que incluyan operadores de unión, intersección o complemento [47].

OWL FULL

OWL Full permite el uso de todos los constructores de OWL además de permitir el uso libre e irrestricto de RDF. Es poco probable que algún software sea capaz de obtener un razonamiento completo para cada característica de OWL Full. La elección entre OWL Lite y OWL DL depende de las necesidades de los usuarios sobre la expresividad de las construcciones, proporcionando OWL DL las más expresivas. La elección entre OWL DL y OWL Full depende principalmente de las necesidades de los usuarios sobre los recursos de metamodelado del esquema RDF (por ejemplo, definir clases de clases, o definir propiedades de clases). Cuando se usa OWL Full en comparación con OWL DL, el soporte en el razonamiento es menos predecible, ya que no existen actualmente implementaciones completas de OWL Full [47].

2.4 Editores de ontologías

Estos editores son herramientas que permiten la codificación de una determinada ontología con base en un lenguaje específico; permiten definir la estructura para la organización de la información de un dominio específico.

Los editores de ontologías más usados actualmente son:

- **Apollo:** Aplicación amigable de modelado de conocimiento. El modelado está basado en torno a los principios básicos tales como clases, instancias, funciones, relaciones, etc. La interfaz de usuario tiene una arquitectura abierta y está escrita en lenguaje de programación Java [48].

- **LinkFactory:** Se trata de una herramienta utilizada para construir sistemas completos de terminología corporativa, capaz de extraer valor significativo de gran cantidad de datos no estructurados almacenados en bases de datos de contenido corporativo [49].

- **OntoEdit versión Libre y Profesional:** Permite crear y gestionar ontologías. Utiliza estándares W3C y ofrece muchas interfases exportables a la mayor parte de lenguajes de representación de ontologías [50].

- **Servidor Ontolingua:** Provee de un entorno de colaboración distribuido para navegar, crear, editar, modificar y utilizar ontologías [51] .

- **Ontosaurus:** Es un navegador web para las bases de conocimiento de LOOM. Proporciona una interfaz gráfica enlazada hacia varias bases de conocimiento [52].

- **OpenKnome:** Piedra angular de los motores de conocimiento *topThing* (clase padre de los conceptos). Es un sistema de gestión de conocimiento y motor de ontologías. Desde el 2001 el código fuente está abierto para la comunidad académica [53].

- **Protégé:** Editor de ontologías y editor de bases de conocimiento. De código abierto, desarrollo en Java, proporciona una arquitectura extensible para la creación de aplicaciones de bases de conocimiento. Las aplicaciones desarrolladas con Protégé son empleadas en la resolución de problemas y toma de decisiones en dominios particulares; emplea una interfaz de usuario que facilita la creación de una estructura de marcos con clases, restricciones e instancias de una forma integrada [22].

- **WebODE:** Herramienta para modelar el conocimiento utilizando ontologías. Facilita la máxima flexibilidad e interoperabilidad con otras aplicaciones de negocios necesarias para las empresas de hoy en día [54].

- **WebOnto:** Java Applet con un servidor Web personalizado que permite a los usuarios navegar y editar modelos de conocimiento sobre la Web [55].

2.5 Motores de inferencia o razonadores

Las tareas de razonamiento son llevadas a cabo por los motores de inferencia, también llamados *razonadores*, que son módulos de software que implementan algoritmos deductivos clásicos optimizados para el subconjunto de la lógica de primer orden que delimita el nivel de la lógica de descripciones que aceptan [56].

Un razonador es una herramienta que aprovecha el gran poder semántico de las ontologías, de forma que puedan extraer conocimiento no expresado de forma explícita en su construcción. El motor de inferencia será el que permita sacar conclusiones de la base de conocimiento y resolver el problema que se le ha planteado.

El motor de inferencia permitirá obtener nuevas sentencias o conclusiones a partir de las sentencias de su base de conocimiento, y determinar, por ejemplo, si una conclusión es deducible a partir de las sentencias de dicha base. También permite determinar si una determinada sentencia es imposible de inferir, es decir, lleva a una contradicción. Otro posible resultado es concluir que una sentencia no es imposible, pero no es deducible de la base de conocimiento.

Estas herramientas son las que finalmente propician gran parte del potencial de las ontologías como herramientas de gestión de la información. Como hemos mencionado, al momento de definir una ontología se pueden encontrar tres elementos principales:

1. Clases e instancias, que son los objetos o elementos que la componen.
2. Propiedades, que representan las relaciones entre los anteriores.
3. Reglas, que se emplean para modelar el conocimiento que no puede recogerse a través de los elementos anteriores.

Si no se emplearan estas últimas se estaría ante una ontología ligera, que no es más que una taxonomía o clasificación de elementos. La no utilización de reglas origina el problema de la poca capacidad expresiva del conocimiento.

De hecho, podría incluso establecerse una clasificación de las distintas tecnologías existentes para el tratamiento de la información, distinguiendo para cada una de ellas el nivel de semántica capturado. Así, por ejemplo, las bases de datos relacionales, si bien resultan de gran utilidad, presentan un poder casi nulo en lo que se refiere a la **recolección** de **aspectos semánticos**.

Un razonador basado en lógica descriptiva asocia dos mecanismos internos en su entendimiento del conocimiento. El primero denominado *TBox* (caja terminológica) y un segundo llamado *ABox* (caja de aserciones). Esta separación es puramente operativa, ya que estas distinciones permiten al razonador operar de manera más eficiente.

- La *TBox* contiene sentencias describiendo conceptos jerárquicos (es decir, relaciones entre conceptos).
- La *ABox* contiene sentencias indicando a dónde pertenecen los individuos en la jerarquía (es decir, relaciones entre individuos y conceptos).

En cuanto a la forma interna de razonamiento, este proceso se basa en la realización de inferencias que permiten realizar deducciones mayores o transitivas con base en axiomas.

A continuación se describen algunos de los diferentes motores de inferencia más usados actualmente:

FaCT++ 1.1.3

Desarrollado por la Universidad de Manchester, es la continuación del razonador de código abierto para *TBoxes* FaCT (Fast Classification of Terminologies); la versión antigua fue desarrollada en Lisp, la versión actual sobre C++ para acelerar los tiempos de procesamiento.

La nueva versión permite el uso de *Aboxes* y ha mejorado su interfaz con DIG. FaCT++ está optimizado para la lógica de OWL DL. Los algoritmos utilizados son optimizaciones del algoritmo *Tableaux* [57].

Su instalación sobre Windows conlleva cierta dificultad al tener la necesidad de instalar la plataforma .NET 2.0, y no se ha encontrado información alguna sobre su instalación, manejo y mensajes de error mostrados por el razonador. No cuenta con el apoyo de ninguna comunidad de desarrolladores y beta-testers asociados, por lo que se encuentra aún en una fase inestable.

RACER 1.9

Racer (Reasoner for Aboxes and Concept Expressions Renamed) fue uno de los primeros razonadores para *Aboxes* que surgió. Aunque fue desarrollado por la Universidad de Hamburgo, actualmente es software propietario. Racer 1.9 soporta OWL DL excepto para los nominales (clases definidas por una enumeración de sus miembros, que implementa como definiciones parciales) y para tipos de datos no estandarizados. Al contrario de las primeras versiones, por defecto no posee UNA (Unique Name Assumption) para ser compatible con OWL. Se basa en el algoritmo *Tableaux* con optimizaciones de inferencias obtenidas [58].

RacerPro

Es el razonador por defecto de Protégé. Incluye un cliente para consultas en OWL-QL (RacerPorter), su propio lenguaje de consultas (nRQL, new Racerpro Query Language) así como

una herramienta de cliente exclusiva (RICE, Racer Interactive Client Environment). Asimismo proporciona una API en Java para poder interactuar directamente (Jracer). RacerPorter es la interfaz gráfica de usuario de RacerPro [59].

Para las *Tbox* soporta las utilidades de consistencia de conceptos y determinación de padres e hijos, basándose en la semántica del lenguaje de representación.

Para las *Abox* soporta las siguientes utilidades: Chequeo de consistencia de la *Abox* respecto de la *Tbox* y chequeo de la clase a que pertenecen las instancias.

Pellet 1.3

Es un razonador de código abierto, desarrollado por Mindswap. Es un razonador exclusivo para OWL DL e implementado en Java, proporciona la interfaz DIG y además RDQL para consultas sobre la información en RDF. Además, dispone de una API para poder ser utilizado directamente desde Java. Al igual que los sistemas anteriores, está basado en el algoritmo *Tableaux* [24].

Algunas de las funcionalidades principales de este razonador son las siguientes:

- **Comprobación de la consistencia:** Se encarga de comprobar que no existen contradicciones en la ontología. La semántica de OWL define una especificación formal para la definición de la consistencia en una ontología empleando Pellet. En terminología DL esta operación consiste en revisar la consistencia de la **Abox** respecto de la **Tbox**.

- **Corrección de los conceptos:** Verifica si es posible que se definan instancias para una clase determinada; de no ser así, toda la ontología resultaría inconsistente.

- **Clasificación:** Observa la relación entre cada clase y crea la jerarquía completa de clases.

- **Realización:** Encuentra las clases más específicas a las que pertenece una instancia; en otras palabras, determina la clase a la que pertenece cada uno de los individuos. Esta operación sólo se puede realizar después de la clasificación, puesto que los tipos directos se definen respecto de la jerarquía de clases.

2.6 Herramientas de gestión de ontologías

Estas herramientas incluyen varios aspectos del proceso de desarrollo de ontologías, tales como edición, navegación, almacenamiento y recuperación.

Lenguaje SPARQL

SPARQL (SPARQL Protocol And RDF Query Language, pronunciado como "sparkle") define un lenguaje de organización y recuperación para RDF/RDFS y OWL. Esta tecnología de consulta permite que las personas puedan centrarse en la información que desean, sin tener en cuenta la tecnología de la base de datos o el formato utilizado para almacenar los datos. Debido a que las consultas en el lenguaje SPARQL expresan objetivos de alto nivel, es fácil extenderlos a orígenes de datos inesperados, o incluso transferirlos a nuevas aplicaciones [33].

El lenguaje de recuperación SPARQL ha sido diseñado para su uso a escala de la Web, de manera que permite hacer consultas sobre orígenes de datos distribuidos, independientemente del formato. Es más fácil crear una consulta sencilla y recuperar información en una sola consulta a través de diferentes almacenes de datos que crear múltiples consultas, además de tener un costo menor y de ofrecer mejores resultados.

Debido a que SPARQL no está ligado a un formato de base de datos específico, puede ser utilizado para beneficiarse de la nueva ola de los datos de la Web 2.0 y de la composición de éstos con otros recursos de la Web Semántica en las aplicaciones. Además, debido a que los orígenes de datos dispares pueden no tener el mismo formato o compartir las mismas

propiedades, SPARQL ha sido diseñado para consultar datos que no son uniformes.

La especificación de SPARQL define un lenguaje de consulta y un protocolo, y trabaja con el resto de las tecnologías esenciales del W3C de la Web Semántica: Infraestructura de Descripción de Recursos (RDF) para la representación de datos; RDF Schema; Lenguaje de Ontologías Web (OWL) para la construcción de vocabularios. SPARQL también usa otros estándares del W3C existentes en las implementaciones de servicios Web, como Lenguaje de Descripción de Servicios Web (WSDL).

Jena

Es un marco de trabajo de Java para construir aplicaciones de Web Semántica. Proporciona un entorno de programación de RDF, RDFS, OWL, SPARQL e incluye un motor de inferencia basado en reglas [32].

Las características más relevantes de Jena son las siguientes:

1. API de RDF y OWL.
2. Lectura y escritura de RDF en RDF/XML.
3. Almacenamiento en memoria y persistente.
4. Motor de consultas SPARQL.

Jena fue diseñado con la intención de dar soporte para OWL Full y para el razonamiento de casos no incluidos en el subconjunto de la sintaxis OWL DL.

SDB es un componente de Jena. Proporciona almacenamiento escalable y consultas de datos RDF usando bases de datos SQL convencionales, para el uso en aplicaciones tanto en J2EE como autónomas. Pueden ser instaladas herramientas para el balanceo, seguridad, respaldo y administración. SDB está diseñado específicamente para el soporte de SPARQL.

Kaon2

Es una infraestructura para gestionar ontologías OWL-DL, SWRL y F-Logic [60]. Proporciona las siguientes funcionalidades:

- Una API de programación para la gestión de ontologías OWL-DL, SWRL y F-Logic.
- Un servidor propio con acceso a ontologías de forma distribuida usando RMI.
- Un motor de inferencia para responder consultas expresadas en sintaxis SPARQL.
- Interfaz DIG, proporcionada a herramientas tales como Protégé.
- Un módulo para extraer instancias de ontologías de bases de datos relacionales.

Powl

Powl es un marco de trabajo en PHP para el análisis sintáctico, almacenamiento, consulta, manipulación, servicio y serialización de bases de conocimiento OWL en un entorno colaborativo Web [61].

Algunas de las características más sobresalientes de Powl son:

- Navegación y edición de ontologías RDFS/OWL de tamaño arbitrario.
- Edición de datos de manera visual.
- Extensible - filosofía *plug-in*.
- Sistema de consulta RDQL.
- Autenticación en el modelo.

2.7 Herramientas de desarrollo Web

Los temas tratados en la siguiente sección abordarán las tecnologías para el desarrollo de páginas Web, dado que sobre este ambiente será construida la interfaz del sistema propuesto.

XHTML (eXtensible Hypertext Markup Language - Lenguaje Extensible de Marcado de Hipertexto*)*

Es una adaptación de HTML , teniendo como principal diferencia que el primero cumple con las especificaciones del estándar XML y a la vez con un objetivo específico.

XHTML surge como una versión más estricta de su predecesor, debido a que los contenidos almacenados en la Web realizados con HTML necesitan una cierta cantidad de recursos para afrontar la complejidad de la sintaxis del mismo. Los dispositivos móviles (PDAs, laptops, teléfonos inteligentes) no contaban con los recursos suficientes para dicho procesamiento y fue entonces que surgió la idea de estandarizar XHTML en los navegadores principales.

XHTML, al estar orientado al uso de un etiquetado estandarizado, exige una serie de requisitos básicos tales como una estructuración coherente dentro del documento, donde se incluirían elementos correctamente anidados, etiquetas en minúsculas, elementos cerrados correctamente, atributos de valores entrecomillados, etc.

Así mismo, XHTML puede incluir otros lenguajes como MathML, SVG (Vectores Gráficos Escalables), entre otros, lo cual le da una funcionalidad mucho más amplia que a HTML [62].

CSS (Cascading Style Sheets - Hojas de Estilo en Cascada)

Es un lenguaje usado para definir la presentación de un documento escrito en XML, HTML o XHTML. La idea es separar la estructura funcional y la presentación de un documento [63].
Las ventajas de utilizar CSS son:

- Se tiene el control de la presentación de un sitio Web completo, con lo que se agiliza de forma considerable la actualización del mismo.
- Los navegadores permiten a los usuarios especificar su propia hoja de estilo local que

puede ser aplicada a un sitio Web, aumentando considerablemente la accesibilidad.

- Una página puede disponer de diferentes hojas de estilo según el dispositivo que la muestre o incluso por selección del usuario. Por ejemplo, para ser impresa, mostrada en un dispositivo móvil, o ser "leída" por un sintetizador de voz.
- El documento HTML en sí mismo es más claro de entender y se consigue reducir considerablemente su tamaño.

Existen tres versiones de CSS: CSS1 y CSS2, con CSS3 en desarrollo por el World Wide Web Consortium (W3C).

- **CSS1:** Las hojas de estilo CSS1 describen el formato del texto y de los componentes de una página (fuente, color, tamaño, etc.).
- **CSS2:** Permite ubicar elementos de XHTML en diferentes capas, cuyo posicionamiento no tiene que seguir el flujo HTML (plantea el modelo de cajas donde cada bloque se define como una caja que se coloca en un lugar concreto). CSS2 incluye a CSS1.
- **CSS3:** aún está en desarrollo por el W3C y los navegadores apenas son capaces de "entenderlo".

Servlets

Los *Servlets* son programas que se ejecutan en un Servidor Web, los cuales fungen como un intermediario entre las peticiones que solicitan los Clientes y las aplicaciones del Servidor. A diferencia de los *Applets*, los *Servlets* se ejecutan en el servidor y no presentan ninguna interfaz gráfica, debido al trabajo "oculto" que realizan [31]. Sus principales funciones son:

- Leer los datos proporcionados por el usuario (ya sea desde un formulario, un *Applet*, etc.)
- Comunicarse con una base de datos, ejecutar llamadas a RMI (Remote Method Invocation) o invocar a otras aplicaciones existentes.
- Devolver el documento al cliente, ya sea en formato de texto (un XHTML), binario (imágenes) o inclusive un formato comprimido.

Los *Servlets* son la mejor opción para desarrollar aplicaciones de servidores Web. Trabajan de forma transparente con las filosofías GET y POST de los formularios XHTML, se comunican perfectamente con los Applets y con clientes programados en cualquier otro lenguaje.

JSPs (Java Server Pages, Páginas de Servidor Java)

Es una tecnología Java que permite generar contenido dinámico para Web, en forma de documentos HTML, XML o de otro tipo.

Las JSPs permiten la utilización de código Java mediante scripts. Además, es posible utilizar algunas acciones JSP predefinidas mediante etiquetas. Estas etiquetas pueden ser enriquecidas mediante la utilización de Bibliotecas de Etiquetas (TagLibs o Tag Libraries) externas e incluso personalizadas [30].

El rendimiento de una página JSP es el mismo que tendría el servidor equivalente, ya que el código es compilado como cualquier otra clase Java. A su vez, la máquina virtual compilará dinámicamente el código máquina de las partes de la aplicación que lo requieran. Esto hace que JSP tenga un buen desempeño y sea más eficiente que otras tecnologías Web que ejecutan el código de una manera puramente interpretada.

PHP (Hypertext Pre-processor)

Es un lenguaje de programación interpretado, diseñado originalmente para la creación de páginas web dinámicas. Es usado principalmente en interpretación del lado del servidor (*server-side scripting*), pero actualmente puede ser utilizado desde una interfaz de línea de comandos o en la creación de otros tipos de programas, incluyendo aplicaciones con interfaz gráfica. Generalmente se ejecuta en un servidor web, tomando el código en PHP como su entrada y creando páginas web como salida. Puede ser desplegado en la mayoría de los servidores web y en casi todos los

sistemas operativos y plataformas sin costo alguno.

Ventajas

- Es un lenguaje multiplataforma.
- Capacidad de conexión con la mayoría de los manejadores de base de datos que se utilizan en la actualidad.
- Amplia documentación.
- Software libre.
- Filosofía orientada a objetos.
- Biblioteca nativa de funciones sumamente amplia.

Servidores Web

Tomcat

Tomcat es el servidor Web que frecuentemente se utiliza cuando se trabaja con Java en la Web; fue desarrollado bajo el proyecto Jakarta en Apache Software Foundation. Tomcat es un administrador de peticiones de los estándares de JSP y Servlets de Sun Microsystems; otros servidores Web populares son Apache HTTP o el servidor Microsoft Internet Information Server (IIS). Tomcat está escrito e integrado en Java 2 Enterprise Edition (J2EE) de Sun Microsystems.

Tomcat no es un servidor de aplicaciones, pues incluye el compilador *Jasper*, que compila JSPs convirtiéndolas en Servlets, por lo que puede funcionar como servidor Web por sí mismo. En sus inicios existió la percepción de que el uso de Tomcat de forma autónoma era sólo recomendable para entornos de desarrollo con requisitos mínimos de velocidad y gestión de transacciones. Hoy en día, ya no existe esa percepción y es usado como servidor Web autónomo en entornos con alto nivel de tráfico y disponibilidad.

Apache

El servidor HTTP Apache es un servidor Web de código abierto para plataformas Unix (BSD, GNU/Linux, etc.), Windows, Macintosh y otras. Este servidor posee las siguientes características:

- Nuevo conjunto de interfaces de programación de aplicaciones (APIs).
- Filtrado, los módulos pueden actuar como filtros de contenido.
- Soporte a Ipv6, la próxima generación de formato de direcciones IP.
- Respuestas a errores en diversos idiomas: cuando se utilizan documentos Server Side Include (SSI), las páginas personalizables de errores se pueden entregar en diversos idiomas.

Apache es usado principalmente para enviar páginas estáticas y dinámicas en la Web. Muchas aplicaciones Web están diseñadas asumiendo como ambiente de implantación a Apache, o que utilizarán características propias de este servidor Web. Apache es el componente de servidor Web en la popular plataforma de aplicaciones LAMP, junto a MySQL y los lenguajes de programación PHP/Perl/Python y Ruby.

La capa frontal (*front end*) del motor de búsqueda Google está basada en una versión modificada de Apache, denominada Google Web Server (GWS).

Mashup

Una aplicación Web híbrida (mashup o remezcla), es un sitio o aplicación Web que usa información de otras aplicaciones Web para crear un nuevo contenido completo. El contenido de un mashup normalmente proviene de sitios Web de terceros a través de una interfaz pública o mediante el uso de una API. Ésta contacta con el sitio proveedor del contenido, por ejemplo Google Maps, y le pide que le envíe los datos que en ese momento requiere el usuario.

De momento, los principales abastecedores de contenido para mashups son grandes servicios como Flickr, eBay, Youtube, Amazon, Yahoo!, Microsoft o Google.

2.8 Visualización de mapas por Internet

A continuación se presenta un recuento de las principales aplicaciones para la visualización de mapas por Internet.

OpenMaps

Esta aplicación libre fue creada por un grupo de voluntarios; contiene mapas topológicos y de carreteras. Dichos mapas tienen la opción de ser descargados para diversos dispositivos, en especial los móviles (GPS, PDA's, etc.). Únicamente cuenta con mapas de países europeos.

ALOV Map

ALOV Map es de uso libre, desarrollado en Java, para visualizar mapas vectoriales y tipo raster; se utiliza igualmente para la publicación de mapas por Internet con visión interactiva. Cuenta con una arquitectura de representación compleja, posee navegación ilimitada y permite el trabajo con múltiples capas.

Google Maps/Earth

Es un SIG que permite visualizar imágenes en 2D del planeta. Cuenta con un *plug-in* y una API en JavaScript que permite visualizar modelos digitales de elevación de terreno en 3D o incluso cargar archivos KML (Key hole Markup Language, para representación de datos en tres dimensiones).

Google Earth permite igualmente visualizar imágenes en 3D del planeta a través de la combinación de imágenes de satélite, mapas y el motor de búsqueda de Google. Entre las funcionalidades de Google Earth encontramos que se pueden realizar búsquedas de hoteles, escuelas o calles y obtener la dirección exacta, proporcionando planos o vistas del lugar objetivo.

Yahoo Maps

Es un portal en línea de Yahoo, de uso libre, para la visualización de mapas digitales a través de Internet. Cuenta con tres formas diferentes de visualización:

- Servicios Web de Mapas Yahoo
- Mapas internacionales
- Mapas por satélite

La API de Yahoo Maps nos permite incrustar fácilmente los mapas que deseemos en nuestra página Web mediante Flash, AJAX o imágenes estáticas de los mapas. Yahoo también provee contenido geográfico relevante para los usuarios, tales como el tráfico en determinadas zonas, el reporte del estado del tiempo, espectáculos próximos a presentarse, entre otros.

2.9 Trabajos relacionados

A continuación se realiza una breve descripción de trabajos afines a la línea de investigación de la presente tesis.

2.9.1 Harmonise

El proyecto Harmonise [65] presenta una solución alternativa al problema de la interoperabilidad en los sistemas de información turística: la creación de un "espacio de armonización" en la que los proveedores de servicios turísticos pueden intercambiar información sin tener que modificar sus fuentes de datos. Esto se logra a través de la reconciliación semántica entre los diferentes

sistemas de información, utilizando una ontología de dominio. Los formatos de datos propietarios son transparentes a la Plataforma Harmonise (HP), que se ocupa del tratamiento de documentos XML que los proveedores determinen exportar. Después, éste es traducido a la terminología especificada por la ontología a través de un mediador, la herramienta de "Armonización" (*Harmonisation tool*), utiliza un mapeo lineal para obtener un representación de datos propia de la ontología turística de Harmonise (HIR: Harmonise Interchange Representation – por sus siglas en inglés). De esta manera, cada proveedor puede visualizar la información de todos los sistemas participantes como una extensión de su propio sistema, sin preocuparse por la heterogeneidad de los datos.

Harmonise es una iniciativa financiada por la Comisión Europea (5th Framework RTD Programme), a través de la "Red de Armonización Turística" (THN: Tourism Harmonisation Network – por sus siglas en inglés), la cual comprende a las principales organizaciones turísticas europeas, propietarias de estándares y sistemas. Oficialmente formalizada en Enero de 2002, entre los países miembros se encuentran Italia, Austria, Francia, Portugal, España, Finlandia, además de otras organizaciones internacionales.

2.9.2 OnTour

El proyecto OnTour [66], consiste en la construcción de un portal que realiza búsquedas semánticas sobre sitios Web específicos para el caso, buscando a la vez resultados óptimos y eficientes a través del uso de tecnologías de la Web Semántica. Los componentes de OnTour son los siguientes:

1. Una ontología diseñada especialmente para este proyecto.
2. Una base de conocimiento con datos provenientes de proveedores de la industria turística; dichos datos cumplen con una estructura específica para el modelo de conocimiento, la ontología.
3. Un mecanismo de inferencia para la recuperación de datos desde la base de conocimiento.

4. Un programa central implementado en JAVA para la lógica del sistema.
5. Una interfaz de usuario o sitio Web implementado en HTML, para la búsqueda y presentación de resultados.

Para la alimentación de la base de conocimiento de OnTour es necesario contar con una herramienta de anotación de metadatos, de manera que la información pueda sea estructurada para esta ontología específicamente.

OnTour fue desarrollado en la Universidad de Innsbruck, en Viena, Austria, a partir de 2004, y auspiciado por la Sociedad de Investigación Austriaca (Austrian Research Promotion Agency, F.F.G.) y DERI International (Digital Enterprise Research Institute).

2.9.3 COTRIN

El proyecto COTRIN [67] (Comprehensive Ontology for the Travel Industry – por sus siglas en inglés), nace como una idea para promover el intercambio eficiente de información entre los diferentes segmentos de la industria turística, de manera que puedan utilizarse las especificaciones basadas en XML de OTA (Open Travel Alliance), con la finalidad de compartir servicios Web autónomos y heterogéneos.

El objetivo principal de este sistema es la creación de paquetes dinámicos de viaje basándose en las preferencias del usuario, a partir de la información contenida en una ontología turística creada específicamente para este proyecto, y a la vez, ajustándose a los estándares que rigen el turismo europeo tales como QoS (Quality of Service).

El diseño del sistema consiste en la separación del mismo en cuatro capas: integración, inferencia, consultas y paquetes dinámicos. La capa de *integración* incluye toda la información proveniente de fuentes heterogéneas, y se accede mediante conectores específicos para cada uno de los tipos de datos que podrán integrarse como instancias en la ontología. La capa de *inferencia* se implementa a través de reglas para la formación de paquetes semánticos. La capa de *consulta*

proporciona una interfaz para la búsqueda de la información almacenada en la base de conocimiento. Finalmente, la capa de *paquetes dinámicos* es la responsable de la lectura de especificaciones para la construcción de paquetes válidos de acuerdo con la petición original del usuario.

COTRIN fue desarrollado en la Universidad de Madeira, en Portugal, bajo la coordinación del Departamento de Matemáticas e Ingeniería.

2.10 Conclusiones

Se han presentado algunas de las herramientas más utilizadas en cada una de las áreas de trabajo que abarca el desarrollo de la tesis. Por supuesto, existen más que no se mencionan en este Estado del Arte por no cubrir con las características básicas requeridas: interoperabilidad, portabilidad, amplia documentación, eficiencia, de fácil uso o aprendizaje y preferentemente libres. En el capítulo siguiente se muestra el análisis comparativo que se realizó para la selección de las herramientas a utilizar en el presente trabajo de tesis.

Capítulo 3. Metodología propuesta

Capítulo 3. Metodología propuesta

En este capítulo se proporciona una descripción general de las diferentes etapas de la metodología propuesta.

El marco general de la metodología se concentra en un conjunto de tareas o actividades contempladas en el desarrollo de la tesis para llevar a buen término la misma, las cuales se muestran en la figura 3.1.

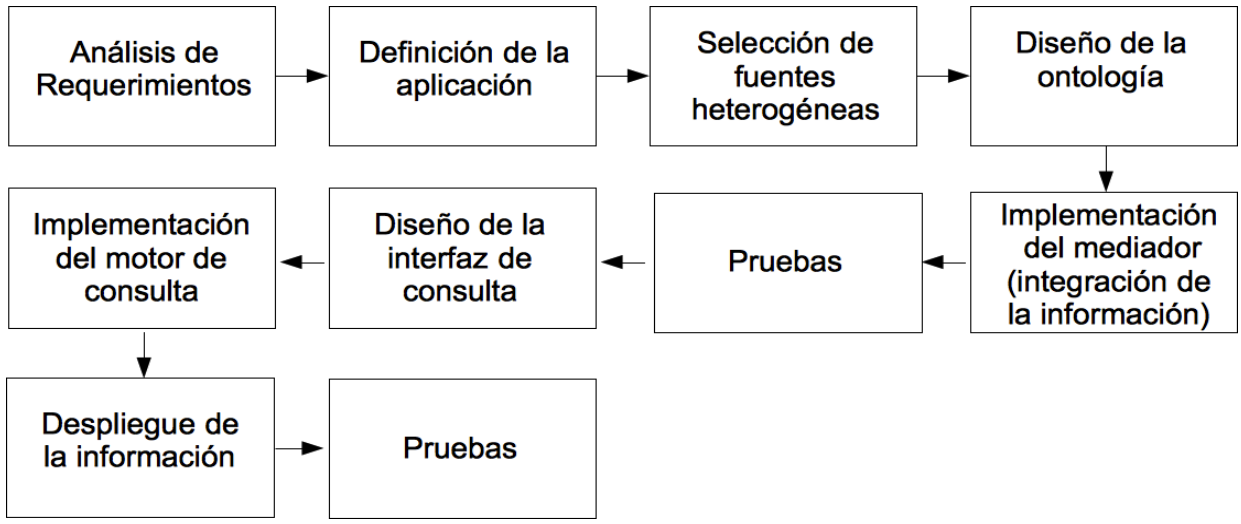

Figura 3.1
Marco general de la metodología

3.1 Análisis de requerimientos

Se requiere una aplicación turística que pueda ser consultada desde la Web, con una estructura semántica de la información como origen de datos, pudiendo provenir ésta de fuentes heterogéneas. A través de dicha aplicación Web podrán realizarse consultas delimitadas en el ámbito turístico, retornando datos tanto alfanuméricos como geográficos. La descripción a grandes rasgos del sistema puede ilustrarse como se presenta en la figura 3.2.

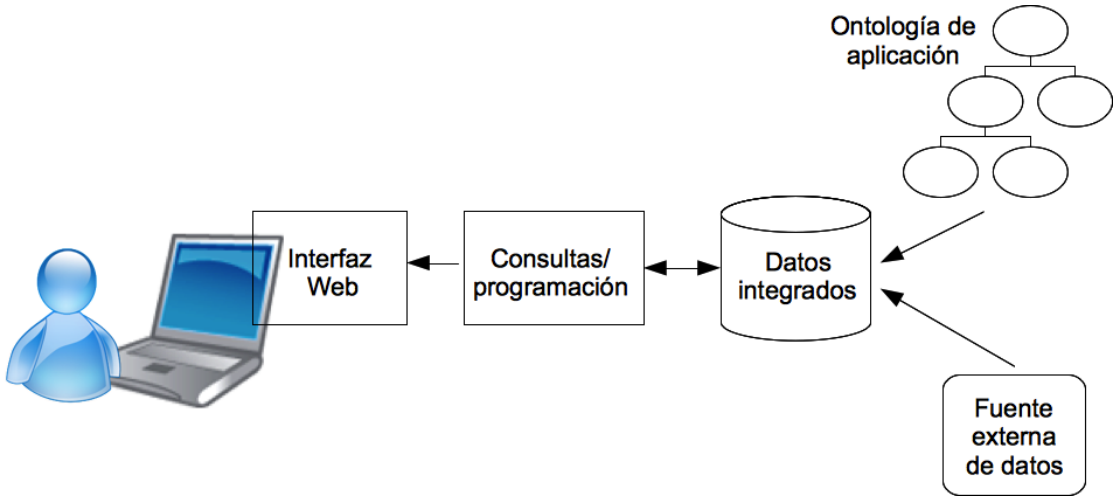

Fig. 3.2

Descripción del funcionamiento general del sistema

Con base en esta descripción, podemos realizar un análisis de los módulos y/o funcionalidades requeridos para el desarrollo del sistema, como se presenta a continuación en la tabla 3.1.

Tabla 3.1
Módulos y/o funcionalidades requeridos para el desarrollo del sistema.

Módulo o funcionalidad	**Herramientas requeridas**
Elaboración de ontología de aplicación	Metodología para el desarrollo de la ontología
	Marco de trabajo para la construcción de la ontología
	Selección de un lenguaje de modelado de ontologías
	Selección de razonador o motor de inferencia para la ontología
Fuente externa de datos por integrar a la ontología	Aplicación para la visualización y/o construcción del formato elegido
Elaboración de módulo	Marco de trabajo para la programación

mediador	Lenguaje de programación
	API para el manejo de información geográfica proveniente de fuente externa de datos
	API para el manejo de información proveniente y destinada a la ontología
Elaboración de módulo de control	Marco de trabajo para la programación
	Lenguaje de programación de aplicaciones Web para generar contenido dinámicamente del lado del cliente y del servidor
Elaboración de módulo de consultas	Marco de trabajo para la programación
	Lenguaje de programación
	API para el manejo de consultas sobre lenguaje ontológico
	Lenguaje de consulta sobre ontologías
Elaboración de la interfaz Web	Marco de trabajo para la programación
	Lenguaje de programación de aplicaciones Web para generar contenido dinámicamente del lado del cliente
	Servicio Web para el despliegue de mapas
	Lenguaje script para la publicación de datos geográficos
	Servidor Web para la aplicación
Adicionalmente, no se omite mencionar la necesidad de una computadora para el usuario con conexión a Internet para accesar al sistema.	

3.2 Definición de la aplicación

El sistema propuesto en el presente trabajo de tesis se encuentra delimitado de la siguiente manera:

Aplicación Web que recupera información turística, geográfica y descriptiva, a partir de fuentes heterogéneas de datos (fuente externa de datos e instancias de la ontología) según los requerimientos del usuario; integra y organiza los datos por medio de una ontología turística de aplicación (previamente diseñada de acuerdo con el diccionario de datos turísticos del INEGI), genera un modelo persistente de datos sobre el cual se realiza la búsqueda solicitada y arroja los resultados al usuario en la interfaz Web, incluyendo la visualización de la localización geográfica de las entidades turísticas solicitadas.

De esta forma, la descripción detallada con módulos y funcionalidades del sistema se muestra en la figura 3.3.

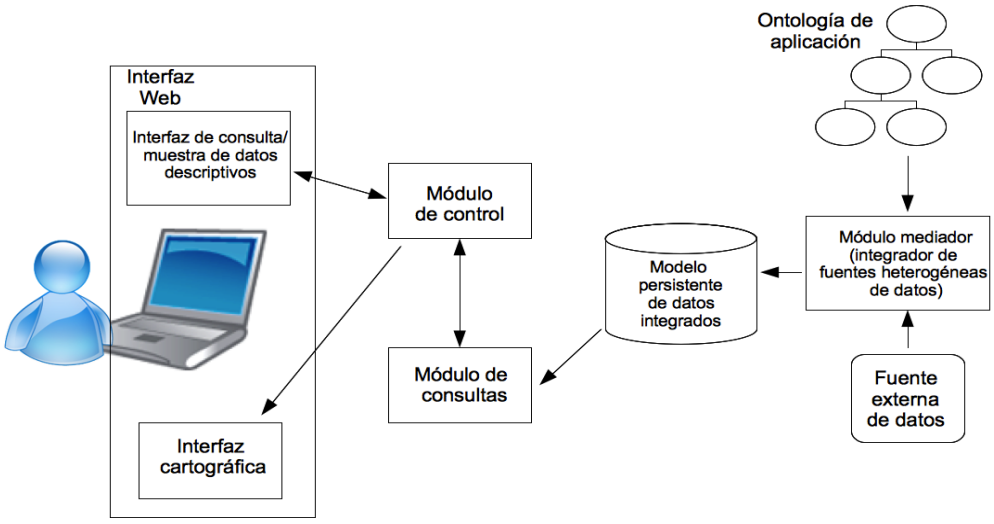

Fig. 3.3

Descripción del sistema por módulos y/o funcionalidades

En cuanto a la selección de las herramientas a utilizar a lo largo del desarrollo de este trabajo, ésta fue realizada con base en su interoperabilidad, portabilidad, documentación, eficiencia, dando preferencia a aquellas que fueran libres y de fácil uso o aprendizaje. A continuación se detalla la selección de dichas herramientas por módulo y/o funcionalidad (ver tabla 3.2).

Tabla 3.2
Selección de herramientas

Módulo o funcionalidad	**Herramienta**	**Descripción**	**Costo de la Licencia**
Elaboración de ontología de aplicación	Methontology	Metodología para el desarrollo de la ontología	Libre
	Protégé	Marco de trabajo para la edición y construcción de la ontología	Libre
	OWL -DL	Selección de un lenguaje de modelado de ontologías	Libre
	Pellet	Selección de razonador o motor de inferencia para la ontología	Libre
Elaboración de shapefile	ESRI ArcMap, ArcCatalog	Aplicación para la visualización y construcción de shapefiles	$1,500.00 US
Elaboración del módulo mediador	NetBeans	Marco de trabajo para la programación	Libre
	JAVA	Lenguaje de programación	Libre
	GeoTools	API para el manejo de información geográfica proveniente del shapefile	Libre
	JDOM	API para el manejo de información proveniente y destinada a la ontología	Libre

Elaboración del módulo de control	JSP, Java Servlets	Lenguaje de programación de aplicaciones Web para generar contenido dinámicamente del lado del cliente y del lado del servidor	Libre
Elaboración de módulo de consultas	JAVA	Lenguaje de programación	Libre
	JENNA	API para el manejo de consultas sobre lenguaje ontológico	Libre
	SPARQL	Lenguaje de consulta sobre ontologías	Libre
Elaboración de la interfaz Web	JSP	Lenguaje aplicaciones Web para generar contenido dinámico del lado del cliente	Libre
	Mash-up para GoogleMaps	Servicio Web para el despliegue de mapas	Libre
	Javascript, JSON	Lenguaje script para la publicación de datos geográficos	Libre
	GlassFish	Servidor Web para la aplicación	Libre

La decisión de utilizar cada una de las herramientas mencionadas fue realizada con base en un análisis comparativo entre las tecnologías existentes actualmente en el mercado:

Methontology

Para el desarrollo de la ontología de aplicación para sitios turísticos se adoptará la metodología Methontology [21], la cual está inspirada en las metodologías de desarrollo de software. Esta metodología divide el proceso en tres fases:

1.- Actividades de administración del proyecto

Planificación: ¿Qué tareas se han de realizar y cuándo?

Control: Garantizar que las tareas se realizan como se deben realizar.

Control de Calidad: Asegurar que el resultado tiene la calidad esperada.

2.- Actividades orientadas al desarrollo

Especificación: Por qué y para qué se desarrolla la ontología, fuentes de conocimiento, alcances.

Conceptualización: Determinar las estructuras que representan el modelo.

Formalización: Transformar el modelo conceptual en un modelo formal o semi computable (lógica, frames).

Implementación: Transformación del modelo a un lenguaje de programación (en este caso OWL, que es generado por Protégé).

Mantenimiento: Actualizar y corregir la ontología.

3.- Actividades de soporte

Adquisición del conocimiento: Uso de procesos estándar de adquisición del conocimiento.

Evaluación: Evaluar las ontologías resultantes.

Integración: Reutilización de ontologías.

Documentación: Detallar cada una de las fases y productos obtenidos en el proceso de creación de la ontología.

Protégé [22]

Este editor de ontologías fue elegido por las siguientes razones:

1. Está basado en la plataforma de desarrollo de Java, con una arquitectura de tres capas, de acceso transparente vía código nativo o de interfaz de programación de aplicaciones.
2. Cuenta con manejo de instancias sobre las clases, restricciones para su generación y

revisión de inconsistencias.

3. La extensibilidad de la herramienta se basa en *plug-ins*.
4. Permite el almacenamiento de ontologías por medio de archivos OWL. Asimismo, permite importar de lenguajes como RDF y XML, y exportar a XML, RDF, FLogic y CLIPS.
5. La representación del conocimiento se basa en marcos, FOL y metaclases; mantiene la representación gráfica basada en una taxonomía y cuenta con un mecanismo de poda gráfica, dadas las restricciones y relaciones.

OWL-DL

OWL [23] ofrece mayor expresividad y características de la que tiene RDF/RDF-Schema, entre otras, la posibilidad de definir restricciones de cardinalidad, propiedad transitiva o la inversa; contiene los constructores del lenguaje con restricciones jerárquicas y resolubilidad (todos los cálculos se resolverán en un tiempo finito), definiciones de clase en forma extensiva y expresiones de clase que incluyan operadores de unión, intersección o complemento.

Pellet

El razonador Pellet [24] fue seleccionado como el motor de inferencias para nuestra aplicación por las siguientes razones:

1. Aplicación de código libre y abierto, con amplio soporte de parte de su comunidad de desarrolladores.
2. Proporciona una interfaz de programación de aplicaciones en Java para su integración.
3. Posee una interfaz nativa para Jena.

ESRI ArcMap y ArcCatalog

A pesar de ser la única de las herramientas en esta tesis que no es de distribución libre, ArcGIS [25] fue seleccionado por ser la que provee las mayores funcionalidades y facilidades para trabajar en la consulta y construcción de los shapefiles, formato de archivo de datos espaciales propietario de ESRI.

NetBeans IDE [26]

Proporciona un entorno de desarrollo muy completo con un número importante de módulos disponibles para extensión; además de ser un producto libre y gratuito sin restricciones de uso.

JAVA

El lenguaje Java [27] proporciona APIs para las principales funcionalidades requeridas en el sistema, además de ser libre, abierto, orientado a objetos, de amplia popularidad y de contar con la documentación necesaria para su implementación.

GeoTools

GeoTools [28] es una biblioteca SIG de código libre que permite desarrollar soluciones adaptadas a los estándares existentes actualmente; proporciona una implementación de las especificaciones del Open Geospatial Consortium para actualizaciones.

Está escrito en el lenguaje de programación Java y se encuentra actualmente en un desarrollo activo al nutrirse de una comunidad de usuarios muy dinámica. Su diseño y concepción modular hace que numerosas implementaciones de software libre en el ámbito de los Sistemas de Información Geográfica hagan uso de los desarrollos de GeoTools.

JDOM

JDOM [29] es una biblioteca de código abierto para manipular datos XML optimizados en Java. A pesar de su similitud con DOM del consorcio World Wide Web (W3C), es una alternativa para modelado de objetos que no está incluido en DOM. La principal diferencia es que mientras DOM fue creado para ser un lenguaje neutral e inicialmente usado para manipulación de páginas HTML con JavaScript, JDOM se creó específicamente para usarse con Java y por tanto beneficiarse de sus características, incluyendo sobrecarga de métodos, colecciones, etc.

JSPs

JavaServer Pages (JSP) [30] es una tecnología Java que permite generar contenido dinámico para la Web, en forma de documentos HTML, XML o de otro tipo. Permiten la utilización de código Java mediante scripts. Además, es posible utilizar algunas acciones JSP predefinidas mediante etiquetas. Estas etiquetas pueden ser enriquecidas mediante la utilización de Bibliotecas de Etiquetas (TagLibs o Tag Libraries) externas e incluso personalizadas.

Servlets

Los *servlets* [31] son objetos que corren dentro y fuera del contexto de un contenedor de *servlets* y extienden su funcionalidad; su uso más común es generar todas páginas web de forma dinámica a partir de los parámetros de la petición que envíe el navegador web.

Jena

Los motivos para elegir Jena [32] como marco de trabajo en la construcción de nuestra ontología de aplicación son los siguientes:

1. Desarrollo en Java, de código libre y abierto, con amplio soporte a usuarios a través de su lista de correos.
2. Proporciona interfaces de programación para RDF, RDFS y OWL.
3. Cuenta con su propio motor de consultas SPARQL (Simple Protocol And RDF Query Language).
4. Permite almacenamiento en memoria y persistente.
5. Posee un motor de inferencias basado en reglas, así como la capacidad de conectarse con un razonador independiente.

SPARQL

SPARQL [33] es un acrónimo recursivo del inglés *Simple Protocol and* RDF *Query Language;* lenguaje estandarizado para la consulta de grafos RDF, normalizado por el RDF Data Access Working Group (DAWG) del Word Wide Web Consortium (W3C). Es una tecnología clave en el desarrollo de la Web Semántica, recomendación oficial del W3C .

Al igual que sucede con SQL, es necesario distinguir entre el lenguaje de consulta y el motor para el almacenamiento y recuperación de los datos. Por este motivo, existen múltiples implementaciones de SPARQL, generalmente ligados a entornos de desarrollo y plataforma tecnológicas.

Google Maps

La aplicación para la visualización de mapas vía Web seleccionado fue Google Maps [34] por las siguientes razones:

1. Realiza localizaciones a partir de latitud y longitud dados.
2. Los mapas pueden ser generados en cualquier página Web mediante los archivos KML.
3. Es posible agregar texto HTML o XML en ventanas de información.

4. Uso no restringido hasta un límite razonable (mayor a 50,000 visitas).
5. Proporciona una interfaz amigable al usuario.

Javascript [35]

Es un lenguaje de programación interpretado, dialecto del estándar ECMAScript, basado en prototipos, imperativo, débilmente tipeado y dinámico. Se utiliza principalmente en su forma del lado del cliente (*client-side*), implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas. Todos los navegadores modernos interpretan el código JavaScript integrado en las páginas web. Para interactuar con una página web se provee al lenguaje JavaScript de una implementación del Document Object Model (DOM).

JSON

JSON [36], acrónimo de *JavaScript Object Notation*, es un formato ligero para el intercambio de datos. JSON es un subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de XML.

Una de las ventajas de JSON sobre XML como formato de intercambio de datos en este contexto es que es mucho más sencillo escribir un analizador sintáctico (parser) de JSON. En JavaScript, un texto JSON se puede analizar fácilmente usando el procedimiento `eval()`, lo cual ha sido fundamental para que JSON haya sido aceptado por parte de la comunidad de desarrolladores AJAX, debido a la ubicuidad de JavaScript en casi cualquier navegador web.

GlassFish

GlassFish Server [37] proporciona un servidor para el desarrollo y puesta en producción de aplicaciones Java EE y Java Web Services. Algunas de sus características más importantes incluyen modularidad y extensibilidad, soporte JEE, desarrollo rápido e iterativo, contenedores de lenguajes script, soporte para Tecnologías de Interoperabilidad de Servicios Web (WSIT), servicios Web de alto rendimiento, seguridad y capacidades de integración.

3.3 Selección de fuentes de datos

Por último, para la selección de fuentes de datos se consideró el uso del tipo de archivo shapefile [38] para ser integrado a la ontología por ser el formato más utilizado en los sistemas de información geográficos libres y propietarios.

De esta manera, la selección de herramientas para cada funcionalidad de la aplicación se presentan en la figura 3.4.

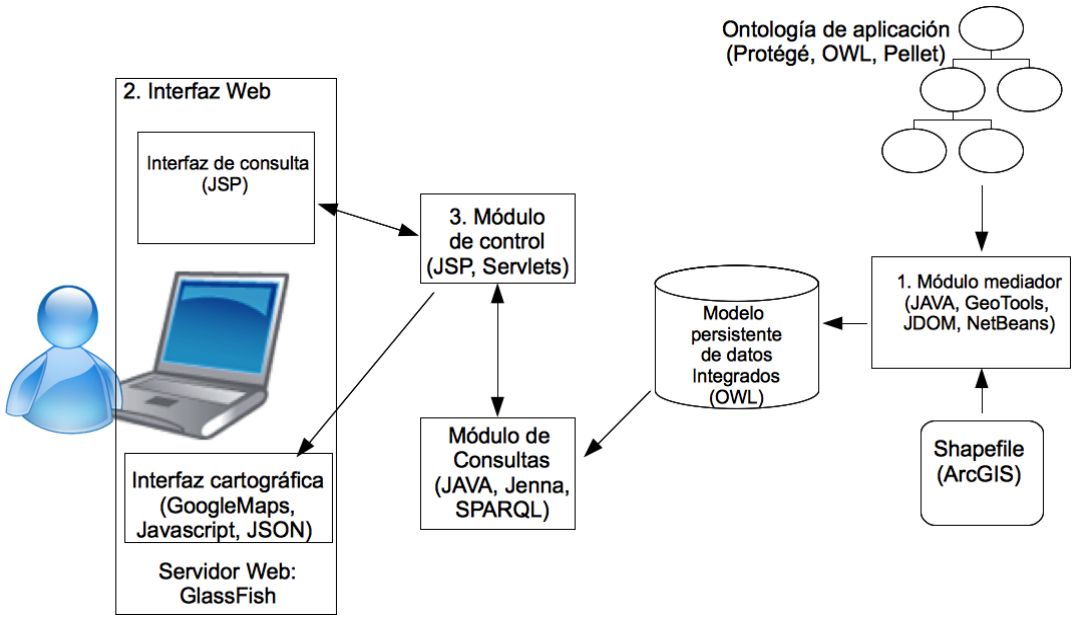

Figura 3.4
Diagrama general del funcionamiento del sistema con las herramientas de desarrollo seleccionadas

3.4 Diseño e implementación de la ontología

Para realizar el diseño de la ontología se utilizó la metodología METHONTOLOGY [21], trabajando las primeras fases de dicha metodología con base en la organización de la información proporcionada por el diccionario de datos turísticos del INEGI [14]. Como es el caso de las bases de conocimiento, la presente ontología con las instancias contenidas en ella es susceptible de ser reutilizada por cualquier aplicación en el dominio de interés turístico.

En el siguiente capítulo de esta tesis podrá encontrarse la documentación del diseño y la implementación de la ontología turística a realizarse.

3.5 Implementación del mediador

El módulo mediador se encarga de integrar fuentes de datos heterogéneas en una sola ontología. Como orígenes de datos, en el presente trabajo se contemplan dos tipos de archivos:

- Una ontología de aplicación modelada de acuerdo con el Diccionario de Datos Turísticos del INEGI, conteniendo instancias de entidades turísticas.
- Un *shapefile* con datos similares a los contenidos en la ontología.

Ambas fuentes de datos son integradas en la estructura de la ontología, a través del módulo **mediador**, que es el encargado de realizar un mapeo entre cada registro del shapefile y las instancias de la ontología, de manera que todas las entradas existentes en el shapefile sean integradas a la ontología como sus instancias, respetándose la organización, jerarquización y restricciones de los datos inherentes a la misma. A partir de todas la instancias integradas de esta manera, se genera un modelo persistente de datos sobre el cual se realizan las búsquedas solicitadas por el usuario.

3.6 Diseño de la interfaz de consultas

El usuario puede acceder a la aplicación a través de una interfaz Web, por medio de la cual es posible realizar una consulta de información turística contenida en la ontología integrada. Las opciones disponibles de consulta de datos son acotadas en el caso de estudio de la aplicación en el siguiente capítulo de esta tesis.

3.7 Implementación del motor de consultas

A este módulo se envían los parámetros solicitados por el usuario, provenientes del módulo de control. De esta manera se realizan las consultas sobre el modelo persistente de datos integrado, del cual se extrae la información que se devuelve de nueva cuenta al módulo de control.

A su vez, el módulo de control se encarga de captar y responder las consultas del usuario mediante la invocación de los procesos requeridos al módulo de consultas, comunicándose con la interfaz Web para la obtención de las consultas solicitadas por el usuario y posteriormente arrojando las respuestas provenientes del módulo de consultas de vuelta a la interfaz Web.

3.8 Despliegue de la información

Como resultado de la consulta realizada por el usuario, se integra en la misma interfaz de consulta un servicio Web que permite visualizar el mapa de la entidad turística solicitada a la vez que son visibles los datos alfanuméricos arrojados en la consulta solicitada por el usuario.

Capítulo 4. Resultados Experimentales

Capítulo 4. Resultados experimentales

En este capítulo se describen la implementación de la ontología turística, el caso de estudio seleccionado y los resultados de la aplicación.

4.1 Diseño e implementación de la ontología

Como se menciona en el capítulo anterior, el diseño de la ontología se realizó siguiendo la metodología Methontology [21]. Dicha metodología se divide en seis fases:

1. Especificación
2. Conceptualización
3. Formalización
4. Integración
5. Implementación
6. Mantenimiento

Otras tareas que se llevan a cabo a lo largo de todo el ciclo de vida de la ontología son:

- Adquisición del conocimiento
- Documentación
- Evaluación

Las primeras dos fases de dicha metodología fueron completadas con base en la organización de la información proporcionada por el diccionario de datos turísticos del INEGI [14]. Sin embargo, para enfocar la implementación de la ontología de manera que satisfaga los requerimientos para este sistema en particular, se observaron aquellas tareas encaminadas específicamente a dicho fin. A continuación se hace una revisión de las tareas documentadas:

4.1.1 Especificación

Para la fase de especificación, Methontology propone una serie de tareas tales como:

- Definición del Propósito de la Ontología

La ontología de aplicación tiene como propósito la recuperación de información descriptiva y geoespacial de sitios turísticos en México.

- Definición de los Alcances

De acuerdo con la clasificación proporcionada por el INEGI, se encuentra limitado a la representación de entidades turísticas de México.

- Vigencia de la Ontología

La vigencia de la ontología se encuentra determinada por los cambios climáticos y desastres naturales presentes en la zona turística.

- Uso que se le dará a la ontología

La presente ontología está pensada para la organización del diccionario de entidades turísticas y posteriormente la creación de un modelo persistente de datos.

Preguntas de competencia:

- ¿Donde está el sitio arqueológico de Edzná?
- ¿El hotel del Mar tiene discoteca?
- ¿Dónde está la cenaduría de San Francisco?
- ¿Qué hoteles tienen playa?
- ¿Dónde puedo comprar artesanías?
- ¿Cuál es la dirección del Baluarte de Santa Rosa?
- ¿Hay algún restaurante con vista al mar?
- ¿Dónde está el paradero del turibús?
- Lista de hoteles por número de estrellas o precios
- Lista de discotecas en la ciudad

- Lista de hostales por precio
- Lista de restaurantes de comida tradicional
- ¿Dónde están los principales museos de la ciudad?
- ¿Dónde están las zonas arqueológicas del área?

4.1.2. Conceptualización

Esta etapa consiste en organizar y convertir una percepción informal del dominio de interés en una especificación formal por medio de una ontología.

Siguiendo las tareas de conceptualización de METHONTOLOGY [68] se muestran los componentes de la ontología (conceptos, atributos, relaciones, constantes, axiomas formales, reglas e instancias) construidos en cada tarea y se ilustra el orden propuesto para crear tales componentes en la figura 4.1.

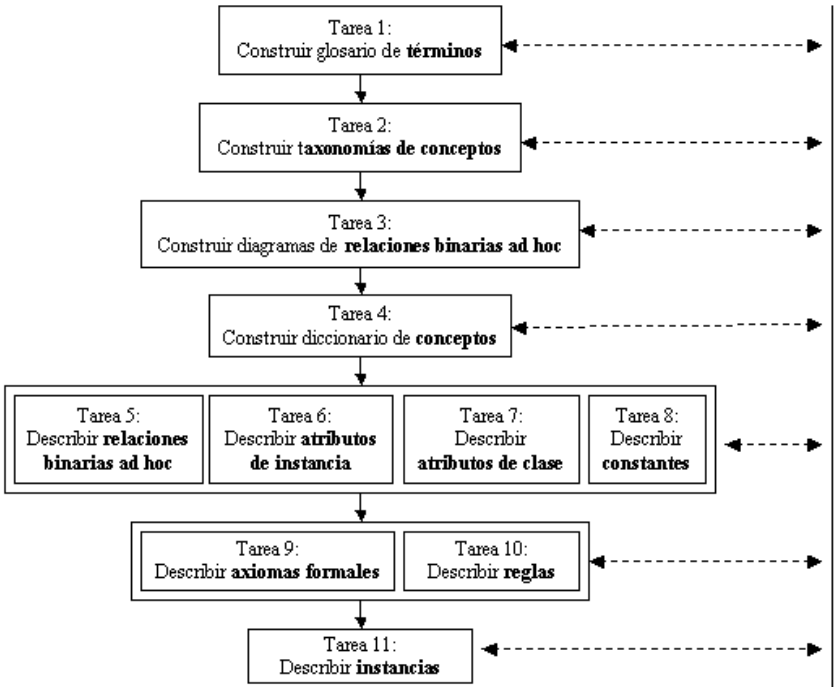

Fig. 4.1

Tareas de la actividad de Conceptualización según METHONTOLOGY

Dentro de la fase de conceptualización, encontramos las siguientes tareas:

Tarea 1: Construir glosario de términos

Los conceptos que son utilizados para la implementación de esta ontología se encuentran definidos en el diccionario de datos turísticos del INEGI [14]. Siendo ésta la institución encargada de la obtención, tratamiento y análisis de información relevante a nivel nacional, se pretende que este trabajo sirva como base para futuras líneas de investigación afines al presentado en esta tesis.

Tarea 2: Construir taxonomía de conceptos

Con base en los términos encontrados en el diccionario del INEGI, se llevó a cabo una clasificación de los mismos, describiendo estos elementos en la tabla 4.1.

Tabla 4.1
Organización de la ontología: conceptos, clases y atributos

Concepto	Clase	Atributos
Sitios Turísticos	Turismo Campeche/Owl Object	▪ descripciónSitio ▪ direccionSitio ▪ paginaWebSitio ▪ latitud ▪ longitud
Espectáculos	Sitios Turísticos	▪ idEspectaculo ▪ capacidadEspectaculo ▪ telefonoEspectaculo ▪ estacionamientoEspectaculo ▪ horarioEspectaculo ▪ emailEspectaculo
Auditorio	Espectáculos	▪ nombreAuditorio
Estadio	Espectáculos	▪ nombreEstadio
Teatro	Espectáculos	▪ nombreTeatro
Foro	Espectáculos	▪ nombreForo
Cine	Espectáculos	▪ nombreCine
Plaza de Toros	Espectáculos	▪ nombrePlazaToros
Zoológico	Espectáculos	▪ nombreZoologico
Centros Nocturnos	Espectáculos	▪ extrasCentroNocturno
Antro	Centros Nocturnos	▪ tipoAntro ▪ tipoMusicaAntro ▪ nombreAntro

Bar	Centros Nocturnos	▪ nombreBar
Mens Club	Centros Nocturnos	▪ nombreMensClub
Parques	Sitios Turísticos	▪ idParque ▪ horarioParque ▪ telefonoParque
Acuáticos	Parques	▪ costoParqueAcuatico
Acuario	Acuaticos	▪ nombreAcuario
Balneario	Acuaticos	▪ numeroToboganes ▪ nombreBalneario
Nado con Delfines	Acuaticos	▪ nombreNadoDelfines
Snorkel	Acuaticos	▪ nombreSnorkel
Deportivos	Sitios Turisticos	▪ costoParqueDeportivo
Campo de Golf	Deportivos	▪ numeroHoyosGolf ▪ nombreCampoGolf
Sitio Para Buceo	Deportivos	▪ nombreBuceo
Sitio Para Navegación	Deportivos	▪ nombreNavegacion
Sitio Para Pesca	Deportivos	▪ nombrePesca
Sitio Para Rapel	Deportivos	▪ nombreRapel
Sitio Para Canotaje	Deportivos	▪ nombreCanotaje
Zona Para Montar a Caballo	Deportivos	▪ nombreZonaCaballo
Zona de Bicicletas	Deportivos	▪ nombreZonaBicicletas
Diversiones	Parques	▪ atraccionesDiversiones ▪ admisionesDiversiones
Recreativos	Parques	
Pista de Patinaje	Recreativos	▪ costoPistaPatinaje ▪ nombrePistaPatinaje
Parque Público	Recreativos	▪ nombrePublico
Temático	Parques	▪ admisionesTematico ▪ atraccionesTematico ▪ nombreTematico
Playa	Sitios Turísticos	▪ idPlaya ▪ nombrePlaya ▪ sitioAcampar ▪ tipodeOla ▪ tipodePlaya ▪ zonaPlaya
Plazas y Mercados	Sitios Turísticos	▪ idPlaza ▪ tipoPlaza ▪ horarioPlaza ▪ telefonoPlaza
Artesanías	Plazas y Mercados	▪ nombreArtesanias
Centro Comercial	Plazas y Mercados	▪ nombreCentroComercial

Servicios	Sitios Turísticos	▪ idServicio ▪ telefonoServicio
Cafe	Servicios	▪ nombreSitio ▪ direccionSitio
Gasolinera	Servicios	▪ nombreGasolinera
Hospedaje	Servicios	▪ costoHospedaje ▪ zonaHospedaje
Casa Huéspedes	Hospedaje	▪ nombreCasaHuespedes
Hotel	Hospedaje	▪ nombreSitio ▪ costoHospedaje ▪ latitud ▪ longitud ▪ idServicio ▪ direccionSitio ▪ descripcionSitio ▪ zonaHospedaje ▪ numeroEstrellasHotel ▪ paginaWebSitio ▪ telefonoServicios ▪ tiposHabitacionHotel
Motel	Hospedaje	▪ nombreMotel
Zona Para Acampar	Hospedaje	▪ nombreZonaParaAcampar
Restaurante	Servicios	▪ codigoVestir ▪ nombreSitio ▪ idServicio ▪ direccionSitio ▪ descripcionSitio ▪ paginaWebSitio ▪ telefonoServicios ▪ ClaseRestaurante ▪ TipoComidaRestaurante
Sitios Culturales	Sitios Turísticos	▪ idCultural
Arquitectura Civil	Sitios Culturales	▪ estadoConservacionCivil ▪ estiloArquitecturaCivil ▪ sigloCivil
Edificio Civil	Arquitectura Civil	▪ nombreEdificioCivil
Mirador	Arquitectura Civil	▪ nombreMirador
Monumento Civil	Arquitectura Civil	▪ arquitectoMonumento ▪ nombreMonumentoCivil
Plazas y Explanadas	Arquitectura Civil	▪ nombrePlazasYExplanadas
Presa	Arquitectura Civil	▪ nombrePresa
Arquitectura Religiosa	Sitios Culturales	▪ estadoReligiosa ▪ estiloReligiosa ▪ sigloReligiosa

Basílica	Arquitectura Religiosa	▪ nombreBasilica
Capilla	Arquitectura Religiosa	▪ nombreCapilla
Catedral	Arquitectura Religiosa	▪ nombreCatedral
Convento	Arquitectura Religiosa	▪ nombreConvento
Templo	Arquitectura Religiosa	▪ nombreTemplo ▪ religionTemplo
Museo	Sitios Culturales	▪ categoríaMuseo ▪ costoMuseo ▪ horarioMuseo ▪ nombreMuseo ▪ telefonoMuseo ▪ tipoMuseo
Sitio Arqueológico	Sitios Culturales	▪ culturaArqueologia ▪ nombreSitioArqueologico ▪ periodoArqueologia
Transportes	Sitios Turísticos	▪ idTransporte ▪ telefonoTransporte
Transporte Aéreo	Transportes	
Aeropuerto	Transporte Aéreo	▪ nombreAeropuerto ▪ operadorAeropuerto
Transporte Marítimo	Transportes	
Terminal Marítima	Transporte Marítimo	▪ nombreTerminalMaritima
Transporte Terrestre	Transportes	
Renta De Autos	Transporte Terrestre	▪ horarioRentaAutos ▪ nombreRentaAutos
Terminal De Autobuses	Transporte Terrestre	▪ nombreTerminalAutobuses
Zona Natural	Sitios Turísticos	▪ idZonaNatural
Cascada	Zona Natural	▪ nombreCascada
Gruta	Zona Natural	▪ nombreGruta
Lago	Zona Natural	▪ nombreLago
Laguna	Zona Natural	▪ nombreLaguna
Monumento Natural	Zona Natural	▪ nombreMonumentoNatural
Parque Nacional	Zona Natural	▪ nombreParqueNacional
Reserva Natural	Zona Natural	▪ nombreReservaNatural
Rio	Zona Natural	▪ nombreRio

Una vez jerarquizada la información, se realizó el diseño de la ontología en Protégé. En el diseño se consideraron ocho clases principales según la clasificación del INEGI, como se puede apreciar en la siguiente figura 4.2.

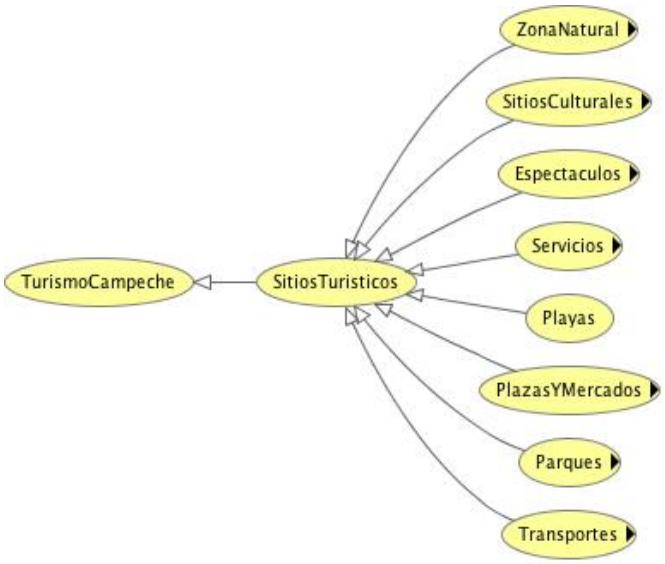

Fig. 4.2
Representación de clases según la clasificación del INEGI

Por cada una de las ocho clases principales existen varias subclases, que pueden estar subdivididas en más clases a su vez, de acuerdo con la organización mostrada en la Tabla 4.1; igualmente se definieron propiedades, restricciones y tipos de datos para la implementación de las instancias de cada clase. En el Anexo A pueden consultarse las subclases implementadas para esta ontología.

Cabe señalar que, dado el alcance acotado para el presente trabajo, la implementación de instancias, propiedades y reglas de la ontología fue delimitada para la clase de **Servicios,** subclases **Hospedaje** (subclase **Hoteles), Café, Restaurant** y **Bar,** para un total de 27 instancias de la clase Hotel, 2 instancias de la clase Café, 2 de la clase Bar y 1 de Restaurant, todas ellas realizadas a través de Protégé, además de otras 10 instancias de la clase Hotel implementadas a través del shapefile que se integra posteriormente a la ontología (ver figura 4.3).

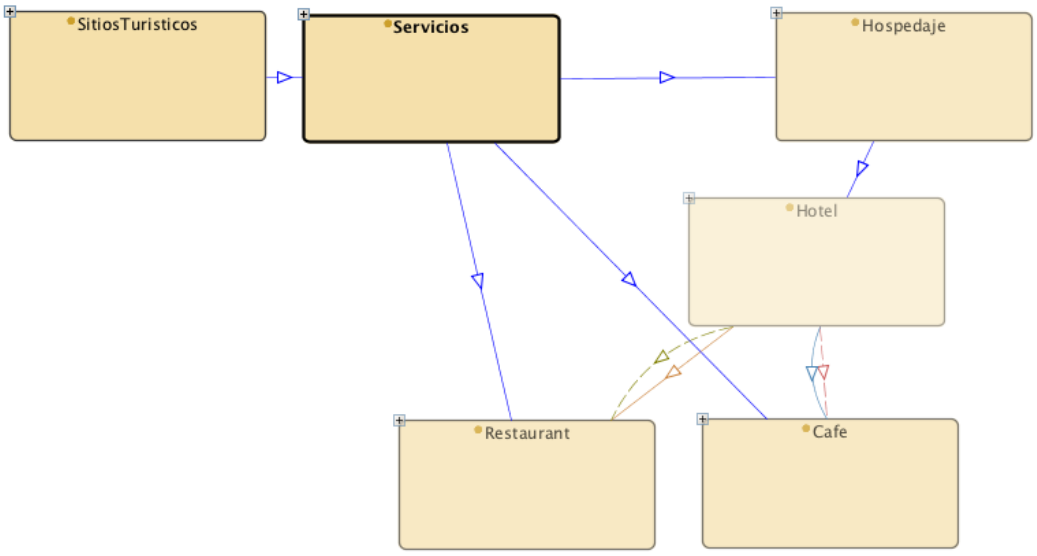

Fig. 4.3
Clases implementadas

Tarea 3: Construir un diagrama de relaciones binarias

A través del diagrama de relaciones binarias se establecen las relaciones existentes entre los conceptos de la misma o de distintas taxonomías de conceptos. En la figura 4.4 se muestra lo anterior.

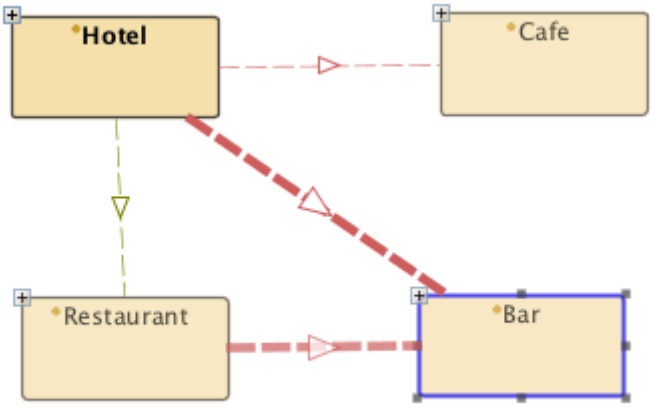

Fig. 4.4
Relaciones binarias

Tarea 4: Construir diccionario de conceptos

El diccionario de conceptos contiene todos los conceptos del dominio, sus relaciones, sus instancias, y sus atributos de clase y de instancia. Las relaciones especificadas para cada concepto son aquellas en las que el concepto es el origen de la misma (ver tabla 4.2).

Tabla 4.2

Diccionario de conceptos

Nombre del concepto	Instancias	Atributos de instancia	Atributos de clase	Relaciones
Hotel	Alhambra America Avenida Baluartes Bambu Campeche Casa_Don_Gustavo Castelmar City_Express_Campeche Del_Mar Del_Paseo Francis_Drake H177 Hacienda_Uayamon Holiday_Inn Hostal_del_Pirata Mayan_Campeche Ocean_View Plaza_Campeche Plaza_Colonial Santa_Marina Uxul_Kah Villa_Campeche	nombreSitio costoHospedaje latitud longitud idServicio direccionSitio descripcionSitio zonaHospedaje numeroEstrellasHotel paginaWebSitio telefonoServicios tiposHabitacionHotel	-	hasABar hasARestaurant hasACafe
Bar	ElPolvorin Laffite	nombreSitio idServicio direccionSitio descripcionSitio paginaWebSitio telefonoServicios	-	-

Restaurant	ElBicentenario	nombreSitio idServicio direccionSitio descripcionSitio paginaWebSitio telefonoServicios ClaseRestaurante TipoComidaRestaurante	-	hasABar -
Cafe	DelParque ElPoquito	nombreSitio idServicio direccionSitio descripcionSitio paginaWebSitio telefonoServicios	-	-

Tarea 5: Describir relaciones binarias

El objetivo de esta tarea es describir en detalle todas las relaciones binarias identificadas en el diagrama de relaciones binarias e incluidas en el diccionario de conceptos. Para cada relación binaria, se debe especificar su nombre, los nombres de sus conceptos origen y destino, su cardinalidad y su relación inversa, si existe (ver tabla 4.3).

Tabla 4.3

Descripción de relaciones binarias

Nombre de la relación	Concepto origen	Cardinalidad máxima	Concepto destino	Relación inversa
hasABar	Hotel	N	Bar	-
hasABar	Restaurant	N	Bar	-
hasACafe	Hotel	N	Café	-
hasARestaurant	Hotel	N	Restaurant	-

Tarea 6: Describir los atributos de instancia

El objetivo de esta tarea es describir en detalle todos los atributos de instancia incluidos en el diccionario de conceptos. Los atributos de instancia describen a las instancias del concepto y sus valores pueden ser distintos para cada una de dichas instancias.

Por cada atributo de instancia, se debe especificar su nombre, el concepto al que el atributo pertenece, su tipo de valor, su rango de valores y sus cardinalidades mínima y máxima (ver tabla 4.4).

Tabla 4.4
Atributos de instancia para el concepto Hotel en OntoCampeche

Nombre de atributo de instancia	Concepto	Tipo de valor	Rango de valores	Cardinalidad
nombreSitio	Hotel	cadena de caracteres	-	(1,1)
costoHospedaje	Hotel	cadena de caracteres	-	(1,1)
latitud	Hotel	cadena de caracteres	-	(1,1)
longitud	Hotel	cadena de caracteres	-	(1,1)
idServicio	Hotel	número entero	-	(1,1)
direccionSitio	Hotel	cadena de caracteres	-	(1,1)
descripcionSitio	Hotel	cadena de caracteres	-	(1,1)
zonaHospedaje	Hotel	cadena de caracteres	-	(1,1)
numeroEstrellasHotel	Hotel	número entero	-	(1,1)
paginaWebSitio	Hotel	cadena de caracteres	-	(1,1)
telefonoServicios	Hotel	cadena de caracteres	-	(1,1)
tiposHabitacionHotel	Hotel	cadena de caracteres	-	(1,1)

Tarea 7: Describir los atributos de clase

El objetivo de esta tarea es describir en detalle los atributos de clase incluidos en el diccionario de conceptos. En esta ontología no se desarrolla esta tarea, ya que no fueron declarados atributos de clase.

Tarea 8: Describir las constantes

El objetivo de esta tarea es describir cada una de las constantes identificadas en el glosario de términos. Para cada constante, se debe especificar su nombre, tipo de valor, valor y unidad de medida en el caso de constantes numéricas. En esta ontología no se desarrolla esta tarea, dado que no maneja constantes.

Tarea 9: Definir Axiomas formales

Para realizar esta tarea, se deben identificar los axiomas formales que son necesarios en la ontología y describirlos de manera precisa. Para cada definición de axioma formal, METHONTOLOGY propone especificar la siguiente información: nombre, descripción en lenguaje natural, expresión lógica que define de manera formal el axioma usando lógica de primer orden, y los conceptos, atributos y relaciones utilizadas en el axioma, así como las variables utilizadas (ver tabla 4.5).

Tabla 4.5

Sección de la tabla de axiomas formales

Nombre Axioma	Descripción	Expresión	Concepto	Relaciones	Variables
Localización de bar	Un Bar puede estar localizado en un Hotel o en un Restaurant	Existe (bar(?B), hotel(?H) y restaurant(?R)) tal que ((hasABar(?R, ?B)) ó (hasABar(?H, ?B)))	Bar Hotel Restaurant	hasABar	?B ?H ?R

La tabla 4.5 muestra un axioma formal de la ontología, la cual establece que "un bar puede estar localizado en un hotel o en un restaurante". Las columnas que corresponden a conceptos y relaciones referidas contienen los conceptos y relaciones utilizados en la expresión formal del axioma. Asimismo, las variables utilizadas son ?B para Bar, ?H para Hotel y ?R para Restaurant.

Tarea 10: Definir reglas

De manera similar a la tarea previa, en esta tarea se debe identificar en primer lugar qué reglas se necesitan en la ontología, y entonces describirlas en la tabla de reglas. Para cada regla, METHONTOLOGY propone incluir la siguiente información: nombre, descripción en lenguaje natural, expresión que describe formalmente la regla, y conceptos, atributos y relaciones utilizados en la regla, así como las variables usadas (ver tabla 4.6).

METHONTOLOGY propone especificar las expresiones de las reglas utilizando el formato si <condiciones> entonces <consecuencia>. La parte izquierda de la regla es una conjunción de condiciones simples, mientras que la parte derecha es una simple expresión de un valor de la ontología.

Tabla 4.6

Sección de tabla de reglas definidas

Nombre de la regla	Descripción	Expresión	Conceptos	Atributos	Relaciones	Variables
Bar en Hotel	Un bar está dentro de un hotel si el bar está dentro de un restaurant que a su vez está dentro de un hotel	Si (Hotel(?x) hasARestaurant(?x,?y) y hasABar(?y,?z)) entonces (hasABar((?x,?z))	Hotel Restaurant Bar	-	hasABar hasARestaurant	?x ?y ?z

Tarea 11: Describir instancias

En este apartado se definen las instancias que aparecen en el diccionario de conceptos. Para cada instancia se define: su nombre, el nombre del concepto al que pertenece y los valores de sus atributos de instancia, si se conocen en la figura 4.5 se muestran las instancias generadas en la ontología; así como en la tabla 4.7 se describe una instancia de hotel.

Tabla 4.7
Sección de la tabla de instancias de OntoCampeche

Nombre de la instancia	Nombre del concepto	Atributos	Valores
Laffite	Bar	nombreSitio idServicio direccionSitio descripcionSitio paginaWebSitio telefonoServicios	Laffite - Hotel Del Mar Bar mayores de 30 años - -

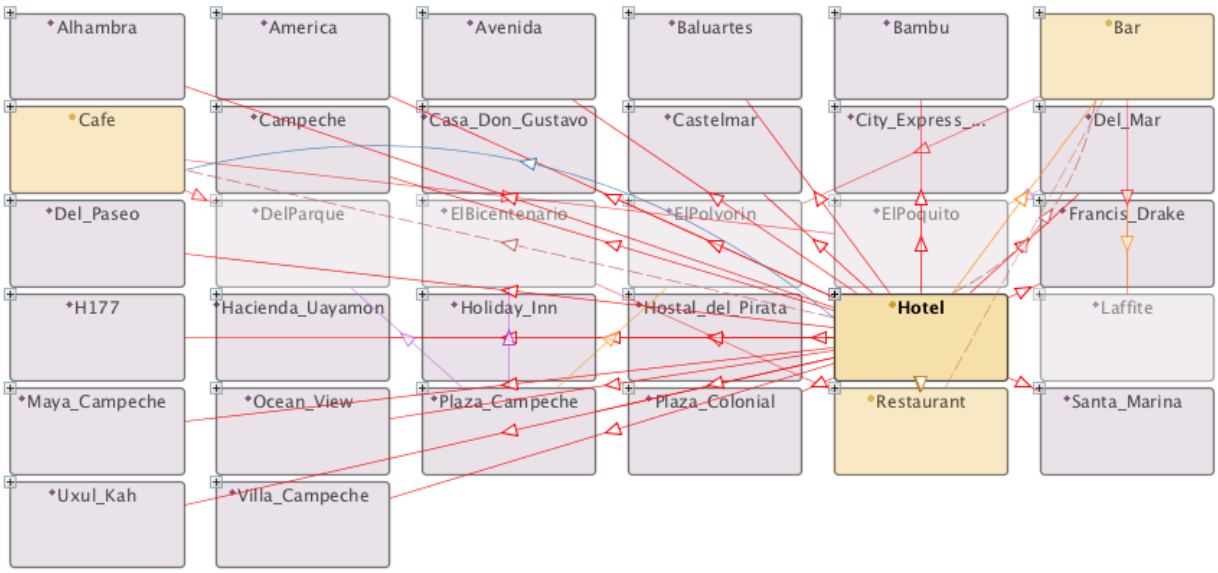

Fig. 4.5
Instancias implementadas a través de Protégé

4.1.3 Formalización

Después de generar el modelo, se realiza la codificación de la ontología utilizando un lenguaje formal; para esto se seleccionó la herramienta de edición Protégé, a través de la cual, al definir la ontología, se genera la codificación en el lenguaje OWL.

4.1.4 Integración

La fase de integración de Methontology consiste propiamente en la reutilización de las definiciones contenidas en el diccionario de datos del INEGI como conceptos básicos para la ontología turística a elaborar en el presente trabajo.

4.1.5 Implementación

La construcción de la ontología se realizó en Protégé, generándose el archivo *OWL* con la estructura de la misma, sus propiedades y especificaciones. Así mismo, se exportaron en este mismo archivo las instancias de hoteles que se dieron de alta para utilizarse en el caso de estudio (ver figura 4.6).

```
1280      >Plaza Campeche</nombreSitio>
1281    </Hotel>
1282    <Hotel rdf:ID="Del_Mar">
1283      <longitud rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
1284      >-90.53913474082947</longitud>
1285      <direccionSitio rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
1286      >Av. Ruiz Cortines No. 51</direccionSitio>
1287      <numeroEstrellasHotel rdf:datatype="http://www.w3.org/2001/XMLSchema#int
1288      >4</numeroEstrellasHotel>
1289      <latitud rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
1290      >19.847744002886362</latitud>
1291      <paginaWebSitio rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
1292      >http://www.delmarhotel.com.mx</paginaWebSitio>
1293      <hasACafe>
1294        <Cafe rdf:ID="ElPoquito"/>
1295      </hasACafe>
1296      <telefonoServicios rdf:datatype="http://www.w3.org/2001/XMLSchema#string
1297      >01(981)8119191</telefonoServicios>
1298      <costoHospedaje rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
1299      >1000</costoHospedaje>
1300      <hasABar>
1301        <Bar rdf:ID="Laffite"/>
1302      </hasABar>
1303      <nombreSitio rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
1304      >Best Western Hotel Del Mar</nombreSitio>
1305    </Hotel>
1306  </rdf:RDF>
```

Fig. 4.6

Fragmento de TurismoCampeche2.owl, con la ontología turística e instancias implementadas

4.1.5.1 Inferencia en la ontología

El mecanismo de inferencia sobre la ontología para esta tesis está integrado en dos partes:

- **Inferencia sobre Tbox:**

  La consistencia, corrección y organización de la ontología quedan determinadas por la definición de propiedades y axiomas a manera de restricciones sobre la taxonomía construída.

- **Inferencia sobre Abox:**

  La generación de información nueva a partir de los datos depositados en la ontología, a través de instancias se manejó para este caso por medio de reglas, utilizando el lenguaje SWRL (SWRL Tab en Protégé [69]), consistente en lógica de primer orden o de predicados [70]. En la figura 4.7, se muestra la forma en que se introdujo el axioma basado en la regla descrita para inferir conocimiento sobre la ontología desarrollada.

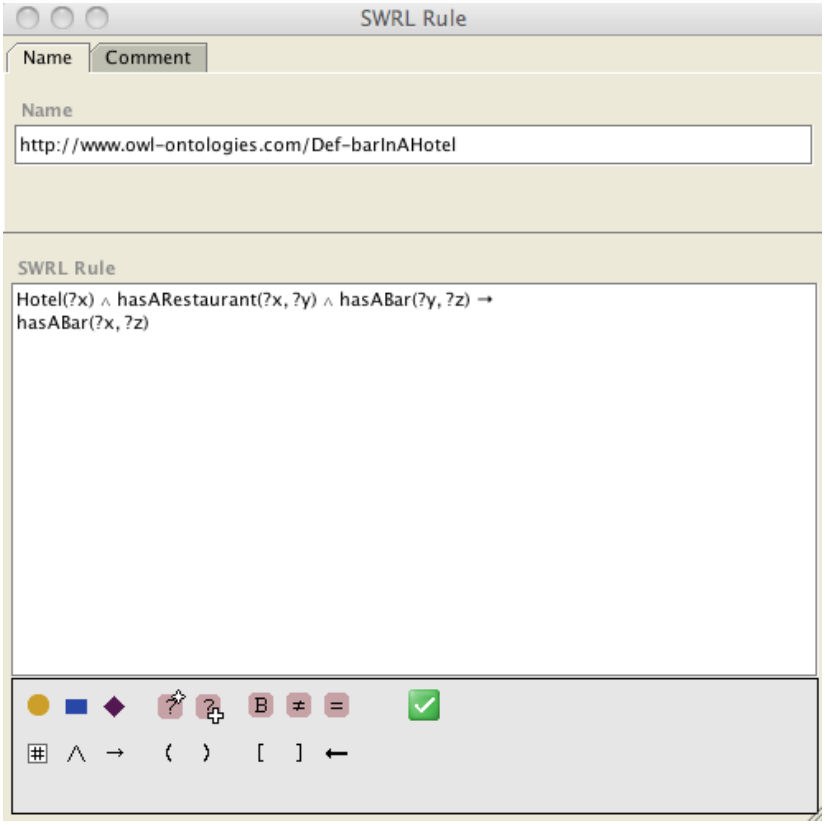

Fig. 4.7
SWRL Tab para construcción de reglas en Protégé

4.1.6 Mantenimiento

Para las actividades de mantenimiento correspondientes a esta fase se considera la documentación generada hasta el momento. En el CD anexo a la presente tesis puede consultarse el archivo completo TurismoCampeche2.owl conteniendo la ontología turística así como las instancias implementadas en Protégé; el Diccionario de datos turísticos del INEGI, base de la conceptualización de esta ontología, así como las estructuras jerárquicas obtenidas a partir del plug-in OWL Viz para Protégé.

4.2 Caso de estudio

La delimitación del caso de estudio, de acuerdo con la definición de la aplicación realizada en el capítulo anterior, queda como sigue:

Aplicación Web que recupera información turística, geográfica y descriptiva, a partir de fuentes heterogéneas de datos (fuente externa de datos e instancias de la ontología) según los requerimientos del usuario; integra y organiza los datos por medio de una ontología turística de aplicación (previamente diseñada de acuerdo con el diccionario de datos turísticos del INEGI), genera un modelo persistente de datos sobre el cual se realiza la búsqueda solicitada y arroja los resultados al usuario en la interfaz Web, incluyendo la visualización de la localización geográfica de las entidades turísticas solicitadas.

El diagrama de casos de uso general para el sistema se ilustra en la figura 4.8:

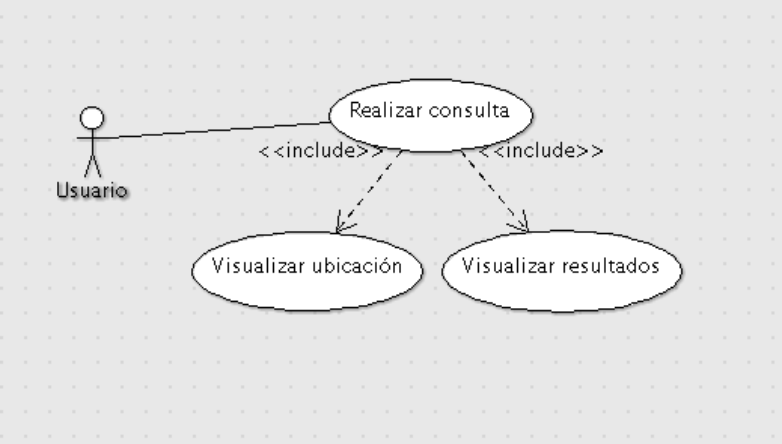

Fig. 4.8

Diagrama de Casos de uso del caso de estudio

Este diagrama representa la funcionalidad con sentido para el usuario del sistema y son arquitectónicamente significativos, como lo son realizar las consultas mismas que se derivan en entregar la información necesaria, así como las marcas georeferenciadas en un mapa para su ubicación.

La aplicación debe ser *web-enabled*, es decir, con interfaz que pueda ser accedida desde un explorador de Internet (Web Browser).

Los requerimientos no funcionales adicionales son la capacidad de respuesta, eficiencia de transferencia de información, seguridad, desempeño, y confiabilidad de los resultados, mismos

que no son del alcance de la presente tesis, pero que deben ser considerados en la implementación para prever problemas a futuro.

4.2.1 Vistas de la arquitectura del caso de estudio

Para la elaboración del caso de estudio se plantea a continuación la arquitectura con diferentes vistas que integran cada uno de los módulos necesarios para lograr la integración de la información, así como su procesamiento y presentación (ver figura 4.9).

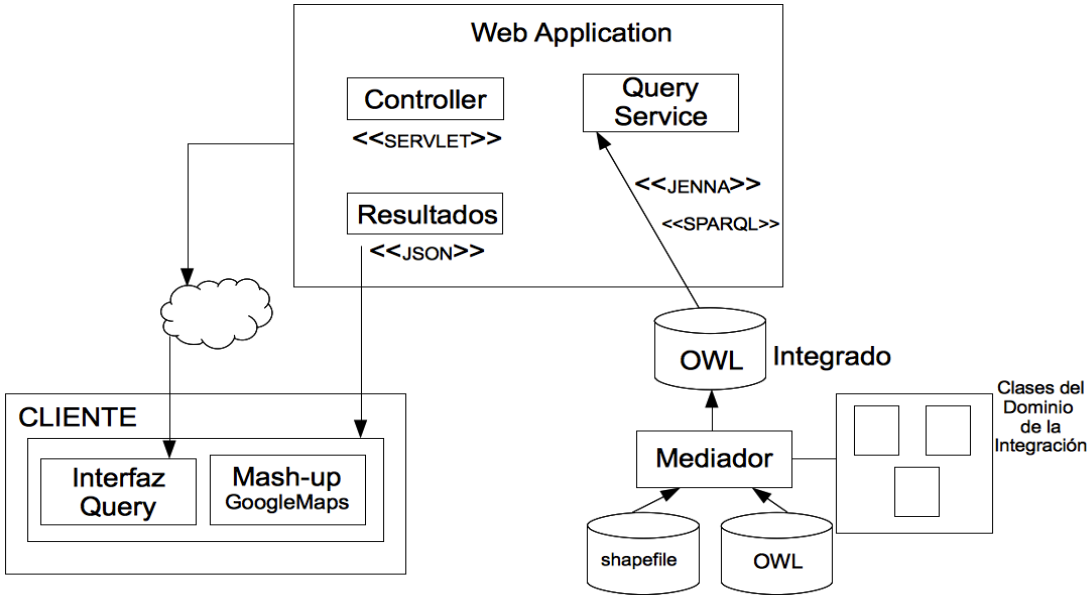

Fig. 4.9

Arquitectura lógica del sistema

La figura previa nos muestra tres áreas de implementación de manera lógica: el cliente, la aplicación web y el mediador, siendo la aplicación Web la responsable de consumir la información generada por el mediador y atender las solicitudes de los clientes.

El **cliente** es básicamente el navegador de Internet, el cual usa como interfaz de consulta un formulario en HTML, así como un *mashup* para la visualización de mapas de Google Maps. Éste

recibe de la aplicación Web la información de las coordenadas de los sitios en formato JavaScript Object Notation (JSON). La comunicación entre el cliente y la aplicación Web se lleva a cabo utilizando el protocolo de solicitud-respuesta denominado hyper text transfer protocol (HTTP). La aplicación se desarrolló utilizando componentes Web definidos por la especificación Java Enterprise Edition (JEE) y utiliza las bibliotecas de Jena y SPARQL para realizar las consultas del modelo persistente de datos con la información de las instancias ontológicas encontradas en las diferentes fuentes. De esta manera, a partir del modelo de datos generado en OWL son aplicadas consultas con criterios específicos proporcionados por el usuario, tal como se observa en el siguiente ejemplo. El siguiente es un fragmento del archivo *owl* sobre el que se realizan las consultas mencionadas:

```
<Hotel rdf:ID="Del_Mar">
    <longitud rdf:datatype="http://www.w3.org/2001/XMLSchema#string">-90.53913474082947</longitud>
    <direccionSitio rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Av. Ruiz Cortines No.
51</direccionSitio>
    <numeroEstrellasHotel rdf:datatype="http://www.w3.org/2001/XMLSchema#int">4</numeroEstrellasHotel>
    <latitud rdf:datatype="http://www.w3.org/2001/XMLSchema#string">19.847744002886362</latitud>
    <paginaWebSitio
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">http://www.delmarhotel.com.mx</paginaWebSitio>
    <hasACafe>
      <Cafe rdf:ID="ElPoquito" />
    </hasACafe>
    <telefonoServicios
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">01(981)8119191</telefonoServicios>
    <costoHospedaje rdf:datatype="http://www.w3.org/2001/XMLSchema#string">1000</costoHospedaje>
    <hasABar>
      <Bar rdf:ID="Laffite" />
    </hasABar>
    <nombreSitio rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Best Western Hotel Del
Mar</nombreSitio>
  </Hotel>
```

Fragmento de Integrado.owl, modelo integrado en una ontología a partir de fuentes heterogéneas

El siguiente código muestra la construcción de una consulta aplicada al archivo *owl* anterior:

```
{criterio:c, nombre:n, precio:p, estrellas:e}
String criterio = request.getParameter("criterio");

if (criterio.equals("n")) {
              "SELECT * "+
        "WHERE "+ "{ "+
                "?Hotel result:nombreSitio ?nombreSitio. "+
                "?Hotel result:longitud ?Longitud. "+
                "?Hotel result:latitud ?Latitud. "+
                 "?Hotel result:costoHospedaje ?costoHospedaje. "+
               "?Hotel result:numeroEstrellasHotel ?numeroEstrellasHotel. "+
                "FILTER regex(str(?nombreSitio), '"+nombre+"', 'i')"+
                 "}";
               }
```

Fragmento de OntoQuery.java, administrador de consultas a través de Jena y SPARQL

La integración y creación de las instancias de la ontología corre a cargo del **mediador,** el cual está diseñado para analizar y procesar un archivo *shapefile* utilizando las bibliotecas de GeoTools, y junto con un conjunto de clases programadas específicamente para el caso, complementar el archivo OWL poblándolo de las instancias provenientes del *shapefile*, al mismo tiempo que se conservan las instancias previas.

La infraestructura física necesaria para la ejecución y prueba del caso de estudio consiste en un servidor con capacidades para ejecutar el Servidor de Aplicaciones compatible con la especificación Java Enterprise Edition, en este caso Glassfish Application Server, instalado en un servidor de desarrollo como se muestra en la figura 4.10.

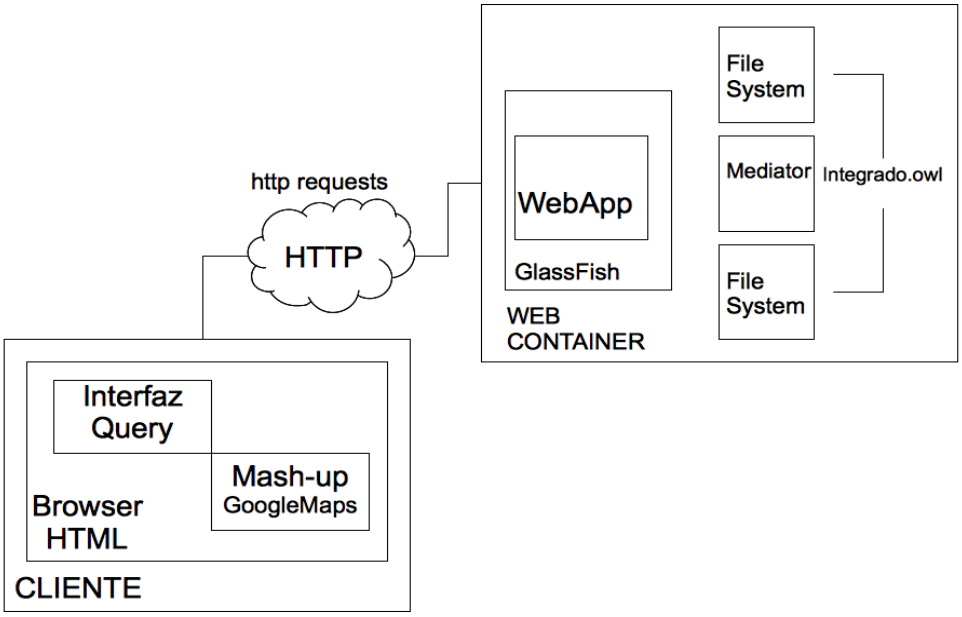

Fig. 4.10

Arquitectura física de la infraestructura del caso de estudio

4.3 Módulo mediador

El término **mediador** es utilizado en la presente tesis para nombrar el conjunto de programas que se encargan de recolectar información de fuentes heterogéneas con información geográfica y

descriptiva e integrarlas como instancias de la ontología en el repositorio OWL-XML ver figura 4.11).

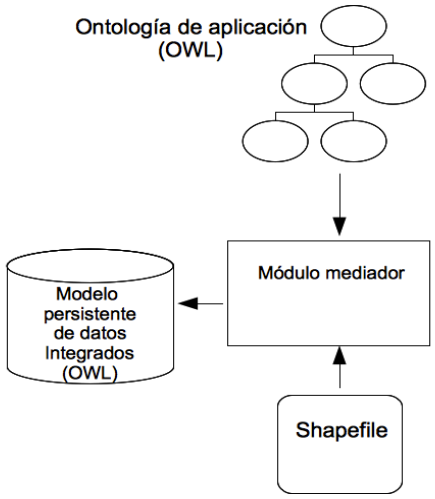

Fig. 4.11

Funcionamiento general del módulo mediador

El mediador implementado para el caso de estudio consta de 2 tipos de clases:

1. Las clases responsables de acceso a las fuentes de datos, utilizando bibliotecas de programación de XML para acceder al repositorio de instancias ontológicas y las bibliotecas de programación (APIs) para acceder a la información alfanumérica contenida en los *shapefiles*.

2. Las clases que representan el **dominio**, es decir, las instancias de la ontología, y que cumplen con el patrón de diseño de *Transfer-Objects* (objetos de transporte o transferencia) para enviar la información entre las clases que utilizarán la información.

En el Anexo B se proporciona el código fuente correspondiente al módulo mediador. A continuación se muestra el diagrama de clases en UML en la figura 4.12.

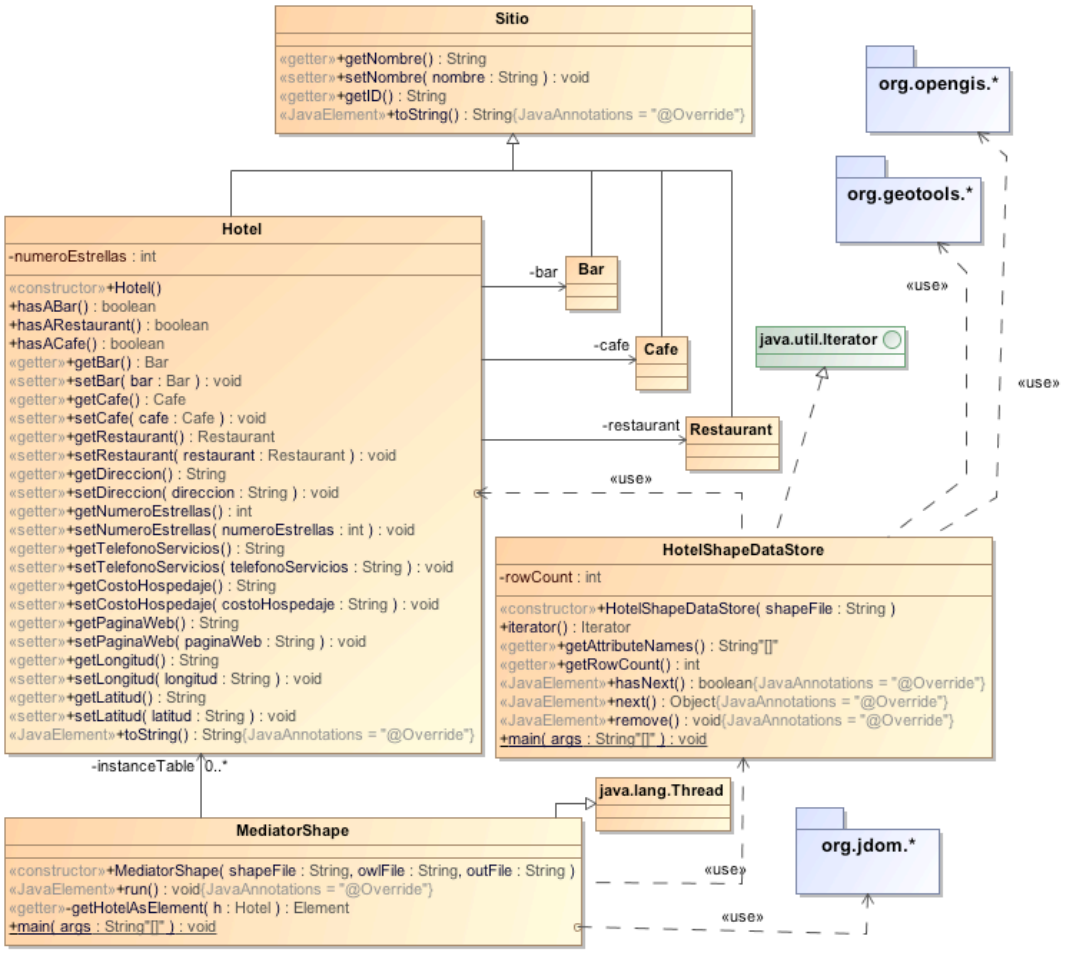

Figura 4.12

Diagrama de clases en UML del módulo mediador.

Se eligió trabajar con estructuras orientadas a objetos para permitir una integración más transparente y de fácil mantenimiento, con clases separadas por funcionalidad para la mejor organización y mantenimiento de los datos.

El mediador está diseñado para poder ejecutar el procesamiento de la recolección e integración de la información en segundo plano, utilizando hilos o procesos ligeros en Java, y de esta manera no interrumpir la ejecución de la aplicación en el contexto que se esté utilizando, ya sea Web, consola (terminal) o escritorio.

```
1.   public static void main(String[] args) {
2.          //ESTE MÉTODO SE USA PARA PROBAR LA INTEGRACION Y MODO DE USO
3.          String shpFile = "hoteles_campeche.shp";
4.          String owlFile = "TurismoCampeche2.owl";
5.          String outFile = "integrado.owl";
6.          MediatorShape ms = new MediatorShape(shpFile,owlFile,outFile);
7.          ms.start();
8.   }
```

En el código de arriba se muestra cómo se utiliza la clase *MediatorShape*, responsable de llevar a cabo todo el proceso de extracción e integración de la información, para lo cual requiere se le envíen como argumentos, las rutas de los archivos con la información a procesar: el *shapefile*, el *OWL* con la ontología actual, así como el archivo de salida resultado de la integración, tipo *OWL* también. Esta clase está implementada como un hilo, es decir, hereda la clase *Thread* de Java para lograr un procesamiento en segundo plano, y utiliza las APIs de la biblioteca **org.jdom** para procesar documentos con formato XML. En el diagrama de secuencia de la figura 4.13 se muestra la secuencia de invocaciones del proceso principal.

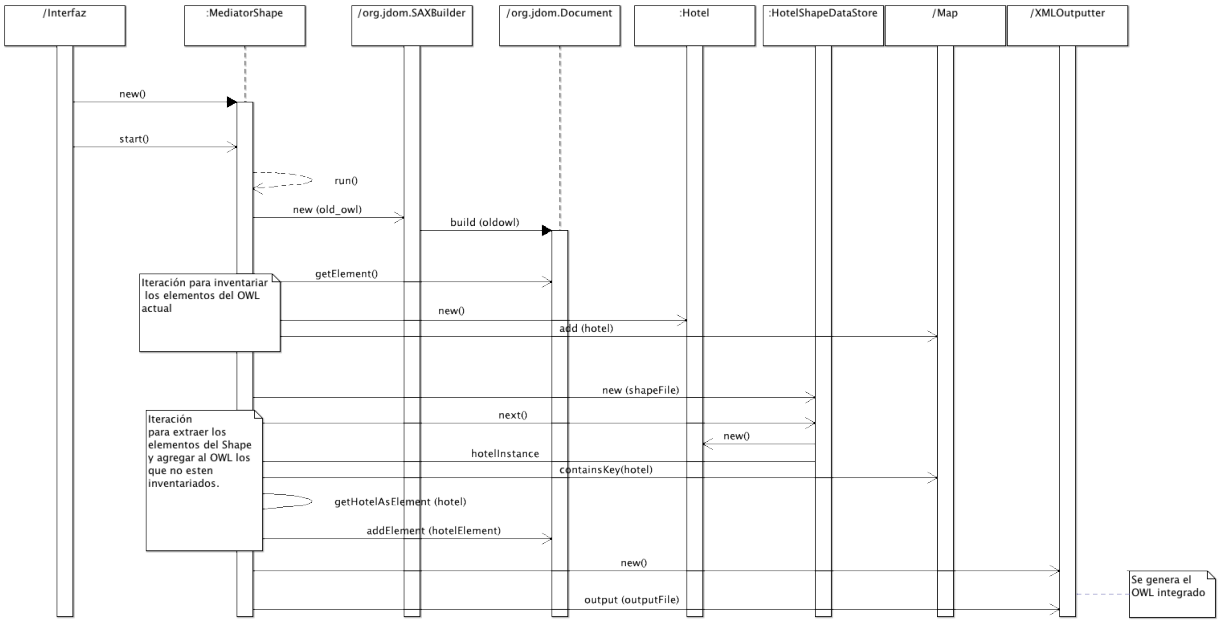

Fig. 4.13

Diagrama de secuencias del módulo mediador

Ambas fuentes de datos, *TurismoCampeche2.owl* y *hotelesCampeche.shapefile* son integradas en la estructura de la ontología a través del módulo **mediador**, que es el encargado de realizar un mapeo entre cada registro del shapefile y las instancias de la ontología, de manera que todas las entradas existentes en el *shapefile* son integradas a la ontología como sus propias instancias, respetándose la organización, jerarquización y restricciones de los datos inherentes a la misma.

Las clases de **dominio** que representan sitios: *Sitio*, *Hotel*, *Bar*, *Cafe*, *Restaurant* son una representación orientada a objetos de la información en proceso; estas clases sirven como medios de contención de la información que es recuperada del *shapefile* y enviadas para ser procesadas como documentos XML de acuerdo con las definiciones ontológicas.

La clase *HotelShapeDataStore* es básicamente un mecanismo de recuperación de información, ya que implementa la interfaz **java.util.Iterator** especificada en el estándar Java porque posee los mecanismos para recuperar y representar la información del *shapefile,* usando clases de dominio, para lo cual utiliza bibliotecas de programación de **GeoTools** y de **OpenGIS**. Esta clase se implementó como un iterador para proporcionar al mediador una forma estándar de recuperar registro a registro.

4.3.1 Archivo *shapefile*

Otro archivo contemplado como entrada de datos para la ontología final fue un *shapefile* conteniendo registros de entidades turísticas con una organización de sus datos descriptivos semejante al archivo *OWL* (ver figura 4.14).

	A	B	C	D	E	F	G	H	I	J
1	Id	Name,C,50	Address,C,50	S	Restaurant,C,50	Bar,C,50	Cafe,C,50	Latitude,C,50	Price,	Longitude,C,50
2	0	Best Western Hotel Del Mar	Adolfo Ruiz Cortines 51	4		Laffite	El Poquito	19.847744002886362	1000	-90.53913474082947
3	0	Plaza Campeche	Calle 10 # 126 esquina circuito baluartes col. Cen	4	El Bicentenario	El Polvorin	Del Parque	19.84766327067685	1290	-90.53390443325043
4	0	Debliz Campeche	Av. Gustavo Diaz Ordaz 55 Col. Ermita	4				19.861235796421067	600	90.51366448402405
5	0	Hacienda Puerta Campeche	Calle 59 entre 16 y 18 Centro	5				19.84890956958513	4141	-90.52142143249512
6	0	El Navegante	Calle 65 entre 14 y 16 Centro	3				19.841441721286454	672	-90.53903818130493

Fig. 4.14

hotelesCampeche.dbf, como parte del archivo shapefile

El shapefile *hotelesCampeche* fue realizado con ESRI ArcMap, de tipo geométrico punto, georreferenciando las ubicaciones originales de los principales hoteles en la Ciudad de San Francisco de Campeche.

En el CD proporcionado con esta tesis se incluye el archivo *shapefile* completo para su visualización y consulta.

4.3.2 Archivo integrado

A partir de los archivos *TurismoCampeche2.owl* y *hotelesCampeche.shapefile* se realizó la integración de ambos en uno solo, *Integrado.owl*, de manera que se respetara la estructura y los usos potenciales de la ontología turística, y al mismo tiempo pudiera darse uso a los datos geográficos contenidos en ambos para su visualización a través de mapas en servicios Web.

En el CD anexo puede consultarse el archivo *owl* producto de la integración de la ontología original y del *shapefile*, realizada a través del módulo mediador.

4.4 Módulo de aplicación Web

La aplicación Web es la responsable de implementar la funcionalidad descrita en los casos de uso (ver figura 4.15) del caso de estudio. A través de esta aplicación probaremos la integración de la información y las consultas, utilizando los mecanismos de consultas sobre el modelo persistente de datos.

La aplicación Web está compuesta de dos partes fundamentales: las vistas que se componen con las interfaces del lado del cliente, el cual básicamente es un navegador Web, y la consulta que es la que se encarga del control de solicitudes y generación de la información a partir del modelo persistente, utilizando una serie de clases que se ejecutan en un servidor de aplicaciones Web. La separación de funcionalidades cumple con el patrón de diseño modelo vista controlador (MVC).

La siguiente figura ilustra los componentes participantes, así como las relaciones entre las bibliotecas utilizadas, archivos y clases.

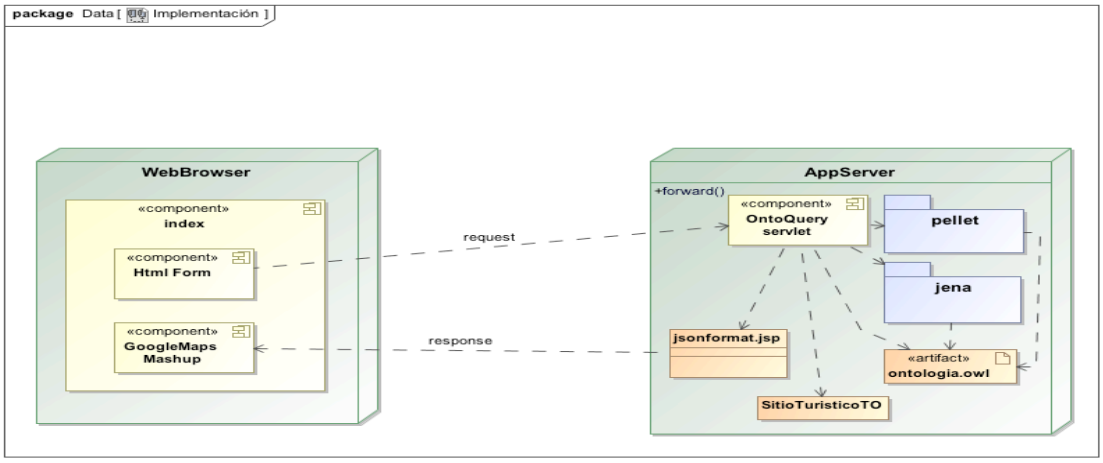

Fig. 4.15

Modelo de componentes en UML de la aplicación Web.

La comunicación entre el cliente (navegador Web) y el servidor (Servidor de Aplicaciones Web) se lleva a cabo utilizando el protocolo *HTTP*, utilizando solicitudes con el comando *POST* del mismo.

Las solicitudes provienen de un formulario en HTML con la información de los sitios que se desean buscar, y el resultado es la población del componente visual responsable de desplegar el mapa con los puntos de los sitios. El proceso que se lleva a cabo del lado del servidor se describe a detalle más adelante (motor de consultas).

El modo de comunicación *solicitud-respuesta* inherente al servidor de aplicaciones Web nos proporciona la pauta para describir las partes que la conforman, tal como se muestra en la figura 4.16.

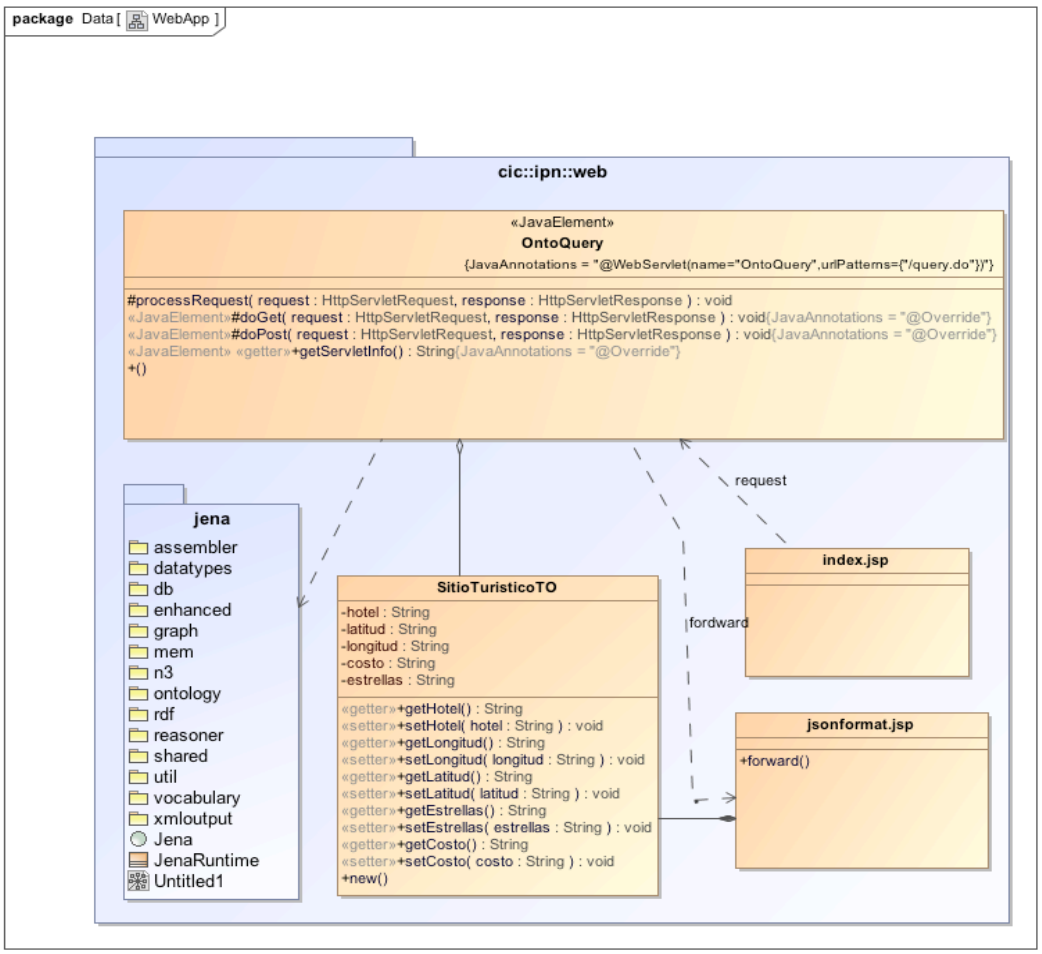

Figura 4.16

Diagrama de clases UML de la aplicación Web.

4.4.1 Motor de Consultas

El motor de consultas es el control en el patrón de diseño MVC; se encarga de analizar las solicitudes y armar las consultas en el lenguaje de consultas para ontologías SPARQL [33], para recuperar la información del modelo persistente de datos, para lo cual utiliza las bibliotecas de Jena.

Las consultas se llevan a cabo sobre el repositorio de instancias ontológicas: *integrado.owl*, utilizando las bibliotecas de Jena y el razonador Pellet (líneas 3 y 12 del siguiente código fuente):

```
1.  OntModel model = null;
2.          // crear un modelo utilizando como razonador OWL_MEM_RULE_INF
3.  model = ModelFactory.createOntologyModel(PelletReasonerFactory.THE_SPEC);
4.  String path= getServletContext().getRealPath("/owl")+"/integrado.owl";
5.  java.io.InputStream in =  FileManager.get().open(path);
6.  model.read(in, "");
7.
8.                  ...
9.
10.         Query query = QueryFactory.create(queryString);
11.         //Ejecutar la consulta y obtener los resultados
12.         QueryExecution qe = QueryExecutionFactory.create(query, model);
```

Ejemplo de creación del modelo con el razonador y fábrica de consultas.

La consulta en SPARQL tiene el siguiente formato:

```
1.  queryString =
2.  "PREFIX result: <http://www.owl-ontologies.com/OntologiaST.owl#> "+
3.     "SELECT * WHERE"+
4.     "{ "+
5.     "?Hotel result:nombreSitio ?nombreSitio. "+
6.     "?Hotel result:longitud ?Longitud. "+
7.     "?Hotel result:latitud ?Latitud. "+
8.     "?Hotel result:costoHospedaje ?costoHospedaje. "+
9.        "?Hotel result:numeroEstrellasHotel ?numeroEstrellasHotel. "+
10.  "FILTER ( ?numeroEstrellasHotel = "+estrellas+")"+
11.  "}";
```

Ejemplo de consulta en SPARQL.

Después de obtener la información y encapsularla utilizando objetos de transferencia con la clase *SitioTuristicoTO*, se delega a una vista responsable de armar la información en formato JSON llamada *jsonformat.jsp*, para retornarla al cliente como respuesta; de esta manera el cliente se encarga de utilizarla en las interfaces, como se muestra en el diagrama de secuencias (ver figura 4.17).

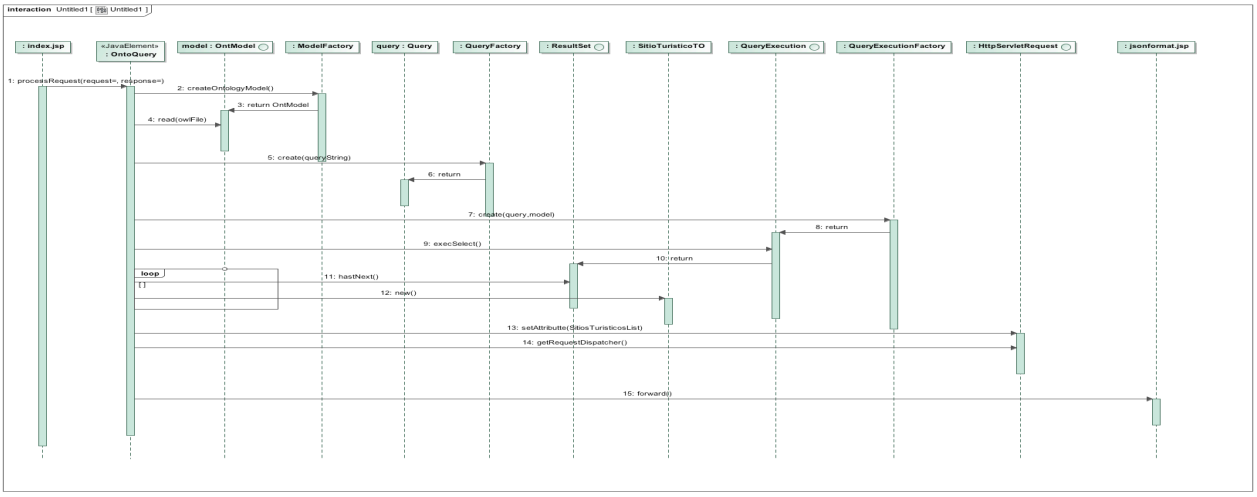

Fig. 4.17
Diagrama de secuencias de UML de la aplicación Web

4.4.2 Interfaces WEB

Las interfaces gráficas de usuario (GUI) implementadas son para visualizar en navegadores Web, es decir, cumplen con el estándar HTML y utilizan tecnologías como JavaScript.

La vista principal de la GUI posee dos componentes: la interfaz de consulta y la interfaz cartográfica (ver figura 4.18).

Fig. 4.18
Composición de la interfaz gráfica de usuario del caso de estudio

4.4.2.1 Interfaz de consulta

La interfaz de consulta es un formulario en HTML (etiqueta <form/>) el cual proporciona la funcionalidad para que el usuario pueda realizar una solicitud de información turística desde su navegador Web; las opciones de búsqueda disponibles para la demostración del caso de estudio son:

- HOTELES POR NOMBRE
- HOTELES POR RANGO DE PRECIOS
- HOTELES POR NÚMERO DE ESTRELLAS

Dichas opciones pueden mezclarse entre sí, al seleccionar la opción "CUALQUIER CRITERIO" (operador lógico OR para la construcción de la consulta resultante en SPARQL), o bien usarse de manera exclusiva mediante la opción "BÚSQUEDA EXACTA" (operador lógico AND para la construcción de l consulta resultante en SPARQL).

La información que viaja hacia la aplicación Web, específicamente el motor de consultas, se utiliza para realizar la consulta en SPARQL sobre el modelo persistente de datos.

Para realizar la solicitud del formulario se utiliza la biblioteca de *jQuery*, que lleva a cabo una solicitud asíncrona de AJAX comunicándose con el componente de control, la clase *OntoQuery*.

```
1.  var jqxhr = $.post("query.do",
2.  {criterio:c, nombre:n, estrellas:e, minimo:p1, maximo:p2},function(data) {
3.        ...
```

Invocación POST con AJAX con la biblioteca de jQuery.

4.4.2.2 Interfaz geográfica

La interfaz de despliegue del mapa con los puntos de los sitios marcados se logra gracias a un servicio (mashup) de Google Maps, el cual recibe en formato de JavaScriptObject Notation (JSON) las coordenadas de los sitios a ubicar. El componente del lado del servidor llamado *jsonformat.jsp* es el responsable de generarlo:

```
1. {
2. "lugares":
3.                   [
4.                     [1,"Hotel del Mar", "19.84605149", "-90.52413433"],
5.                     [2, "Hotel Francis Drake", "19.84411389", "-90.51875919"]
6.                   ]
7. }
```

JavaScript Object Notation (JSON) para los sitios resultado de las consultas.

Este contenido es procesado del lado del navegador utilizando JavaScript y la biblioteca *jQuery*:

```
1. var r = jQuery.parseJSON(data);
```

Procesamiento del contenido en JSON con la biblioteca de jQuery.

Los datos que debe arrojar el sistema por cada búsqueda son (máximo 3 resultados):

- Nombre
- Número de estrellas
- Precio por habitación
- Ubicación geográfica

Esta información deberá ser desplegada en cada punto dentro del mashup de Google Maps a partir de las coordenadas geográficas recuperadas, como se muestra a continuación:

```
1.  for(var i=0; i < lugares.length; i++){
2.      lugares[i] = new google.maps.LatLng(
3.                    json_object.lugares[i][2],json_object.lugares[i][3]);
4.      titles[i] = json_object.lugares[i][1];
5.  }
```

Procesamiento del contenido en JSPN con la biblioteca de jQuery.

Dichas coordenadas son insertadas dentro del mashup (línea 9 del siguiente código) junto con una ventana con la información recuperada (líneas 11 y 13)

```
1.   var marker2=new google.maps.Marker({
2.        position: lugares[contador],
3.        title: titles[contador],
4.        map: map,
5.        draggable: false,
6.        animation: google.maps.Animation.DROP
7.  });
8.
9.      markers.push(marker2);
10.
11. var contenido='<div><h3>'+ titles[contador]+'</h3><p>Por: Datos tomados de <a
    href="owl/integrado.owl">OWL</a>!</p></div>';
12.
13.     var infowindow = new google.maps.InfoWindow({
14.             content: contenido,
15.             maxWidth: 100
16.     });
17.
18.     google.maps.event.addListener(marker2, 'click', function() {
19.           infowindow.open(map,marker2);
20.     });
```

Inserción de puntos en el mash-up de googleMaps y ventana informativa para cada punto

4.5 Resultados de la aplicación

En esta sección se presentan los resultados obtenidos en la implementación de la aplicación Web para la localización de sitios turísticos en la ciudad de Campeche. En la figura 4.19 se muestra la interfaz de usuario para realizar consultas:

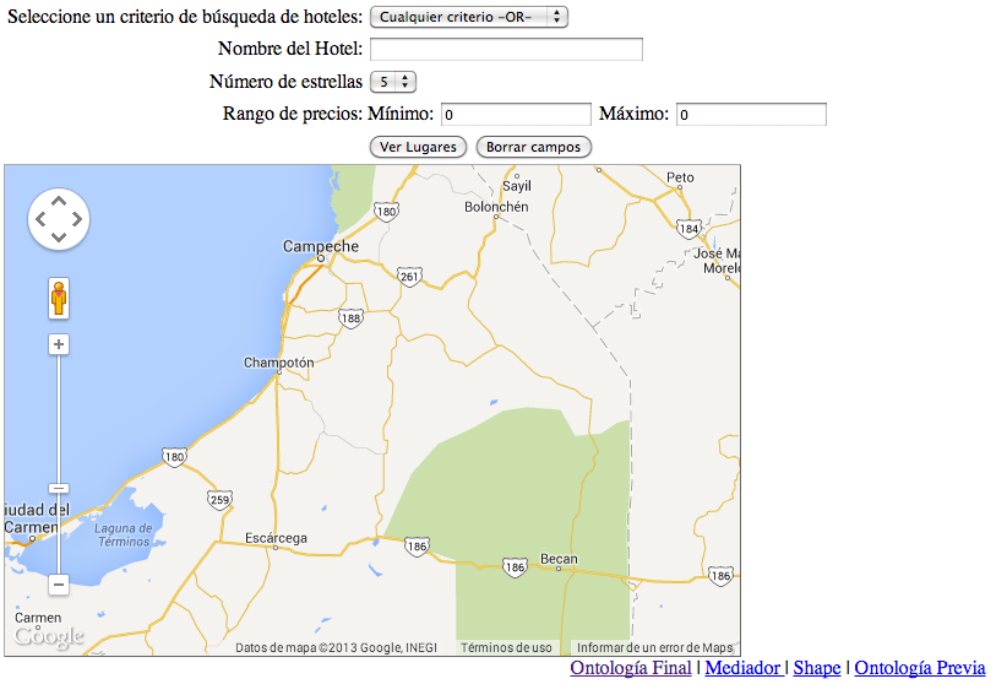

Fig. 4.19

Interfaz de usuario para consultas OntoCampeche

Para esta aplicación se aceptan tres criterios de búsqueda, que pueden ser definidos en el campo "Seleccione un criterio de búsqueda de hoteles: ".

1. Nombre del Hotel: Se puede introducir todo o parte del nombre del hotel que se desee localizar.
2. Estrellas: Realiza búsqueda de hoteles por número de estrellas proporcionado.
3. Rango de precios: Es posible introducir un precio mínimo y un precio máximo como rango de búsqueda de un hotel en particular.

Estos tres criterios de búsqueda pueden ser mezclados entre sí, mediante la selección de la primera opción de la lista desplegable, "Cualquier criterio -OR-", que utilizará todos los campos llenados por el usuario para armar la consulta correspondiente, o bien pueden ser utilizados de manera individual, mediante la opción "Búsqueda exacta -AND-", la cual únicamente toma en cuenta el criterio seleccionado para construir la consulta resultante en SPARQL.

En el caso de "Búsqueda exacta -AND-", si la búsqueda solicitada fue por **Nombre del Hotel**, es posible introducir todo o parte del nombre del hotel buscado en el siguiente campo, "Nombre del Hotel". Los demás campos son ignorados por el sistema que arroja los resultados correspondientes en el mash-up de Google Maps incluído en la interfaz, tal como se muestra en la figura 4.20.

Fig. 4.20

Búsqueda por nombre de hotel

En el segundo caso, si la consulta requerida fue por número de **Estrellas**, es posible seleccionar el número de estrellas deseadas en la lista del campo "Número de Estrellas". A continuación el sistema arroja la consulta solicitada en el mashup de Google Maps (ver figura 4.21).

Fig. 4.21

Búsqueda por número de estrellas

Para el último caso, la búsqueda por rango de precios, se selecciona en el primer campo la opción "Rango de Precios" y se definen las cantidades mínima y máxima en pesos mexicanos en los campos correspondientes, tal como se muestra en la figura 4.22.

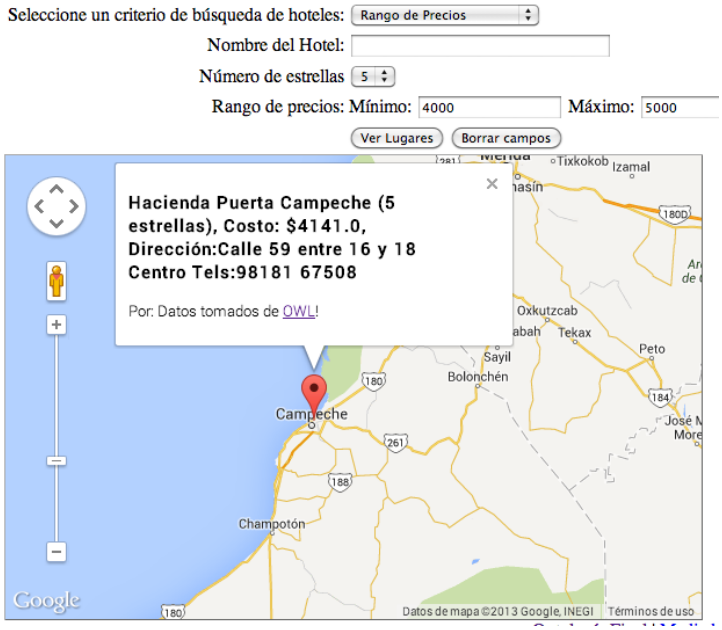

Fig. 4.22

Búsqueda por rango de precios

Los resultados aquí presentados son susceptibles de comprobación, a través de su comparación con los archivos de origen.

4.6 Comparación de resultados con aplicaciones afines

En cuanto a la comparación de la aplicación desarrollada en esta tesis, con respecto a buscadores de hoteles existentes en la Web actualmente, se realizó un análisis de la obtención de resultados, seleccionando la búsqueda de un hotel en particular, **Monkey Hotel y Hostal**.

A manera de antecedente, cabe mencionar que este hotel se encuentra ubicado en el corazón de la Ciudad de Campeche, justo frente a la plaza principal, siendo uno de los favoritos del turismo nacional e internacional por su relación calidad-costo, así como por su excelente ubicación.

Uno de los primeros buscadores de hoteles visitado fue Viajes México [71]; en dicho sitio se puede acceder a un menú de destinos turísticos para la localización de hoteles y paquetes de viaje. Sin embargo, Campeche no figura en la lista disponible. El mismo caso fue detectado en los sitios de Viajes Vive México [72] y Mundo Joven [73] (ver figura 4.23).

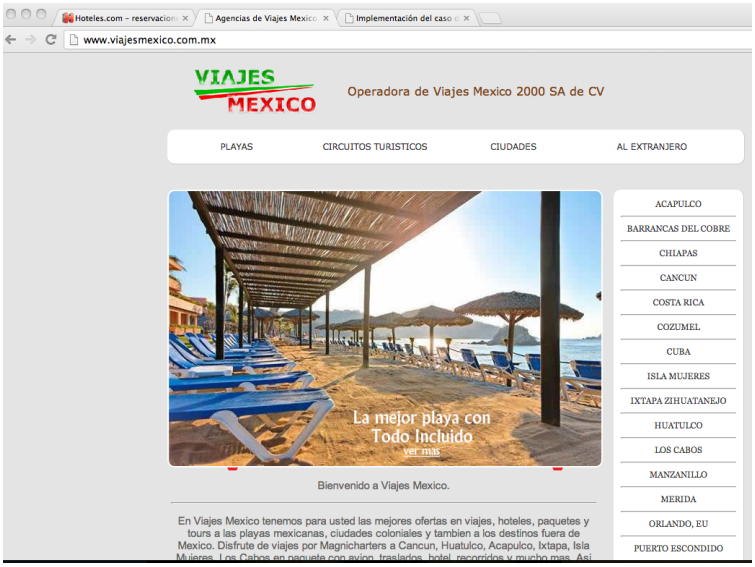

Fig. 4.23
Búsqueda de hoteles campechanos en viajesmexico.com.mx

En el sitio de despegar.com [74], como se puede apreciar en la figura 4.24 es posible realizar la búsqueda de hoteles por nombre:

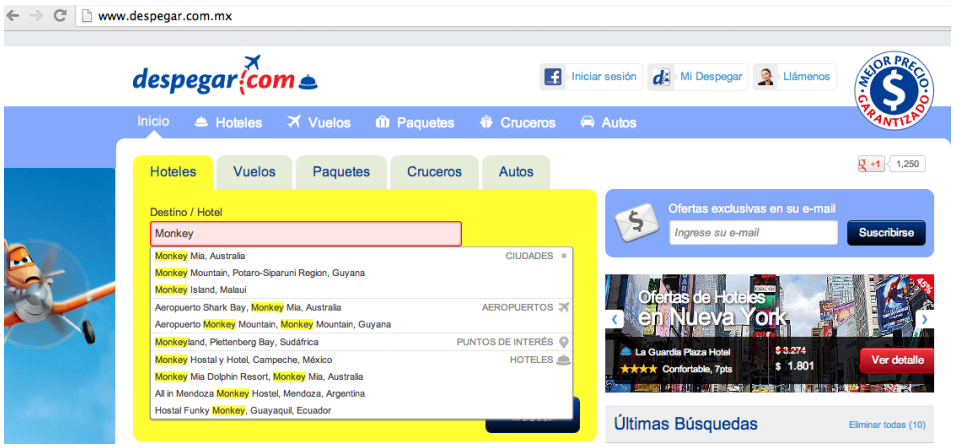

Fig. 4.24
Búsqueda de hoteles campechanos en despegar.com.mx

Los resultados obtenidos no muestran a primera vista el costo de los servicios ni los datos de contacto del hotel solicitado; el sitio indica que para acceder al precio, se debe acceder a la página de pago; sin embargo, sin importar la fecha seleccionada, el sistema indica que la disponibilidad del hotel se encuentra agotada. De esta manera, no fue posible verificar precios ni datos de contacto del hotel seleccionado (ver figura 4.25).

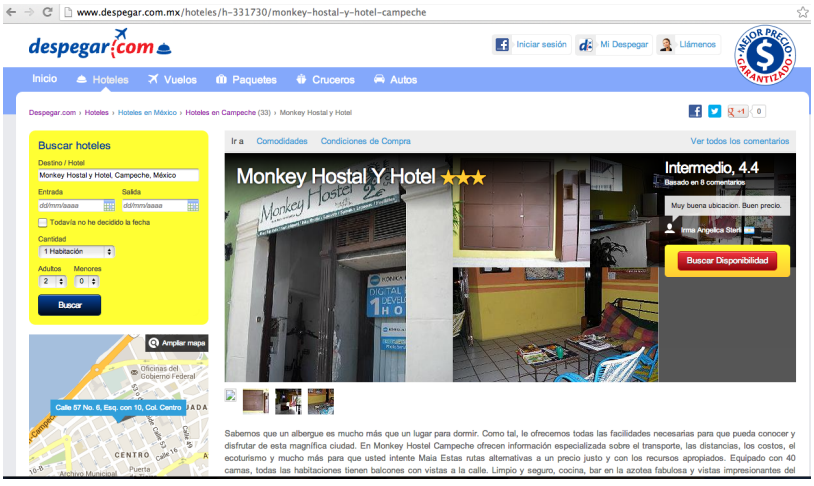

Fig. 4.25
Resultados en despegar.com.mx

En el caso de Hoteles.com [75], aunque fue posible localizar doce hoteles campechanos en el sistema, no se encontró el hotel seleccionado en particular para estas pruebas, Monkey Hostal, tal como se muestra en la figura 4.26.

Fig. 4.26
Búsqueda de hoteles campechanos en hoteles.com

En el sitio Hospedarse.com [76], la interfaz de usuario proporciona la facilidad de realizar búsquedas por nombre de ciudad, más no por nombres o características de hoteles. Para Campeche aparecieron diez hoteles disponibles, pero no el Monkey Hostel; los datos de contacto desplegados en los resultados no proporcionan el contacto directo con el hotel, de manera que las reservaciones se deben hacer a través del sitio Hospedarse.com (ver figura 4.27 y 4.28).

Fig. 4.27
Búsqueda de hoteles campechanos en hospedarse.com

Fig. 4.28
Resultados preliminares en hospedarse.com

En los resultados obtenidos con la aplicación del caso de estudio, **OntoCampeche**, se puede observar la búsqueda realizada por nombre aproximado (query REGEX en SPARQL, equivalente a LIKE en SQL), arrojando en primera instancia nombre, precio y datos de contacto del hotel deseado, además de su localización geográfica en el *mash-up* de Google Maps, con lo que proporciona ventajas al consumidor directo en cuanto a la obtención de resultados directos e inmediatos, tal como se muestra en la figura 4.29.

Fig. 4.29
Resultados preliminares en el sistema realizado OntoCampeche

En este punto es necesario señalar que la intención primera en el diseño e implementación de la presente aplicación, fue la explotación de las principales características, usos y ventajas de una ontología, tales como la integración de datos a partir de fuentes heterogéneas, la inferencia de información, entre otras; el caso de estudio resultante no fue pensado para competir abiertamente con otras aplicaciones afines existentes en el mercado, sino como un punto de partida en el trabajo futuro de las líneas de investigación manejadas en esta tesis.

Capítulo 5. Conclusiones y trabajos futuros

Capítulo 5. Conclusiones y trabajos futuros

En este capítulo se presentan las conclusiones finales del trabajo, así como las diferentes propuestas para investigaciones futuras.

5.1 Conclusiones

En la presente tesis se ha descrito el desarrollo de una aplicación para la localización de entidades turísticas a través de Internet, utilizando una ontología para la organización e integración de la información disponible a partir de fuentes heterogéneas; dicha integración de datos es la parte medular y la aportación más importante del presente trabajo, ya que el éxito de todo sistema de consultas recae justamente en la información que éste sea capaz de recuperar, procesar e inferir para el usuario, por lo cual las ontologías resultan una herramienta clave.

Dicha integración fue realizada a partir de fuentes heterogéneas de datos, a través de los estándares más utilizados actualmente, teniendo en mente la posibilidad de incrementar en trabajos futuros tanto sus alcances como las fuentes de datos susceptibles de ser utilizadas.

Para el desarrollo del caso de estudio presentado, se utilizó una ontología de aplicación en el dominio turístico a partir del diccionario de datos del INEGI; siendo esta institución la que rige los estándares de trabajo en localización espacial a nivel nacional, se espera que otras líneas de investigación compartirán esta característica inicial con el presente trabajo, lo cual favorece la posibilidad del trabajo cooperativo entre diferentes investigaciones e instituciones por igual.

De esta manera, a través de la integración de datos provenientes de fuentes heterogéneas, quedan cubiertos dos de los principales objetivos planteados al principio de este trabajo, que son la construcción de una ontología de aplicación para promover la reutilización de las estructuras generadas de manera semántica, así como la posibilidad de ampliaciones posteriores al mismo o a

través de otros sistemas. Los otros dos objetivos propuestos quedan cumplidos igualmente en esta tesis, y por supuesto abiertos a la extensión y mejora: una aplicación híbrida accesible desde la Web para la recuperación y visualización de datos geográficos y descriptivos, con el ejemplo funcional de la ciudad de Campeche.

Así mismo, adicionalmente a los objetivos planteados inicialmente para esta tesis, se implementó la inferencia de datos a partir de información existente en la ontología, a través de la utilización de reglas de inferencia especificadas para dicho fin, lo cual presenta un área de oportunidad para el establecimiento de líneas de investigación afines a lo realizado hasta el momento en el presente trabajo.

No obstante haber llevado a buen término la conclusión de dichos objetivos, quedan algunos elementos de la aplicación susceptibles de ser optimizados, los cuales se presentan más adelante como trabajo futuro.

5.2 Limitaciones

A pesar de haber cumplido satisfactoriamente con los objetivos planteados al principio de esta tesis, se pueden considerar las siguientes limitaciones:

- A pesar de que las entradas de las instancias provenientes del *shapefile* se realizan de manera automática, las instancias registradas en el archivo *owl* son hechas de manera manual.

- La interfaz Web fue diseñada con las funcionalidades indispensables para el funcionamiento básico del sistema, teniendo en mente la realización de pruebas fundamentales para la puesta en marcha de la aplicación resultante.

5.3 Trabajos futuros

Las restricciones encontradas en el sistema obtenido parten de la aspiración de lograr una integración a mayor escala que la alcanzada hasta este momento en el presente trabajo; tomar un archivo de cualquier formato e integrarlo transparente, eficiente y óptimamente dentro de la ontología diseñada. A partir de ese punto, se puede decir que el sistema es fácilmente escalable desde el punto de vista de la interacción con el usuario.

Hasta este momento, el sistema presentado es capaz de integrar un archivo tipo *shapefile* a una ontología existente (*.owl*), alimentando esta última con un mayor número de instancias en su contenido final. Para lograr esto se hace un mapeo directo entre las estructuras conocidas de cada uno de los archivos; sin embargo, existen tantas fuentes de datos en tan diversos formatos que solucionar el problema de una integración efectiva resulta un reto interesante dentro de la línea de investigación de este trabajo.

Otro punto medular es la explotación efectiva del motor de inferencias contenido en la aplicación para la generación real de información útil para el usuario; en el presente trabajo el razonador es empleado, tanto en la construcción de la ontología para garantizar su consistencia (Tbox) como en la inferencia de datos mediante el uso de reglas (Abox); estas últimas operaciones fueron realizadas sobre información limitada a las instancias mínimas necesarias para la realización de pruebas en el sistema.

Una vez realizadas estas observaciones, los tópicos restantes se refieren a detalles de la presentación y ampliación de la información contenida en la ontología:

- Contenido: El número de instancias contenidas en la ontología se limitó a aquellas existentes para el caso de estudio; sin embargo, el sistema queda abierto a la actualización y crecimiento con la introducción de más instancias a través de la ontología original (Protégé), o bien mediante el archivo *shapefile*.

- Presentación: Así mismo, el diseño de la interfaz Web se acotó a las funcionalidades estrictamente necesarias para la puesta en marcha de la aplicación y la realización de las pruebas fundamentales.

Como se puede observar, la solución de cada una de las propuestas aquí presentadas brindan la oportunidad de abrir líneas de investigación afines a estos temas, con lo cual se pueden realizar diversos proyectos fomentando la investigación en esta área.

De igual forma, con este trabajo se busca proponer acercamientos a la solución de problemas como la integración, heterogeneidad e interoperabilidad semántica entre las diferentes aplicaciones y fuentes de datos geoespaciales, contribuyendo de esta manera al área de las Ciencias de la Información Geográfica (GIScience).

Referencias

[1] M. Tsou, Z. Peng: "Internet GIS: Distributed Geographic Information Services for the Internet". John Wiley and Sons, 2003.

[2] Available: http://www.guiaroji.com.mx/

[3] Available: http://www.geonames.org/

[4] Available: http://maps.google.com.mx/

[5] Baden, S., et al; Mashup: Web application hybrid; report by University of Alberta, Ca.

[6] Available: W3C Web Services, http://www.w3.org/2002/ws/

[7] Ontologías - Available: http://www.w3c.es/Divulgacion/GuiasBreves/WebSemantica

[8] World Tourism Organization. WTO. Tourism 2020 Vision: http://www.unwto.org/facts/eng/vision.htm

[9] Lake, D. American Go Online for Travel Information, in CNN. 2001

[10] J. Cardoso. E-Tourism: Creating Dynamic Packages using Semantic Web Processes (2006).

[11] The Sixth Framework Programme, European Commision, 2002 edition. Available: http://ec.europa.eu/research/fp6/index_en.cfm

[12] V. Alexiev, M. Breu. Information Integration with Ontologies: Experiences from an Industrial Showcase, Ed. Wiley, April 2005.

[13] Semantic Web - Available: http://www.w3.org/standards/semanticweb/

[14] Diccionario de Datos Turísticos, Instituto Nacional de Estadística Geografía e Informática (INEGI). Available: http://www.cp-idea.org/documentos/normasEspecificaciones/tur_1000a.pdf

[15] M. Hossein Eftekhari, Zeynab Barzegar, and M. T. Isaai. Web 1.0 to Web 3.0 Evolution: Reviewing the Impacts on Tourism Development and Opportunities. *HCITOCH, volume 6529 of Lecture Notes in Computer Science, page 184-193. Springer,* (*2010*)

[16] Available: http://www.w3.org/XML/

[17] Buccella, A., Cechich A., Nieves, B; Ontology based data integration methods: a framework for comparison. Revista Colombiana de Computación, Volume 6, Number 1, 2005.

[18] T. R. Gruber. Toward Principles for the Design of Ontologies. Used for Knowledge Sharing.

Standford Knowledge Systems Laboratory.

[19] Guarino, N. (1998) Formal Ontology in Information Systems. International Conference on Formal Ontology in Information Systems (FOIS'98). Trento, Italy. IOS Press, Amsterdam, pp 3–18.

[20] Available: http://www.w3.org/2001/sw/WebOnt/

[21] Fernández-López M, Gómez-Pérez A, Juristo N (1997) METHONTOLOGY: From Ontological Art Towards Ontological Engineering. Spring Symposium on Ontological Engineering of AAAI. Stanford University, California, pp 33–40

[22] Available: http://protege.stanford.edu/

[23] Available: http://www.w3.org/TR/owl-features/

[24] Available: http://clarkparsia.com/pellet/

[25] Available: http://www.esri.com/software/arcgis

[26] Available: http://netbeans.org/

[27] Available: http://www.java.com/

[28] Available: http://www.geotools.org/

[29] Available: http://www.jdom.org/

[30] Available: http://www.oracle.com/technetwork/java/javaee/jsp/index.html

[31] Available: http://www.oracle.com/technetwork/java/index-jsp-135475.html

[32] Available: http://jena.apache.org/

[33] Available: http://www.w3.org/TR/rdf-sparql-query/

[34] Available: https://maps.google.com.mx/

[35] Available: https://developer.mozilla.org/es/docs/JavaScript

[36] Available: http://www.json.org/

[37] Available: http://glassfish.java.net/es/

[38] Available: http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf

[39] Available: http://www.jdto.org/

[40] Borst, W. N. Construction of Engineering Ontologies. University of Tweenty. Ensechede, NL- Centre for Telematica and Information Technology. 1997.

[41] R. Studer, V.R. Benjamins, D. Fensel. Knowledge Engineering: Principles and Methods. Data and Knowledge Engineering. 25: 161-197. 1998.

[42] Available: http://www.w3.org/html/

[43] Available: http://www.w3.org/XML/Schema.html

[44] Available: http://www.w3.org/RDF/

[45] Available: http://www.w3.org/TR/rdf-schema/

[46] Available: http://www.w3.org/TR/2004/REC-owl-features-20040210/#s3

[47] Available: http://www.w3.org/TR/2004/REC-owl-features-20040210/#s4

[48] Available: http://apollo.open.ac.uk/

[49] Ceusters, W., Martens, P. LinKFactory® : an Advanced Formal Ontology Management System. Language and Computing (L&C). Zonnegem, Belgium

[50] Sure, Y., Erdmann, M., Angele, J., Staab, S., Studer, R., Wenke, D., OntoEdit: Collaborative Ontology Development for the Semantic Web. Institute AIFB, University of Karlsruhe, 76128 Karlsruhe, Germany.

[51] Available: http://www.ksl.stanford.edu/software/ontolingua/

[52] Available: http://www.isi.edu/isd/ontosaurus.html

[53] Available: http://ontology4.us/english/OntoPage/Subject/sub_OpenKnoME,,rel_isi,class_OntologyTool,.Developer,ASC,att_Developer,att_City,att_Country,att_Web,.html

[54] Available: http://mayor2.dia.fi.upm.es/oeg-upm/index.php/en/technologies/60-webode

[55] Available: http://projects.kmi.open.ac.uk/webonto/

[56] Available: http://owlapi.sourceforge.net/reasoners.html

[57] Available: http://code.google.com/p/factplusplus/

[58] Available: http://www.sts.tu-harburg.de/~r.f.moeller/racer/

[59] Available: http://www.racer-systems.com/products/racerpro/

[60] Available: http://kaon2.semanticweb.org/

[61] Available: http://sourceforge.net/projects/powl/

[62] Available: http://www.w3.org/MarkUp/

[63] Available: http://www.w3.org/Style/CSS/

[64] Torres M., Quintero R., Moreno-Ibarra M., Menchaca-Mendez R., Guzman, G., "GEONTO-MET: An approach to conceptualizing the geographic domain", International Journal of Geographical Information Science, 2011, 1633-1657.

[65] Dell Erba M. *et al*., HARMONISE: a Solution for Data Interoperability; eCommerce and Tourism Research Laboratory, ITC-1rst, Italy.

[66] Siorpaes, K., OnTour System Design, Digital Enterprise Reaserch Institue.

[67] Cardoso, J. "Developing Dynamic Packaging Systems using Semantic Web Technologies", Transactions on Infromation Sceince and Applications, Vol. 3(4), April 2006, pp. 729-736, ISSN:1970-0832.

[68] Corcho O., Fernández-López M., Gómez-Pérez A., López-Cima A., Construcción de ontologías legales con la metodología METHONTOLOGY y la herramienta WebODE. Facultad de Informática - Universidad Politécnica de Madrid, 2005.

[69] O'Connor, M. SWRLTab: A Development Environment for working with SWRL Rules In Protégé-OWL. Stanford Medical Informatics, Stanford University, 2007.

[70] Available: http://www.w3.org/Submission/SWRL/

[71] Available: http://www.viajesmexico.com.mx/

[72] Available: http://viajesvivemexico.com.mx/

[73] Available: http://www.mundojoven.com.mx/

[74] Availabe: http://www.despegar.com.mx/

[75] Available: http://www.hoteles.com/

[76] Available: http://www.hospedarse.com/

Anexo A. Descripción de la ontología

A continuación se ilustran las ocho clases correspondientes a la clasificación realizada para la ontología utilizada, según el diccionario de datos turísticos del INEGI.

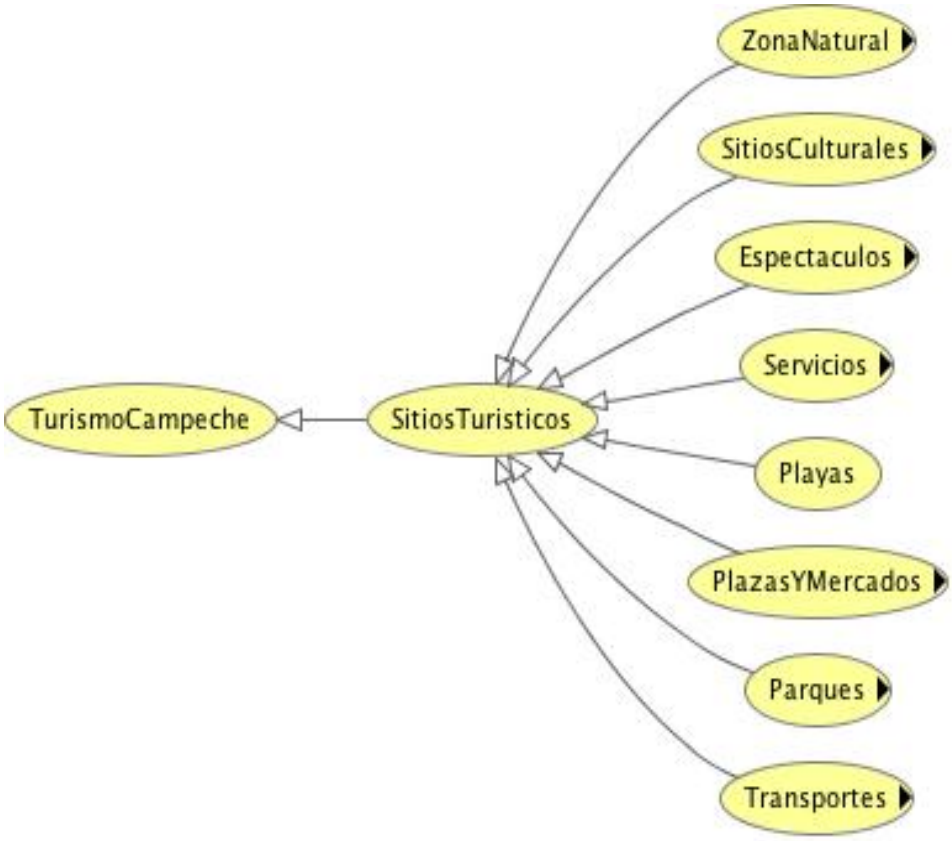

Fig. A.1:

Ocho subclases de SitiosTurísticos

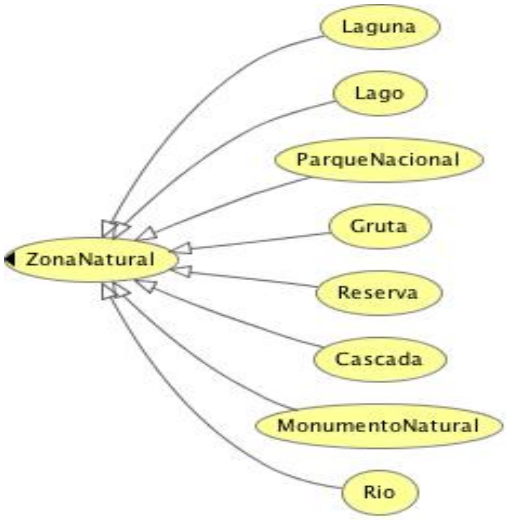

Fig. A.2:

Sublcases de ZonaNatural

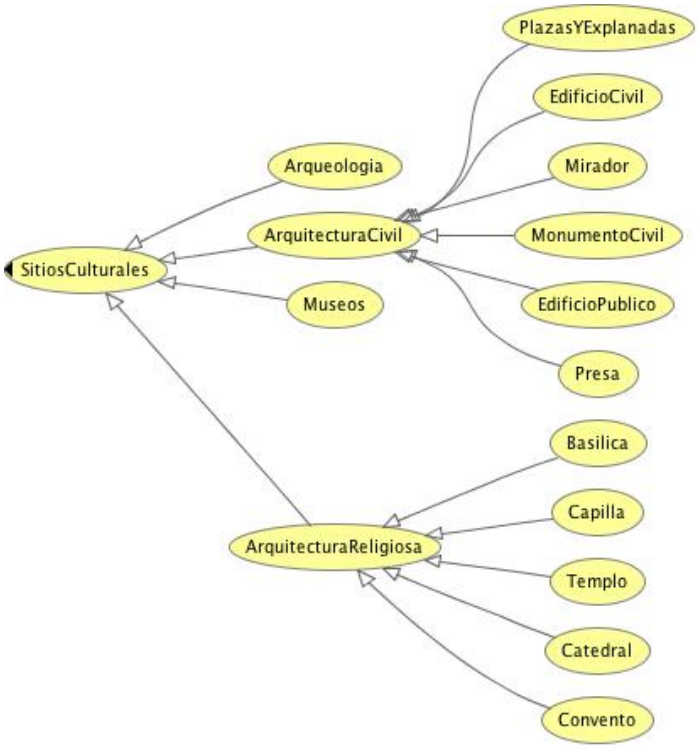

Fig. A.3:

Subclases de SitiosCulturales

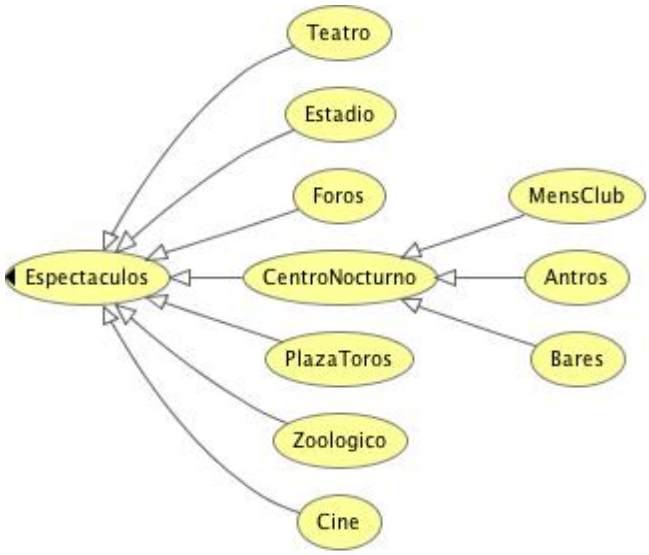

Fig. A.4:

Sublcases de Espectáculos

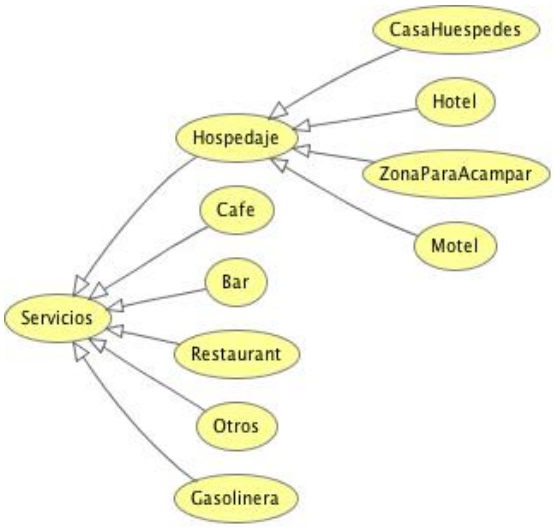

Fig. A.5:

Subclases de Servicios

Fig. A.6:

Clase Playa, sin subclases definidas

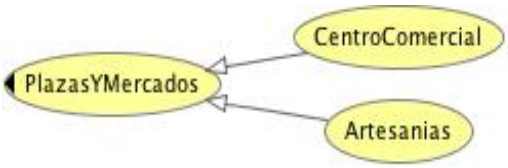

Fig. A.7:

Subclases de PlazasYMercados

Fig. A.8:

Subclases de Parques

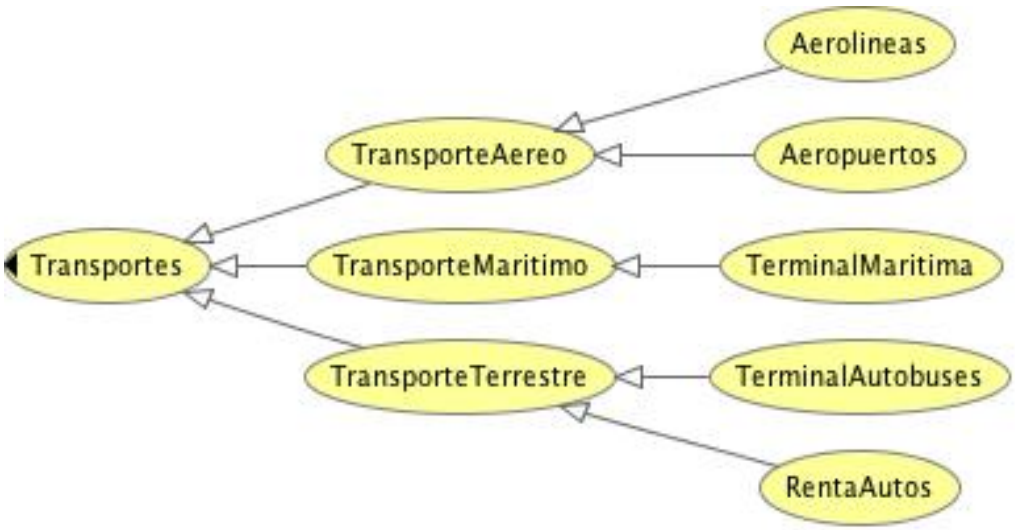

Fig. A.9:

Subclases de Transportes

Anexo B.

Código fuente del módulo mediador, *MediatorShape.java*.

```
package ipn.cic.ontogis;
import ipn.cic.ontogis.instance.Hotel;
import java.io.FileOutputStream;
import java.io.IOException;
import java.util.HashMap;
import java.util.Iterator;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;

import org.jdom.*;
import org.jdom.input.*;
import org.jdom.output.*;

public class MediatorShape  extends Thread{
   private Document doc;    //documento DOM del OWL
   private String shapeFile;
   private String owlFile;
   private String outFile;
   public static final Namespace nsbase = Namespace.getNamespace("", "http://www.owl-
ontologies.com/OntologiaST.owl#");
   public static final Namespace rdf =
Namespace.getNamespace("rdf","http://www.w3.org/1999/02/22-rdf-syntax-ns#");

   private HashMap <String, Hotel> instanceTable;
```

```
  public MediatorShape(String shapeFile, String owlFile, String outFile) {
     this.shapeFile = shapeFile;
     this.owlFile = owlFile;
     this.outFile = outFile;
   }
  @Override
  public void run() {

     //PASO 1. inventariar las instancias del owl actual en un Map
     // CREANDO EL ARBOL DOM DEL OWL ACTUAL
     try {
        SAXBuilder builder = null;
        builder = new SAXBuilder();
        builder.setExpandEntities(true);
        builder.setIgnoringBoundaryWhitespace(true);
        try {
           doc = builder.build(owlFile);
        } catch (IOException ex) {
           Logger.getLogger(MediatorShape.class.getName()).log(Level.SEVERE, null, ex);
        }
        Element root = doc.getRootElement();
        List hoteles = root.getChildren("Hotel",nsbase);
System.out.println("Instancias de Hotel en actual OWL:"+hoteles.size());
        Iterator i = hoteles.iterator();
        instanceTable = new HashMap<String, Hotel>();
        while(i.hasNext()) {
           Element e = (Element) i.next();
          String ID = e.getAttributeValue("ID", rdf);
           String nombreSitio = e.getChildText("nombreSitio",nsbase);
```

```
            System.out.println("ID: "+ID+", nombresitio: "+nombreSitio);
            Hotel hotel = new Hotel();
            hotel.setNombre(nombreSitio);
            instanceTable.put(hotel.getNombre(), hotel);
          }

         //PASO 2. Parsear el Shape File y comparar con la tabla de instancias
          HotelShapeDataStore hsds = new HotelShapeDataStore(shapeFile);
          Iterator it = hsds.iterator();
          while(it.hasNext() ) {
            Hotel h = (Hotel) it.next();
            if (instanceTable.containsKey(h.getNombre())) {
              System.out.println("Row del Shape en OWL encontrado...");
            } else {
              Element addContent = root.addContent(getHotelAsElement (h));
            }
          }
System.out.println("Instancias de Hotel en nuevo
OWL:"+root.getChildren("Hotel",nsbase).size());
         XMLOutputter outputter = new XMLOutputter(Format.getPrettyFormat());
          outputter.output(doc, new FileOutputStream(outFile));
       } catch (JDOMException e) {
          System.out.println("MediatorShape:run():"+e.getMessage());
       } catch (IOException e) {
          System.out.println("MediatorShape:run():"+e.getMessage());
       }
    }

    private  Element getHotelAsElement(Hotel h) {
              // Creamos una nueva etiqueta
```

```
        Element hotelNuevo = new Element("Hotel",nsbase);
        // Añadimos un atributo
        hotelNuevo.setAttribute("ID", h.getID(),rdf);
        //añadimos el elemento longitud
        Element longitud = new Element("longitud",nsbase);
        longitud.setAttribute("datatype","http://www.w3.org/2001/XMLSchema#string", rdf);
        longitud.setText(h.getLongitud());
        hotelNuevo.addContent(longitud);
        //añadimos el elemento direccionSitio
        Element direccionSitio = new Element("direccionSitio",nsbase);
      direccionSitio.setAttribute("datatype","http://www.w3.org/2001/XMLSchema#string",
rdf);
        direccionSitio.setText(h.getDireccion());
        hotelNuevo.addContent(direccionSitio);
        //añadimos el elemento numeroEstrellasHotel
        Element numeroEstrellasHotel = new Element("numeroEstrellasHotel",nsbase);

numeroEstrellasHotel.setAttribute("datatype","http://www.w3.org/2001/XMLSchema#int", rdf);
        numeroEstrellasHotel.setText(h.getNumeroEstrellas()+"");
        hotelNuevo.addContent(numeroEstrellasHotel);
        //añadimos el elemento latitud
        Element latitud = new Element("latitud",nsbase);
        latitud.setAttribute("datatype","http://www.w3.org/2001/XMLSchema#string", rdf);
        latitud.setText(h.getLatitud());
        hotelNuevo.addContent(latitud);
        //añadimos el elemento paginaWebSitio
        Element paginaWebSitio = new Element("paginaWebSitio",nsbase);
        paginaWebSitio.setAttribute("datatype","http://www.w3.org/2001/XMLSchema#string",
rdf);
        paginaWebSitio.setText(h.getPaginaWeb());
```

```
        hotelNuevo.addContent(paginaWebSitio);
        //añadimos el elemento telefonoServicios
        Element telefonoServicios = new Element("telefonoServicios",nsbase);
        telefonoServicios.setAttribute("datatype","http://www.w3.org/2001/XMLSchema#string",
rdf);
        telefonoServicios.setText(h.getTelefonoServicios());
        hotelNuevo.addContent(telefonoServicios);
        //añadimos el elemento costoHospedaje
        Element costoHospedaje = new Element("costoHospedaje",nsbase);
        costoHospedaje.setAttribute("datatype","http://www.w3.org/2001/XMLSchema#string",
rdf);
        costoHospedaje.setText(h.getCostoHospedaje());
        hotelNuevo.addContent(costoHospedaje);
        //añadimos el elemento nombreSitio
        Element nombreSitio = new Element("nombreSitio",nsbase);
        nombreSitio.setAttribute("datatype","http://www.w3.org/2001/XMLSchema#string", rdf);
        nombreSitio.setText(h.getNombre());
        hotelNuevo.addContent(nombreSitio);

        if (h.hasACafe()) {
           //añadimos el elemento hasACafe
           Element hasACafe = new Element("hasACafe",nsbase);
           Element Cafe = new Element("Cafe",nsbase);
           Cafe.setAttribute("ID",h.getCafe().getID(), rdf);
           Element nombreSitioCafe = new Element("nombreSitio",nsbase);

nombreSitioCafe.setAttribute("datatype","http://www.w3.org/2001/XMLSchema#string", rdf);
           nombreSitioCafe.setText(h.getCafe().getNombre());
           Cafe.addContent(nombreSitioCafe);
           hasACafe.addContent(Cafe);
```

```
            hotelNuevo.addContent(hasACafe);
        }

        if (h.hasABar()) {
            //añadimos el elemento hasABar
            Element hasABar = new Element("hasABar",nsbase);
            Element Bar = new Element("Bar",nsbase);
            Bar.setAttribute("ID",h.getBar().getID(), rdf);
            Element nombreSitioBar = new Element("nombreSitio",nsbase);

nombreSitioBar.setAttribute("datatype","http://www.w3.org/2001/XMLSchema#string", rdf);
            nombreSitioBar.setText(h.getBar().getNombre());
            Bar.addContent(nombreSitioBar);
            hasABar.addContent(Bar);
            hotelNuevo.addContent(hasABar);
        }

        if(h.hasARestaurant()) {
            //añadimos el elemento hasARestaurant
            Element hasARestaurant = new Element("hasARestaurant",nsbase);
            Element Restaurant = new Element("Restaurant",nsbase);
            Restaurant.setAttribute("ID",h.getRestaurant().getID(), rdf);
            Element nombreSitioRestaurant = new Element("nombreSitio",nsbase);

nombreSitioRestaurant.setAttribute("datatype","http://www.w3.org/2001/XMLSchema#string",
rdf);
            nombreSitioRestaurant.setText(h.getRestaurant().getNombre());
            Restaurant.addContent(nombreSitioRestaurant);
            hasARestaurant.addContent(Restaurant);
            hotelNuevo.addContent(hasARestaurant);
```

```
            }
            return hotelNuevo;
    }

    public static void main(String[] args) {
        String shpFile = "hoteles_campeche.shp";
        String owlFile = "TurismoCampeche2.owl";
        String outFile = "integrado.owl";
        MediatorShape ms = new MediatorShape(shpFile,owlFile,outFile);
        ms.start();
    }
}
```