UNIVERSIDAD DE VALLADOLID

ESCUELA TÉCNICA SUPERIOR

INGENIEROS DE TELECOMUNICACIÓN

TRABAJO FIN DE MASTER

MASTER UNIVERSITARIO EN INVESTIGACIÓN

EN TECNOLOGÍAS DE LA INFORMACIÓN Y LAS COMUNICACIONES

Diseño de una ontología para aplicaciones en el dominio de la movilidad sostenible: el coche compartido

Autor:

D. Pablo Sauras Pérez

Tutores:

Dr. D. Diego Rafael Llanos Ferraris
Dr. D. Arturo González Escribano

Valladolid, Julio de 2012

TÍTULO:	**Diseño de una ontología para aplicaciones en el dominio de la movilidad sostenible: el coche compartido**
AUTOR:	**D. Pablo Sauras Pérez**
TUTORES:	**Dr. D. Diego Rafael Llanos Ferraris** **Dr. D. Arturo González Escribano**
DEPARTAMENTO:	**ATC, CCIA**

Tribunal

PRESIDENTE:	**Dr. D. Pablo de la Fuente Redondo**
VOCAL:	**Dr. D. Valentín Cardeñoso Payo**
SECRETARIO:	**Dr. D. Jesús Vegas Hernández**
FECHA:	**Julio de 2012**
CALIFICACIÓN:	

Resumen del TFM

El presente Trabajo Fin de Master diseña y desarrolla una ontología en lenguaje OWL *(Ontology Web Language)* para una aplicación en el dominio de la movilidad sostenible, como es el coche compartido. Así, se trata de presentar una base para la aplicación de tecnologías de Web Semántica en este tipo de sistemas. De esta forma, se podría facilitar la interoperabilidad entre los distintos portales existentes de coche compartido, de forma que los distintos usuarios puedan llegar a encontrar compañeros de viaje entre usuarios pertenecientes a otros portales. Esto supondría una posible solución para el problema de falta de masa crítica que hace que este tipo de iniciativas de movilidad sostenible no tengan el éxito deseado (al menos en España).

El diseño de una ontología común a este tipo de aplicaciones facilitaría aun más esta interoperabilidad entre los portales, ya que estos podrían adaptar sus datos almacenados en sus bases de datos a formato RDF *(Resource Description Framework)* tomando como base esta ontología. Así, todos los portales "hablarían un mismo lenguaje". Si además los distintos portales publicaran sus datos siguiendo los principios de *Linked Open Data*, se podría crear una nube de conjuntos de datos relacionados que permitiría la interoperabilidad anteriormente mencionada.

La ontología propuesta se ha diseñado teniendo en cuenta la experiencia profesional del autor de este trabajo en los sistemas de coche compartido, que ha servido para realizar una conceptualización del funcionamiento de los mismos. Esta conceptualización ha servido

como base para el diseño y desarrollo de la ontología propuesta.

Asimismo, se ha realizado una evaluación de la ontología mediante dos tipos de análisis. El primer análisis muestra que la ontología propuesta y la taxonomía que define son consistentes. El segundo análisis muestra que la ontología es capaz de responder a una serie de preguntas planteadas (llamadas "preguntas de competencia") que la ontología ha de ser capaz de responder.

Mediante este análisis se puede considerar esta ontología como una base para facilitar la interoperabilidad entre los portales de coche compartido, aumentando así la probabilidad de los usuarios de encontrar compañeros de viaje; lo que supondría una mejora de la masa crítica de las aplicaciones de coche compartido y, por tanto, de la movilidad sostenible.

Palabras clave

Coche compartido, Ontología, OWL, Web Semántica, *Linked Open Data*, Interoperabilidad.

Abstract

This M.Sc. Thesis proposes the design and development of an OWL *(Web Ontology Language)* ontology for the sustainable mobility domain, in particular for carpooling applications. In this way, this project presents a basis to apply Semantic Web technologies in such carpooling systems. This could bring us to an interoperability scenario between existing carpooling portals, so that users belonging to different carpooling sites could find carpoolers of other sites. This could lead to solve the lack of critical mass problem that affects to this kind of mobility initiatives.

Designing a common ontology for these carpooling applications will further facilitate this interoperability, as they may map their relational databases data to RDF *(Resource Description Framework)* using this ontology. Thus, all carpooling portals would "speak the same language". Moreover, if the different carpooling portals publish their data taking into account the *Linked Open Data* principles, a carpooling dataset cloud could be created. This would also benefit to the aforementioned interoperability.

The professional background in carpooling systems of the author of this project has been taken into account in the conceptualization of carpooling applications, that has been the basis for the proposed ontology design and development.

In order to evaluate the proposed ontology, two analysis were carried out. The first one shows that the ontolology and the taxonomy that it defines are consistent. The second one shows that the ontology is able to answer some competency questions, that establish the scope and utility of the ontology.

This analysis allows this ontology to be considered as a valid way to bring the interoperability between carpooling portals or websites. This could increase the matching probability, which would enhance the critical mass of carpooling initiatives and therefore benefit the sustainable mobility.

Keywords

Carpooling, Ontology, OWL, Semantic Web, Linked Open Data, Interoperability.

Agradecimientos

Quisiera aprovechar estas líneas para agradecer el apoyo recibido a todas aquellas personas que han estado a mi lado de una u otra forma durante la realización de este Master.

A mis padres y hermanos, por creer siempre en mi y apoyarme en todo momento... parte de este trabajo es vuestro.

A toda mi familia, por todo el apoyo y cariño recibido.

A mis amigos de Oviedo y Valladolid, en especial a mis compañeros de piso Javi y Jesús.

A La Torzida, por llenar de música y diversión mi vida, haciéndola mucho más fácil.

A GMV, excompañeros y amigos, por introducirme en el mundo profesional de los sistemas de transporte sostenibles.

Y cómo no, a mis tutores Diego R. Llanos y Arturo González por mostrarme todo su apoyo y confianza durante toda mi etapa en el Master.

GRACIAS A TODOS.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

En el mundo existen 700 millones de vehículos circulando, y las previsiones indican que esta cifra aumentará considerablemente, hasta alcanzar, en los próximos 40 años, la cifra de 3000 millones [1]. Teniendo en cuenta esto, cabe pensar que el tráfico en las carreteras, sobre todo en las grandes ciudades, sufrirá un aumento considerable. Esto se traducirá en problemas de circulación, en un aumento de la siniestralidad y en una peor calidad de vida de los ciudadanos, causada entre otros factores, por el estrés que los problemas de movilidad suponen. A su vez, la emisión de gases que los automóviles emiten a la atmósfera se verá incrementada, más aun si se producen atascos, ya que estos incrementan un 16 % las emisiones de CO_2 a la atmósfera.

Por otra parte, actualmente en España circulan casi 30 millones de vehículos, con una media de ocupación de 1,2 personas por automóvil y desplazamiento [2, 3]. Teniendo en cuenta que en un vehículo se suele contar con 4 plazas destinadas a pasajeros, esto supone un verdadero gasto, tanto energético como económico. En la figura 1.1 se muestran unas cifras comparativas del coste de un viaje urbano en vehículo privado, de 5 kilómetros de longitud, por persona y año, según la ocupación [4].

EFECTOS	VEH.PRIVADO CON 1 OCUPANTE	VEH.PRIVADO CON 4 OCUPANTES
COMBUSTIBLE	124 Litros gasolina	28,5 Litros gasolina
INVERNADERO CO_2	347 Kg	79,8 kg
EMISIONES	18,7 Kg	4,3 kg
OCUPACIÓN DE ESPACIO CIRCULANDO	59.520 m².hora	14.880 m².hora
OCUPACIÓN DE ESPACIO APARCANDO	8.928 m².hora	432 m².hora

Figura 1.1: Cifras comparativas según la ocupación del vehículo. m^2.hora = metro cuadrado ocupado durante una hora

Según estudios del IDAE (Instituto para la Diversificación y Ahorro de Energía) [5], en el año 2007 más del 25 % de las emisiones de CO_2 a la atmósfera en España se debieron al transporte. Además el coche es, por detrás del avión, el medio de transporte que más

consumo energético produce por viajero transportado y kilómetro recorrido (ver figura 1.2).

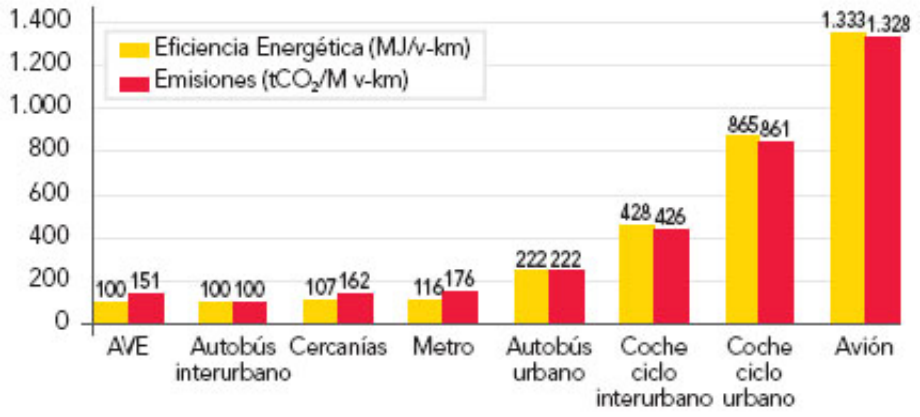

Figura 1.2: Consumo energético comparado de los diferentes modos de transporte

Una de las posibles soluciones para estos problemas de circulación es el uso del coche compartido, que consiste en compartir vehículo entre personas con trayectos y horarios similares. De esta forma, se ocuparían los asientos vacíos anteriormente mencionados; conllevando esto una serie de beneficios, como pueden ser una reducción del tráfico, un ahorro económico de los usuarios y una mejora de su calidad de vida (ya que disminuye las situaciones de estrés, al poder turnarse los ocupantes al conducir), así como una mejora en el medio ambiente. Además, se trata de una solución barata y de fácil implantación.

A pesar de las ventajas que este tipo de sistemas ofrecen, en España no está arraigada la cultura del coche compartido. Uno de los motivos es la dificultad que tienen las personas en encontrar y contactar con otras con las que poder compartir vehículo.

Sin embargo, existen multitud de aplicaciones y portales Web de coche compartido, tales como deAaB [6], Amovens [7], Compatir.org [8] o BlaBlaCar [9]. A pesar de ello, aun no se ha creado la masa crítica suficiente como para que el coche compartido se instaure en nuestro país. Aumentar esta masa crítica es, pues, una necesidad para que este tipo de sistemas pueda llegar a funcionar con cierto éxito.

Una de las posibles formas de conseguir esto es llegar a aumentar las opciones de encontrar compañeros de viaje. La interoperabilidad entre los portales de coche compartido podría ayudar a este propósito, ya que aumentaría el número de usuarios potenciales con los que poder compartir vehículo.

Esta interoperabilidad la puede facilitar el uso de tecnologías de Web Semántica [10]. En particular, la inclusión de las distintas fuentes de datos de los portales de coche compartido en la nube de *Linked Data* [11]. Pero para conseguir esto, se plantean unos pasos previos, como el diseño de una ontología, el mapeado de los datos de las bases de datos a formato RDF *(Resource Description Framework)*, etc.

El presente Trabajo Fin de Master propone la realización de uno de estos pasos, consistente en un primer diseño e implementación de una ontología en lenguaje OWL *(Ontology Web Language)* [12] para una aplicación en el dominio de la movilidad sostenible como es el coche compartido, que pueda servir como primera fase para el desarrollo de este tipo de aplicaciones, haciendo uso de las tecnologías de Web Semántica. De esta forma, teniendo en cuenta la ontología implementada e incluso colaborando en su desarrollo a lo largo de su ciclo de vida, los portales existentes podrían adaptar y publicar sus datos, siguiendo las reglas de diseño de *Linked Open Data* [13]. Esto podría facilitar la interoperabilidad entre las fuentes de datos de los distintos portales, de forma que un usuario de un portal podría ser capaz de encontrar un compañero de viaje de otro portal. Esto provocaría un aumento de las probabilidades de encontrar compañeros de viaje, produciéndose un posible aumento de la deseada masa crítica para las iniciativas de coche compartido.

Ya que, como se ha dicho, este trabajo propone una ontología de una aplicación concreta como es el coche compartido, en la siguiente sección se realiza un primer estudio de este tipo de sistemas, que se ampliará en el capítulo 2.

1.1. El Coche Compartido

El concepto de *Carpooling*, también conocido como *Ridesharing* se remonta a la década de los 70, cuando aparecieron los primeros proyectos formales para fomentar este tipo de prácticas [14]. Desde entonces, compartir el coche para realizar trayectos comunes se ha convertido en una práctica, que si bien no es demasiado popular en España, en otros países de Europa y en Estados Unidos, tiene un nivel de aceptación bastante elevado.

En ciudades de Estados Unidos como Washington o San Francisco, en las que se practica el denominado *Casual Carpool*, un tipo de *Carpool* informal y flexible en el que existen unos "puntos de recogida" de pasajeros a los que acuden los conductores que desean compartir su vehículo, ponen de manifiesto la aceptación que este tipo de sistemas tiene.

En España, en ciudades como Madrid o Barcelona, se permite la circulación de los vehículos compartidos por los carriles VAO (Vehículos de Alta Ocupación). En los alrededores de Barcelona, incluso, se está estudiando implantar descuentos para los vehículos con dos o más ocupantes. Se pone de manifiesto, así, la intención de reducir el uso del vehículo privado por una sola persona.

En esta línea de fomentar el uso del coche compartido, se han desarrollado diversos proyectos y aplicaciones que intentan facilitar el contacto entre personas para compartir el coche, tanto a nivel nacional como internacional. Se nombran a continuación algunas de estas aplicaciones.

Zimride [15]

Esta compañía, afincada en Palo Alto (California, EEUU), es uno de los referentes mundiales en este tipo de sistemas de coche compartido. Fundada en 2007, ha logrado alcanzar la cifra de 300.000 usuarios, repartidos entre usuarios públicos y usuarios pertenecientes a organizaciones. A ello le ha ayudado su clara inclinación por el uso de la red social Facebook, para poder llegar al mayor número de personas posibles. Además, a lo largo de los años, se ha podido observar una evolución del servicio prestado, siendo cada vez más el número de viajes publicados e incluso permitiendo búsquedas no sólo de coches, si no de servicios de autobús.

iCarpool [16]

Esta compañía ha logrado ser reconocida en el Congreso Mundial de ITS (*Intelligent Transport Systems*) del año 2009 celebrado en Estocolmo, con un premio dado por ITS America [17] dentro del llamado *ITS Congestion Challenge* en el que se buscaban las mejores iniciativas para la reducción de la congestión del tráfico. Su evolución desde entonces le ha permitido ofrecer más servicios, como servicios *Real-Time RideSharing*, *School Pool* o *VanPool*.

Avego [18]

Esta compañía ha desarrollado una aplicación para iPhone, de forma que gracias a su GPS, se pueda compartir el vehículo. Avego desarrolla el denominado *"ridesharing dinámico"*, consistente en un sistema de compartir coche en tiempo real, en el que, en cualquier momento, un usuario pueda ver si hay cerca otro con el que compartir coche desde su móvil. Sobre este concepto de *"ridesharing dinámico"* también se puede obtener información en [19]. Su evolución le ha permitido ofrecer servicios de *Vanpool* y gestión de flotas.

Compartir.org [8]

Se trata de una empresa española que da servicio a algunos ayuntamientos de España. De esta forma, los ayuntamientos ofrecen los servicios facilitados por Compartir.org para que los ciudadanos puedan compartir coche. A pesar de todo, el éxito de Compartir SL ha sido escaso, siendo el número de usuarios registrados menor que el deseado.

Blablacar [9]

Este portal es la versión española de Comuto.com [20]. Al igual que otros portales de este estilo, los usuarios registrados pueden buscar compañeros para compartir coche. Por otra parte, ofrece servicios de búsquedas para desplazamientos a campus universitarios y a eventos.

Amovens [7]

Amovens es un portal de coche compartido español. Así, los usuarios registrados, pueden buscar compañeros de viaje. Este portal también ofrece sus servicios a empresas, administraciones, universidades y eventos.

deAaB [6]

deAaB es un portal de viaje compartido en coche ofrecido por la EMT (Empresa Municipal de Transportes) de Madrid [21] a empresas de la Comunidad de Madrid. De esta forma, se pretende fomentar el uso del coche compartido en distintos centros de trabajo y polígonos de la comunidad. Así, los usuarios registrados, pertenecientes a distintos centros de trabajo, pueden buscar compañeros de viaje, tanto en sus centros de trabajo como en agrupaciones, formadas por centros de trabajo o empresas que estén próximas. De esta forma se pretende reducir los efectos negativos (como los indicados anteriormente) de la denominada movilidad obligada (aquella que se produce por motivos laborales).

Como se acaba de ver, existen diversos portales de coche compartido. Sin embargo, y como por otra parte parece lógico, dentro de un portal, únicamente los usuarios registrados en el mismo pueden publicar sus viajes y buscar compañeros de viaje. Esto limita las posibilidades de encontrar compañeros de viaje únicamente a usuarios registrados dentro del mismo portal.

La idea principal de este Trabajo Fin de Master es llegar a conseguir que usuarios de distintos portales de coche compartido puedan encontrar compañeros de viaje pertenecientes a otros portales, como se muestra en la figura 1.3. Para esto, se ha pensado en el uso de la Web Semántica como medio para conseguirlo.

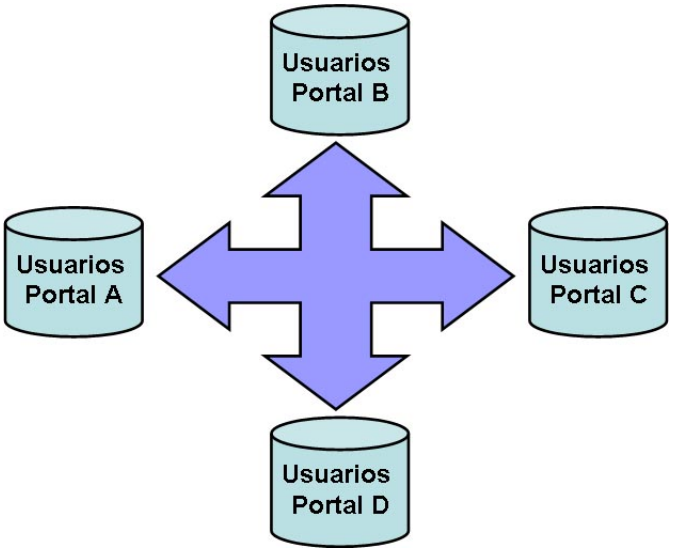

Figura 1.3: Interoperabilidad entre portales de coche compartido

En la siguiente sección se hace un breve estudio de la Web Semántica, intentando encontrar posibles puntos de unión entre la interoperabilidad deseada de los portales de coche compartido y esta tecnología.

1.2. La Web Semántica

En el año 2001 Tim Berners-Lee [10] pensó en la Web Semántica como una extensión de la Web conocida hasta el momento, en la que los contenidos y documentos que la forman están diseñados para ser leídos por humanos, priorizando la presentación de dichos datos sobre su significado. En contraposición, la Web Semántica se centra en darle un significado a los datos, de forma que la información pueda ser interpretada automáticamente por las máquinas *(machine-readable)*. Para conseguir esto, la Web Semántica debe proporcionar una infraestructura que permita a las máquinas acceder a una estructura común de datos, de forma que se consiga la integración y enlazado de los mismos en una "gran base de datos" procedentes de diferentes fuentes, siendo posible incluso la reutilización, combinación e integración de los mismos, facilitando la interoperabilidad entre distintos sistemas y fuentes de datos. Además, la Web Semántica ha de establecer una de una serie de reglas que hagan que las máquinas puedan "razonar" sobre los datos que han de interpretar.

Para describir la información de los recursos de la Web de forma que ésta pueda ser interpretada por las máquinas, la Web Semántica hace uso del lenguaje RDF *(Resource Description Framework)* [22]. Así, RDF proporciona un modelo estándar para el intercambio de información en la Web entre diferentes fuentes sin que ésta pierda su significado. Para ello, RDF codifica el significado de los datos que representa en un conjunto de triplas. La tripla es el elemento básico de RDF y está formada por tres componentes: sujeto - predicado - objeto. Así, una tripla se puede representar como dos nodos (sujeto y objeto), unidos por una relación (predicado). Estas triplas se pueden representar en forma de grafos dirigidos de forma que se pueda ver la estructura de los datos (como se muestra en la figura 1.4). Para lograr que los datos descritos por las triplas estén disponibles en la Web y se puedan referenciar, RDF dota a cada elemento de la tripla (sujeto, predicado y objeto) de un identificador único, llamado URI *(Universal Resource Identifier)* [23], que facilita la creación de enlaces entre los datos procedentes de diferentes fuentes, permitiendo la combinación e integración de estos datos.

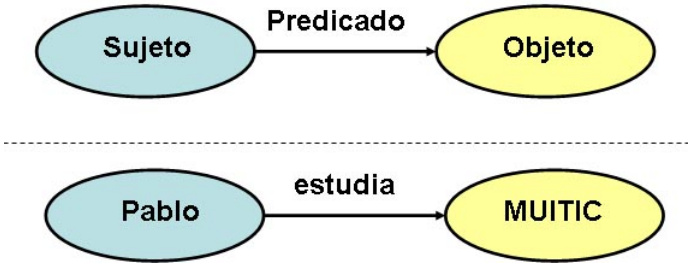

Figura 1.4: Grafo de una tripla

Para realizar consultas sobre los datos representados en formato RDF, la Web Semántica hace uso del lenguaje de consulta SPARQL *(SPARQL Protocol and RDF Query Language)* [24]. De esta forma, las búsquedas mediante SPARQL se realizan sobre los grafos RDF; pudiendo extraer así información sobre los datos descritos en diferentes fuentes y las relaciones entre ellos.

Otro de los componentes básicos de la Web Semántica lo constituyen las ontologías. Una ontología es una definición de un modelo de la realidad que se quiere describir. De esta forma, una ontología introduce un vocabulario describiendo varios aspectos de la realidad o dominio que se está modelando y provee una especificación explícita del significado de ese vocabulario [25]. Para ello, las ontologías definen los conceptos (clases) de lo que está describiendo, así como sus propiedades, atributos, relaciones y restricciones o limitaciones en el uso de estas relaciones.

Las ontologías permiten realizar, pues, un análisis del dominio que se quiere describir. Por otra parte, las ontologías facilitan el compartir el conocimiento del dominio a tratar, permitiendo la reutilización del mismo, la integración de los datos y la interoperabilidad entre los diferentes sistemas. En el capítulo 3 se profundiza algo más sobre las ontologías.

Como indica Tim Berners-Lee en [13], la Web Semántica no sólo se trata de introducir y acceder a los datos en la web; sino que es necesario establecer relaciones y enlaces entre las diferentes fuentes de datos, para que, efectivamente, nos encontremos ante una "gran base de datos", la Web de Datos. *Linked Data*, basándose en unos principios de diseño establecidos en [13] (los denominados *"Principios de Linked Data"*), hace posible este entrelazado entre distintas fuentes de datos, de forma que se puedan llegar a encontrar datos relacionados navegando a través de esta Web de Datos. Esto es posible, entre otros, gracias al uso de URIs para identificar los objetos que se están describiendo (primer principio).

El proyecto *Linked Open Data* [26], coordinado por W3C *(World Wide Web Consortium)* [27], nace en 2007 con el propósito de facilitar conjuntos de datos abiertos, gratuitos, pertenecientes a diferentes fuentes, a toda la Web, de forma que cualquiera pueda acceder a esa información. Así, en el año 2007 nació la denominada *"Nube de Linked Data"* (ver figura 1.5) formada por conjuntos de datos pertenecientes a diferentes fuentes, publicados siguiendo los principios de *Linked Data* mencionados anteriormente. Partiendo de una cifra inicial de doce conjuntos distintos de datos, la nube ha ido evolucionando uniéndose nuevos conjuntos de datos de distintas fuentes hasta llegar a una cifra de 295 actualmente [11].

En esta nube se encuentran disponibles conjuntos de datos pertenecientes a distintas áreas, tales como medios de comunicación, publicaciones, ciencias de la vida, datos geográficos, etc. Además, existen ciertas fuentes de datos que proporcionan información que abarca varios dominios. Son las llamadas *"cross-domain data sources"* (como la conocida

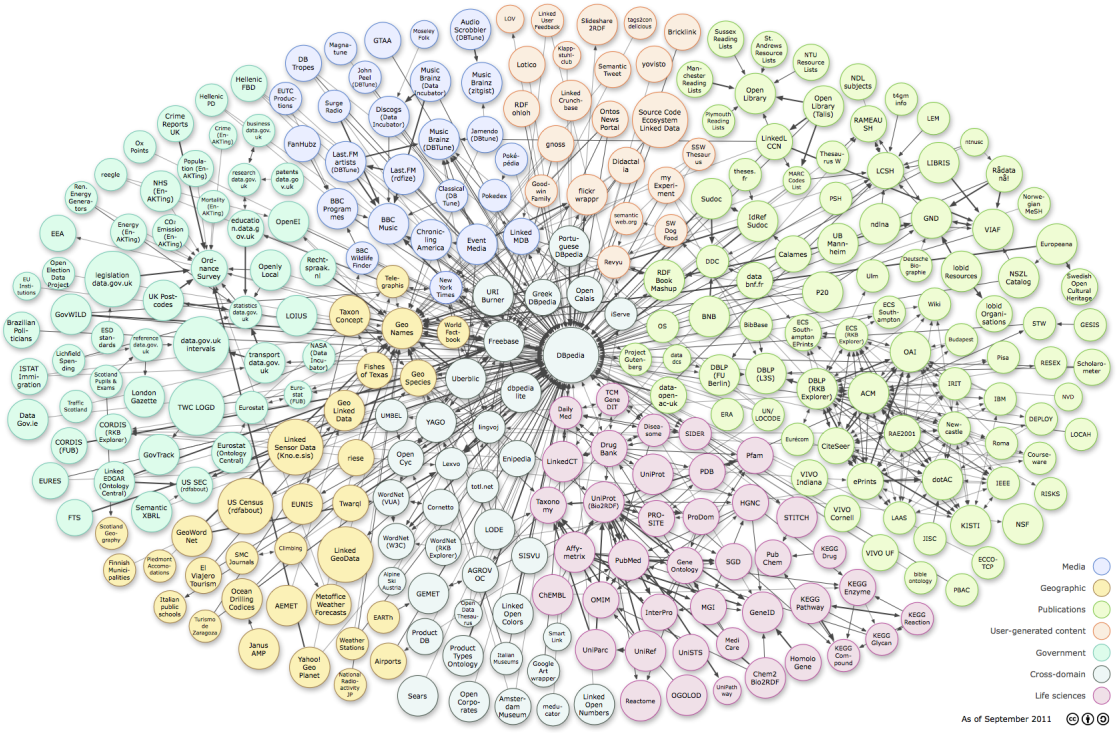

Figura 1.5: Nube de *Linked Data* en Septiembre de 2011

DBpedia [28]), cuya existencia es fundamental para mantener la nube como un único espacio de datos y evitar la fragmentación de la misma en distintas "islas" de conocimiento independientes [29].

Ya que la Web Semántica (la "Web de Datos") permite al interrelación entre datos de diversas fuentes, facilitando la interoperabilidad entre ellas, parece que puede ayudar a resolver el problema mencionado en la sección 1.1 de la interoperabilidad entre los portales de coche compartido. Este es, pues, el principal motivo del uso de tecnologías de Web Semántica en el para aplicaciones de coche compartido. Como primer paso para alcanzar este objetivo, se propone el diseño de una ontología para el coche compartido, de forma que los distintos miembros de la comunidad (portales) puedan tener una base común sobre la que trabajar, mapear sus datos, etc.

1.3. Motivación

En las secciones anteriores (1.1 y 1.2) se ha realizado un primer estudio del coche compartido y de la Web Semántica. Respecto al coche compartido (sección 1.1), se pueden extraer tres puntos destacables:

- En España (y en el resto del mundo) existen iniciativas, en forma de portales web para el fomento del coche compartido. A través de estos portales, los usuarios regis-

trados pueden publicar viajes que quieran compartir, así como buscar compañeros de viaje.

- En España no está arraigada la cultura del coche compartido. Esto es debido a que no se cuenta con una masa crítica suficiente para el crecimiento de este tipo de sistemas.

- Si se consiguiera que los usuarios de un portal puedan buscar viajes ofrecidos por usuarios de otro portal distinto, las probabilidades de encontrar compañeros de viaje aumentaría, así como la masa crítica de los sistemas de coche compartido.

Por otra parte, tras realizar un primer estudio de la Web Semántica (sección 1.2), se pueden extraer los siguientes puntos relevantes:

- La idea de integración y enlazado de los datos pertenecientes a diferentes fuentes de datos, como si de una "gran base de datos" se tratara, que promueve la Web Semántica es considerada de utilidad y adecuada para conseguir el tercer punto de la lista anterior.

- Los portales dedicados al coche compartido podrían publicar sus datos según las reglas de diseño de *Linked Open Data*. Así, podría ser posible la integración, enlazado y reutilización de esos datos.

- Aunque los portales podrían realizar un mapeado de sus datos a formato RDF, se ve necesaria la definición previa de una ontología común del dominio, que defina los conceptos, relaciones y atributos del mismo. De esta forma, se puede facilitar la interoperabilidad de los datos de cada portal de una forma más sencilla, ya que todos partirían de una base común sobre la que basarse.

Por otra parte, tras un estudio del estado del arte de aplicaciones y dominios de actuación de la web semántica ([30], [31], [32]), no se han observado aplicaciones de coche compartido que hagan uso de ella. Además, tampoco se han encontrado fuentes de datos u ontologías que tengan que ver con el dominio del coche compartido ([33], [34]). Nos encontramos, pues, ante un elemento motivador adicional, que es la originalidad de la propuesta.

Como se ha dicho en el tercer punto de la lista anterior, el diseño e implementación de una ontología se ve necesario como primer paso para la consecución del objetivo general de la interoperabilidad entre portales de coche compartido. Es precisamente este diseño e implementación de una ontología para aplicaciones de coche compartido el objetivo final de este Trabajo Fin de Master.

1.4. Enunciado y Objetivos

Como se ha venido motivando a lo largo de este capítulo, el objetivo y contribución de este Trabajo Fin de Master puede enunciarse como sigue:

"El presente Trabajo Fin de Master tiene como objetivo el diseño e implementación de una ontología para aplicaciones de coche compartido, pertenecientes al dominio de la movilidad sostenible."

En particular, se implementará una ontología en lenguaje OWL [12], utilizando como herramienta para ello el editor de ontologías Protégé v3.4.7 (disponible en [35]).

Este objetivo viene motivado por la necesidad de interoperabilidad entre portales de coche compartido para así aumentar las probabilidades de encontrar compañeros de viaje, y consecuentemente, la masa crítica de este tipo de iniciativas de movilidad sostenible. La definición de una ontología para este dominio se considera el primer paso para conseguir esta interoperabilidad mediante el uso de tecnologías de Web Semántica.

Por otra parte, para llegar al objetivo de este Trabajo Fin de Master es necesario alcanzar ciertos objetivos parciales, como la conceptualización previa del dominio, la implementación de la ontología o la evaluación de la misma, que forman parte de la metodología que se ha seguido en la realización de este trabajo, como se indica en la siguiente sección.

1.5. Metodología

El objetivo de este Trabajo Fin de Master, mencionado en la sección anterior, es el diseño e implementación de una ontología en lenguaje OWL para aplicaciones de coche compartido (*Carpooling* o *Ridesharing* en inglés). Para conseguir dicho objetivo, la realización de este trabajo ha seguido una metodología que se basa en metodologías conocidas de diseño de ontologías mencionadas en la sección 3.1. Además, como se ha mencionado en la sección anterior, la metodología seguida en este trabajo ha permitido extraer unos resultados intermedios que responden a objetivos parciales del trabajo. La consecución de estos resultados forman el resultado final de este Trabajo Fin de Master.

Así, las fases que se han seguido en la realización de este trabajo se dividen básicamente en:

- **Estudio del contexto y estado del arte.** Se realiza un estudio del contexto en el que se quiere realizar este trabajo, que es la aplicación de tecnologías de la Web Semántica a aplicaciones del dominio de la movilidad sostenible, en particular del coche compartido. Se realiza un estudio del estado del arte tanto del de los sistemas de coche compartido como de la Web Semántica, con el fin de identificar retos del dominio y la viabilidad de la aplicación de las tecnologías de Web Semántica para la resolución de estos retos. Podemos dividir el estudio del estado del arte en dos partes:
  - **Estudio del estado del arte del de sistemas coche compartido.** La experiencia profesional de tres años propia del autor de este Trabajo Fin de Master, en la que ha participado activamente como desarrollador principal en proyectos de coche compartido como la plataforma MOVILOC Comp@rte [36] de la

empresa GMV Sistemas [37] o el portal de coche compartido deAaB [6] de la EMT (Empresa Municipal de Transportes) de Madrid [21], han supuesto una base muy importante para el estudio del estado del arte de estos sistemas y la identificación de sus requisitos y retos en el ámbito de este trabajo. Así, se identifica el problema de la falta de masa crítica que haga que los sistemas de coche compartido tengan el éxito deseado, y se piensa en la interoperabilidad entre los portales de coche compartido como posible solución.

- **Estudio del estado del arte de la Web Semántica.** En el marco de la realización del Master en Investigación en Tecnologías de la Información y las Comunicaciones [38], se ha cursado la asignatura Tecnologías Emergentes en Sistemas Telemáticos [39] que sin duda ha servido como base para el estudio del estado del arte del mundo de la Web Semática. Partiendo de esa base, se ha pensado en la aplicación de la Web Semántica para abordar el problema de la interoperabilidad entre portales de coche compartido anteriormente mencionado. Se ha seguido estudiando esta temática con el fin de plantear la originalidad y viabilidad de la propuesta que realiza este Trabajo Fin de Master. Así, se ha tomado la decisión de qué reto abordar (en este caso la implementación de una ontología en lenguaje OWL [12]) y qué herramienta utilizar para la consecución de ese reto (en este caso el editor de ontologías Protegé [35]).

- **Conceptualización de una aplicación de coche compartido.** Como paso previo se ha realizado una conceptualización de una aplicación de coche compartido. Se han identificado conceptos, relaciones entre ellos y atributos, que puedan definir el dominio tratado. El resultado se presenta en forma de mapa conceptual del dominio, como se ve en la figura 4.1 del capítulo 4.

- **Diseño e implementación de una ontología para una aplicación de coche compartido.** Una vez realizada la conceptualización, se ha implementado la ontología en lenguaje OWL [12], utilizando el editor de ontologías Protegé [35]. Durante esta implementación se ha continuado realimentando el paso de estudio del estado del arte de la Web Semántica, con el fin, por ejemplo, de encontrar ontologías que pudieran servir durante el proceso de implementación (como, por ejemplo la ontología WGS84 *(World Geodetic System 84)* [40], de utilidad para representar la posición de objetos).

- **Evaluación de la ontología.** Se realiza una evaluación de la ontología implementada. Para ello, primeramente se realiza un estudio de sus consistencia, utilizando el razonador RACER [41]. Posteriormente, se comprueba si la ontología desarrollada es capaz de responder unas preguntas de competencia que ha de responder la ontología desarrollada (indicadas en la sección 4.2 del capítulo 4).

- **Análisis y discusión de resultados.** Se analizan y discuten los resultados obtenidos tras la implementación y evaluación de la ontología. Los resultados de esta fase servirán como entrada para la fase siguiente.

- **Conclusiones y propuesta de trabajos futuros.** Tras un análisis y discusión de los resultados obtenidos en este Trabajo Fin de Master, se identifican una serie de

conclusiones y trabajos futuros. Estos trabajos futuros pueden servir como base y motivación para futuros trabajos tanto de investigación como de desarrollo.

1.6. Estructura y Contenidos

Este Trabajo Fin de Master se estructura como sigue:

- Introducción.
- Ánálisis de un sistema de coche compartido
- Conceptos de Ontologías
- Ontología para el coche compartido
- Análisis de resultados
- Consideraciones finales

Además, se incluye una parte final formada por Apéndices y Bibliografía.

El capítulo 1 de **Introducción** presenta un primer análisis del problema que trata este Trabajo Fin de Master, motivando el mismo. Además, presenta el enunciado y los objetivos del trabajo, así como una descripción de la metodología seguida para la realización del mismo.

Una vez presentado el problema que aborda este trabajo, en el capítulo 2 **(Análisis de un sistema de coche compartido)** se realiza un análisis de un sistema de coche compartido, teniendo en cuenta la experiencia profesional del autor de este trabajo. Se presenta asimismo un esbozo de una arquitectura de este tipo de sistemas y se identifican conceptos, actores, relaciones y atributos que pueden ser importantes en la posterior conceptualización del sistema.

El tercer capítulo está dedicado al estudio de **Conceptos de Ontologías**. En él se presenta de forma teórica los conceptos más relevantes para este trabajo relativos a las ontologías y el desarrollo de las mismas.

Una vez presentado el problema y estudiado el contexto del mismo, el capítulo 4 **(Ontología para el coche compartido)** detalla el proceso de diseño e implementación de la ontología creada para una aplicación de coche compartido.

El análisis y evaluación de esta ontología se detalla en el capítulo 5, **Análisis de resultados**.

En el último capítulo de **Consideraciones finales** se extraen las principales conclusiones de este Trabajo Fin de Master, así como se realiza una discusión de las mismas y se identifican unos trabajos futuros de interés para la evolución de este trabajo.

La sección de apéndices muestra el código de la ontología desarrollada, así como el código de las búsquedas SPARQL que se han utilizado para el análisis realizado en el capítulo 5.

Por último, la bibliografía cita las fuentes bibliográficas consultadas en la elaboración de este trabajo.

Capítulo 2

Análisis de un sistema de coche compartido

En el capítulo 1 se ha introducido y motivado el objetivo de este Trabajo Fin de Master, que es el diseño e implementación de una ontología para una aplicación del dominio de la movilidad sostenible, como es el coche compartido. El primer paso, pues, consiste en realizar un análisis de este tipo de aplicaciones, identificando puntos y conceptos considerados de importancia para el desarrollo de la ontología. Este análisis se ha realizado teniendo en cuenta la experiencia profesional de tres años adquirida por el autor de este trabajo, así como el análisis de algunos portales de coche compartido, como los mencionados en la sección 1.1.

2.1. Arquitectura de un sistema de coche compartido

Los portales de coche compartido ofrecen a sus usuarios registrados la posibilidad de publicar viajes que vayan a realizar con el fin de encontrar compañeros de viaje. Asimismo, los usuarios pueden realizar búsquedas de compañeros de viaje entre los viajes publicados por los usuarios de la web.

El funcionamiento de este tipo de portales se suele basar en una arquitectura en la que una aplicación web realizar consultas a una base de datos propia, en la que se encuentran todos los datos de los usuarios, viajes, reservas, etc del portal. El esquema de una de estas arquitecturas correspondientes a una aplicación web de coche compartido [36] se puede ver en la figura 2.1.

Como se ve en la figura 2.1, la arquitectura cuenta con una capa de aplicación web, en la que se define el núcleo del sistema. El núcleo de la aplicación realiza llamadas a una capa intermedia de servicios web, que se encarga de realizar las transacciones con la base de datos del sistema, que guarda toda la información relativa a los usuarios del mismo, sus viajes, reservas, etc. El resultado de estas transacciones son devueltos por la capa de servicios web para ser mostrados en la aplicación.

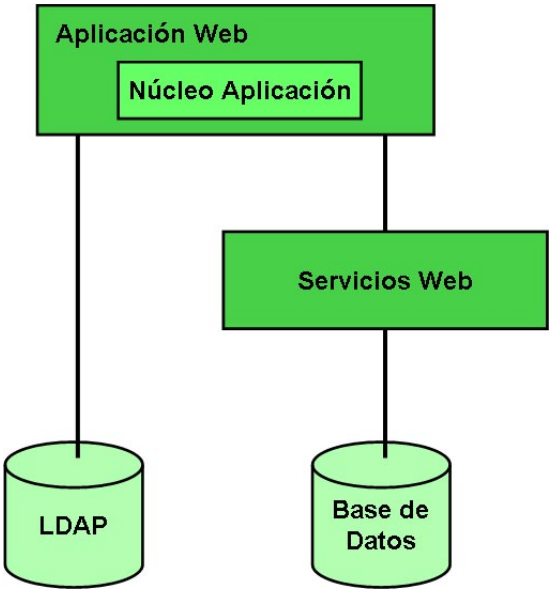

Figura 2.1: Arquitectura básica de una aplicación web de coche compartido

Puesto que para acceder a la aplicación es necesario que un usuario esté registrado y autenticado en el sistema, cabe la posibilidad de que el el proceso de autenticación se realice contra un LDAP (*Lightweight Directory Access Protocol*) [42]; de esta forma se proporciona mayor seguridad a la hora de guardar información, como pueden ser las contraseñas de los usuarios. Una vez autenticado, se guarda la información necesaria acerca del usuario que inicia la sesión, para que éste pueda utilizar la aplicación.

2.2. Conceptos de en un sistema de coche compartido

2.2.1. Definición de Usuarios

Como se ha indicado a lo largo de este trabajo, los portales de coche compartido ofrecen a los usuarios registrados la posibilidad de encontrar compañeros de viaje. Se comienza pues, definiendo qué roles pueden tomar estos usuarios:

- **Conductor.** Un usuario que actúa como conductor es un usuario que ofrece plazas en su vehículo (normalmente un coche) para compartir con otros usuarios (pasajeros) del portal.

- **Pasajero.** Un usuario que actúa como pasajero viaja en el vehículo de un usuario que actúa como conductor. Normalmente no tiene vehículo.

Estas son las definiciones básicas de ambos roles. Sin embargo, cabe destacar que el mismo usuario puede actuar en algunas ocasiones como conductor y en otras como pasajero.

2.2.2. Funcionalidades básicas

Una vez un usuario se ha registrado en el portal web, éste puede realizar distintas funcionalidades básicas:

- **Crear Viaje.** Un usuario tiene la posibilidad de crear un viaje que quiera compartir. Cabe destacar que esta función puede ser realizada por ambos roles: un conductor oferta plazas en su coche y un pasajero demanda plazas en un coche.

- **Buscar viajes.** Un usuario, a traves de un formulario de búsqueda, puede buscar compañeros con los que compartir viaje. Un conductor puede buscar pasajeros que demanden una plaza para ofrecerles una en su coche, y un pasajero puede buscar conductores que ofrezcan plazas en su coche.

- **Realizar reservas de un viaje.** Una vez realizada la búsqueda, un conductor o un pasajero pueden ponerse en contacto con otro usuario para poder viajar juntos. Una vez que el otro usuario acepta la petición de reserva, esta se hace efectiva.

En la figura 2.2 se muestra un esquema de estas funcionalidades básicas, a las que se les ha añadido la funcionalidad de **Gestionar viajes y reservas**, ya que un usuario puede querer ver la información relativa a sus viajes, reservas y usuarios asociados a ellos. Si bien es cierto que se pueden definir otras funcionalidades, como aceptar peticiones, rechazarlas o cancelar viajes; la definición de las mismas no forman parte del objetivo de este trabajo.

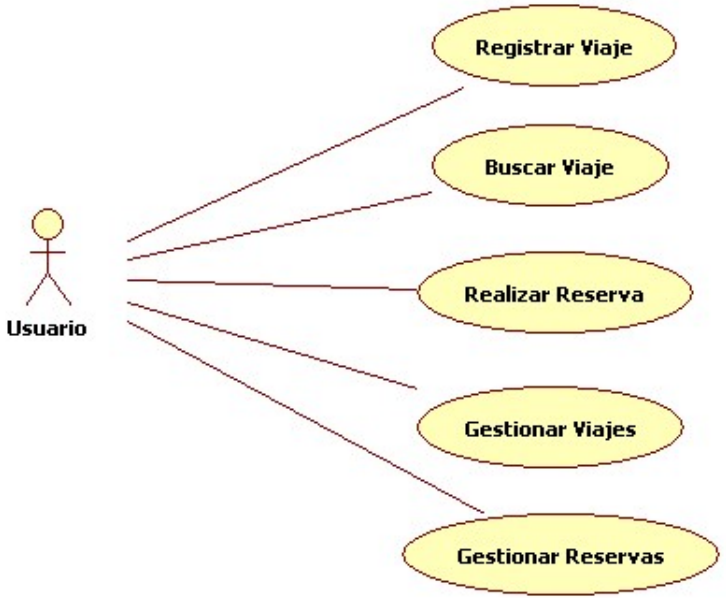

Figura 2.2: Funcionalidades básicas de un usuario

Centrando, pues, nuestra atención en estas funcionalidades, es conveniente explicar el comportamiento de un usuario dentro del portal.

En la figura 2.3 se muestran las funcionalidades de un escenario en el que un conductor crea un viaje, ofertando un número de plazas en su coche. Por otra parte, un pasajero realiza una búsqueda de viajes, encontrando (supongamos) el viaje que el conductor ha creado. En ese momento le realiza al conductor una petición de reserva de plaza, que el conductor acepta. En ese momento, la reserva de plaza del pasajero queda asociada al viaje del conductor.

Figura 2.3: Escenario 1. Pasajero busca Conductor

La figura 2.4 muestra un escenario un tanto distinto, en el que un pasajero crea un viaje demandando una potencial plaza en un coche de un conductor. Por otra parte, un conductor realiza una búsqueda de viajes con el objetivo de encontrar pasajeros a los que poder llevar en su coche. El conductor encuentra el viaje del pasajero y le realiza una oferta de plaza en su coche. El pasajero acepta esa oferta, por lo que queda asociada a ese viaje.

Se ha visto, pues, que nos encontramos ante dos escenarios. Sin embargo, en el capítulo 4 se diseñará la ontología teniendo en cuenta el primero de ellos, ya que el segundo es muy parecido.

2.2.3. Tipos de viajes

Como se ha dicho, una de las funcionalidades básicas de un usuario es la de “Crear Viaje”, ya sea como pasajero o como conductor. Cabe pues, señalar que pueden existir distintos tipos de viaje, según distintos criterios:

- Según el sentido del recorrido un viaje puede ser:

Figura 2.4: Escenario 2. Conductor busca Pasajero

- **Viaje de Ida.** El sentido del recorrido del viaje es únicamente desde su origen hacia su destino.
- **Viaje de Ida y Vuelta.** El sentido del recorrido del viaje va de origen a destino y después de destino a origen.

- Según la frecuencia del viaje:
  - **Viaje Puntual.** Es un viaje que se realiza un día concreto.
  - **Viaje Regular.** Es un viaje que se realiza de forma frecuente. Este tipo de viajes se puede dividir en otros dos tipos:
    - **Viaje Frecuente.** Es un viaje regular que se realiza durante un tiempo determinado; es decir, tiene una fecha de inicio y una fecha de fin. Por ejemplo, un viaje que se va a realizar desde el 1 de Agosto de 2012 hasta el 16 de Septiembre de 2012, los Lunes, Miércoles y Viernes es un viaje de este tipo.
    - **Viaje Habitual.** Es un viaje regular que se realiza siempre, habitualmente; es decir, todos los días. Un viaje de este tipo se puede dar, por ejemplo, en viajes de casa al trabajo, que se suelen hacer todos los días, sin límite definido de tiempo (sin fecha de inicio ni de fin).

2.2.4. Grupo de usuarios

Como se ha visto en la sección 1.1, algunos portales de coche compartido ofrecen servicios a grupos de usuarios, ya sean colegios, universidades, centros de trabajo, eventos, etc. Estos grupos de usuarios tienen en común normalmente el destino de los viajes que realizan. Esto implica que los usuarios pertenecientes a un grupo podrán buscar compañeros de viaje dentro de ese grupo; de forma que los usuarios que no pertenecen a él no pueden realizar búsquedas sobre ellos.

2.2.5. Definición de atributos

En este capítulo se está viendo que un portal de coche compartido está formado por usuarios que pueden actuar como conductores o como pasajeros. Dependiendo de esos roles, un usuario puede tener coche, pertenecer a un grupo, ofrecer un viaje o realizar una reserva de un viaje. El siguiente paso en este análisis de conceptos de un sistema de coche compartido es definir ciertos atributos y características que tienen los conceptos más importantes identificados hasta ahora: usuario, vehículo, grupo, viaje y reserva.

- **Usuario.** En general, un usuario de portal de coche compartido puede tener estos atributos básicos:
  - **Identificador de usuario.** Es un número único que identifica al usuario dentro del portal.
  - **Nombre.** Nombre del usuario.
  - **Apellidos.** Apellidos del usuario.
  - **Login.** Login del usuario. Normalmente es único. Aunque la autentificación en los portales se suele hacer con el e-mail del usuario, en ocasiones se pide el login.
  - **E-mail.** Dirección de e-mail del usuario. Normalmente es requerida en el proceso de autentificación del usuario en el portal. A esta dirección de e-mail se le envía al usuario las notificaciones del portal (peticiones de reserva, cancelaciones, etc).
  - **Número de teléfono.** Número de teléfono del usuario. En ocasiones utilizado como medio informativo de contacto o incluso para enviar mensajes de aviso, en caso de tratarse de un teléfono móvil.
  - **Contraseña de acceso.** Es la contraseña de acceso necesaria para que el usuario pueda autenticarse en el portal.
  - **Puntuación.** En la mayor parte de los portales, los usuarios pueden evaluar a otros usuarios, por lo que un atributo de puntuación puede ser de utilidad.
- **Vehículo.** De forma general, un vehículo asociado a un conductor puede tener estos atributos básicos:
  - **Identificador de vehículo.** Es un número único que identifica al vehículo en el portal.
  - **Matrícula.** Es la matrícula del vehículo. En ocasiones este atributo no se utiliza, por motivos de privacidad de datos.
  - **Marca.** Es la marca del vehículo.
  - **Plazas.** Es el número de plazas que tiene el vehículo. Aunque como se verá a continuación, no tienen por qué coincidir con el número de plazas que se ofertan en un viaje.
  - **Confort.** Es el nivel de confort de un vehículo. Por ejemplo, básico, cómodo, de lujo, ...

- **Grupo.** Es el grupo al que puede pertenercer un usuario. Sus atributos pueden ser:
  - **Identificador del grupo.** Es un número único que identifica al grupo dentro del portal.
  - **Nombre del grupo.**
  - **Localización del grupo.** Típicamente se trata de la dirección del grupo, aunque puede referirse a polígonos, zonas, etc. Es de utilidad, por ejemplo, para grupos formados por usuarios pertenecientes a empresas.
- **Viaje.** Un viaje que un usuario publica con el fin de encontrar compañeros con los que compartirlo, puede tener estos atributos básicos:
  - **Identificador de viaje.** Es un número único que identifica al viaje dentro del portal.
  - **Origen.** Es el lugar de partida del viaje.
  - **Destino.** Es el lugar de llegada del viaje.
  - **Hora de Salida.** Es la hora a la que se parte del origen.
  - **Hora de Vuelta.** En caso de tratarse de un viaje de ida y vuelta, se indica también la hora de vuelta.
  - **Plazas ofertadas.** En caso de que el usuario que crea el viaje sea un conductor, se indican las plazas que se ofertan en su vehículo para ese viaje. Como se ha dicho, este número de plazas puede no coincidir con el número de plazas del vehículo del conductor, ya que puede ofertar menos plazas de las que tiene el mismo.
  - **Precio.** En algunos portales los conductores pueden poner un precio por pasajero a su viaje, con el fin de compartir gastos.

  Por otra parte, según el tipo de viaje, podemos tener otros atributos adicionales:

  - **Viaje Puntual.**
    - **Fecha del viaje.** Al tratarse de un viaje que se va a realizar un único día, se indica la fecha en la que se va a realizar el viaje.
  - **Viaje Frecuente.**
    - **Fecha de inicio.** Es la fecha en la que se comienza a realizar ese viaje.
    - **Fecha de fin.** Es la fecha en la que se termina de realizar ese viaje.
    - **Días de la semana.** Son los días de la semana en los que se realiza ese viaje.
  - **Viaje Habitual.** En este caso no hay atributos adicionales ya que no hay restricciones de duración de viaje (se realiza todos los días).
- **Reserva.** Una reserva de un viaje puede tener estos atributos básicos:
  - **Identificador de reserva.** Es un número único que identifica la reserva dentro del portal.

- **Lugar de recogida.** En este caso el lugar de recogida no tiene por qué coincidir con el origen del viaje sobre el que se realiza la reserva.
- **Lugar de destino.** En este caso el lugar de destino no tiene por qué coincidir con el destino del viaje sobre el que se realiza la reserva.
- **Fecha.** Día para el que se realiza la reserva.
- **Hora de recogida.** Hora de encuentro propuesta para el viaje. No tiene por qué coincidir, pero lo normal es que sí lo haga.

Por otra parte, estamos hablando de orígenes y destinos. Éstos son puntos geográficos que constan de los siguientes atributos básicos:

- **Dirección.** Es la dirección del punto (calle, ciudad, código postal, ...).
- **Latitud.** La latitud del punto, normalmente en coordenadas cartesianas.
- **Longitud.** La longitud del punto, normalmente en coordenadas cartesianas.

Estos son los atributos y características consideradas básicas en un portal de coche compartido. Se podrían indicar características adicionales, como por ejemplo, flexibilidad en la hora de salida; pero para los objetivos de este trabajo, bastan con los anteriormente indicados. En la figura 2.5 se muestra un esquema con las relaciones básicas entre los conceptos vistos hasta ahora (teniendo en cuenta el escenario en el que un conductor crea un viaje y un pasajero realiza una reserva de ese viaje).

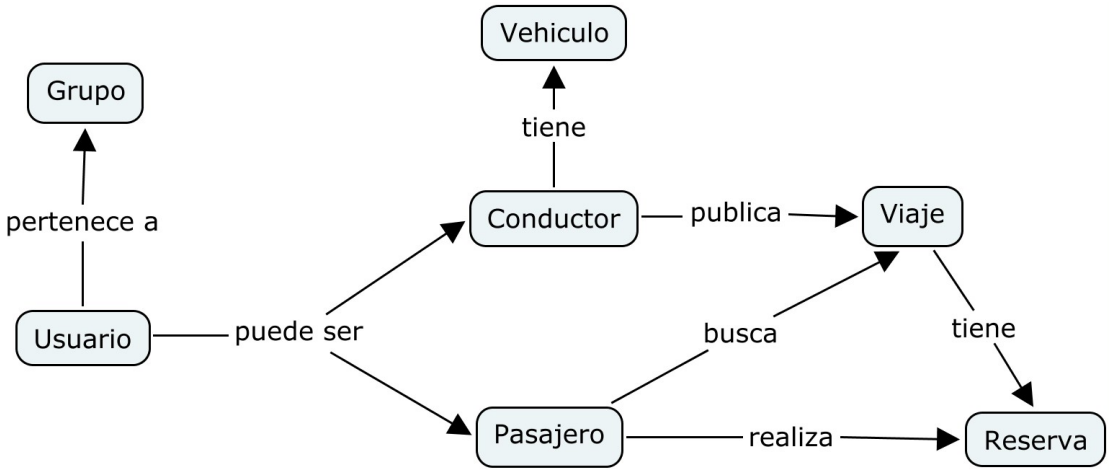

Figura 2.5: Relaciones entre conceptos de coche compartido

2.2.6. Búsqueda de viajes

Una de las funcionalidades básicas que se ha indicado en un portal de coche compartido es la búsqueda de viajes. Un usuario, mediante un formulario de búsqueda puede realizar una búsqueda para encontrar compañeros de viaje. Los resultados de una búsqueda se pueden mostrar tanto en formato tabla como en mapa. En la figura 2.6 se muestra un resultado de una búsqueda en un portal de coche compartido.

Figura 2.6: Resultado de una búsqueda de viajes

Por otra parte, estas búsquedas se pueden realizar teniendo en cuenta diversos factores de filtrado, que coinciden en su mayor parte con los atributos indicados en la sección anterior

- **Origen** del viaje.
- **Destino** del viaje.
- **Fecha** del viaje.
- **Hora de Salida** del viaje.

De forma adicional, se pueden indicar otros parámetros de búsqueda, como pueden ser:

- **Sentido del trayecto.** Ya sea viaje de ida o viaje de ida y vuelta; en cuyo caso se puede indicar también la hora de vuelta.
- **Tipo de viaje.** Ya sea viaje puntual, frecuente o habitual.
- **Distancia al origen.** Flexibilidad de distancia al origen sobre el que se busca. Por ejemplo, un usuario puede querer realizar una búsqueda en un radio de 1 km. de un origen concreto. En ese caso, se le mostrarían los viajes cuyos orígenes estén dentro de ese radio.
- **Distancia al destino.** De la misma forma que con el origen, un usuario puede querer realizar una búsqueda en un radio a un determinado destino.
- **Flexibilidad en la hora de salida.** Teniendo como base una hora de salida, un usuario puede querer que le muestren viajes que tengan horas de salida dentro de un rango. Por ejemplo, una búsqueda con hora de salida de las 12:00 con 30 minutos de flexibilidad, encontraría viajes con hora de salida entre las 11:30 y las 12:30.

- **Flexibilidad en la hora de vuelta.** Se trata del mismo caso que el anterior.

En la figura 2.7 se muestra un modelo de formulario de búsqueda viaje (extraído de [7]).

Figura 2.7: Ejemplo de un formulario de búsqueda de viaje

En este capítulo se ha relizado un análisis de los sistemas de coche compartido. Así, se ha visto cómo estos sistemas hacen uso de sus propias bases de datos, en las que almacenan la información relativa a sus usuarios, viajes y reservas. De la misma forma, se han descrito las principales funcionalidades de los usuarios, así como se han identificado una serie de atributos y características básicas tanto de usuarios, como de viajes y reservas. Este análisis sirve como base para el desarrollo de la ontología propuesta en este trabajo que se verá en el capítulo 4. Pero antes de describir el desarrollo de la ontología, se ve necesario plantear una serie de conceptos teóricos importantes para el desarrollo de la misma. Estos conceptos se presentan en el capítulo siguiente.

Capítulo 3

Conceptos de Ontologías

En el capítulo 2 se han analizado los conceptos, atributos, propiedades y relaciones más importantes del dominio del coche compartido. Partiendo de este análisis, en el capítulo 4 se realiza el diseño e implementación de una ontología para este dominio. En este capítulo se analizan los conceptos más destacados referentes a ontologías que se utilizan en el capítulo 4.

3.1. Ontologías

Las ontologías constituyen uno de los elementos clave de la Web Semántica [10]. Tienen un origen filosófico, constituyento estas una rama de la metafísica cuyo objetivo era el estudio de la estructura del mundo, determinando las entidades y tipos de entidades que existían [25].

En el mundo de la informática, las ontologías se utilizan para capturar el conocimiento de un dominio concreto de interés, modelando y describiendo los conceptos que conforman el dominio y las relaciones que existen entre ellos [43]. De esta forma, una ontología define de forma típica una taxonomía del dominio y establece una serie de reglas de inferencia. Así, los agentes software, por ejemplo, pueden llegar a realizar razonamientos sobre los conceptos del dominio. Otro de los objetivos de una ontología es facilitar la interoperabilidad y reutilización de recursos entre agentes o miembros de un mismo dominio. Una de las formas de conseguir esta interoperabilidad entre los miembros del dominio es crear ontologías compartidas, en las que distintos miembros del dominio participen de forma activa en su desarrollo [44].

Por otra parte, una ontología está compuesta por ([45], [46]):

- **Clases** o conceptos del dominio para el que se quiere realizar la ontología. La jerarquización de las clases forman lo que se conoce como taxonomía. Por ejemplo, en el caso del coche compartido un **`Usuario`** es una clase. Ya que un **`Usuario`** puede ser **`Conductor`** o **`Pasajero`**, se pueden considerar especializaciones (subclases) de la clase **`Usuario`**. Esto constituye una jerarquía de clases, también llamada taxonomía.

- **Propiedades** de las clases que forman la ontología, que describen sus características, atributos o también relaciones entre individuos. Por ejemplo, un **`Usuario`** puede tener una propiedad llamada **`Nombre`**. Por otra parte, la propiedad **`tieneCoche`** puede describir el hecho de que un **`Conductor`** tiene un **`Coche`**.
- **Restricciones** sobre estas propiedades. Se podría indicar, por ejemplo, que un **`Conductor`** tiene al menos un **`Coche`**.
- **Axiomas**, que son sentencias siempre ciertas que pueden servir para restringir, verificar información o deducir información nueva.
- **Individuos**, que son realizaciones concretas de una clase. Por ejemplo, **`Pablo`** puede ser un individuo (o instancia) de la clase **`Conductor`**.

En la figura 3.1 se pueden identificar alguno de estos elementos descritos.

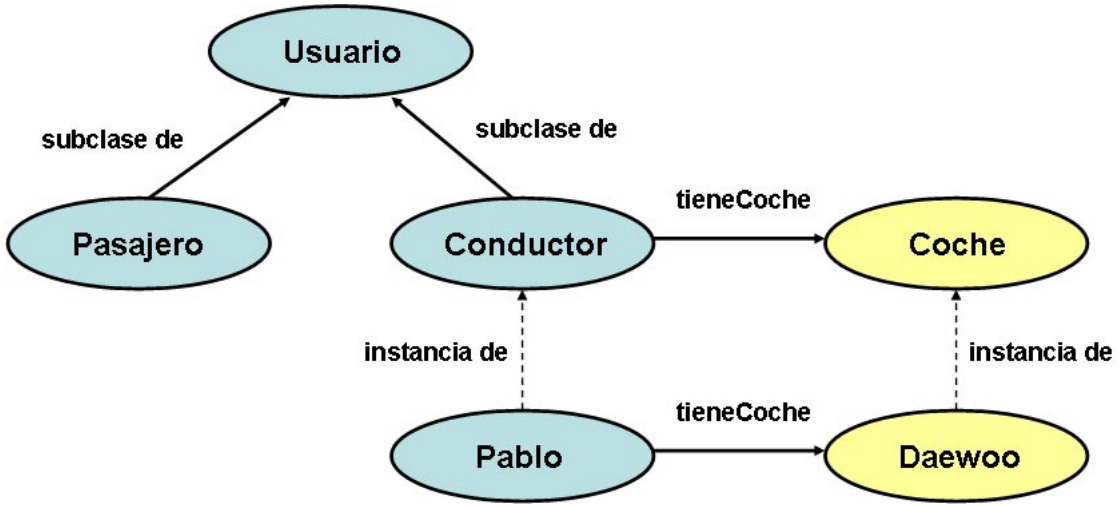

Figura 3.1: Elementos de una ontología

En la web Semántica existen una serie de lenguajes formales para la representación de ontologías [46], tales como los descritos a continuación.

3.1.1. RDFS

RDF Schema *(Resource Description Framework Schema)* ([47]) es la extensión de RDF que define primitivas para la creación de ontologías. Así, con RDF Schema se puede definir jerarquías de clases con herencia múltiple. De esta forma, se pueden establecer relaciones entre clases, subclases, propiedades e instancias.

Siguiendo con el ejemplo de la figura 3.1, con RDFS se puede indicar que **`Conductor`** y **`Pasajero`** son subclases de la clase **`Usuario`** mediante **`rdfs:subClassOf`**, tal y como se ve en la figura 3.2.

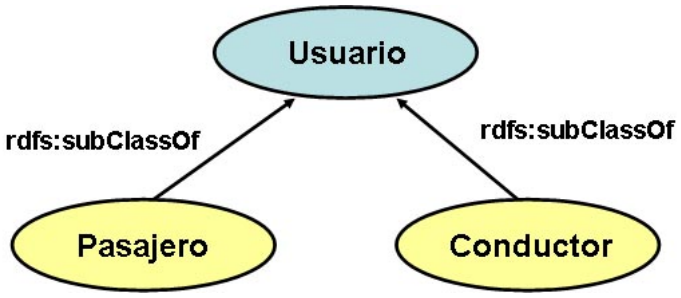

Figura 3.2: Elementos de una ontología

Sin embargo, cabe destacar que RDFS es un lenguaje limitado, que no permite expresar restricciones más concretas (como por ejemplo la cardinalidad) sobre los elementos de la ontología descrita. Se hace, pues, necesaria la creación de nuevos lenguajes que aumenten la expresividad de RDFS.

3.1.2. DAML+OIL

DAML+OIL *(DARPA Agent Markup Language + Ontology Interchange Language)* ([48], [49], [50]) es un lenguaje que ofrece expresiones más sofisticadas de clasificaciones y propiedades de los recursos de las que nos ofrece RDFS. De hecho DAML+OIL es un lenguaje que está basado en RDFS, en el que se redefinen muchas de sus descripciones y se añaden otras para mejorar el lenguaje y aportar propiedades y mecanismos para que se puedan definir ontologías que luego pueden ser empleadas por razonadores, que pueden realizar inferencias sobre la información que proporciona la ontología descrita.

3.1.3. OWL

OWL *(Ontology Web Language)* ([12]) es un lenguaje de especificación de ontologías que surge como extensión del modelo RDFS, proporcionando mayor significado y semántica a las ontologías. Además, OWL es una revisión de DAML+OIL, incorporando parte de sus funcionalidades y añadiendo nuevas. Actualmente existe una extensión de OWL, llamada OWL2 [51].

Con OWL se puede describir una jerarquía de clases y subclases, formando una taxonomía. De la misma forma, OWL permite establecer jerarquías de propiedades, así como definición de axiomas.

Una característica muy importante de OWL es que está basado en Lógica Descriptiva *(Description Logic - DL)* [52] como formalismo para la representación del conocimiento. Esto permite realizar razonamientos (utilizando sistemas razonadores, como se verá en el capítulo 5) complicados sobre la jerarquía de clases y propiedades que forman la ontología.

Por otra parte, OWL se puede clasificar en tres tipos, según su nivel de expresividad:

- **OWL Lite.** Es la versión más simple de todas. Su utilidad se reduce para situaciones en las que únicamente se vaya a definir una jerarquía de clases simple en las que son necesarias únicamente restricciones simples.
- **OWL DL.** Posee mayor expresividad que la anterior. Esta basada en la anteriormente mencionada Lógica Descriptiva, lo que permite que sean razonadas automáticamente por razonadores, que computan de forma automática la jerarquía de clases y buscan posibles inconsistencias.
- **OWL Full.** Es la versión que cuenta con mayor expresividad de todas. Está pensada para utilizarse en situaciones que requieran de una gran expresividad, perdiendo las garantías computacionales; no pudiendo ser tratadas por un sistema razonador.

Para la realización de este trabajo, se ha tenido en cuenta, además, las siguientes propiedades importantes del lenguaje de ontologías OWL:

- Todas las clases son subclases de la clase raíz **`owl:Thing`**.
- **Clases disjuntas** son aquellas clases en las que una instancia no puede pertenecer a dos clases a la vez. Es decir, si un individuo es instancia de una clase, no lo puede ser de su clase disjunta. Por ejemplo, como veremos en el capítulo 4, las clases **`Conductor`** y **`Pasajero`** son disjuntas, ya que en nuestro caso un usuario o es conductor o es pasajero. Esto es importante especificarlo, ya que las clases en OWL por defecto se asumen solapadas.
- Existen dos tipos principales de propiedades:
  - **Propiedades de objeto,** que relacionan una instancia de una clase con una instancia de otra clase. Por ejemplo, en la figura 3.1 mostrada anteriormente, la propiedad **`tieneCoche`** es una propiedad de objeto, ya que relaciona una instancia de la clase **`Conductor`** con una instancia de la clase **`Coche`**.
  - **Propiedades de tipo de datos,** que relacionan una instancia de una clase con un tipo de datos (entero, flotante, string, etc). Por ejemplo, si queremos referirnos a la edad de **`Pablo`**, que es instancia de la clase **`Conductor`**, esto lo podemos describir con la propiedad **`tieneEdad`**, siendo la edad un entero (con valor 30, por ejemplo).
- Una propiedad de objeto puede tener su **propiedad inversa**. Si una propiedad relaciona una **instancia a** con una **instancia b**, entonces su inversa relaciona la **instancia b** con la **intancia a**. Por ejemplo, como se ve en la figura 3.3, una instancia de la clase **`Conductor`** se relaciona con una instancia de la clase **`Coche`** a través de la propiedad **`tieneCoche`**; su inversa podría ser **`esCocheDe`**, que relaciona la instancia de **`Coche`** con la instancia de **`Conductor`**.
- Una propiedad es **funcional** si para un individuo, este se relaciona como máximo con un individuo de otra clase a través de esa propiedad. Por ejemplo, suponiendo que un **`Conductor`** puede tener únicamente un **`Coche`** registrado en un portal

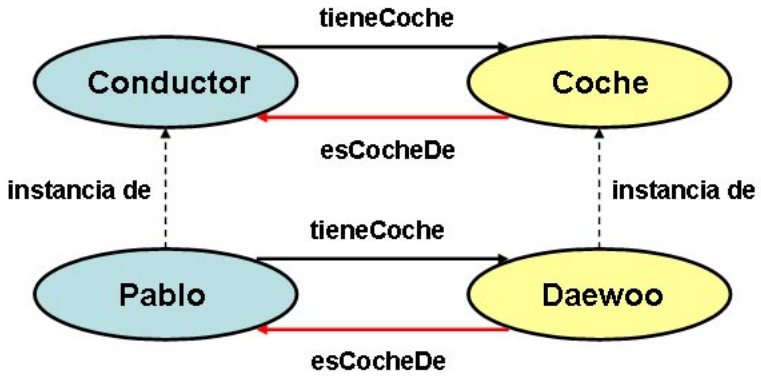

Figura 3.3: Elementos de propiedad inversa

de coche compartido, la propiedad **`tieneCoche`** puede ser considerada **funcional**. Esto quiere decir que si el **`Pablo tieneCoche Coche1`** y **`Pablo tieneCoche Coche2`**, entonces, necesariamente **`Coche1`** y **`Coche2`** son el mismo, ya que **`tieneCoche`** es **funcional**.

De la misma forma, una propiedad es **funcional inversa** si su inversa es **funcional**, cumpliéndose la misma definición que para propiedad funcional. Así, si la propiedad **`esCocheDe`** es **inversa,** quiere decir que si **`Coche1 esCocheDe Pablo`** y **`Coche1 esCocheDe PabloS`**, entonces **`Pablo`** y **`PabloS`** son el mismo **`Conductor`**. Esto se puede ver en la figura 3.4.

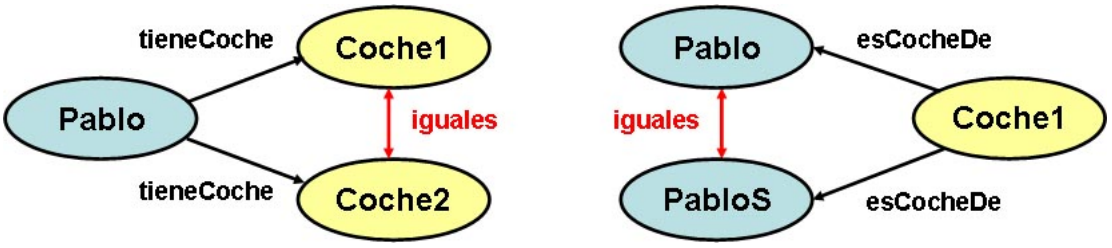

Figura 3.4: Propiedad funcional y funcional inversa

- Existen otros tipos de propiedades, como son las **propiedades simétricas,** que son propiedades que establecen relaciones bidireccionales entre individuos; o las **propiedades transitivas,** que son propiedades que cumplen la propiedad transitiva.
- Es posible definir el **dominio** y **rango** de una propiedad. Así, las propiedades establecen relaciones entre individuos que van desde el **dominio** hasta el **rango** de la propiedad. Adicionalmente, cuando una propiedad tiene múltiples individuos como dominio o rango, estos se tratan como la **unión** de ellos.
- Para definir una clase se puede tener en cuenta una serie de restricciones sobre sus propiedades:

- **Restricciones de valor.** Estas pueden ser:
  - **Existencial** ($\exists$) **(`owl:someValuesFrom`),** que establece la relación entre un individuo de una clase con un individuo de otra o un tipo de datos a través de **por lo menos** una relación establecida por la propiedad. Es decir, **al menos** existe una relación de este tipo; sin embargo, puede haber otros individuos de otras clases que se relacionen también a través de esta propiedad.
  - **Universal** ($\forall$) **(`owl:allValuesFrom`),** que establece si existe una relación entre instancias o una instancia y un tipo de datos, ésta relación **únicamente** puede ser a través de la propiedad sobre la que actúa la restricción. Nótese que esto no implica que dicha relación tenga que existir obligatoriamente.
  - **`owl:hasValue`** ($\ni$)**.** Establece la relación de entre un individuo de una clase y un indiduo concreto de otra a través de la propiedad sobre la que actúa la restricción. La diferencia con **`owl:someValuesFrom`** es que mientras que la primera establece la relacion con algún individuo de otra clase, esta establece la relación con un individuo determinado (es decir, tiene un valor concreto).
- **Restricciones de cardinalidad.** Se utilizan para restringir el número de relaciones que se pueden establecer a través de una determinada propiedad. Estas pueden establecer el **máximo** número de relaciones **(`owl:maxCardinality`)** ($\leq$), el **mínimo (`owl:minCardinality`)** ($\geq$) o el número **exacto** de relaciones **(`owl:cardinality`)** (=).

- Se pueden establecer **condiciones necesarias** y **condiciones necesarias y suficientes** en la descripción de las clases. De esta forma, las **condiciones necesarias** establecen que si un individuo pertenece a una clase, entonces tiene que cumplir las condiciones establecidas por las **condiciones necesarias**. Sin embargo, puede haber individuos de otras clases que también cumplan esa condición.

  Por otra parte, las **condiciones necesarias y suficientes** establecen que si un individuo cumple esa condición, entonces obligatoriamente tiene que pertenecer a la clase donde se define dicha condición, no pudiendo pertenecer a otras.

  Las clases en las que se definen **condiciones necesarias y suficientes** se denominan **clases definidas**, mientras que las clases donde no se definen estas condiciones se denominan **clases primitivas.**

A lo largo del desarrollo de la ontología propuesta se han tenido en cuenta algunas de las propiedades que se acaban de mencionar, así como otras que pueden encontrarse en [12] o [43]. Además, se han tenido en cuenta aspectos metodológicos en dicho desarrollo, como los descritos a continuación.

3.1.4. Metodologías para el desarrollo de ontologías

El proceso de diseño y desarrollo de ontologías es un proceso complejo e iterativo, en el que no existe una metodología "correcta" de desarrollo [45]. Sin embargo, de manera formal, se han propuesto ciertas metodologías para el desarrollo de ontologías, que se describen brevemente a continuación:

METHONTOLOGY [53]

Esta metodología divide el desarrollo de una ontología en las siguientes fases:

1. **Especificación.** El objetivo de esta fase es producir un **documento de especificación** de la ontología (formal o informal) que detalle al menos:
   - El **propósito** de la ontología, incluyendo los posibles escenarios de uso, quién va a utilizarla, etc.
   - El **nivel de formalidad** de la ontología, que depende del nivel de formalidad utilizado en la codificación de los términos de la ontología y su significado. Este nivel de formalidad puede ser altamente informal, semi-informal, semi-formal o altamente formal.
   - El **alcance** de la ontología, que incluye los términos representados, sus características y granularidad.
2. **Adquisición de conocimiento.** Esta fase se produce durante todo el ciclo de vida de la ontología, adquiriendo mayor relevancia durante las primeras fases de especificación y siendo algo más "débil" a medida que el proceso de desarrollo de la ontología va avanzando (cuando la mayor parte del conocimiento está adquirido). Consulta de fuentes bibliográficas, consultas con expertos del dominio, etc, son algunas de las formas comunes de adquisición de conocimiento.
3. **Conceptualización.** El objetivo de esta fase es estructurar el conocimiento en un modelo conceptual que describa el problema y su solución. En esta fase se desarrolla un glosario de términos que incluyen conceptos, instancias y propiedades del dominio.
4. **Integración.** La integración y reutilización de términos utilizados en ontologías existentes es útil en la medida en que no se repiten esfuerzos anteriormente llevados a cabo y se acelera el proceso de desarrollo de la ontología.
5. **Implementación.** Consiste en la codificación en lenguaje formal y computable de la ontología.
6. **Evaluación.** Esta fase en realidad se lleva a cabo durante todo el desarrollo de la ontología, en la que se comprueba que la ontología desarrollada es "correcta".
7. **Documentación.** Se trata de una fase que ha de ser llevada a cabo durante todo el ciclo de desarrollo de la ontología. De hecho, cada fase descrita anteriormente, debe producir la documentación correspondiente (documento de especificaciones, de adquisición de conocimiento, de conceptualización, etc).

***On-To-Knowledge* [54]**

La metodología seguida para el desarrollo de ontologías sigue las siguientes fases principales (mostradas en la figura 3.5, adaptada de [54]):

1. **Estudio de viabilidad.** Esta fase se ha de realizar antes de que comience el desarrollo de la ontología propiamente dicho. En ella se identifica el problema a resolver, así como sus posibles soluciones. El desarrollo de esta fase ayuda a analizar la viabilidad del proyecto que implica el desarrollo de la ontología.

2. **Arranque de la ontología.** En esta fase se genera un documento de especificaciones de requisitos de la ontología. Esto conlleva detallar factores tales como:

   - **Objetivo de la ontología.**
   - **Dominio y alcance** de la ontología.
   - **Aplicaciones que van a usar la ontología.**
   - **Fuentes de conocimiento** que se van a usar para conceptualizar la ontología.
   - **Usuarios y escenarios** a los que va dirigido la ontología.
   - **Preguntas de competencia** que debe responder la ontología.
   - **Ontologías existentes** candidatas para ser reutilizadas.

3. **Refinamiento.** Partiendo de los resultados de la fase de arranque, en esta fase se realiza una conceptualización de los términos de la ontología, para posteriormente pasar a su formalización e implementación añadiendo relaciones y axiomas.

4. **Evaluación.** Esta fase consiste en comprobar la utilidad de la ontología. Para ello, se comprueba que la ontología cumple los requisitos establecidos en el documento de especificaciones de requisitos generado en la fase de arranque. Asimismo, se comprueba si la ontología responde a las preguntas de competencia establecidas en la misma fase. Por otra parte, se puede evaluar la ontología junto a la aplicación en la que se va a utilizar. Los resultados de esta fase servirán para volver a refinar la ontología si es preciso. Así, el proceso que conlleva las fases de refinamiento y evaluación se puede convertir en un proceso cíclico.

5. **Mantenimiento.** Ya que la ontología debe responder a sus especificaciones, es posible que éstas cambien a lo largo de su ciclo de vida. Se hace necesario, pues, un mantenimiento de la misma con el fin de poder actualizarla adecuadamente.

Metodología de la Universidad de Stanford en la guía de Protégé [45]

La metodología propuesta en esta guía, establece una serie de pautas sencillas para el desarrollo de ontologías con el editor de ontologías Protégé [35], que es el que se ha utilizado en este Trabajo Fin de Master. Al igual que en las otras metodologías, se destaca que el desarrollo de una ontología es un proceso iterativo. En particular, se proponen los siguientes pasos que componen dicho proceso:

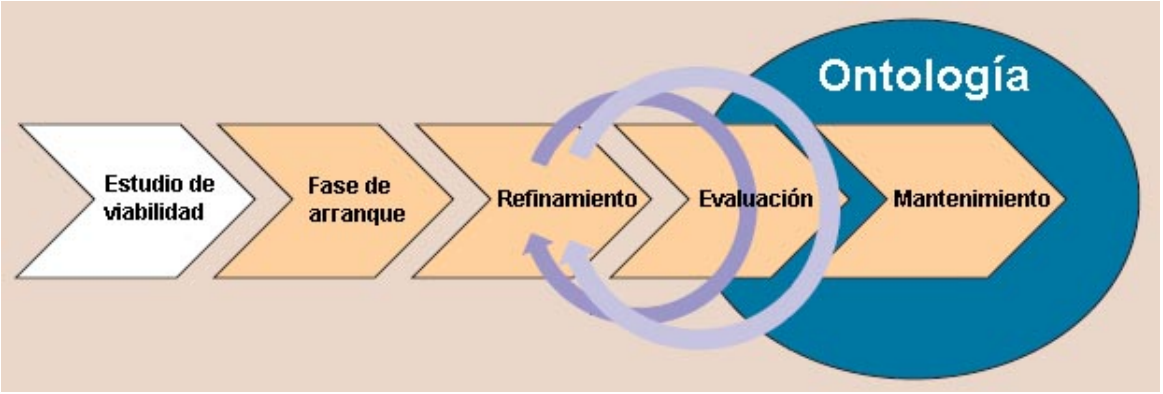

Figura 3.5: Fases de la metodología *On-To-Knowledge*

1. **Determinar el dominio y alcance de la ontología.** En esta fase se establecen cuestiones relativas al dominio para el que se va a desarrollar la ontología, para qué se va a utilizar, qué preguntas ha de responder (preguntas de competencia) o a quién va dirigida (¿quién la va a usar?), así como quién se va a encargar de su mantenimiento. Aunque las respuestas a estas preguntas pueden evolucionar durante el desarrollo de la ontología, es importante plantearlas desde el principio, ya que es un punto de partida muy importante, que ayuda a delimitar y focalizar el objetivo que se quiere alcanzar.

2. **Considerar la reutilización de ontologías existentes.** De esta forma, se pueden reaprovechar recursos de ontologías existentes que se consideren de utilidad para nuestro dominio y objetivo. A su vez, esto puede llegar a facilitar la interconexión de la ontología desarrollada con otras aplicaciones que utilicen estas ontologías reutilizadas.

3. **Enumerar términos importantes para la ontología.** Es útil realizar una lista de los términos o conceptos de los que se quiere tratar en la ontología, anotando también sus propiedades. Esto ayuda a continuar con los siguientes pasos del proceso de desarrollo.

4. **Definir las clases y la jerarquía de clases.** En [55] se distinguen tres posibles enfoques para desarrollar la jerarquía de clases de una ontología:

   - ***Top-down.*** Se comienza definiendo los conceptos generales (clases) y se continúa con las "especializaciones" de estos (subclases).
   - ***Bottom-up.*** En este caso se comienza por la definición de clases más específicas y luego se agrupan para formar conceptos más generales.
   - ***Combinación.*** Se trata de una combinación de las dos anteriores, en la que primero se definen los conceptos más destacados de la ontología y luego se generalizan y especializan según sea el caso.

   El uso de estos enfoques depende de la visión particular del desarrollador de la ontología. Aunque, sea cual sea el enfoque utilizado, lo primero que se suele hacer

es definir las clases de la ontología, que son conceptos de la lista del paso anterior que pueden existir por sí mismos. Estas clases formarán parte de una jerarquía de clases (clases y subclases), que forma la taxonomía de la ontología.

5. **Definir las propiedades de las clases.** Dado que las clases por sí solas no presentan información suficiente como para responder a las preguntas de competencia del primer paso, se considera necesario proveer a estas clases de propiedades, que son heredadas por las subclases de éstas. Así, se pueden diferenciar entre clases "intrínsecas", "extrínsecas", partes (si el objeto es compuesto) y relaciones entre las instancias de una clase y otros términos.

6. **Definir las restricciones de las propiedades**, ya sea tipo de valor, valores admitidos, número de valores (cardinalidad), dominio, rango u otras restricciones.

7. **Crear instancias.** El último paso consiste en crear instancias individuales de las clases en la jerarquía, indicando también los valores de las propiedades de las instancias.

A parte de la especificación de los pasos, [45] explica aspectos importantes a tener en cuenta en el desarrollo de una ontología, al igual que otras referencias como [43] o [56], en las que se utiliza también Protégé como herramienta de desarrollo de ontologías.

Teniendo en cuenta los conceptos vistos en este capítulo, en el siguiente capítulo se describe el desarrollo de la ontología propuesta en este trabajo.

Capítulo 4

Ontología para el coche compartido

En los capítulos anteriores se ha planteado una posible solución para lograr la interoperabilidad entre las aplicaciones existentes de coche compartido utilizando técnicas relacionadas con la Web Semántica. Se ha considerado, además, la conveniencia de plantear un primer paso para lograr esta interoperabilidad, que es el desarrollo de una ontología para este tipo de aplicaciones ([57]).

En este capítulo se describen los pasos seguidos para el desarrollo de una ontología para uno de los casos de uso planteados en el capítulo 2, en el que un conductor registra un viaje y un pasajero realiza una búsqueda de viajes y solicita a un conductor una plaza en su coche para viajar con él. Asimismo, se sigue una metodología basada en las nombradas en el capítulo 3; en mayor medida la metodología de la Universidad de Stanford en la guía de Protégé [45] (ya que es la herramienta utilizada para el desarrollo de la ontología), aunque también se tiene en cuenta la metodología *On-To-Knowledge* [54].

4.1. Estudio de viabilidad

A lo largo del capítulo 1 se ha motivado el objetivo de este Trabajo Fin de Master, constituyendo esta motivación el estudio de viabilidad [54] de la ontología propuesta. Así, se pretende ayudar a resolver el problema de la falta de masa crítica en los sistemas de coche compartido. Si se consiguiera la interoperabilidad entre los distintos portales existentes, de forma que usuarios de un portal puedan encontrar viajes de usuarios de otro portal diferente, las posibilidades de encontrar compañeros de viaje aumentarían. Esto se podría conseguir utilizando los principios de la Web Semántica y *Linked Data*. Así, los portales podrían realizar una conversión de sus datos almacenados en bases de datos a formato RDF y publicar sus conjuntos de datos. Sin embargo, antes de realizar esta conversión, se ve necesario el desarrollo de una ontología común que pueda servir como base, facilitando esto la interoperabilidad entre los conjuntos de datos de los distintos portales. Esto, unido a que no se han encontrado referencias que planteen explícitamente la aplicación de tecnologías de Web Semántica en aplicaciones de coche compartido, hacen suponer que la ontología propuesta puede ser viable.

4.2. Fase de arranque

Una vez estudiada la viabilidad del proyecto que implica el desarrollo de la ontología a desarrollar, el siguiente paso es describir una serie de especificaciones básicas de la misma. Constituye esto la fase de arranque del proceso de desarrollo de una ontología [54], en el que se detallan aspectos tales como el objetivo de la ontología, el dominio y alcance de la misma, etc, tal y como se ha indicado en el capítulo 3.

El objetivo de la ontología propuesta es describir el funcionamiento de una aplicación dentro del dominio de la movilidad sostenible, como es el coche compartido. Las fuentes de conocimiento consultadas que han servido como base, han sido personas expertas en sistemas de coche compartido (el propio autor, sus tres años de experiencia y los profesionales con los que ha tratado, tales como la EMT de Madrid [21]), así como el conocimiento adquirido por el estudio de distintos portales de coche compartido (como Zimride [15], Amovens [7] o deAaB [6]).

La ontología propuesta tiene el objetivo de servir como base para que aplicaciones de coche compartido puedan utilizarla, de forma que sus usuarios puedan encontrar compañeros de viaje (incluso pertenecientes a otros portales). Aunque el desarrollo de una aplicación que utilice esta ontología sea un trabajo futuro (tal y como se indica en el capítulo 6), los usuarios de la ontología son los mismos que los de la aplicación (es decir, usuarios de portales de coche compartido).

Ya que los sistemas de coche compartido utilizan datos relativos a información geográfica e información relativa a fechas y horas, algunas ontologías son potencialmente reutilizables, tales como:

- **WGS84** [1] ***(World Geodetic System 84)*** **[40].** De utilidad para describir objetos que tengan una posición espacial.
- **OWL-Time** [2] **[58].** De utilidad para describir conceptos en los que intervengan fechas, horas, tiempos, etc.
- **Temporal OWL** [3] **[59].** De utilidad para describir conceptos en los que intervengan fechas, horas, tiempos, etc, al igual que la anterior (se trata de una simplificación de OWL-Time).

En la tabla 4.1 se pueden ver estos puntos descritos en la fase de arranque, pudiendo constituir esta tabla el documento de salida de esta fase, tal y como indica [54].

Por otra parte, en la fase de arranque se especifican una serie de preguntas de competencia, que básicamente son preguntas que la ontología ha de ser capaz de responder. Estas preguntas se especifican en la siguiente subsección.

[1] `http://www.w3.org/2003/01/geo/wgs84_pos#`
[2] `http://www.w3.org/2006/time#`
[3] `http://swrl.stanford.edu/ontologies/built-ins/3.3/temporal.owl#`

Objetivo de la ontología	Describir el funcionamiento de un sistema de coche compartido
Dominio y alcance	Movilidad sostenible Búsqueda de viajes
Aplicaciones	Aplicaciones de coche compartido
Fuentes de conocimiento	Expertos en movilidad sostenible Desarrolladores de aplicaciones de coche compartido Aplicaciones existentes de coche compartido
Usuarios y casos de uso	Usuarios de aplicaciones de coche compartido Caso de uso: Búsqueda de viajes registrados por conductores
Preguntas de competencia	Ver subsección 4.2.1
Ontologías potencialmente reutilizables	WSG84 OWL-Time Temporal OWL

Tabla 4.1: Especificación de requisitos de la ontología

4.2.1. Preguntas de Competencia

Uno de los pasos de la fase de arranque consiste en enumerar una serie de preguntas de competencia que el sistema que utilice la ontología debería ser capaz de responder. Establecer estas preguntas pueden ayudar determinar el alcance y contenido de la ontología, pudiéndose a su vez a identificar conceptos, relaciones y atributos que pueden ser de importancia para la misma.

En el capítulo 2 se describieron una serie de funcionalidades básicas que pueden realizar los usuarios de una aplicación de coche compartido. Una de esas funcionalidades es la de **Buscar Viajes**. Son precisamente preguntas relacionadas con la búsqueda de viajes y reservas las que conforman las preguntas de competencia de la ontología propuesta, mostradas en la tabla 4.2. La realización de estas preguntas vienen motivadas por el funcionamiento de las aplicaciones de coche compartido. De esta forma:

- Las preguntas **P1** y **P2** responden a una funcionalidad básica de este tipo de aplicaciones, que es la búsqueda de compañeros de viaje (de viajes registrados en la aplicación). Como se ha indicado en el capítulo 2 los parámetros de filtrado más comunes son los de Origen, Destino, Fecha y Hora del viaje. Es la búsqueda de viajes teniendo en cuenta estos parámetros lo que se han tenido en cuenta en las preguntas **P1** y **P2**.

- La pregunta **P3** viene motivada porque un conductor puede querer saber cuáles son

Identificador	Pregunta de competencia
P1	Viajes según origen, destino u origen y destino
P2	Viajes según fecha, hora o fecha y hora
P3	Reservas asociadas a un conductor
P4	Reservas realizadas para un viaje
P5	Reservas realizadas por un pasajero
P6	Viajes asociados a un grupo
P7	Reservas asociadas a un grupo

Tabla 4.2: Preguntas de competencia para la ontología

las reservas que tiene. Además, también puede querer saber las reservas que tienen sus viajes; siendo esta la motivación de la pregunta **P4**.

- De la misma forma, un pasajero puede querer tener información relativa a las reservas que ha realizado, constituyendo esto la motivación de la pregunta **P5**.
- Como se ha indicado en el capítulo 2 los usuarios pueden formar parte de grupos. Conocer los viajes y reservas de un grupo tiene interés en el caso de que los usuarios quieran restringir sus búsquedas a viajes de su grupo. Esto motiva las preguntas **P6** y **P7**.

Del análisis de estas preguntas se pueden extraer, además, ciertos conceptos que pueden ser importantes para nuestra ontología como son Conductor, Pasajero, Grupo, Viaje, Reserva, Origen, Destino, Fecha y Hora. Como se verá posteriormente, estos conceptos son básicos a la hora de conceptualizar la ontología propuesta. Aun así, cabe destacar que un análisis algo más profundo de los sistemas de coche compartido puede llegar a proponer búsquedas más complejas, como búsqueda por distancia a un origen o un destino, búsqueda según una flexibilidad horaria, etc (algunas de ellas indicadas en el capítulo 2). Sin embargo, este tipo de búsquedas más complejas se considera, como se dice en el capítulo 6, un trabajo futuro que se propone para llevar a cabo el desarrollo de una aplicación de coche compartido que haga uso de esta ontología.

4.3. Refinamiento de la ontología

Partiendo de las conclusiones obtenidas en la fase de arranque, la fase de refinamiento trata de realizar una conceptualización de la ontología para luego proceder a su desarrollo e implementación. Cabe destacar que en esta fase de desarrollo de la ontología, el idioma utilizado es el Inglés.

Por otra parte, esta fase de refinamiento puede ser llevada a cabo las veces que se consideren necesarias, teniendo en cuenta los resultados de la posterior fase de evaluación

de la ontología, descrita en el capítulo 5.

Cabe destacar que los pasos seguidos en esta fase de refinamiento son los indicados en la metodología propuesta por la Universidad de Stanford en la guía de Protégé [45].

Se recuerda, asimismo, que en el caso de estudio que se está tratando, un conductor registra viajes y un pasajero los busca y realiza reservas sobre los viajes del conductor.

4.3.1. Conceptualización

La conceptualización de la ontología propuesta se basa en los resultados obtenidos en la fase de arranque, en la que se han identificado conceptos importantes como Conductor, Pasajero, Viaje o Reserva. De la misma forma, en la sección 2.2 del capítulo 2 se identificaron funcionalidades básicas, atributos y relaciones que se establecen en un sistema de coche compartido. El resultado de esto se muestra en la figura 2.5 de dicho capítulo.

Clases y relaciones entre clases

Partiendo de la base anteriormente mencionada, se puede comenzar a conceptualizar la ontología propuesta, definiendo clases y relaciones entre clases (llamadas propiedades de objeto).

Comencemos por los usuarios y los tipos de usuarios que forman parte de una aplicación de coche compartido:

- **Usuario `(User).`** Es un usuario registrado en una aplicación de coche compartido. Un usuario puede formar parte de un grupo **`(Group)`** de usuarios; es decir, tiene un grupo **`(hasGroup)`**. Además, existen dos tipos (subclases) de usuarios:
  - **Conductor `(Driver).`** Un usuario que actúa como conductor registra viajes **`(Trip)`** para compartir; es decir, tiene viajes **`(hasTrip)`**. Además, en algunos portales el conductor puede dar información adicional, como son los datos de su vehículo **`(Vehicle)`**; es decir, un conductor puede tener un vehículo **`(hasVehicle)`**.
  - **Pasajero `(Passenger).`** Un usuario que actúa como pasajero, realiza búsquedas de viajes, y si encuentra alguno para compartir, realiza una petición de reserva **`(Booking)`** para ese viaje. Es decir, un pasajero tiene reservas **`(hasReservation)`**.

Se acaba de ver que un usuario puede registrar viajes o relizar reservas de dichos viajes. Nos encontramos, pues, ante dos nuevas clases:

- **Viaje `(Trip).`** Como se ha dicho, un conductor tiene viajes registrados para compartir. Es decir, un viaje está asociado a un conductor **`(isTripOf)`** Los viajes pueden tener reservas asociadas **`(hasBooking)`**, que han sido realizadas por pasajeros que quieren compartir ese viaje. Por otra parte, dependiendo de la frecuencia con que se realice un viaje, este puede ser de dos tipos (subclases):

- **Viaje Puntual `(OneDayTrip).`** Es un viaje que se realiza un único día.
- **Viaje Regular `(RegularTrip).`** Es un viaje que se realiza de forma repetida. A su vez, este tipo de viajes puede clasificarse según el rango de tiempo en el que se repite el viaje en dos subclases:
  - **Viaje Frecuente `(FrequentTrip).`** Con duración determinada, este tipo de viajes se repite unos determinados días de la semana en un rango de tiempo determinado (con una fecha de inicio y una fecha de fin).
  - **Viaje Habitual `(HabitualTrip).`** Es un viaje que se realiza todos los días, sin restricción de rango de tiempo (no tiene fecha de inicio ni de fin).

- **Reserva `(Booking).`** Como se ha dicho, un pasajero realiza una reserva de un viaje registrado por un conductor. Es decir, una reserva está asociada a un viaje **`(isBookingOf)`** y tiene pasajeros asociados a ella **`(isReservationOf)`**.

Por otra parte, se ha especificado que un usuario puede formar parte de un grupo. Además, un conductor puede tener un coche registrado. Por tanto, nos encontramos ante otras dos clases:

- **Grupo `(Group).`** Como se ha dicho, un grupo está formado por usuarios **`(hasMember)`**.
- **Vehículo `(Vehicle).`** Un conductor puede tener un vehículo asociado y viceversa **`(isVehicleOf)`**. Además, un vehículo puede ser de varios tipos (subclases). En este caso, se indican dos: **Coche `(Car)`** y **Monovolumen `(Minivan)`**.

En la figura 4.1 se muestra un esbozo de esta conceptualización.

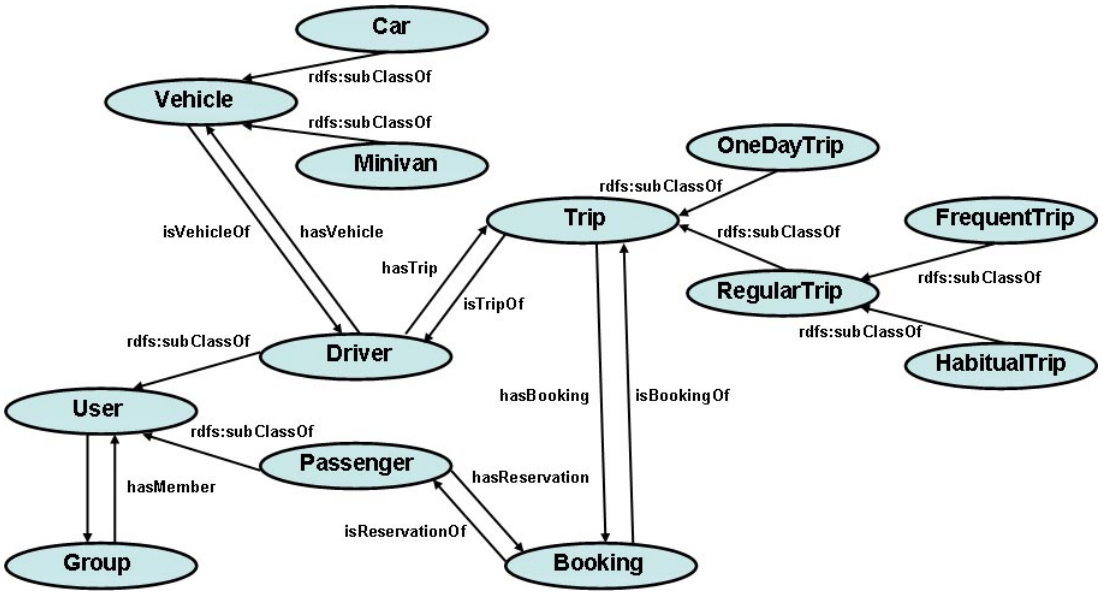

Figura 4.1: Conceptualización de clases de la ontología

Atributos de las clases

Cada una de las clases y subclases descritas, tienen una serie de atributos, que se relacionan con las clases mediante las llamadas propiedades de tipo de datos:

- **`User`.** Un usuario puede tener los siguientes atributos básicos (mostrados también en la figura 4.2):
  - **Identificador `(ID)`** de usuario, que se relaciona con la clase **`User`** mediante la propiedad **`hasID`**.
  - **Nombre `(Name)`** del usuario, que se relaciona con la clase **`User`** mediante la propiedad **`hasName`**.
  - **Apellido `(Surname)`** del usuario, que se relaciona con la clase **`User`** mediante la propiedad **`hasSurname`**.
  - **Login `(Login)`** del usuario, que se relaciona con la clase **`User`** mediante la propiedad **`hasLogin`**.
  - **E-mail `(Email)`** del usuario, que se relaciona con la clase **`User`** mediante la propiedad **`hasEmail`**.
  - **Contraseña de acceso `(Password)`** del usuario, que se relaciona con la clase **`User`** mediante la propiedad **`hasPassword`**.
  - **Teléfono `(Phone)`** del usuario, que se relaciona con la clase **`User`** mediante la propiedad **`hasPhone`**.
  - **Puntuación `(Rating)`** del usuario, que se relaciona con la clase **`User`** mediante la propiedad **`hasRating`**.

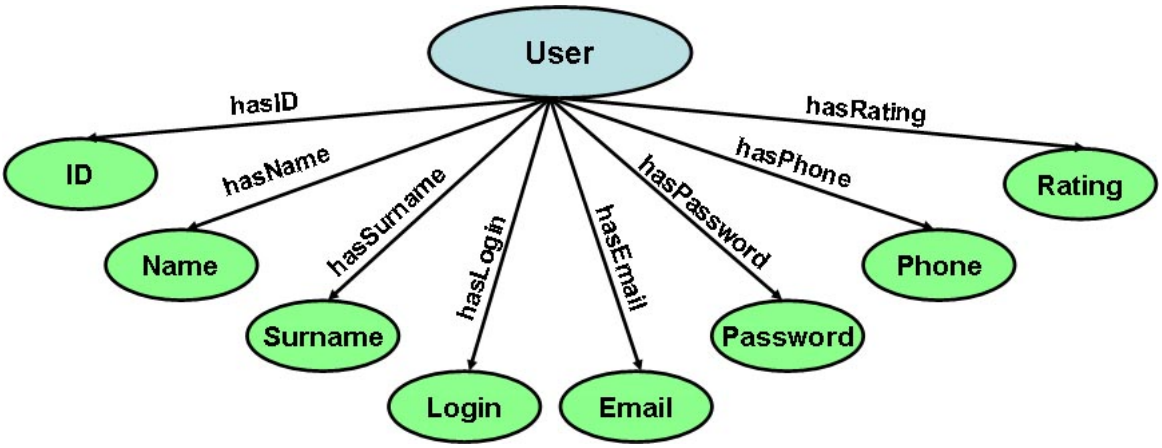

Figura 4.2: Atributos de la clase **`User`**

- **`Group`.** Un grupo de usuarios puede tener los siguientes atributos básicos (mostrados también en la figura 4.3):
  - **Identificador `(ID)`** de grupo, que se relaciona con la clase **`Group`** mediante la propiedad **`hasID`**.

- **Nombre (`Name`)** del grupo, que se relaciona con la clase **`Group`** mediante la propiedad **`hasName`**.
- **Localización (`Location`)** del grupo, que se relaciona con la clase **`Group`** mediante la propiedad **`hasLocation`**.

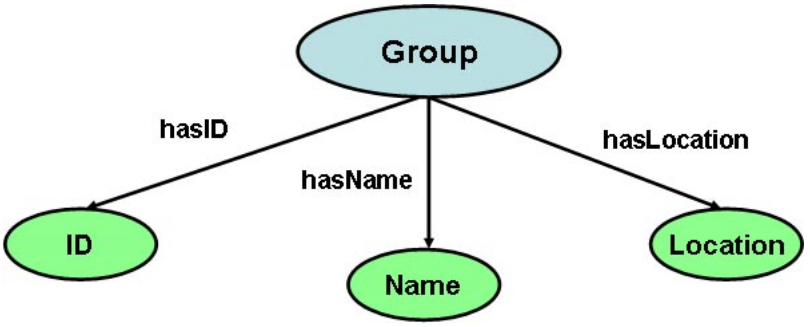

Figura 4.3: Atributos de la clase **`Group`**

- **`Vehicle`.** Un vehículo de un conductor puede tener los siguientes atributos básicos (mostrados también en la figura 4.4):
  - **Identificador (`ID`)** de vehículo, que se relaciona con la clase **`Vehicle`** mediante la propiedad **`hasID`**.
  - **Matrícula (`PlateNr`)**, que se relaciona con la clase **`Vehicle`** mediante la propiedad **`hasPlateNr`**.
  - **Marca (`Brand`)**, que se relaciona con la clase **`Vehicle`** mediante la propiedad **`hasBrand`**.
  - **Plazas (`Seats`)**, que se relaciona con la clase **`Vehicle`** mediante la propiedad **`hasSeats`**.
  - **Confort (`Comfort`)**, que se relaciona con la clase **`Vehicle`** mediante la propiedad **`hasComfort`**.

- **`Trip`.** Un viaje tiene los siguientes atributos básicos, comunes a todos los tipos de viajes (mostrados también en la figura 4.5):
  - **Identificador (`ID`)** de viaje, que se relaciona con la clase **`Trip`** mediante la propiedad **`hasID`**.
  - **Origen (`Origin`)** del viaje, que se relaciona con la clase **`Trip`** mediante la propiedad **`hasOrigin`**. Más adelante se verá cómo se puede conceptualizar el origen.
  - **Destino (`Destination`)** del viaje, que se relaciona con la clase **`Trip`** mediante la propiedad **`hasDestination`**. Más adelante se verá cómo se puede conceptualizar el destino.
  - **Hora de Salida (`DepartTime`)** del viaje, que se relaciona con la clase **`Trip`** mediante la propiedad **`hasDepartTime`**.

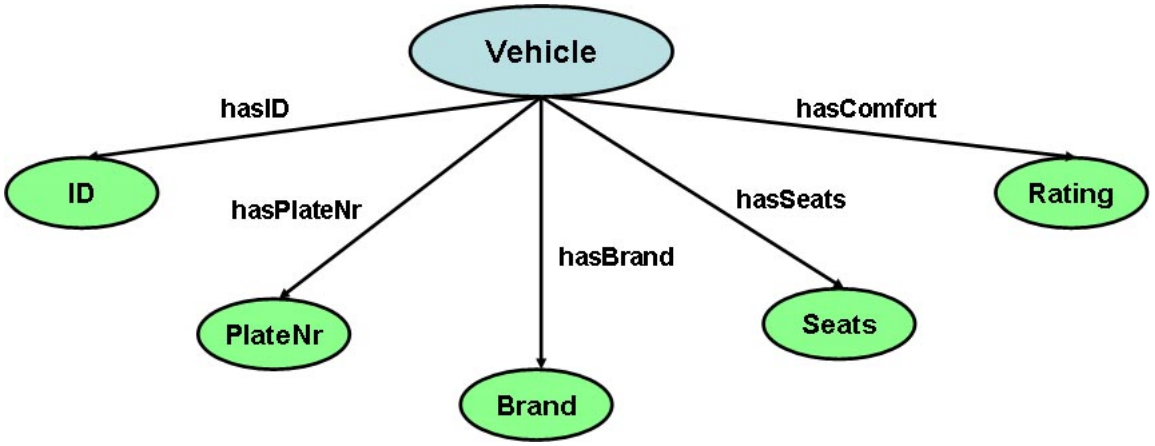

Figura 4.4: Atributos de la clase **`Vehicle`**

- **Hora de Vuelta `(ReturnTime)`** del viaje (si la tiene), que se relaciona con la clase **`Trip`** mediante la propiedad **`hasReturnTime`**.
- **Plazas Ofertadas `(AvailableSeats)`** del viaje, que se relaciona con la clase **`Trip`** mediante la propiedad **`hasAvailableSeats`**.
- **Precio `(Price)`** del viaje (si se establece), que se relaciona con la clase **`Trip`** mediante la propiedad **`hasPrice`**.

Por otra parte, las subclases de la clase **`Trip`** tienen los siguientes atributos:

- **`OneDayTrip`.** Ya que se trata de un viaje que se realiza puntualmente, se pueden definir los atributos relativos a la fecha de salida del viaje y la fecha de vuelta (en caso de haberla) del mismo.
  - **Fecha de Salida `(DepartDate)`** del viaje, que se relaciona con la clase **`OneDayTrip`** mediante la propiedad **`hasDepartDate`**.
  - **Fecha de Vuelta `(ReturnDate)`** del viaje (si la tiene), que se relaciona con la clase **`OneDayTrip`** mediante la propiedad **`hasReturnDate`**.
- **`FrequentTrip`.** En este caso los atributos característicos de esta subclase son los referentes al rango de fechas en el que se realiza el viaje y los días de la semana en que se repiten.
  - **Fecha de Comienzo `(StartDate)`** del viaje, que se relaciona con la clase **`FrequentTrip`** mediante la propiedad **`hasStartDate`**.
  - **Fecha de Fin `(EndDate)`** del viaje, relacionado con la clase **`FrequentTrip`** mediante la propiedad **`hasEndDate`**.
  - **Dia de la Semana `(DaysOfTheWeek)`** en los que se realiza el viaje frecuente, que se relaciona con la clase **`FrequentTrip`** mediante la propiedad **`hasDaysOfTheWeek`**. Más adelante se verá cómo se pueden conceptualizar los días de la semana.

- **`HabitualTrip`.** En este caso, al tratarse de un tipo de viaje en el que no existen mayores restricciones en cuanto a fechas se refiere, no se ve necesario aportar atributos nuevos.

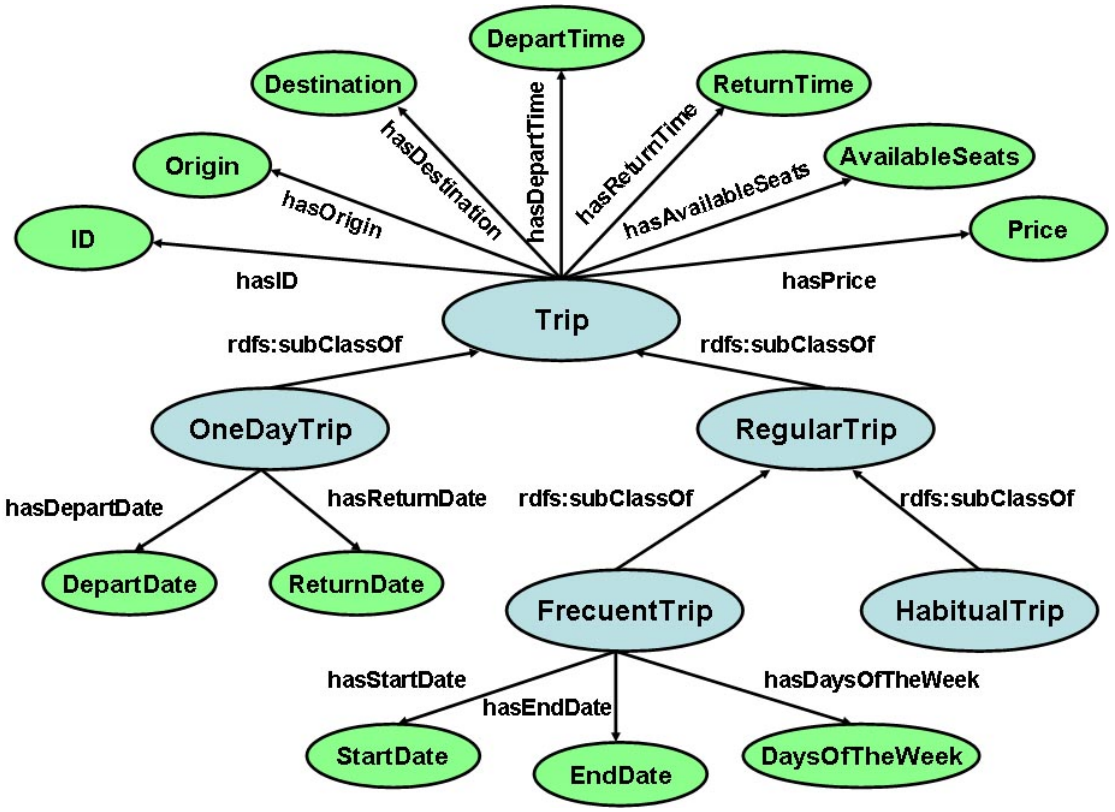

Figura 4.5: Atributos de la clase **`Trip`** y sus subclases

- **`Booking`.** Una reserva de un viaje tiene los siguientes atributos básicos (mostrados también en la figura 4.6):
  - **Identificador `(ID)`** de la reserva, que se relaciona con la clase **`Booking`** mediante la propiedad **`hasID`**.
  - **Origin `(Origin)`** de la reserva, que se relaciona con la clase **`Booking`** mediante la propiedad **`hasOrigin`**. Más adelante se verá cómo se puede conceptualizar el origen.
  - **Destino `(Destination)`** de la reserva, que se relaciona con la clase **`Booking`** mediante la propiedad **`hasDestination`**. Más adelante se verá cómo se puede conceptualizar el destino.
  - **Fecha de Salida `(DepartDate)`** de la reserva, que se relaciona con la clase **`Booking`** mediante la propiedad **`hasDepartDate`**.
  - **Hora de Salida `(DepartTime)`** de la reserva, que se relaciona con la clase **`Booking`** mediante la propiedad **`hasDepartTime`**.

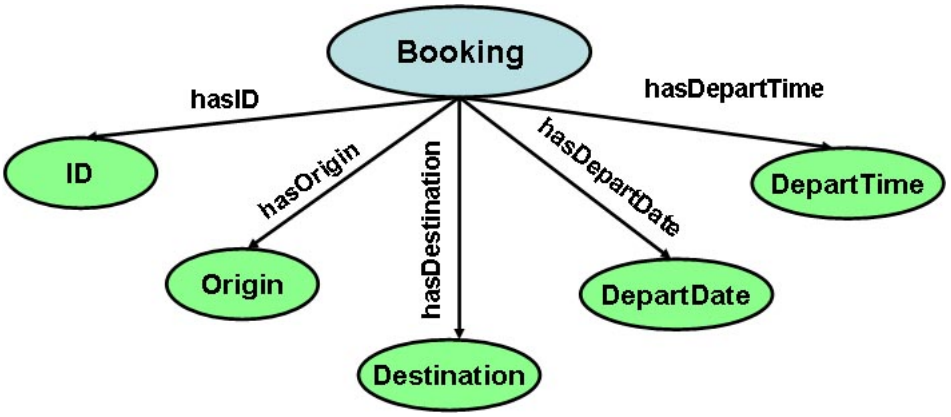

Figura 4.6: Atributos de la clase **`Booking`**

Por otra parte, estamos hablando de orígenes y destinos de viajes y reservas; y localización de grupos. Éstos son básicamente puntos geográficos. Un punto geográfico puede ser considerado como una clase **`(Point)`** y constar de los siguientes atributos básicos (mostrados también en la figura 4.7):

- **Dirección `(Address)`,** que se puede relacionar con la clase **`Point`** mediante la propiedad **`hasAddress`**.
- **Latitud `(Latitude)`,** que se puede relacionar con la clase **`Point`** mediante la propiedad **`hasLat`**.
- **Longitud `(Longitude)`,**que se puede relacionar con la clase **`Point`** mediante la propiedad **`hasLong`**.

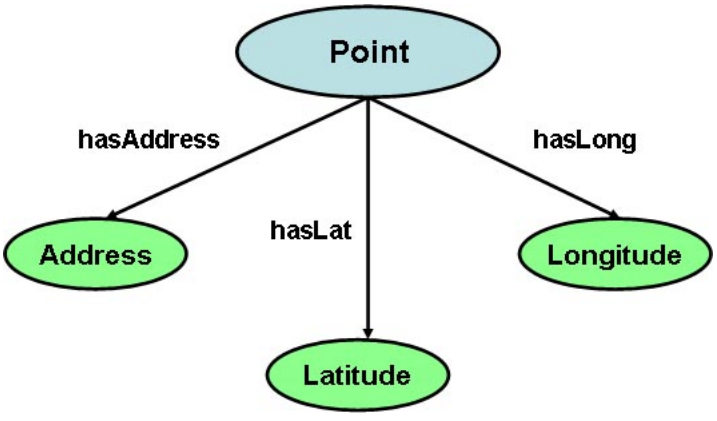

Figura 4.7: Atributos de la clase **`Point`**

Sin embargo, como se ha planteado en la fase de arranque, hay ontologías que son potencialmente reutilizables, como la WGS84, que describe precisamente puntos geográ-

ficos. En la siguiente subsección se plantea la reutilización de algunas ontologías existentes y se describe cómo afecta su reutilización al modelo conceptual de la ontología que acabamos de plantear.

4.3.2. Reutilización de ontologías

La reutilización de ontologías resulta de utilidad en la medida de que se puede aprovechar soluciones existentes a retos que nos encontramos a la hora de desarrollar una ontología [45], pudiendo incluso ampliarla de acuerdo a nuestras necesidades específicas.

Como ya se ha comentado en la fase de arranque, en la ontología propuesta se utilizan conceptos relacionados con puntos geográficos o fechas y horas. A este respecto, se han considerado tres ontologías potencialmente reutilizables.

WGS84 *(World Geodetic System 84)* [40]

Se trata de una ontología básica para la representación de puntos geográficos mediante el uso del sistemas de coordenadas cartográficas WGS84 [4]. Así, se especifica la latitud, longitud y altitud de los puntos.

La jerarquía de clases de esta ontología, mostrada en la figura 4.8, define una clase **`(SpatialThing)`** que se refiere a cualquier cosa que tenga una extensión espacial. Además, esta clase tiene como subclase **`Point`** que describe cualquier punto geográfico. La clase **`(SpatialThing)`** (y por tanto también **`Point`**) tienen como propiedades **latitud `(lat)`, longitud `(long)` y altitud `(alt)`.** Por ejemplo, la ciudad de Valladolid tiene como latitud *41.697526* y como longitud *-4.735107*.

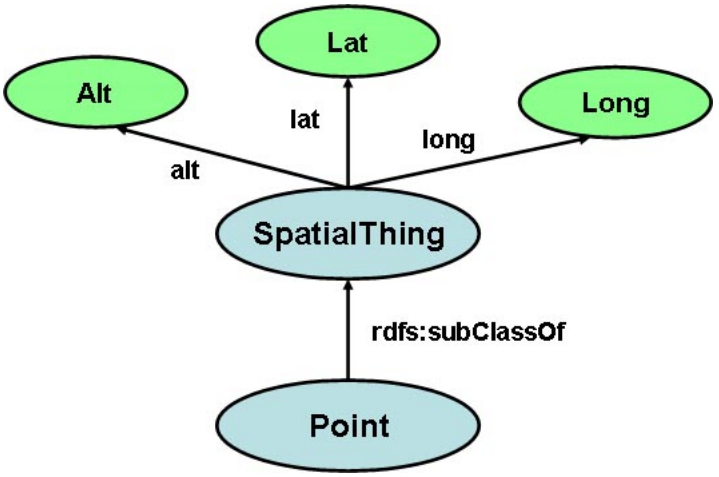

Figura 4.8: Clases y propiedades de WGS84

[4] `http://www.w3.org/2003/01/geo/wgs84_pos#`

Ya que en la ontología que se propone en este Trabajo Fin de Master se cuenta con orígenes o destinos, parece que la reutilización de esta ontología tiene sentido. De esta forma, las propiedades **hasOrigin** y **hasDestination** de las clases **Trip** o **Booking** pueden relacionar las instancias de dichas clases con instancias de la clase **Point** que tiene una latitud y una longitud. El nombre de la calle o del lugar tomado como origen o destino podría indicarse a través de la propiedad **rdfs:label.**

OWL-Time [58]

OWL-Time [5] es una ontología que ha sido desarrollada para presentar y describir conceptos temporales (contenido temporal de páginas web y propiedades temporales de servicios web). De esta forma, la ontología proporciona un vocabulario para expresar hechos y conceptos en los que existan relaciones entre instantes e intervalos de tiempo, así como información sobre la duración, la fecha o la hora. En definitiva, se trata de proporcionar un significado semántico a los conceptos que tienen que ver con el tiempo, hasta ahora especificado mediante un tipo de dato (como *dateTime*) [60].

La descripción de estos instantes o intervalos de tiempo permite obtener más información de la que da el tipo de datos especificado en XML Schema *dateTime* [61]. De esta forma, mediante el uso de esta ontología sería posible describir, por ejemplo, que un evento se produce *antes* de una fecha determinada o en un *intervalo* de años, o que el evento se *solapa* en el tiempo con otro evento.

En la figura 4.9 se muestra parte de la jerarquía de clases que define esta ontología, donde destaca la clase **TemporalEntity**, que tiene como subclases la clase **Instant** y la clase **Interval**, que se refieren a un instante puntual de tiempo y a un rango de tiempo entre dos instantes, respectivamente.

Finalmente no se ha reutilizado OWL-Time para el desarrollo de la ontología propuesta en este trabajo, dejando su posible reutilización como un trabajo futuro que se puede llegar a abordar.

Temporal [59]

En este caso, la ontología Temporal [6] se trata de una versión un tanto simplificada de la anterior, en la que se simplifica tanto el número de clases como de propiedades. En la figura 4.10 se puede ver la jerarquía de clases que define y las propiedades de objeto.

Al igual que con la anterior, no se ha reutilizado esta ontología para el desarrollo de la propuesta en este trabajo, dejando su posible reutilización como un trabajo futuro que se puede llegar a abordar.

[5] `http://www.w3.org/2006/time#`
[6] `http://swrl.stanford.edu/ontologies/built-ins/3.3/temporal.owl#`

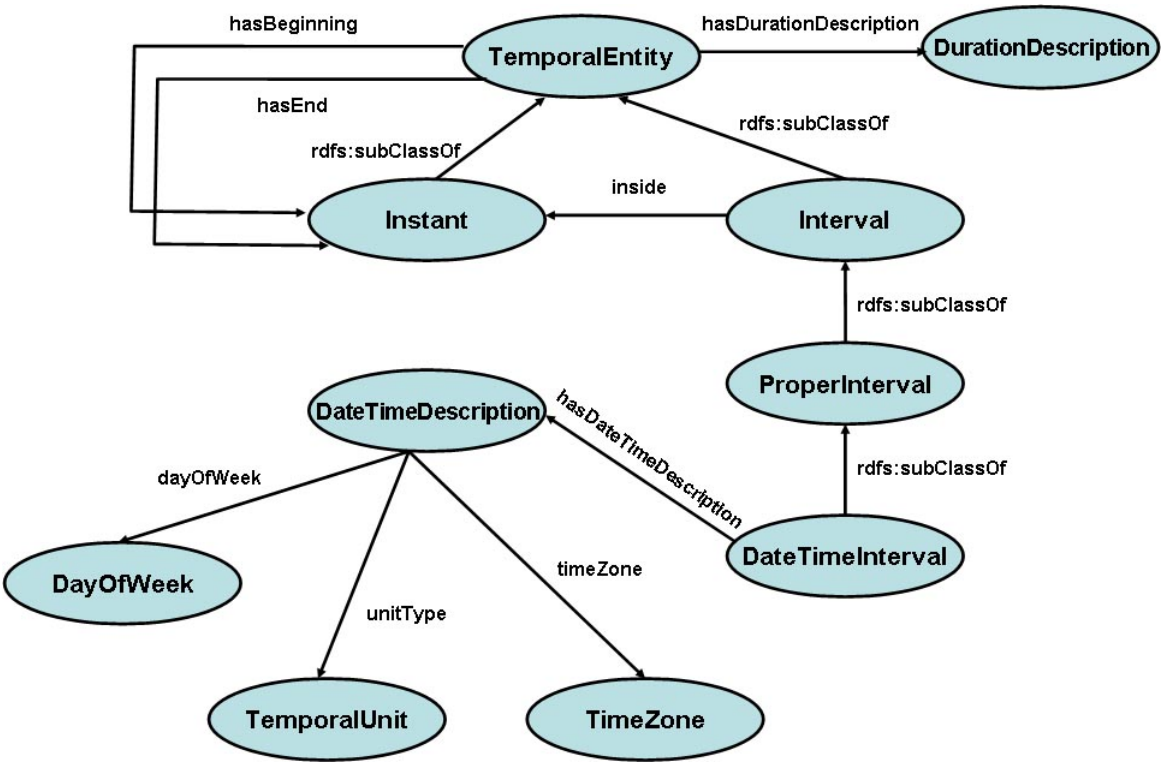

Figura 4.9: Clases y propiedades de OWL-Time

4.3.3. Clases y jerarquía de clases

En la conceptualización de la ontología propuesta (figura 4.1 de la subseccion 4.3.1) se definieron las clases y subclases principales de la misma. Además, se ha explicado también el modo en el que se reutiliza la ontología WGS84. Antes de proceder a indicar la jerarquía de clases de la ontología propuesta, cabe destacar la implementación de una clase que resulta de utilidad para describir los días de la semana en los que se va a llevar a cabo un **viaje frecuente (`FrequentTrip`)**. Se trata de la clase **días de la semana (`DaysOfTheWeek`)**, que es una clase enumerada, que tiene como instancias los siete días de la semana y se relaciona con la clase **`FrequentTrip`** mediante la propiedad **`hasDaysOfTheWeek`**.

La figura 4.11 muestra la jerarquía de clases y las relaciones principales entre clases de la ontología propuesta en este trabajo. Como se ve, se trata de una jerarquía de clases cuya clase raíz es **`owl:Thing`**. Además, las clases del mismo nivel son disjuntas, de forma que instancias de una clase no puedan ser instancias de otra del mismo nivel. Aunque esto no tiene por qué ser del todo cierto, en este trabajo se ha considerado de esta forma. Así, se puede pensar en el caso de la clase **`Driver`** y la clase **`Passenger`**. Como se ha indicado en el capítulo 2, en los portales de coche compartido un usuario puede actuar en ocasiones como conductor y en otras como pasajero (las clases no serían entonces disjuntas). Sin embargo, la implementación de la ontología propuesta considera que un usuario únicamente puede actuar como pasajero o como conductor, tratándose, pues, de

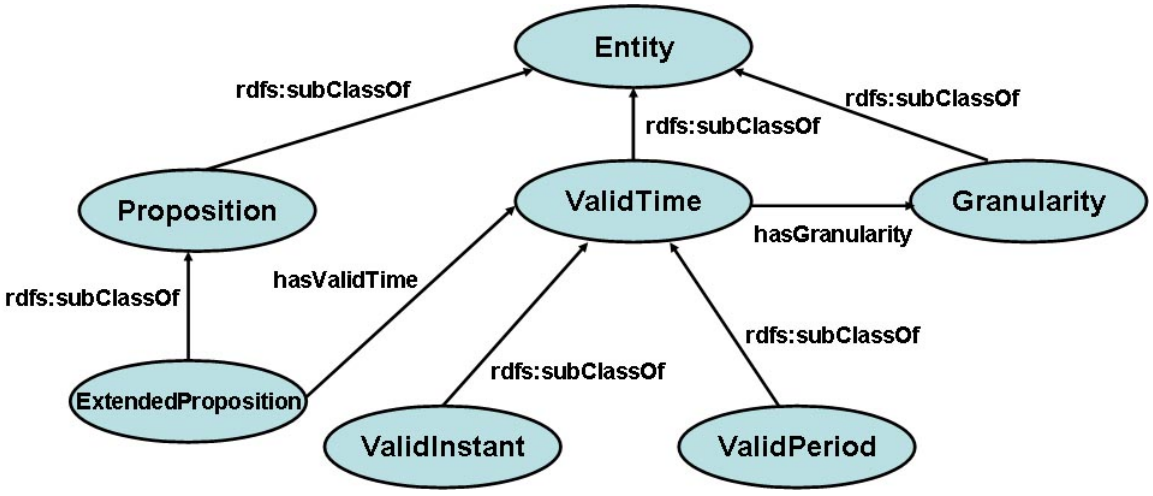

Figura 4.10: Clases y propiedades de Temporal

clases disjuntas. En la descripción de trabajos futuros del capítulo 6 se considera el caso que se acaba de explicar.

4.3.4. Propiedades y restricciones

Las clases por sí solas no proporcionan la información suficiente como para responder las preguntas de competenca que se plantearon en la fase de arranque de la ontología. Es necesario establecer propiedades de las clases y restricciones sobre las mismas.

En el capítulo 3 se vió que OWL definía dos tipos principales de propiedades: de objeto y de tipo de datos. En el primer caso, se relaciona una instancia de una clase con una instancia de otra clase. En el segundo caso se relaciona una instancia de una clase con un tipo de datos (por ejemplo, entero, flotante, etc). Además, también se definieron lo que eran las propiedades funcionales, transitivas, simétricas e inversas.

En las tablas 4.3 y 4.4 se indican las propiedades de objeto y de tipo de datos respectivamente, definidas para la ontología propuesta. Se indican también el dominio y rango de las propiedades, aunque en [43] se recomienda no usar el dominio y rango muy a menudo, ya que en ontologías algo grandes puede causar problemas “inesperados”, ya que no son tratados como restricciones. Aun así esto se ha hecho, ya que se considera que la ontología queda más clara.

Para ambos tipos de propiedades, las decisiones que se han tomado han tenido en cuenta los conceptos vistos en el capítulo 3 referentes al desarrollo de ontologías, así como los referentes a los sistemas de coche compartido del capítulo 2.

Una vez realizado un listado de las propiedades de objeto y de tipo de datos, el siguiente paso es completar las definición de las clases de la ontología con restricciones. Para ello,

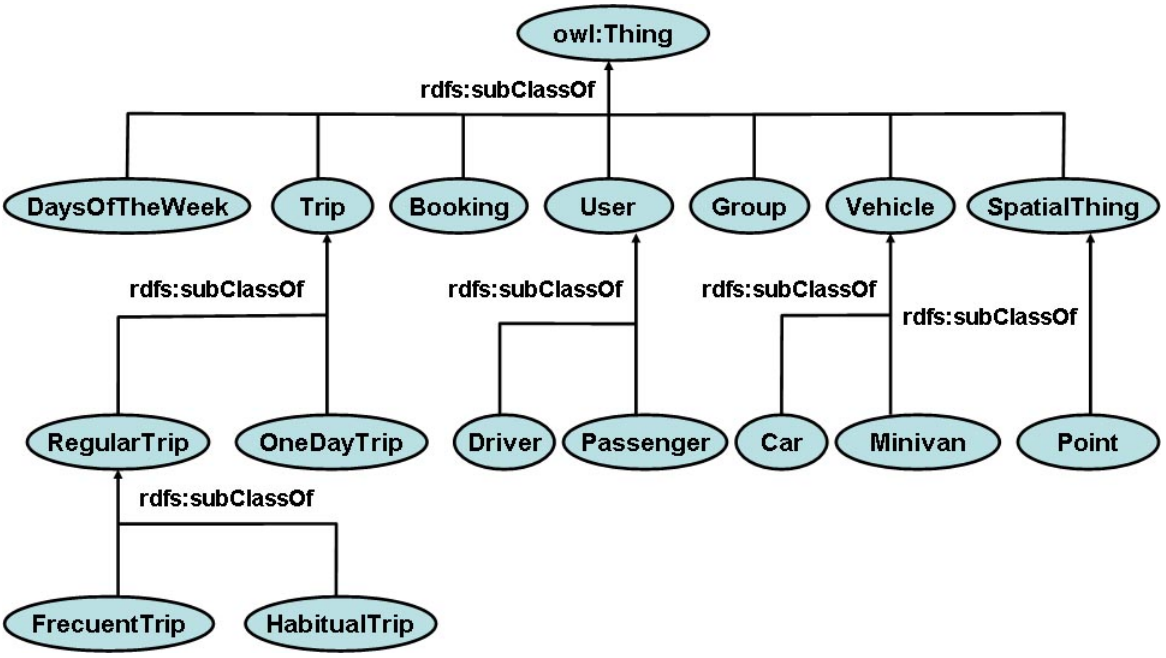

Figura 4.11: Jerarquía de clases de la ontología

se indican las condiciones necesarias y las condiciones necesarias y suficientes que ha de cumplir cada clase. A continuación se indica la definición de cada una de las clases de la ontología propuesta.

Clase `User`

- **Superclase.** `owl:Thing.`
- **Subclases.** `Driver, Passenger.`
- **Clases disjuntas.** `Vehicle, Group, DaysOfTheWeek, Booking, Trip.`
- **Condiciones necesarias y suficientes.**
  - `Driver` $\cup$ `Passenger`
- **Condiciones necesarias.**
  - `hasName = 1`
  - `hasPassword = 1`
  - `hasPhone = 1`
  - `hasRating` $\leq$ `1`
  - `hasSurname = 1`

Propiedad	Característica	Inversa	Dominio	Rango
hasBooking		isBookingOf	Trip	Booking
hasDaysOfTheWeek			FrequentTrip	DaysOfTheWeek
hasDestination			$Trip \cup Booking$	geo:Point
hasGroup		hasMember	User	Group
hasMember		hasGroup	Group	User
hasOrigin			$Trip \cup Booking$	geo:Point
hasReservation		isReservationOf	Passenger	Booking
hasTrip		isTripOf	Driver	Trip
hasVehicle	Funcional	isVehicleOf	Driver	Vehicle
isBookingOf		hasBooking	Booking	Trip
isReservationOf		hasReservation	Booking	Passenger
isTripOf		hasTrip	Trip	Driver
isVehicleOf	Funcional inversa	hasVehicle	Vehicle	Driver

Tabla 4.3: Propiedades de objeto

Clase `Driver`

- **Superclase.** `User`
- **Subclases.** Ninguna
- **Clases disjuntas.** `Passenger`
- **Condiciones necesarias y suficientes.**
  - `User`
  - `∃ hasTrip Trip`
  - `∀ hasVehicle Vehicle`
- **Condiciones necesarias.** Ninguna.

Clase `Passenger`

- **Superclase.** `User`
- **Subclases.** Ninguna.
- **Clases disjuntas.** `Driver`
- **Condiciones necesarias y suficientes.**
  - `User`
  - `∃ hasReservation Booking`
- **Condiciones necesarias.** Ninguna.

Propiedad	**Característica**	**Dominio**	**Rango**
hasAvailableSeats		Trip	xsd:int
hasBrand		Vehicle	xsd:string
hasComfort		Vehicle	xsd:int
hasDepartDate		$OneDayTrip \cup Booking$	xsd:date
hasDepartTime		$Trip \cup Booking$	xsd:time
hasEmail	Funcional	User	xsd:string
hasEndDate		FrequentTrip	xsd:date
hasID	Funcional	$User \cup Vehicle \cup Group \cup Trip \cup Booking$	xsd:int
hasLogin	Funcional	User	xsd:string
hasName		User	xsd:string
hasPassword		User	xsd:string
hasPhone		User	xsd:string
hasPlateNr	Funcional	Vehicle	xsd:string
hasPrice		Trip	xsd:decimal
hasRating		User	xsd:decimal
hasReturnDate		OneDayTrip	xsd:date
hasReturnTime		Trip	xsd:time
hasSeats		Vehicle	xsd:int
hasSurname		User	xsd:string

Tabla 4.4: Propiedades de tipo de datos

Clase `Group`

- **Superclase.** `owl:Thing.`
- **Subclases.** Ninguna.
- **Clases disjuntas.** `User, Trip, DaysOfTheWeek, Booking, Vehicle.`
- **Condiciones necesarias y suficientes.**
  - `hasLocation` $\leq$ `1`
  - `hasMember` $\geq$ `1`
- **Condiciones necesarias.**
  - `hasName = 1`

Clase `Vehicle`

- **Superclase.** `owl:Thing.`

- **Subclases.** `Car, Minivan.`
- **Clases disjuntas.** `Group, User, Booking, DaysOfTheWeek, Trip.`
- **Condiciones necesarias y suficientes.**
  - `Car ∪ Minivan`
  - `isVehicleOf = 1`
- **Condiciones necesarias.**
  - `hasBrand = 1`
  - `hasComfort = 1`
  - `hasSeats = 1`

Clase `Car`

- **Superclase.** `Vehicle.`
- **Subclases.** Ninguna.
- **Clases disjuntas.** `Minivan.`
- **Condiciones necesarias y suficientes.** Ninguna.
- **Condiciones necesarias.** Ninguna.

Clase `Minivan`

- **Superclase.** `Vehicle.`
- **Subclases.** Ninguna.
- **Clases disjuntas.** `Car.`
- **Condiciones necesarias y suficientes.** Ninguna.
- **Condiciones necesarias.** Ninguna.

Clase `Trip`

- **Superclase.** `owl:Thing.`
- **Subclases.** `OneDayTrip, RegularTrip.`
- **Clases disjuntas.** `User, Vehicle, Booking, DaysOfTheWeek, Group.`
- **Condiciones necesarias y suficientes.**
  - `OneDayTrip ∪ RegularTrip`

- **Condiciones necesarias.**
  - $\exists$ `hasBooking Booking`
  - `isTripOf = 1`
  - `hasAvailableSeats = 1`
  - `hasPrice` $\leq$ `1`
  - `hasOrigin = 1`
  - `hasDestination = 1`
  - `hasDepartTime = 1`
  - `hasReturnTime` $\leq$ `1`

Clase `OneDayTrip`

- **Superclase.** `Trip.`
- **Subclases.** Ninguna
- **Clases disjuntas.** `RegularTrip.`
- **Condiciones necesarias y suficientes.**
  - `Trip`
  - `hasDepartDate = 1`
  - `hasReturnDate` $\leq$ `1`
- **Condiciones necesarias.** Ninguna.

Clase `RegularTrip`

- **Superclase.** `Trip.`
- **Subclases.** `FrequentTrip, HabitualTrip.`
- **Clases disjuntas.** `OneDayTrip.`
- **Condiciones necesarias y suficientes.**
  - `Trip`
  - `HabitualTrip` $\cup$ `FrequentTrip`
- **Condiciones necesarias.** Ninguna.

Clase `FrequentTrip`

- **Superclase.** `RegularTrip.`
- **Subclases.** Ninguna.
- **Clases disjuntas.** `HabitualTrip.`
- **Condiciones necesarias y suficientes.** Ninguna.
- **Condiciones necesarias.**
  - `RegularTrip`
  - `hasStartDate = 1`
  - `hasEndDate = 1`
  - `hasDaysOfTheWeek` $\geq$ `1`

Clase `HabitualTrip`

- **Superclase.** `RegularTrip.`
- **Subclases.** Ninguna.
- **Clases disjuntas.** `FrequentTrip.`
- **Condiciones necesarias y suficientes.** Ninguna.
- **Condiciones necesarias.** Ninguna.

Clase `Booking`

- **Superclase.** `owl:Thing.`
- **Subclases.** Ninguna.
- **Clases disjuntas.** `User, Group, Vehicle, Trip, DaysOfTheWeek.`
- **Condiciones necesarias y suficientes.**
  - `isBookingOf = 1`
  - `isReservationOf = 1`
- **Condiciones necesarias.**
  - `hasDepartDate = 1`
  - `hasDepartTime = 1`
  - `hasOrigin = 1`
  - `hasDestination = 1`

Clase `DaysOfTheWeek`

- **Superclase.** `owl:Thing`.
- **Subclases.** Ninguna.
- **Clases disjuntas.** `User, Group, Vehicle, Trip, Booking`.
- **Condiciones necesarias y suficientes.** `{Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday}`.
- **Condiciones necesarias.** Ninguna.

4.3.5. Crear instancias

El paso previo a la evaluación de la ontología consiste en la creación de instancias de las clases de la misma. De esta forma, se podrá comprobar, tal y como se verá en el capítulo 5, si la ontología es capaz de responder a las preguntas de competencia especificadas en la fase de arranque.

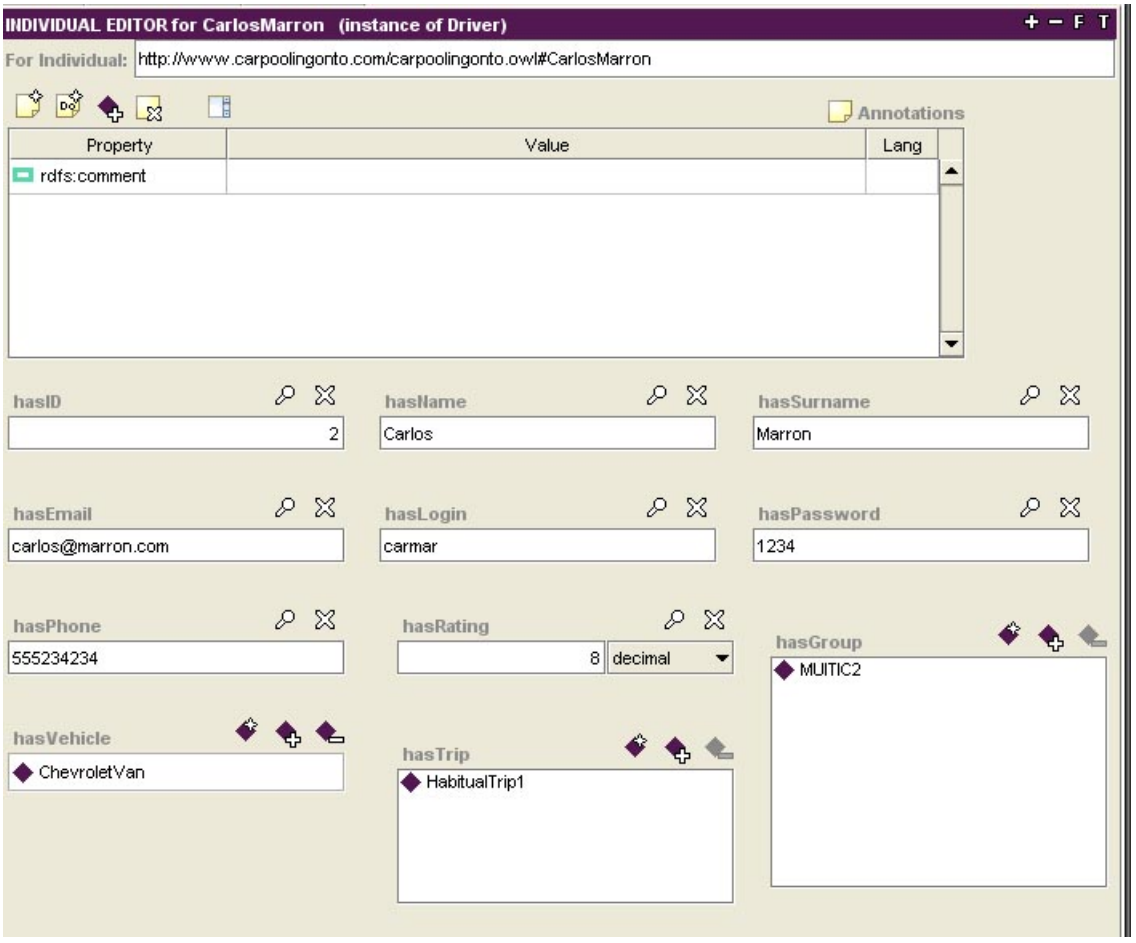

Figura 4.12: Ejemplo de creación de una instancia

En el caso que nos ocupa, las instancias se han introducido manualmente utilizando la opción de instanciación diponible en Protégé, tal y como se puede ver en el ejemplo de la figura 4.12. Se tratan, por tanto, instancias "inventadas", que ayudarán a realizar una primera evaluación de la ontología. Sin embargo, tal y como se indica en el capítulo 6, es deseable que en el futuro se puedan realizar pruebas sobre datos reales.

Como se ve en la figura 4.12, el proceso de instanciación con Protégé es muy sencillo, ya que únicamente hay que rellenar el formulario correspondiente con los valores de cada propiedad en cada caso. En dicha figura se muestra la instanciación de un **Conductor (`Driver`)**, que tiene una serie de propiedades de tipo de datos y otras de objeto. En la figura 4.13 se muestra un esquema conceptual en el que se ven las instancias creadas, las clases a las que correspoden y las propiedades que las relacionan.

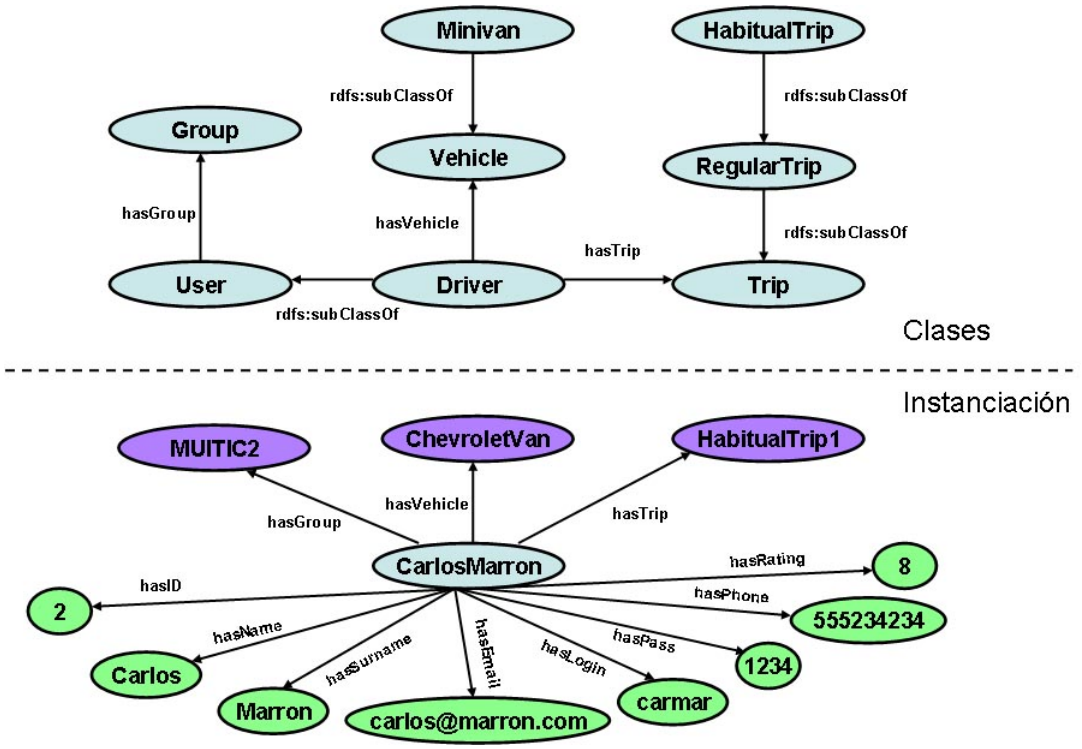

Figura 4.13: Esquema de una instancia de la clase **`Driver`**

Una vez creadas las instancias, se puede pasar a la siguiente fase en el desarrollo de la ontología, que es la evaluación. Los resultados de esta fase se analizan en el siguiente capítulo. Este análisis no sólo sirve para ver si la ontología es "correcta", sino también para obtener una realimentación sobre la misma, que sirve para decidir si es conveniente depurar la fase de refinamiento de la ontología, con lo que se podrían realizar distintas iteraciones en el proceso.

Capítulo 5

Análisis de resultados

La fase de evaluación de la ontología desarrollada sirve para establecer si la ontología cumple con los requisitos establecidos en la fase de arranque. Principalmente esta evaluación se corresponde con el análisis de las respuestas a las preguntas de competencia. Estas repuestas se han obtenido realizando búsquedas mediante el lenguaje SPARQL [24].

Sin embargo, en este trabajo se ha añadido a la evaluación de la ontología la comprobación de la misma mediante razonadores de lógica descriptiva. En el caso que nos ocupa, se ha utilizado el razonador RACER [41] para comprobar la consistencia de la ontología.

El análisis de los resultados de ambas evaluaciones se describe en las siguientes secciones.

5.1. Análisis de consistencia de la ontología

El análisis de la consistencia de la ontología propuestas es posible gracias a que OWL-DL está basado en la lógica descriptiva y puede ser procesada por un razonador. El editor de ontologías Protégé cuenta con la opción de realizar este análisis mediante el razonador DIG *(DL Implementation Group)* [62] o el Pellet [63]. En nuestro caso se ha utilizado el razonador DIG. Por otra parte, el razonador RACER permite la comunicación con Protégé para poder llevar a cabo este análisis (se puede ver una explicación más amplia en [43]).

La consistencia de la ontología creada se comprueba mediante tres tipos de análisis de consistencia:

- **Comprobación de taxonomía.** Esta comprobación permite al razonador computar e inferir la jerarquía de clases de la ontología. Un resultado positivo sería que la jerarquía de clases inferida por el razonador coincida con la propia diseñada.
- **Comprobación de la consistencia.** Basándose en la descripción (condiciones) que se han especificado en las clases de la ontología, el razonador comprueba si existe alguna clase que no pueda tener instancias (en cuyo caso se trata de una clase inconsistente). Un resultado positivo sería que el razonador no encuentra clases que no son consistentes.

- **Comprobación de instancias**. En este caso se comprueba que las instancias de una clase cumplan con las especificaciones de la misma. Un resultado positivo sería que las instancias inferidas coincidan con las diseñadas.

A continuación se presenta un análisis de los resultados obtenidos.

Comprobación de taxonomía

- **Acción realizada.** Se invoca al razonador para que extraiga la taxonomía inferida.
- **Resultado esperado.** La taxonomía diseñada coincide con la inferida.
- **Resultado obtenido.** La taxonomía diseñada y la inferida coinciden (ver figura 5.1).
- **Valoración.** Positiva.

Comprobación de la consistencia

- **Acción realizada.** Se invoca al razonador para que compruebe que todas las clases pueden tener instancias.
- **Resultado esperado.** No existen clases inconsistentes.
- **Resultado obtenido.** No se identifican clases inconsistentes (ver figura 5.1).
- **Valoración.** Positiva.

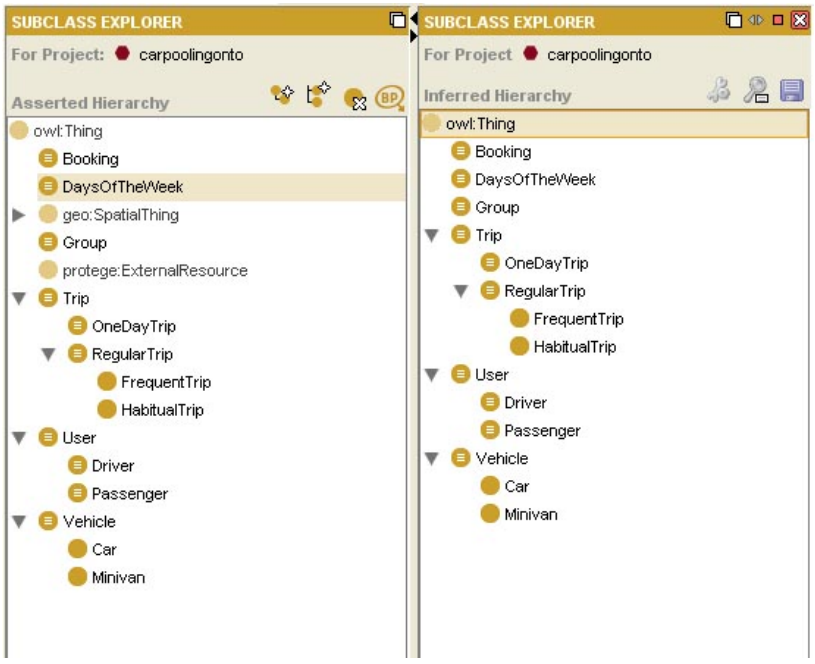

Figura 5.1: Comprobación de taxonomía y consistencia

Comprobación de instancias

- **Acción realizada.** Se invoca al razonador para que compruebe que todas las instancias cumplen con las especificaciones de sus clases correspondientes.
- **Resultado esperado.** Las instancias cumplen con las especificaciones de las clases.
- **Resultado obtenido.** No se identifican instancias que no cumplan con las especificaciones de las clases (ver figura 5.2).
- **Valoración.** Positiva.

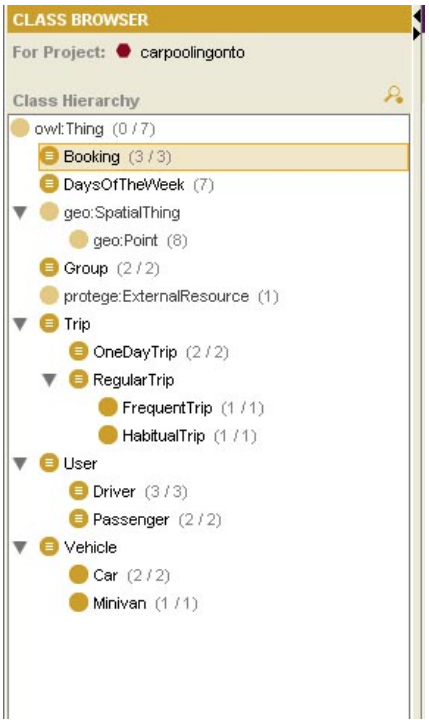

Figura 5.2: Comprobación de instancias

Cabe destacar que la comprobación de la consistencia de la ontología se fue realizando poco a poco durante el desarrollo de la misma. Esto ha servido para detectar inconsistencias y poder subsanarlas rápidamente.

5.2. Análisis de las preguntas de competencia

El último paso en la evaluación de la ontología propuesta consiste en comprobar que esta es capaz de responder a las preguntas de competencia planteadas en la fase de arranque (sección 4.2). Esto se ha realizado mediante búsquedas SPARQL, que se pueden ver en el apéndice B. Recordemos, asimismo, que estas preguntas se refieren a búsquedas de viajes y reservas.

Pregunta 1. Viajes según origen, destino u origen y destino

Esta pregunta de competencia se puede dividir en tres "subpreguntas":

- Búsqueda de viajes según un origen determinado.
- Búsqueda de viajes según un destino determinado.
- Búsqueda de viajes según origen y destino (que coincidan los dos).

Así, estas búsquedas muestran como salida información importante sobre los viajes encontrados, como datos del conductor, origen del viaje, destino del mismo, fechas, y horas. Cabe destacar que otros campos de salida son posibles. A continuación se muestra un fragmento de una de estas búsquedas en el que se ve cómo se aplica un filtrado sobre el origen, identificando su latitud y su longitud, que se toman como parámetros extraídos, por ejemplo, de un buscador cartográfico integrado en la futura aplicación (que por otra parte utilizan la mayoría de los portales de coche compartido).

```
...
?ori geo:lat ?latori.
?ori geo:long ?longori.
FILTER (?latori="41.655038"^^xsd:double
&& ?longori="-4.720752"^^xsd:double)
...
```

Por otra parte, se encontraron resultados erróneos en algunas búsquedas, ya que en ocasiones no se mostraban todos los valores que se debiera. El motivo es que algunos viajes no contaban por ejemplo, con fecha u hora de vuelta (es decir, se trataba de viajes sólo de ida). Esto se subsanó refinando la búsqueda correspondiente. En particular se utilizó **OPTIONAL** en las búsquedas, como se muestra en el siguiente fragmento de búsqueda.

```
...
?trip co:hasDepartTime ?departTime.
OPTIONAL
{
?trip co:hasReturnTime ?returnTime.
}
?trip co:hasOrigin ?ori.
...
```

El resultado de las búsquedas fue el esperado. Por lo tanto, la ontología es capaz de responder a la pregunta de competencia 1. Sin embargo, se considera de utilidad estudiar en el futuro la posibilidad de realizar búsquedas según un rango de distancia a origen o destino, tal y como se menciona en el capítulo 2.

Pregunta 2. Viajes según fecha, hora o fecha y hora

Al igual que la anterior, esta pregunta de competencia consta de tres "subpreguntas":

- Búsqueda de viajes para una fecha.
- Búsqueda de viajes para una hora.
- Búsqueda de viajes para una fecha y una hora.

Los datos de salida de estas búsquedas son similares al de la pregunta anterior. Sin embargo, es destacable el hecho de que para la búsqueda de viajes que sean frecuentes (que se repiten determinados días de la semana durante un periodo de tiempo), se debe realizar un filtrado que tenga en cuenta que la fecha buscada se encuentra entre las fechas de comienzo y fin del viaje y además que el día de la semana coincida. Normalmente en las aplicaciones de calendario esta información está disponible, por lo que no se considera que pueda ser un problema. A continuación se muestra un fragmento de este tipo de búsqueda.

```
...
?trip co:hasDaysOfTheWeek ?day.
?day rdfs:label ?dayweek
FILTER ( ?startDate <= "2012-06-25"^^xsd:date && ?endDate >=
"2012-06-25"^^xsd:date && ?dayweek = "Monday")
...
```

El resultado de las búsquedas fue el esperado. Por lo tanto, la ontología es capaz de responder a la pregunta de competencia 2. Sin embargo se considera de utilidad para futuros trabajos estudiar la posibilidad de realizar búsquedas según una flexibilidad horaria, tal y como se menciona en el capítulo 2.

Pregunta 3. Reservas asociadas a un conductor

Para un conductor puede ser interesante ver todas las reservas que tiene asociadas, para llevar un control sobre ellas. Esta pregunta aborda esta situación. Ya que un conductor tiene viajes registrados y esos viajes tienen reservas asociadas, la búsqueda realizada sigue este camino. Su salida muestra información relativa a los viajes y a las reservas y pasajeros asociados a esas reservas. Aunque, una vez más, esta salida puede ser modificada.

El resultado de las búsquedas fue el esperado. Por lo tanto, la ontología es capaz de responder a la pregunta de competencia 3.

Pregunta 4. Reservas realizadas para un viaje

De la misma forma, puede ser de utilidad para un conductor ver las reservas asociadas a cada viaje de forma individual. Tomando como parámetro de búsqueda el identificador del viaje, es posible consultar esta información.

El resultado de las búsquedas fue el esperado. Por lo tanto, la ontología es capaz de responder a la pregunta de competencia 4.

Pregunta 5. Reservas realizadas por un pasajero

Al igual que un conductor puede querer saber los pasajeros que tiene asociados a través de reservas, el caso inverso también es posible. Un pasajero puede querer consultar los datos relativos a sus reservas (viaje al que corresponden, horas, fechas, datos del conductor, etc)

El resultado de las búsquedas fue el esperado. Por lo tanto, la ontología es capaz de responder a la pregunta de competencia 5.

Pregunta 6. Viajes asociados a un grupo

La búsqueda de viajes restringidos al grupo al que pertenece un usuario es la motivación del planteamiento de esta pregunta. Para ello, se pueden realizar las mismas búsquedas que en las preguntas de competencia 1 y 2, pero restringiéndolas a un grupo determinado, mediante un filtro (tal y como se indica a continuación).

```
...
FILTER regex(?groupName, "MUITIC2")
...
```

El resultado de las búsquedas fue el esperado. Por lo tanto, la ontología es capaz de responder a la pregunta de competencia 6.

Pregunta 7. Reservas asociadas a un grupo

De la misma forma, se puede realizar una búsqueda de reservas asociadas a un grupo. La utilidad de esto (según la experiencia del autor de este trabajo), radica en la posibilidad de realizar estadísticas de uso del coche compartido dentro de un grupo. Así, por ejemplo, se podrían computar los viajes registrados y las reservas sobre esos viajes, dando esto una idea de los viajes reales compartidos.

El resultado de las búsquedas fue el esperado. Por lo tanto, la ontología es capaz de responder a la pregunta de competencia 7.

En este capítulo se ha realizado un breve análisis de los resultados obtenidos en la evaluación de la ontología. Así, se ha visto que la ontología desarrollada es consistente y es capaz de responder a las preguntas de competencia planteadas. Aunque otras preguntas son posibles, como se indica en el siguiente capítulo, este análisis nos da una idea de que la ontología propuesta puede resultar de utilidad para su aplicación en sistemas de coche compartido.

Capítulo 6

Consideraciones finales

Con el presente Trabajo Fin de Master se ha propuesto una ontología para una aplicación en el dominio de la movilidad sostenible como es el coche compartido. Este se considera el primer paso para poder desarrollar una aplicación de coche compartido que utilice tecnologías de Web Semántica, con el fin de promover la interoperabilidad entre distintas aplicaciones de coche compartido que utilicen esta ontología.

Este capítulo indica las conclusiones principales que se han obtenido en este trabajo. De la misma forma, se trata de discutir estas conclusiones, así como se indican una serie de trabajos futuros que pueden llevarse a cabo como continuación de este trabajo.

6.1. Conclusiones

Este Trabajo Fin de Master propone el uso de la Web Semántica como posible solución a los problemas de falta de masa crítica de las aplicaciones y sistemas de coche compartido. De esta forma, se trata de facilitar la interoperabilidad entre aplicaciones de coche compartido, que permitiría a usuarios de distintos portales encontrar viajes registrados por usuarios de otros portales. Esto permitiría aumentar las probabilidades de encontrar compañeros de viaje, y por tanto, la masa crítica de este tipo de aplicaciones.

Para lograr esta interoperabilidad, los distintos portales tendrían que convertir sus datos almacenados en bases de datos a fuentes de datos en formado RDF, siguiendo los principios de *Linked Data*. Sin embargo, como primer paso para lograr este objetivo, se ve necesario el diseño, desarrollo e implementación de una ontología para el coche compartido, de tal forma que los distintos proveedores de datos (portales de coche compartido), puedan tomar como base esta ontología para convertir sus datos. La ventaja del uso de una ontología común es que facilitaría la interoperabilidad entre los portales, ya que todos "hablarían un lenguaje común". En contraposición, si los portales convirtieran sus fuentes de datos de forma directa (como se indica por ejemplo en [64] o [65]), se podría ganar en tiempo de conversión, pero se perdería en interoperabilidad, ya que los portales tendrían que conocer cuál es la estructura de los datos de cada portal.

En este trabajo, pues, se propone una ontología de coche compartido para favorecer esta interoperabilidad. Para llegar a una implementación de esta ontología se ha realizado un estudio de los sistemas de coche compartido, identificando los actores que intervienen, así como las principales funcionalidades que realizan estos actores. Así, esta ontología propone el escenario más común en este tipo de aplicaciones, en el que un usuario que actúa como conductor publica un viaje que quiere compartir, y un usuario que actúa como pasajero busca un viaje publicado por un conductor, sobre el que realiza una reserva de viaje. Si bien es cierto que otro escenario es posible, donde los usuarios puedan actuar como pasajeros y conductores indistintamente, este se considera un trabajo futuro que es interesante abordar, ya que responde también a la realidad de las aplicaciones de coche compartido.

La ontología propuesta se ha desarrollado utilizando el editor de ontologías Protégé [35], ya que es ampliamente utilizado en distintos proyectos (como se puede comprobar en [66] o [35], por ejemplo). Además, ha sido presentado en la asignatura [39] cursada en este Master, por lo que se aplican los conocimientos adquiridos en la misma. Además, la ontología se ha desarrollado en lenguaje OWL [12], en particular en OWL-DL, que permite dotar a la ontología de gran expresividad y además ser tratada por sistemas razonadores (que usan la lógica descriptiva) para computar la consistencia de la misma. Asimismo, se ha seguido una metodología de desarrollo basada en metodologías de ingeniería de ontologías tales como *On-To-Knowledge* [54] o la propuesta por la Universidad de Stanford en [43]. En particular:

- Se ha realizado un estudio de viabilidad del trabajo desarrollado, donde se ha identificado el problema a resolver dentro de las aplicaciones de coche compartido: la interoperabilidad entre portales.
- Se ha hecho un análisis previo de la ontología a desarrollar, definiendo aspectos como su dominio, alcance o preguntas que ha de ser capaz de resolver la ontología (denominadas *preguntas de competencia*). Estas preguntas están relacionadas principalmente con la búsqueda de viajes publicados por los conductores en un portal de coche compartido.
- Se ha realizado una conceptualización de la ontología, identificando los conceptos y clases principales, así como las relaciones entre estos conceptos, sus características y sus atributos.
- Se ha formalizado dicha conceptualización, definiendo una jerarquía de clases (taxonomía) para la ontología.
- Se han identificado las propiedades de objeto que relacionan los individuos o instancias de dichas clases. Asimismo, se han identificado las propiedades de tipo de datos de las mismas.
- Se han establecido axiomas sobre estas propiedades, como dominio, rango u otro tipo de restricciones (de valor, cardinalidad, etc).
- Se ha evaluado la "validez" de la ontología mediante:

- Uso de un sistema razonador como es RACER [41], haciendo uso del razonador DIG.
- Búsquedas SPARQL que muestren que la ontología desarrollada es capaz de responder a las preguntas de competencia, así como a otras adicionales que se puedan plantear.

Todo esto muestra la utilidad de la principal contribución de este Trabajo Fin de Master, que es la propuesta de una ontología para aplicaciones de movilidad sostenible, como es el coche compartido. Adicionalmente, se puede identificar otro tipo de contribución en este trabajo, que es el planteamiento de la utilidad del uso de tecnologías de la Web Semántica en aplicaciones de coche compartido. Esto se ha de ver complementado con el desarrollo de una aplicación de coche compartido que haga uso de la ontología desarrollada y de otras tecnologías de Web Semántica. De esta forma, se podrá plantear la validez de esta solución y quizás (idealmente) formar una "comunidad" entre los portales de coche compartido de forma que se logre la interoperabilidad entre ellos.

Aun así, caben destacar algunos aspectos que han de tenerse en cuenta relacionados con este trabajo, que se discuten en la siguiente sección.

6.2. Discusión y trabajos futuros

Como se ha indicado en la sección anterior, la contribución principal de este Trabajo Fin de Master es el desarrollo de una ontología para aplicaciones de coche compartido. Como se indica en el capítulo 3, el proceso de desarrollo de una ontología es un proceso iterativo, en el que la ontología desarrollada va evolucionando hasta llegar a un estado estable. Se considera, pues, que el proceso de desarrollo de la ontología presentada en este trabajo puede llegar a pasar por más iteraciones, fruto de una evolución más profunda. Así, la ontología propuesta describe un escenario donde es el conductor el que publica los viajes y el pasajero el que los busca y reserva. Esto hace que el rol de conductor y pasajero esté claramente diferenciado. Aunque este es el caso más común, en el capítulo 2 se comenta que puede darse el caso en que un usuario pueda actuar en ocasiones como conductor y en otras como pasajero. Es más, puede darse el caso de que sea un pasajero el que publique un viaje y un conductor el que realice búsquedas para encontrar pasajeros con quien compartir las plazas de su coche. Este escenario podría ser planteado en futuras iteraciones de la ontología planteada, añadiendo algunos cambios en la misma.

Por otra parte, la evaluación de la ontología se ha realizado teniendo en cuenta datos e instancias "inventadas", que han servido para poder comprobar si la ontología desarrollada es capaz de responder a las preguntas de competencia iniciales. Sin embargo, se considera de gran utilidad el plantear una evaluación más profunda de la ontología (o su futura evolución) planteada, en la que se haga uso de datos reales. Incluso se considera un paso importante contar con otras personas que puedan realizar la evaluación de la ontología planteada.

Para esto, se considera de gran utilidad contar con:

- El punto de vista de otros miembros de la comunidad del coche compartido; es decir, otros portales que puedan hacer uso de la ontología desarrollada. Esto serviría para refinar aun más el diseño y desarrollo de la ontología.
- El desarrollo de una aplicación de coche compartido que haga uso de esta ontología y de tecnologías de Web Semántica. Esta aplicación tendría que interoperar con distintas fuentes de datos, por lo que contar con esas fuentes (o un extracto de ellas) sería de gran utilidad. Incluso se considera de utilidad realizar una prueba que muestre cómo mapear alguna fuente de datos existente a formato RDF teniendo en cuenta la ontología desarrollada.
- Usuarios que utilicen la aplicación desarrollada con el fin de evaluar su utilidad.

Por otra parte, aunque en esta versión no se ha utilizado, se puede plantear la posibilidad de realizar un tratamiento de los datos referentes al tiempo (fecha, hora, etc) algo más complejo, utilizando para ello alguna de las ontologías mencionadas en el capítulo 4, como **OWL-Time [58]** o **Temporal [60]**. De esta forma, podrían llegar a plantearse búsquedas con flexibilidad de horario, como se ha indicado en el capítulo 5.

Por último, ya que en las aplicaciones de este tipo se manejan datos personales, el problema de la privacidad de estos datos es algo que debe ser considerado. Referencias bibliográficas como [67] plantea una posible solución a esto basada en una ontología llamada *Privacy Preference Ontology (PPO)*. Aun así, se considera que se debe realizar un estudio más profundo sobre este asunto.

Apéndice A

Código de la ontología

El código de la ontología desarrollada se encuentra adjunto en el CD-ROM que se incluye en este Trabajo Fin de Master. De la misma forma, también se incluye el proyecto del editor de ontologías Protégé.

Apéndice B

Búsquedas SPARQL

A continuación se muestran las pruebas realizadas para comprobar que la ontología propuesta es capaz de responder a las preguntas de competencia planteadas en la fase de arranque del desarrollo de la ontología. Estas pruebas se corresponden con el análisis de resultados descrito en el capítulo 5.

B.1. Búsquedas para la Pregunta 1

La pregunta de competencia 1 trata de encontrar viajes según su origen, destino, u origen y destino. Para ello, se realizan búsquedas teniendo en cuenta la latitud y longitud de los puntos buscados.

Viajes para un origen determinado. Por ejemplo viajes cuyo origen sea *Calle Paraíso, Valladolid*, que tiene *latitud = 41.655038 y longitud = -4.720752*.

```
PREFIX co: <http://www.carpoolingonto.com/carpoolingonto.owl#>
SELECT ?name ?surname ?email ?origin ?destination ?startDate
?endDate ?departDate ?departTime ?returnDate ?returnTime
FROM <http://www.carpoolingonto.com/carpoolingonto.owl>
WHERE {
{
?trip rdf:type co:FrequentTrip.
?trip co:isTripOf ?cond.
?cond co:hasName ?name.
?cond co:hasSurname ?surname.
?cond co:hasEmail ?email.
?trip co:hasStartDate ?startDate.
?trip co:hasEndDate ?endDate.
?trip co:hasDepartTime ?departTime.
OPTIONAL
{
?trip co:hasReturnTime ?returnTime.
}
```

```
?trip co:hasOrigin ?ori.
?ori rdfs:label ?origin.
?trip co:hasDestination ?dest.
?dest rdfs:label ?destination.
?ori geo:lat ?latori.
?ori geo:long ?longori.
FILTER (?latori="41.655038"^^xsd:double && ?longori="-4.720752"^^xsd:double)
}
UNION
{
?trip rdf:type co:OneDayTrip.
?trip co:isTripOf ?cond.
?cond co:hasName ?name.
?cond co:hasSurname ?surname.
?cond co:hasEmail ?email.
?trip co:hasDepartDate ?departDate.
?trip co:hasDepartTime ?departTime.
OPTIONAL
{
?trip co:hasReturnDate ?returnDate.
?trip co:hasReturnTime ?returnTime.
}
?trip co:hasOrigin ?ori.
?ori rdfs:label ?origin.
?trip co:hasDestination ?dest.
?dest rdfs:label ?destination.
?ori geo:lat ?latori.
?ori geo:long ?longori.
FILTER (?latori="41.655038"^^xsd:double && ?longori="-4.720752"^^xsd:double)
}
UNION
{
?trip rdf:type co:HabitualTrip.
?trip co:isTripOf ?cond.
?cond co:hasName ?name.
?cond co:hasSurname ?surname.
?cond co:hasEmail ?email.
?trip co:hasDepartTime ?departTime.
OPTIONAL
{
?trip co:hasReturnTime ?returnTime.
}
?trip co:hasOrigin ?ori.
?ori rdfs:label ?origin.
?trip co:hasDestination ?dest.
```

```
?dest rdfs:label ?destination.
?ori geo:lat ?latori.
?ori geo:long ?longori.
FILTER (?latori="41.655038"^^xsd:double && ?longori="-4.720752"^^xsd:double)
}
}
ORDER BY(?departTime)
```

Viajes para un destino determinado. Por ejemplo viajes cuyo destino sea *La Cistérniga, Valladolid*, que tiene *latitud = 41.618266 y longitud = -4.689102*.

```
PREFIX co: <http://www.carpoolingonto.com/carpoolingonto.owl#>
SELECT ?name ?surname ?email ?origin ?destination ?startDate
?endDate ?departDate ?returnDate ?departTime ?returnTime
FROM <http://www.carpoolingonto.com/carpoolingonto.owl>
WHERE {
{
?trip rdf:type co:FrequentTrip.
?trip co:isTripOf ?cond.
?cond co:hasName ?name.
?cond co:hasSurname ?surname.
?cond co:hasEmail ?email.
?trip co:hasStartDate ?startDate.
?trip co:hasEndDate ?endDate.
?trip co:hasDepartTime ?departTime.
OPTIONAL
{
?trip co:hasReturnTime ?returnTime.
}
?trip co:hasOrigin ?ori.
?ori rdfs:label ?origin.
?trip co:hasDestination ?dest.
?dest rdfs:label ?destination.
?dest geo:lat ?latdest.
?dest geo:long ?longdest.
FILTER (?latdest="41.618266"^^xsd:double && ?longdest="-4.689102"^^xsd:double)
}
UNION
{
?trip rdf:type co:OneDayTrip.
?trip co:isTripOf ?cond.
?cond co:hasName ?name.
?cond co:hasSurname ?surname.
?cond co:hasEmail ?email.
```

```
?trip co:hasDepartDate ?departDate.
?trip co:hasDepartTime ?departTime.
OPTIONAL
{
?trip co:hasReturnDate ?returnDate.
?trip co:hasReturnTime ?returnTime.
}
?trip co:hasOrigin ?ori.
?ori rdfs:label ?origin.
?trip co:hasDestination ?dest.
?dest rdfs:label ?destination.
?dest geo:lat ?latdest.
?dest geo:long ?longdest.
FILTER (?latdest="41.618266"^^xsd:double && ?longdest="-4.689102"^^xsd:double)
}
UNION
{
?trip rdf:type co:HabitualTrip.
?trip co:isTripOf ?cond.
?cond co:hasName ?name.
?cond co:hasSurname ?surname.
?cond co:hasEmail ?email.
?trip co:hasDepartTime ?departTime.
OPTIONAL
{
?trip co:hasReturnTime ?returnTime.
}
?trip co:hasOrigin ?ori.
?ori rdfs:label ?origin.
?trip co:hasDestination ?dest.
?dest rdfs:label ?destination.
?dest geo:lat ?latdest.
?dest geo:long ?longdest.
FILTER (?latdest="41.618266"^^xsd:double && ?longdest="-4.689102"^^xsd:double)
}
}
ORDER BY(?departTime)
```

Viajes según origen y destino. Por ejemplo viajes cuyo origen sea *Calle Santiago, Valladolid*, que tiene *latitud = 41.650437 y longitud = -4.728885*; y cuyo destino sea *Calle de Miriam Blasco, Valladolid*, que tiene *latitud = 41.639774 y longitud = -4.768152*.

```
PREFIX co: <http://www.carpoolingonto.com/carpoolingonto.owl#>
SELECT ?name ?surname ?email ?origin ?destination ?startDate
```

```
?endDate ?departDate ?returnDate ?departTime ?returnTime
FROM <http://www.carpoolingonto.com/carpoolingonto.owl>
WHERE {
{
?trip rdf:type co:FrequentTrip.
?trip co:isTripOf ?cond.
?cond co:hasName ?name.
?cond co:hasSurname ?surname.
?cond co:hasEmail ?email.
?trip co:hasStartDate ?startDate.
?trip co:hasEndDate ?endDate.
?trip co:hasDepartTime ?departTime.
OPTIONAL
{
?trip co:hasReturnTime ?returnTime.
}
?trip co:hasOrigin ?ori.
?ori rdfs:label ?origin.
?trip co:hasDestination ?dest.
?dest rdfs:label ?destination.
?dest geo:lat ?latdest.
?dest geo:long ?longdest.
?ori geo:lat ?latori.
?ori geo:long ?longori.
FILTER (?latdest="41.639774"^^xsd:double && ?longdest="-4.768152"^^xsd:double &&
?latori="41.650437"^^xsd:double && ?longori="-4.728885"^^xsd:double)
}
UNION
{
?trip rdf:type co:OneDayTrip.
?trip co:isTripOf ?cond.
?cond co:hasName ?name.
?cond co:hasSurname ?surname.
?cond co:hasEmail ?email.
?trip co:hasDepartDate ?departDate.
?trip co:hasDepartTime ?departTime.
OPTIONAL
{
?trip co:hasReturnDate ?returnDate.
?trip co:hasReturnTime ?returnTime.
}
?trip co:hasOrigin ?ori.
?ori rdfs:label ?origin.
?trip co:hasDestination ?dest.
?dest rdfs:label ?destination.
```

```
?dest geo:lat ?latdest.
?dest geo:long ?longdest.
?ori geo:lat ?latori.
?ori geo:long ?longori.
FILTER (?latdest="41.639774"^^xsd:double && ?longdest="-4.768152"^^xsd:double
&& ?latori="41.650437"^^xsd:double && ?longori="-4.728885"^^xsd:double)
}
UNION
{
?trip rdf:type co:HabitualTrip.
?trip co:isTripOf ?cond.
?cond co:hasName ?name.
?cond co:hasSurname ?surname.
?cond co:hasEmail ?email.
?trip co:hasDepartTime ?departTime.
OPTIONAL
{
?trip co:hasReturnTime ?returnTime.
}
?trip co:hasOrigin ?ori.
?ori rdfs:label ?origin.
?trip co:hasDestination ?dest.
?dest rdfs:label ?destination.
?dest geo:lat ?latdest.
?dest geo:long ?longdest.
?ori geo:lat ?latori.
?ori geo:long ?longori.
FILTER (?latdest="41.639774"^^xsd:double && ?longdest="-4.768152"^^xsd:double
&& ?latori="41.650437"^^xsd:double && ?longori="-4.728885"^^xsd:double)
}
}
ORDER BY(?departTime)
```

B.2. Búsquedas para la Pregunta 2

La pregunta de competencia 2 trata de encontrar viajes para una fecha determinada, una hora, o una fecha y una hora.

Viajes para una fecha determinada. Por ejemplo se buscan viajes para el *Lunes 25 de Junio de 2012*.

```
PREFIX co: <http://www.carpoolingonto.com/carpoolingonto.owl#>
SELECT ?name ?surname ?email ?origin ?destination ?startDate
?endDate ?departDate ?returnDate ?departTime ?returnTime
```

```
FROM <http://www.carpoolingonto.com/carpoolingonto.owl>
WHERE {
{
?trip rdf:type co:FrequentTrip.
?trip co:isTripOf ?cond.
?cond co:hasName ?name.
?cond co:hasSurname ?surname.
?cond co:hasEmail ?email.
?trip co:hasStartDate ?startDate.
?trip co:hasEndDate ?endDate.
?trip co:hasDepartTime ?departTime.
OPTIONAL
{
?trip co:hasReturnTime ?returnTime.
}
?trip co:hasOrigin ?ori.
?ori rdfs:label ?origin.
?trip co:hasDestination ?dest.
?dest rdfs:label ?destination.
?trip co:hasDaysOfTheWeek ?day.
?day rdfs:label ?dayweek
FILTER ( ?startDate <= "2012-06-25"^^xsd:date && ?endDate >=
"2012-06-25"^^xsd:date && ?dayweek = "Monday")
}
UNION
{
?trip rdf:type co:OneDayTrip.
?trip co:isTripOf ?cond.
?cond co:hasName ?name.
?cond co:hasSurname ?surname.
?cond co:hasEmail ?email.
?trip co:hasDepartDate ?departDate.
?trip co:hasDepartTime ?departTime.
OPTIONAL
{
?trip co:hasReturnDate ?returnDate.
?trip co:hasReturnTime ?returnTime.
}
?trip co:hasOrigin ?ori.
?ori rdfs:label ?origin.
?trip co:hasDestination ?dest.
?dest rdfs:label ?destination.
FILTER ( ?departDate = "2012-06-25"^^xsd:date)
}
UNION
```

```
{
?trip rdf:type co:HabitualTrip.
?trip co:isTripOf ?cond.
?cond co:hasName ?name.
?cond co:hasSurname ?surname.
?cond co:hasEmail ?email.
?trip co:hasDepartTime ?departTime.
OPTIONAL
{
?trip co:hasReturnTime ?returnTime.
}
?trip co:hasOrigin ?ori.
?ori rdfs:label ?origin.
?trip co:hasDestination ?dest.
?dest rdfs:label ?destination.
}
}
ORDER BY(?departTime)
```

Viajes para una hora de salida. Por ejemplo para las *8:00 am*.

```
PREFIX co: <http://www.carpoolingonto.com/carpoolingonto.owl#>
SELECT ?name ?surname ?email ?origin ?destination ?startDate
?endDate ?departDate ?returnDate ?departTime ?returnTime
FROM <http://www.carpoolingonto.com/carpoolingonto.owl>
WHERE {
{
?trip rdf:type co:FrequentTrip.
?trip co:isTripOf ?cond.
?cond co:hasName ?name.
?cond co:hasSurname ?surname.
?cond co:hasEmail ?email.
?trip co:hasStartDate ?startDate.
?trip co:hasEndDate ?endDate.
?trip co:hasDepartTime ?departTime.
OPTIONAL
{
?trip co:hasReturnTime ?returnTime.
}
?trip co:hasOrigin ?ori.
?ori rdfs:label ?origin.
?trip co:hasDestination ?dest.
?dest rdfs:label ?destination.
```

```
FILTER (?departTime = "08:00:00"^^xsd:time)
}
UNION
{
?trip rdf:type co:OneDayTrip.
?trip co:isTripOf ?cond.
?cond co:hasName ?name.
?cond co:hasSurname ?surname.
?cond co:hasEmail ?email.
?trip co:hasDepartDate ?departDate.
?trip co:hasDepartTime ?departTime.
OPTIONAL
{
?trip co:hasReturnDate ?returnDate.
?trip co:hasReturnTime ?returnTime.
}
?trip co:hasOrigin ?ori.
?ori rdfs:label ?origin.
?trip co:hasDestination ?dest.
?dest rdfs:label ?destination.
FILTER (?departTime = "08:00:00"^^xsd:time)
}
UNION
{
?trip rdf:type co:HabitualTrip.
?trip co:isTripOf ?cond.
?cond co:hasName ?name.
?cond co:hasSurname ?surname.
?cond co:hasEmail ?email.
?trip co:hasDepartTime ?departTime.
OPTIONAL
{
?trip co:hasReturnTime ?returnTime.
}
?trip co:hasOrigin ?ori.
?ori rdfs:label ?origin.
?trip co:hasDestination ?dest.
?dest rdfs:label ?destination.
FILTER(?departTime = "08:00:00"^^xsd:time)
}
}
ORDER BY(?departTime)
```

Viajes según fecha y hora. Por ejemplo, para el *Lunes 25 de Junio de 2012 a las 8:00 am*.

```
PREFIX co: <http://www.carpoolingonto.com/carpoolingonto.owl#>
SELECT ?name ?surname ?email ?origin ?destination ?startDate
?endDate ?departDate ?returnDate ?departTime ?returnTime
FROM <http://www.carpoolingonto.com/carpoolingonto.owl>
WHERE {
{
?trip rdf:type co:FrequentTrip.
?trip co:isTripOf ?cond.
?cond co:hasName ?name.
?cond co:hasSurname ?surname.
?cond co:hasEmail ?email.
?trip co:hasStartDate ?startDate.
?trip co:hasEndDate ?endDate.
?trip co:hasDepartTime ?departTime.
OPTIONAL
{
?trip co:hasReturnTime ?returnTime.
}
?trip co:hasOrigin ?ori.
?ori rdfs:label ?origin.
?trip co:hasDestination ?dest.
?dest rdfs:label ?destination.
?trip co:hasDaysOfTheWeek ?day.
?day rdfs:label ?dayweek
FILTER ( ?startDate <= "2012-06-25"^^xsd:date && ?endDate >=
"2012-06-25t't'^^xsd:date && ?departTime = "08:00:00"^^xsd:time && ?dayweek = "Monday")
}
UNION
{
?trip rdf:type co:OneDayTrip.
?trip co:isTripOf ?cond.
?cond co:hasName ?name.
?cond co:hasSurname ?surname.
?cond co:hasEmail ?email.
?trip co:hasDepartDate ?departDate.
?trip co:hasDepartTime ?departTime.
OPTIONAL
{
?trip co:hasReturnDate ?returnDate.
?trip co:hasReturnTime ?returnTime.
}
?trip co:hasOrigin ?ori.
?ori rdfs:label ?origin.
?trip co:hasDestination ?dest.
```

```
?dest rdfs:label ?destination.
FILTER ( ?departDate = "2012-06-25"^^xsd:date && ?departTime = "08:00:00"^^xsd:time)
}
UNION
{
?trip rdf:type co:HabitualTrip.
?trip co:isTripOf ?cond.
?cond co:hasName ?name.
?cond co:hasSurname ?surname.
?cond co:hasEmail ?email.
?trip co:hasDepartTime ?departTime.
OPTIONAL
{
?trip co:hasReturnTime ?returnTime.
}
?trip co:hasOrigin ?ori.
?ori rdfs:label ?origin.
?trip co:hasDestination ?dest.
?dest rdfs:label ?destination.
FILTER(?departTime = "08:00:00"^^xsd:time)
}
}
ORDER BY(?departTime)
```

B.3. Búsquedas para la Pregunta 3

La pregunta de competencia 3 trata de ver la información relativa a las reservas asociadas a un determinado conductor. En este caso, se utiliza el *login* del conductor como parámetro de búsqueda (en el ejemplo *psauper*).

```
PREFIX co: <http://www.carpoolingonto.com/carpoolingonto.owl#>
SELECT ?log ?date ?or ?de ?name ?surname ?email ?time ?dori ?ddest
FROM <http://www.carpoolingonto.com/carpoolingonto.owl>
WHERE

?user co:hasLogin ?log.
?user co:hasTrip ?viaje.
?viaje co:hasOrigin ?origen.
?origen rdfs:label ?or.
?viaje co:hasDestination ?destino.
?destino rdfs:label ?de.
?viaje co:hasBooking ?booking.
?booking co:hasOrigin ?dorigin.
?dorigin rdfs:label ?dori.
```

```
?booking co:hasDestination ?ddestination.
?ddestination rdfs:label ?ddest.
?booking co:hasDepartDate ?date.
?booking co:hasDepartTime ?time.
?booking co:isReservationOf ?passenger.
?passenger co:hasName ?name.
?passenger co:hasSurname ?surname.
?passenger co:hasEmail ?email.
FILTER regex(?log,"psauper")

ORDER BY ?date ASC(?time)
```

B.4. Búsquedas para la Pregunta 4

La pregunta de competencia 4 trata de encontrar las reservas asociadas a un determinado viaje (identificado mediante su identificador). En este caso se buscan las reservas asociadas al viaje con identificador 1.

```
PREFIX co: <http://www.carpoolingonto.com/carpoolingonto.owl#>
SELECT ?date ?tori ?tdest ?name ?surname ?email ?time ?dori ?ddest
FROM <http://www.carpoolingonto.com/carpoolingonto.owl>
WHERE {
{
?trip rdf:type co:FrequentTrip.
?trip co:hasID ?id.
?trip co:hasOrigin ?torigin.
?torigin rdfs:label ?tori.
?trip co:hasDestination ?tdestination.
?tdestination rdfs:label ?tdest.
?trip co:hasBooking ?booking.
?booking co:hasOrigin ?dorigin.
?dorigin rdfs:label ?dori.
?booking co:hasDestination ?ddestination.
?ddestination rdfs:label ?ddest.
?booking co:hasDepartDate ?date.
?booking co:hasDepartTime ?time.
?booking co:isReservationOf ?passenger.
?passenger co:hasName ?name.
?passenger co:hasSurname ?surname.
?passenger co:hasEmail ?email.
FILTER (?id = "1"^^xsd:int)
}
UNION
{
```

```
?trip rdf:type co:OneDayTrip.
?trip co:hasID ?id.
?trip co:hasOrigin ?torigin.
?torigin rdfs:label ?tori.
?trip co:hasDestination ?tdestination.
?tdestination rdfs:label ?tdest.
?trip co:hasBooking ?booking.
?booking co:hasOrigin ?dorigin.
?dorigin rdfs:label ?dori.
?booking co:hasDestination ?ddestination.
?ddestination rdfs:label ?ddest.
?booking co:hasDepartDate ?date.
?booking co:hasDepartTime ?time.
?booking co:isReservationOf ?passenger.
?passenger co:hasName ?name.
?passenger co:hasSurname ?surname.
?passenger co:hasEmail ?email.
FILTER (?id = "1"^^xsd:int)
}
UNION
{
?trip rdf:type co:HabitualTrip.
?trip co:hasID ?id.
?trip co:hasOrigin ?torigin.
?torigin rdfs:label ?tori.
?trip co:hasDestination ?tdestination.
?tdestination rdfs:label ?tdest.
?trip co:hasBooking ?booking.
?booking co:hasOrigin ?dorigin.
?dorigin rdfs:label ?dori.
?booking co:hasDestination ?ddestination.
?ddestination rdfs:label ?ddest.
?booking co:hasDepartDate ?date.
?booking co:hasDepartTime ?time.
?booking co:isReservationOf ?passenger.
?passenger co:hasName ?name.
?passenger co:hasSurname ?surname.
?passenger co:hasEmail ?email.
FILTER (?id = "1"^^xsd:int)
}
}
ORDER BY ?date ASC(?time)
```

B.5. Búsquedas para la Pregunta 5

La pregunta de competencia 5 trata de encontrar la información de las reservas asociadas a un pasajero. En este caso se buscan las reservas del pasajero con *login* .

```
PREFIX co: <http://www.carpoolingonto.com/carpoolingonto.owl#>
SELECT ?log ?date ?time ?dori ?ddest ?name ?surname ?email
FROM <http://www.carpoolingonto.com/carpoolingonto.owl>
WHERE
{
?user co:hasLogin ?log.
?user co:hasReservation ?booking.
?booking co:hasOrigin ?dorigin.
?dorigin rdfs:label ?dori.
?booking co:hasDestination ?ddestination.
?ddestination rdfs:label ?ddest.
?booking co:hasDepartDate ?date.
?booking co:hasDepartTime ?time.
?booking co:isBookingOf ?trip.
?trip co:isTripOf ?driver.
?driver co:hasName ?name.
?driver co:hasSurname ?surname.
?driver co:hasEmail ?email.
FILTER regex(?log,"mape")
}
ORDER BY ?date ASC(?time)
```

B.6. Búsquedas para la Pregunta 6

La pregunta de competencia 6 trata de encontrar los viajes asociados a un determinado grupo de usuarios. En este caso se realiza una búsqueda de los viajes asociados al grupo *MUITIC2*, sin restricción alguna de fecha, hora, origen o destino.

```
PREFIX co: <http://www.carpoolingonto.com/carpoolingonto.owl#>
SELECT ?name ?surname ?email ?origin ?destination ?startDate
?endDate ?departDate ?returnDate ?departTime ?returnTime
FROM <http://www.carpoolingonto.com/carpoolingonto.owl>
WHERE {
{
?trip rdf:type co:FrequentTrip.
?trip co:isTripOf ?cond.
?cond co:hasName ?name.
?cond co:hasSurname ?surname.
?cond co:hasEmail ?email.
```

```
?cond co:hasGroup ?group.
?group co:hasName ?groupName.
?trip co:hasStartDate ?startDate.
?trip co:hasEndDate ?endDate.
?trip co:hasDepartTime ?departTime.
OPTIONAL
{
?trip co:hasReturnTime ?returnTime.
}
?trip co:hasOrigin ?ori.
?ori rdfs:label ?origin.
?trip co:hasDestination ?dest.
?dest rdfs:label ?destination.
FILTER regex(?groupName, "MUITIC2")
}
UNION
{
?trip rdf:type co:OneDayTrip.
?trip co:isTripOf ?cond.
?cond co:hasName ?name.
?cond co:hasSurname ?surname.
?cond co:hasEmail ?email.
?cond co:hasGroup ?group.
?group co:hasName ?groupName.
?trip co:hasDepartDate ?departDate.
?trip co:hasDepartTime ?departTime.
OPTIONAL
{
?trip co:hasReturnDate ?returnDate.
?trip co:hasReturnTime ?returnTime.
}
?trip co:hasOrigin ?ori.
?ori rdfs:label ?origin.
?trip co:hasDestination ?dest.
?dest rdfs:label ?destination.
FILTER regex(?groupName, "MUITIC2")
}
UNION
{
?trip rdf:type co:HabitualTrip.
?trip co:isTripOf ?cond.
?cond co:hasName ?name.
?cond co:hasSurname ?surname.
?cond co:hasEmail ?email.
?cond co:hasGroup ?group.
```

```
?group co:hasName ?groupName.
?trip co:hasDepartTime ?departTime.
OPTIONAL
{
?trip co:hasReturnTime ?returnTime.
}
?trip co:hasOrigin ?ori.
?ori rdfs:label ?origin.
?trip co:hasDestination ?dest.
?dest rdfs:label ?destination.
FILTER regex(?groupName, "MUITIC2")
}
}
ORDER BY(?departTime)
```

B.7. Búsquedas para la Pregunta 7

La pregunta de competencia 7 trata de encontrar todas las reservas asociadas a un grupo de usuarios. En este caso se realiza una búsqueda de los viajes asociados al grupo *MUITIC2*.

```
PREFIX co: <http://www.carpoolingonto.com/carpoolingonto.owl#>
SELECT ?log ?date ?time ?dori ?ddest ?name ?surname ?email
FROM <http://www.carpoolingonto.com/carpoolingonto.owl>
WHERE
{
?user co:hasGroup ?group.
?group co:hasName ?gname.
?user co:hasLogin ?log.
?user co:hasReservation ?booking.
?booking co:hasOrigin ?dorigin.
?dorigin rdfs:label ?dori.
?booking co:hasDestination ?ddestination.
?ddestination rdfs:label ?ddest.
?booking co:hasDepartDate ?date.
?booking co:hasDepartTime ?time.
?booking co:isBookingOf ?trip.
?trip co:isTripOf ?driver.
?driver co:hasName ?name.
?driver co:hasSurname ?surname.
?driver co:hasEmail ?email.
FILTER regex(?gname,"MUITIC2")
}
ORDER BY ?date ASC(?time)
```

B.8. Pregunta global adicional

Mostrar todos los viajes registrados por los conductores. Se extraen algunos datos importantes de estos viajes, como son nombre del conductor, apellidos, dirección de email, origen del viaje, destino, etc.

```
PREFIX co: <http://www.carpoolingonto.com/carpoolingonto.owl#>
SELECT ?name ?surname ?email ?origin ?destination ?startDate ?endDate ?departDate
?departTime ?returnDate ?returnTime
FROM <http://www.carpoolingonto.com/carpoolingonto.owl>
WHERE {
{
?trip rdf:type co:FrequentTrip.
?trip co:isTripOf ?cond.
?cond co:hasName ?name.
?cond co:hasSurname ?surname.
?cond co:hasEmail ?email.
?trip co:hasStartDate ?startDate.
?trip co:hasEndDate ?endDate.
?trip co:hasDepartTime ?departTime.
OPTIONAL
{
?trip co:hasReturnTime ?returnTime.
}
?trip co:hasOrigin ?ori.
?ori rdfs:label ?origin.
?trip co:hasDestination ?dest.
?dest rdfs:label ?destination.
}
UNION
{
?trip rdf:type co:OneDayTrip.
?trip co:isTripOf ?cond.
?cond co:hasName ?name.
?cond co:hasSurname ?surname.
?cond co:hasEmail ?email.
?trip co:hasDepartDate ?departDate.
?trip co:hasDepartTime ?departTime.
OPTIONAL
{
?trip co:hasReturnDate ?returnDate.
?trip co:hasReturnTime ?returnTime.
}
```

```
?trip co:hasOrigin ?ori.
?ori rdfs:label ?origin.
?trip co:hasDestination ?dest.
?dest rdfs:label ?destination.
}
UNION
{
?trip rdf:type co:HabitualTrip.
?trip co:isTripOf ?cond.
?cond co:hasName ?name.
?cond co:hasSurname ?surname.
?cond co:hasEmail ?email.
?trip co:hasDepartTime ?departTime.
OPTIONAL
{
?trip co:hasReturnTime ?returnTime.
}
?trip co:hasOrigin ?ori.
?ori rdfs:label ?origin.
?trip co:hasDestination ?dest.
?dest rdfs:label ?destination.
}
}
```

Bibliografía

[1] Carlos Ghosn (CEO Renault y Nissan). Declaraciones. Automobile Club Francia, The Economist, Noviembre 2008.

[2] PMUS: Guía práctica para la elaboración e implantación de Planes de Movilidad Urbana Sostenible. IDAE: Instituto para la Diversificación y Ahorro de Energía, Julio 2006.

[3] PTT: Guía práctica para elaboración e implantación de Planes de Transporte al centro de Trabajo. IDAE: Instituto para la Diversificación y Ahorro de Energía, Julio 2006.

[4] Coche Compartido: Recomendaciones para compartir el coche. IDAE: Instituto para la Diversificación y Ahorro de Energía, Septiembre 2001.

[5] Guía Práctica de la Energía. Consumo Eficiente y Responsable. IDAE: Instituto para la Diversificación y Ahorro de Energía, Julio 2011.

[6] deAaB. http://www.deaab.com. Última visita Junio 2012.

[7] Amovens. http://www.amovens.com. Última visita Mayo 2012.

[8] Compartir.org. http://www.compartir.org. Última visita Mayo 2012.

[9] Blablacar.es. http://www.blablacar.es/. Última visita Mayo 2012.

[10] Tim Berners-Lee, James Hendler, and Ora Lassila. The Semantic Web. *Scientific American*, Mayo 2001.

[11] The Linking Open Data cloud diagram. http://richard.cyganiak.de/2007/10/lod/. Última visita Junio 2012.

[12] W3C Recommendation. *OWL Web Ontology Language Guide*, Febrero 2004. http://www.w3.org/TR/owl-guide/. Última visita Junio 2012.

[13] Tim Berners-Lee. Linked Data - Design Issues, Junio 2009. http://www.w3.org/DesignIssues/LinkedData.html. Última visita Junio 2012.

[14] Eric Ferguson. The rise and fall of the American carpool: 1970-1990. *Transportation*, 24(4):349–376, 1997.

[15] Zimride. http://www.zimride.com/. Última visita Mayo 2012.

[16] iCarpool. http://www.icarpool.com/. Última visita Junio 2012.

[17] ITSA (Intelligent Transport Society of America). http://www.itsa.org. Última visita Junio 2012.

[18] Avego. http://www.avego.com/. Última visita Mayo 2012.

[19] Stephan Hartwig. Empty Seats Traveling. *Nokia Research Center*, Febrero 2007. Disponible en web http://research.nokia.com/files/tr/NRC-TR-2007-003.pdf. Última visita Junio 2012.

[20] Comuto.com. http://www.comuto.com/. Última visita Mayo 2012.

[21] EMT (Empresa Municial de Transportes) de Madrid. http://www.emtmadrid.es/. Última visita Junio 2012.

[22] W3C Recommendation. *RDF Primer*, Febrero 2004. http://www.w3.org/TR/rdf-primer. Última visita Junio 2012.

[23] Tim Berners-Lee, Roy T. Fielding, and Larry Masinter. RFC 3986 Uniform Resource Identifier (URI): Generic Syntax. Technical report, Internet Engineering Task Force (IETF), Enero 2005.

[24] W3C Working Draft. *SPARQL 1.1 Query Language*, Enero 2012. http://www.w3.org/TR/sparql11-query/. Última visita Junio 2012.

[25] Ian Horrocks. Ontologies and the semantic web. *Commun. ACM*, 51(12):58–67, Diciembre 2008.

[26] Linked Data. http://linkeddata.org/. Última visita Junio 2012.

[27] World Wide Web Consortium (W3C). http://www.w3.org/. Última visita Junio 2012.

[28] DBpedia. http://dbpedia.org. Última visita Junio 2012.

[29] Christian Bizer. The Emerging Web of Linked Data. *Intelligent Systems, IEEE*, 24(5):87–92, Septiembre - Octubre 2009.

[30] W3C Semantic Web Activity. http://www.w3.org/2001/sw/. Última visita Junio 2012.

[31] semanticweb.org. http://semanticweb.org. Última visita Junio 2012.

[32] Semantic Web Challenge. http://challenge.semanticweb.org/. Última visita Junio 2012.

[33] Data Hub. http://thedatahub.org/. Última visita Junio 2012.

[34] Swoogle Semantic Web Search Engine. http://swoogle.umbc.edu/. Última visita Junio 2012.

[35] The Protégé Ontology Editor and Knowledge Acquisition System. http://protege.stanford.edu/. Última visita Junio 2012.

[36] Pablo Sauras Pérez. Estudio y mejora de un modelo para compartir vehículos. Proyecto Fin de Carrera Ingeniería de Telecomunicación. Universidad de Valladolid, Marzo 2010.

[37] Web Corporativa de GMV. http://www.gmv.com. Última visita Junio 2012.

[38] Master en Investigación en Tecnologías de la Información y las Comunicaciones. http://muitic.tel.uva.es. Última visita Junio 2012, 2012.

[39] Tecnologías Emergentes en Sistemas Telemáticos. http://emina.tel.uva.es/mediawiki/index.php/TES_2011-2012. Última visita Junio 2012.

[40] WGS84 Geo Positioning Ontology. http://www.w3.org/2003/01/geo/. Última visita Junio 2012.

[41] The Racer Systems GmbH & Co. KG. http://www.racer-systems.com/. Última visita Julio 2012.

[42] RFC 4510 - Lightweight Directory Access Protocol (LDAP): Technical Specification Road Map. Technical report, Internet Engineering Task Force (IETF), 2006.

[43] Matthew Horridge, Holger Knublauch, Alan Rector, Stevens Robert, and Chris Wroe. *A Practical Guide To Building OWL Ontologies Using The Protégé-OWL Plugin and CO-ODE Tools*. Universidad de Manchester, 2004.

[44] Pascal Hitzler and Frank van Harmelen. A Reasonable Semantic Web. *Semantic Web Journal*, 1(1):39–44, 2010.

[45] Natalya F. Noy and Deborah L. McGuinness. *Ontology Development 101: A Guide to Creating Your First Ontology*. Universidad de Stanford, Marzo 2001.

[46] Asunción Gómez-Perez and Oscar Corcho. Ontology languages for the Semantic Web. *Intelligent Systems, IEEE*, 17(1):54 – 60, Enero - Febrero 2002.

[47] W3C Recommendation. *RDF Vocabulary Description Language 1.0: RDF Schema*, Febrero 2004. Disponible en http://www.w3.org/TR/rdf-schema/. Última visita Junio 2012.

[48] DAML.org. http://www.daml.org/. Última visita Junio 2012.

[49] Deborah L. McGuinness, Richard Fikes, James Hendler, and Lynn Andrea Stein. DAML+OIL: an ontology language for the Semantic Web. *Intelligent Systems, IEEE*, 17(5):72 – 80, Septiembre/Octubre 2002.

[50] WC3 Note. *DAML+OIL (March 2001) Reference Description*, Diciembre 2001. Disponible en http://www.w3.org/TR/daml+oil-reference. Última visita Junio 2012.

[51] W3C Recommendation. *OWL 2 Web Ontology Language Quick Reference Guide*, Octubre 2009. http://www.w3.org/TR/owl2-quick-reference/. Última visita Junio 2012.

[52] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The description logic handbook: theory, implementation, and applications*. Cambridge University Press, New York, NY, USA, 2003.

[53] Mariano Fernández López, Asunción Gómez-Perez, and Natalia Juristo. METHONTOLOGY: from Ontological Art towards Ontological Engineering. In *Proceedings of the AAAI97 Spring Symposium*, pages 33–40, Stanford, USA, Marzo 1997.

[54] Steffen Staab, Rudi Studer, Hans-Peter Schnurr, and York Sure. Knowledge processes and ontologies. *Intelligent Systems, IEEE*, 16(1):26 – 34, Enero/Febrero 2001.

[55] Mike Uschold and Michael Grüninger. Ontologies: principles, methods and applications. *Knowledge Engineering Review*, 11(2):93–155, 1996.

[56] Natalya F. Noy, Michael Sintek, Stefan Decker, Monica Crubézy, Ray W. Fergerson, and Mark A. Musen. Creating Semantic Web Contents with Protégé-2000. *IEEE Intelligent Systems*, 16(2):60–71, Marzo 2001.

[57] Raji Ghawi and Nadine Cullot. Database-to-Ontology Mapping Generation for Semantic Interoperability. In *Third International Workshop on Database Interoperability (InterDB 2007)*, 2007.

[58] W3C Working Draft. *Time Ontology in OWL*, Septiembre 2006. http://www.w3.org/TR/owl-time/. Última visita Junio 2012.

[59] Martin J. O'Connor and Amar K. Das. A Method for Representing and Querying Temporal Information in OWL. In *Biomedical Engineering Systems and Technologies*, pages 97–110. Springer, 2010.

[60] Jonas Tappolet and Abraham Bernstein. Applied Temporal RDF: Efficient Temporal Querying of RDF Data with SPARQL. In *Proceedings of the 6th European Semantic Web Conference on The Semantic Web: Research and Applications*, ESWC 2009, pages 308–322, Berlin, Heidelberg, 2009. Springer-Verlag.

[61] W3C Recommendation. *XML Schema Part 0: Primer Second Edition*, Octubre 2004. http://www.w3.org/TR/xmlschema-0/. Última visita Junio 2012.

[62] Sean Bechhofer and Peter F. Patel-Schneider. *DIG 2.0: The DIG Description Logic Interface Overview*. DIG Working Group Note, Mayo 2006. http://dig.cs.manchester.ac.uk/overview.html. Última visita Junio 2012.

[63] Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz. Pellet: A practical OWL-DL reasoner. *Web Semantics: Science, Services and Agents on the World Wide Web*, 5(2):51 – 53, 2007.

[64] Dimitrios-Emmanuel Spanos, Periklis Stavrou, and Nikolas Mitrou. Bringing Relational Databases into the Semantic Web: A Survey. *Semantic Web Journal*, 2010.

[65] Satya S. Sahoo, Wolfgang Halb, Sebastian Hellmann, Kingsley Idehen, Ted Thibodeau Jr, Sören Auer, Juan Sequeda, and Ahmed Ezzat. A Survey of Current Approaches for Mapping of Relational Databases to RDF. W3C Incubator Group, Enero 2009.

[66] Guillermo Vega Gorgojo. *Ontoolcole/Ontoolsearch: un sistema centrado en el educador para la búsqueda de servicios CSCL basado en ontologías*. PhD thesis, Universidad de Valladolid, Julio 2007.

[67] Owen Sacco and Alexandre Passant. A Privacy Preference Ontology (PPO) for Linked Data. In *Linked Data on the Web Workshop at WWW2011*, 2011.

[68] Dean Allemang and James Hendler. *Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL*. Morgan Kaufmann, Mayo 2008.

[69] Amit B. Kothari. Genghis - A Multiagent Carpooling System. B.Sc. Dissertation Work, Universidad de Bath, Reino Unido, Mayo 2004.

[70] IDAE: Instituto para la Diversificación y Ahorro de Energía. http://www.idae.es/. Última visita Julio 2012.

[71] Mike Uschold. Building Ontologies: Towards a Unified Methodology. In *16th Annual Conf. of the British Computer Society Specialist Group on Expert Systems*, Cambridge, UK, Diciembre 1996.

[72] Zuo zhihong and Zhou mingtian. Web Ontology Language OWL and its description logic foundation. In *Proceedings of the Fourth International Conference on Parallel and Distributed Computing, Applications and Technologies, 2003. PDCAT 2003*, pages 157 – 160, Agosto 2003.

[73] Sebastian Tramp, Norman Heino, Sören Auer, and Philipp Frischmuth. Making the semantic data web easily writeable with RDFauthor. In *Proceedings of the 7th international conference on The Semantic Web: Research and Applications - Volume Part II*, ESWC 2010, pages 436–440, Berlin, Heidelberg, 2010. Springer-Verlag.

[74] Nigel Shadbolt, Wendy Hall, and Tim Berners-Lee. The Semantic Web Revisited. *Intelligent Systems, IEEE*, 21(3):96 –101, Enero - Febrero 2006.

[75] Victor Richard Benjamins, Jesús Contreras, Oscar Corcho, and Asunción Gómez-Pérez. Six Challenges for the Semantic Web. In *KR2002 Semantic Web Workshop*, 2002.

[76] Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked Data - The Story So Far. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 4(2):1–22, 2009.

[77] Viorel Milea, Flavius Frasincar, and Uzai Kaymak. tOWL: A Temporal Web Ontology Language. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 42(1):268 – 281, Febrero 2012.