DESARROLLO DE ONTOLOGÍAS PARA SU USO EN PERFILES DE USUARIO EN EL ENTORNO IMS

ANEXOS

DIEGO FABIÁN GALLEGO FERNÁNDEZ

JONNY ALBERTO CABRERA PAZOS

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Telemática
Servicios Avanzados de Telecomunicaciones
Popayán, Noviembre de 2009

DESARROLLO DE ONTOLOGÍAS PARA SU USO EN PERFILES DE USUARIO EN EL ENTORNO IMS

DIEGO FABIÁN GALLEGO FERNÁNDEZ

JONNY ALBERTO CABRERA PAZOS

Anexos del trabajo de Grado presentado como requisito para optar al título de Ingeniero en Electrónica y Telecomunicaciones

Directora: I.E. Mary Cristina Carrascal Reyes

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Telemática
Servicios Avanzados de Telecomunicaciones
Popayán, Noviembre de 2009

TABLA DE CONTENIDO

ANEXO A ENTORNO DE DESARROLLO ONTOLOGÍCO 1

A.1 LENGUAJES ONTOLÓGICOS 1

A.1.1 Ontolingua 1

A.1.2 OKBC Protocol (*Open Knowledge Base Connectivity Protocol*) 1

A.1.3 FLogic (*Frame Logic*) 1

A.1.4 OCML (Operational Conceptual Modeling Language) 1

A.1.5 XML 2

A.1.6 RDF y RDF Schema 2

A.1.7 SHOE (Simple HTML Ontology Extensions) 2

A.1.8 OIL 2

A.1.9 DAML+OIL 3

A.1.10 OWL 3

A.2 LENGUAJES DE CONSULTA EN ONTOLOGÍAS 4

A.2.1 RQL (*RDF Query Language*) 4

A.2.2 RDQL (*RDF Data Query Language*) 4

A.2.3 SeRQL (*Sesame RDF Query Language*) 4

A.2.4 DQL (*Daml Query Language*) 4

A.2.5 OWL-QL (OWL Query Language) 4

A.2.6 SPARQL 5

A.3 HERRAMIENTAS DE RECUPERACIÓN DE INFORMACIÓN ONTOLÓGICA. 5

A.3.1 Ontobroker 5

A.3.2 Jena 5

A.3.3 Sesame 5

A.3.4 KAON Tool 6
A.4 EDITORES DE ONTOLOGÍAS 6
A.4.1 WebOnto 6
A.4.2 OntoEdit 6
A.4.3 OilEd 6
A.4.4 Protégé 6
ANEXO B CASO DE ESTUDIO 8
B.1 MODELO DE CASOS DE USO EXTENDIDOS 8
B.1.1 Caso de uso Crear Instancia 8
B.1.2 Caso de uso Actualizar instancia 8
B.1.3 Caso de uso Eliminar Instancia 9
B.1.4 Caso de uso Consultar Instancia 10
B.1.5 Caso de uso Remover Instancia de Propiedad 10
B.1.6 Caso de uso Consultar Instancias de Propiedad 11
B.1.7 Caso de uso Adicionar Instancia a Propiedad 12
B.1.8 Caso de uso Actualizar Instancias de Propiedad 12
B.1.9 Caso de uso Gestionar Servicio 13
B.2 DIAGRAMA DE PAQUETES DEL CASO DE ESTUDIO 14
B.3 DIAGRAMA DE CLASES DEL CASO DE ESTUDIO 16
B.4 DESCRIPCIÓN DE LA ONTOLOGIA DE PERFILES DE USUARIO 17
ANEXO C CONFIGURACION DEL SDS PARA LA PROVICION DE SERVICIOS 20
C.1 DESCRIPCIÓN DE LA PERSPECTIVA *PROVISIONING* DE ECLIPSE 20
C.1.1 Configuración del DNS 21
C.1.2 Configuración del HSS 22
C.1.3 Configuración del CSCF 26

ANEXO D IMPLEMENTACIÓN DEL SERVICIO DE PRUEBA 29
D.1 MODELO DE CASOS DE USO EXTENDIDOS 29
D.1.1 Caso de uso Aceptar invitación 29
D.1.2 Caso de uso Confirmar recepción de mensaje 29
D.1.3 Caso de uso Finalizar Sesión 30
D.1.4 Caso de uso Gestionar Mensajes. 30
D.1.5 Caso de uso Crear Cliente REST 31
D.1.6 Caso de uso Crear Recurso 31
D.1.7 Caso de uso Eliminar Recurso 32
D.1.8 Caso de uso Actualizar Recurso 32
D.1.9 Caso de uso Recuperar Recurso 33
D.2 DIAGRAMA DE PAQUETES DEL SERVICIO 33
D.3 DIAGRAMA DE CLASES DEL SERVICIO. 34
ANEXO E DESARROLLO DE CLIENTES IMS CON SDS 36
E.1 PLATAFORMA PARA EL DESARROLLO DE CLIENTES IMS 36
E.1.1 Configuración del Perfil de usuario ICP 36
E.1.2 Adaptadores ICP 39
E.1.3 Creación de un IProfile y supervisión de estado del ICP 39
E.1.4 Creación de un IService. 40
E.1.5 Creación un ISession 41
E.1.6 Envío de Mensajes 42
E.2 Instalación de clientes ICP sobre móviles con sistema operativo Symbian 44
REFERENCIAS 45

LISTA DE FIGURAS

Figura 1 Arquitectura de integración entre JENA y Protege-OWL API 7

Figura 2 Diagrama de paquetes del gestor de perfiles de usuario. 14

Figura 3 Diagrama de clases del gestor de perfiles de usuario 16

Figura 4 Diagrama de clases de la ontología de perfiles de usuario 17

Figura 5 Selección de perspectiva Provisioning. 20

Figura 6 Perspectiva *Provisioning* 21

Figura 7 DNS Provisioning – adicionar un Nuevo registro DNS 21

Figura 8 Herramienta para la configuración del HSS. 22

Figura 9 HSS Provisioning – Crear un nuevo iFC. 23

Figura 10 HSS Provisioning – Crear un nuevo SPT. 24

Figura 11 HSS Provisioning – Crear un nuevo Perfil de Servicio. 25

Figura 12 HSS Provisioning – Crear un nuevo Perfil de Usuario. 26

Figura 13 Selección de la perspectiva Visual Network. 26

Figura 14 Perspectiva Visual Network. 27

Figura 15 Opciones para la entidad CSCF. 27

Figura 16 Preferencias de configuración del CSCF 28

Figura 17 Diagrama de paquetes del Servicio de Prueba. 34

Figura 18 Diagrama de clases del Servicio de Prueba. 34

Figura 19 Panel de control – ICP Settings 36

Figura 20 Interfaz gráfica del ICP *Settings* 37

Figura 21 Selección del perfil IMS. 37

Figura 22 Configuración de dirección del P-CSCF 38

Figura 23 Datos para el registro del Usuario. 38

Figura 24 Registro del Usuario. 39

Figura 25 Creación de un IProfile y supervisión de estado del ICP. 40

Figura 26 Creando un IService.....41

Figura 27 Diagrama de secuencia de creación de una ISession. 42

Figura 28 Diagrama de secuencia para el envío de mensajes.....43

Figura 29 Diagrama de Clases del Cliente. 43

Figura 30 Diagrama de Paquetes del Cliente.....44

LISTA DE TABLAS

Tabla 1 Clases de la ontología de perfil de usuario 18

Tabla 2 Propiedades de tipo objeto de la ontología de perfiles de usuario 18

Tabla 3 Propiedades de los datos simples de la ontología de perfiles de usuario 19

ANEXO A ENTORNO DE DESARROLLO ONTOLOGÍCO

A.1 LENGUAJES ONTOLÓGICOS

Aunque las ontologías son una tecnología reciente, varios lenguajes se han utilizado para desarrollarlas y con el correr del tiempo se han venido depurando para dar origen a lenguajes más completos para su especificación.

Los lenguajes de especificación de ontologías pueden dividirse en dos grupos: los lenguajes tradicionales empleados en los sistemas de representación de conocimiento y los más recientes, los lenguajes de especificación basados en Web.

Los lenguajes de ontologías tradicionales se diferencian unos de otros por la forma como representan el conocimiento, ya sea basado en *frames* (marcos), lógica descriptiva, predicados de primer y segundo orden, u orientación a objetos. Dentro de los más representativos tenemos a:

A.1.1 Ontolingua

Es un lenguaje de ontologías basado en KIF (*Knowledge Interchange Format,* Formato de Intercambio de Información) y en FO (*Frame Ontology*, Ontología de Marcos) empleado en el Ontolingua Server, para intercambiar conocimiento. Este permite la descripción de ontologías basándose en *frames*, lo que dio origen a términos como clase, instancia, subclase, además de que brinda la posibilidad de declarar axiomas [1].

A.1.2 OKBC Protocol (*Open Knowledge Base Connectivity Protocol*)

Es un protocolo el cual se basa en el GFP (*Generic Frame Protocol*, Protocolo Genérico de Marcos) es decir, basado en *frames*. Es utilizado como complemento de lenguajes de representación de conocimiento, y permite, como otros sistemas basados en clases, constantes, atributos, *frames* e instancias que sirven de base de conocimiento. A su vez ofrece una interfaz para al conocimiento al igual que funciones utilizadas para acceder a través de la red a un conocimiento compartido [2].

A.1.3 FLogic (*Frame Logic*)

Es un lenguaje basado en *frames* que hace uso de lógica de primer orden. Algunas de sus características son semejantes a los lenguajes de orientación a objeto como la herencia, objetos complejos, tipos polimórficos etc. Con FLogic se han desarrollado desde bases de datos a ontologías y se puede combinar con otros programas de lógica para interactuar con la información almacenada en la ontología [3].

A.1.4 OCML (Operational Conceptual Modeling Language)

Es un lenguaje también basado en *frames* cuya sintaxis permite declarar relaciones, funciones, reglas, clases e instancias. A este lenguaje se le añadió un módulo de mecanismos de lógica y una interfaz para poder interactuar con el conocimiento. Una de las ventajas que tiene este lenguaje es que es compatible con estándares como Ontolingua [4].

Dentro del grupo de especiación web de ontologías existen varios lenguajes, siendo los más utilizados aquellos basados en etiquetas XML. Lenguajes como OIL (*Ontology Inference Layer*), DAML+OIL

(*DARPA's Agent Markup Language + Ontology Inference Layer*) y OWL (*Ontology Web Language, Lenguaje de Ontologias Web*) tienen como bases a lenguajes como XML(*eXtensible Markup Language*, lenguaje de marcas extensible), RDF (*Resource Description Framework*, Marco de Descripción de Recursos) y RDF Schema todos estos se describen a continuación:

A.1.5 XML

Es la base fundamental en la que se apoyan varios de los lenguajes para la construcción de ontologías, sin embargo, aunque se incluye aquí como un lenguaje de especificación de ontologías, en realidad no lo es, dado que su orientación principal es el intercambio de información, permitiendo únicamente extraer datos, sin brindar un contenido semántico. A su vez XML presenta una serie de problemas por basarse en la estructura de los documentos, ya que cuando un esquema XML cambia pueden invalidarse consultas anteriores al mismo documento, lo cual, lo hace poco escalable [5].

A.1.6 RDF y RDF Schema

Es un lenguaje de etiquetado, cuya sintaxis se basa en XML, que define un modelo de datos para describir recursos (cualquier objeto identificable por un URI), basado en tres partes sujeto, predicado y objeto: El sujeto es todo aquello de lo que se puede decir algo, y son referenciados por medio de un URI; el predicado define atributos, aspectos, características o representan una relación que describe a un recurso; los objetos posibilitan establecer valores a las propiedades de un recurso específico [5].

Por otra parte, RDF Schema es un vocabulario RDF que brinda un mecanismo para definir clases, objetos y propiedades; relaciones entre clases y propiedades; y, restricciones de dominio y rango sobre las propiedades, con lo cual se acerca más al concepto de ontología, permitiendo realizar consultas y un razonamiento automático. Sin embargo, RDFS no es lo suficientemente expresivo para representar ontologías complejas, puesto que no fue diseñada como solución final para representar conocimiento, sino como un núcleo para ser extendido [5].

A.1.7 SHOE (Simple HTML Ontology Extensions)

Fue el primer lenguaje de etiquetado para definir ontologías en el entorno Web, cuyo propósito es el de potenciar y mejorar los motores de búsqueda de Internet. SHOE define unas etiquetas que son agregadas a los documentos HTML, que permiten estructurar clases y relaciones entre clases, así como reglas de inferencia, sin embargo, no dispone de mecanismos para expresar negaciones o disyunciones [6].

A.1.8 OIL

Es el primer lenguaje desarrollado para especificar ontologías y permitir su intercambio, basado en estándares de la W3C (*World Wide Web Consortium*). Fue diseñado por un grupo de investigación dentro de proyecto OnToKnowledge, basándose en la lógica descriptiva (declaración de axiomas o reglas) y sistemas basados en marcos (taxonomía de clases y atributos) [5].

OIL está estructurado en tres niveles: un nivel de objeto (donde residen las instancias); un primer nivel meta con las definiciones y descripciones de la ontología y por último un segundo nivel meta donde reside información sobre las características de la ontología [5].

OIL posee algunas limitaciones dado que no ofrece la posibilidad de sobrescribir valores heredados de una superclase, a la vez, presenta falta de expresividad en la declaración de axiomas (reglas); y, no soporta dominios concretos como por ejemplo números enteros, cadenas de caracteres, etc. [5].

A.1.9 DAML+OIL

Es un lenguaje de representación de ontologías, desarrollado por dos grupos de trabajo, el de OIL y DARPA (*US Defense Advanced Research Projects Agency*), quienes anteriormente habían desarrollado el lenguaje DAML con el objetivo de mejorar el nivel de expresividad que ofrecía RDFS. Por tanto DAML+OIL unifica estos dos lenguajes, para concentrar esfuerzos en el desarrollo de un lenguaje unificado para la definición de ontologías [7][5].

Aunque DAML+OIL tiene muchas coincidencias con OIL, también tiene sus diferencias, la principal es que se trata de un lenguaje que se aleja del modelo de marcos (taxonomía de clases y atributos) de OIL, para acercarse más a una base de lenguajes de lógica descriptiva. Por lo tanto, DAML+OIL es un lenguaje que permite realizar una mejor descripción de clasificaciones y propiedades de los recursos que las que ofrecía RDF y RDFS [7][8].

A nivel práctico DAML+OIL demuestra ser más útil como soporte para ontologías que, por ejemplo, RDF Schema, sin embargo aún tiene ciertas carencias en cuanto a formatos de intercambio y modelado de ontologías. Por otro lado, aunque DAML+OIL es soportado por un gran número de herramientas y aplicaciones, pero su complejidad conceptual dificulta su aprendizaje y uso, algo que se intentó solventar con el desarrollo de OWL. No obstante, se desarrollaron muchas aplicaciones que utilizan DAML-OIL y también existen herramientas para convertir DAML a OWL [8].

A.1.10 OWL

Es un lenguaje de marcado derivado de DAML+OIL, cimentado sobre RDFS y codificado en XML, que permite publicar y compartir ontologías en la World Wide Web. OWL fue desarrollado por la W3C y oficialmente es el lenguaje estándar para especificar ontologías y compartir conocimiento en la web [5] [9].

OWL se descompone en tres sublenguajes, siendo estos:

- OWL Lite es el más sencillo de los tres, el cual permite clasificar conceptos jerárquicamente con restricciones simples, haciéndole el más adecuado para los principiantes en la creación de ontologías [9].

- OWL DL este sublenguaje ya tiene todo el vocabulario OWL completo, haciéndolo perfecto para aquellos usuarios que requieren la máxima expresividad del lenguaje y una buena eficiencia computacional [9].

- OWL full es el más completo de los tres sublenguajes, permitiendo la misma expresividad que OWL DL y con toda la capacidad sintáctica que ofrece RDF, sin embargo no ofrece ninguna garantía computacional, dado que añade una gran carga de procesamiento [9].

OWL por medio del uso de URIs para el etiquetado y la estructura para la descripción para la Web que RDF ofrece, brinda las siguientes capacidades a las ontologías [5]:

- Capacidad para ser distribuida a través de muchos sistemas.

- Escalabilidad para las necesidades de la Web.
- Compatibilidad con estándares de la Web para accesibilidad y extensibilidad.
- Abierto y extensible.

A.2 LENGUAJES DE CONSULTA EN ONTOLOGÍAS

Hasta el momento se han realizado varios esfuerzos para generar un lenguaje estándar de consulta dentro de una ontología, que ha dado origen a varios desarrollos, algunos de cuales nombramos a continuación:

A.2.1 RQL (*RDF Query Language*)

Lenguaje de consulta para RDF y RDF *Schema* basado en OQL *(Object Query Language*). RQL proporciona un mecanismo para consultar y seleccionar los nodos del modelo que se quieran recuperar, basado en las relaciones semánticas dentro del RDF *Schema*, tales como relaciones de clase/instancia, clase/propiedad o el dominio y rango de una propiedad, haciendo más sencillo recuperar información modelo [10].

A.2.2 RDQL (*RDF Data Query Language*)

Fue desarrollado por HP para que fuese el lenguaje de consulta para RDF dentro de su colección de herramientas para manipulación de ontologías Jena, cuyo modelo de consulta está orientado a los datos, por lo cual, sólo se pueden hacer consultas sobre la información que hay en el modelo, por lo que la inferencia o razonamiento no es posible [11].

A.2.3 SeRQL (*Sesame RDF Query Language*)

Este lenguaje fue desarrollado por los administradores de Sesame, que combina las características de diferentes lenguajes como RQL, RDQL, añadiendo algunas más propias. Brinda soporte de RDF *Schema*, XML, y coincidencias por URI's [12].

A.2.4 DQL (*Daml Query Language*)

Se trata de un lenguaje formal y protocolo para conducir el diálogo entre un agente cliente que realiza consultas y un agente que contesta los requerimientos, haciendo uso para ello de conocimiento representado en DAML + OIL. Sus consultas siguen un patrón de *"Query"* y *"Answer"* descritos en una ontología dql.daml, la cual describe como deben estar formadas las consultas como las respuestas [13].

A.2.5 OWL-QL (OWL Query Language)

Es una actualización del lenguaje anterior (DQL). En este caso, se trata de un lenguaje formal y protocolo para conducir el diálogo entre un agente cliente que realiza consultas y un agente que contesta los requerimientos, haciendo uso para ello de conocimiento representado en OWL [14].

A.2.6 SPARQL

Es un lenguaje de consulta orientado a la Web 2.0, que ofrece un lenguaje de interrogación preciso para RDF y un protocolo de recuperación que permite la integración de recursos distribuidos. SPARQL es el lenguaje de consulta estándar dentro de la Web semántica, cuyo objetivo es facilitar la identificación de información en la red. SPARQL permite que los desarrolladores y usuarios finales puedan representar y utilizar los resultados obtenidos en las búsquedas a través de una gran variedad de información como son datos personales, redes sociales y metadatos sobre recursos digitales como música e imágenes [15].

A.3 HERRAMIENTAS DE RECUPERACIÓN DE INFORMACIÓN ONTOLÓGICA.

Entre las herramientas que han alcanzado una gran madurez gracias a su desarrollo por parte de los investigadores de la Web Semántica, se encuentran los sistemas de almacenamiento de ontologías. Mediante estos sistemas es posible mantener las ontologías en bases de datos e ir añadiendo nueva información, y con la ayuda de razonadores probar la consistencia de la ontología. La mayoría de los sistemas de almacenamiento están orientados a descripciones de conceptos escritas en RDF, aunque algunas han ido actualizándose a los últimos lenguajes de especificación.

Las principales herramientas en esta área son:

A.3.1 Ontobroker

Es un servicio de consulta ontológica que proporciona una serie de lenguajes y herramientas para mejorar las consultas, accesos y servicios de inferencia en la web y solucionar el problema de cuello de botella en el acceso a la información existente en Internet debido a la creciente cantidad de información y a la libertad en la representación de la información. Ontobroker tiene una arquitectura con tres elementos centrales: una interfaz de consulta, un motor de inferencia que proporciona respuestas y un agente web encargado de colectar el conocimiento requerido de la web. Cada uno de estos elementos va acompañado de un lenguaje de formalización: el lenguaje de consulta para formular consultas, el lenguaje de representación para formular ontologías, y el lenguaje de anotación para dotar a los documentos web de información ontológica [16].

A.3.2 Jena

Es una colección de herramientas desarrolladas por Hewlett-Packard para construir aplicaciones en la Web semántica. Esta colección, provee un API (*Application Programming Interface*, Interface de Programación de Aplicaciones) para RDF y OWL, un motor de consultas en SPARQL, soporte de ontologías en RDFS, DAML + OIL y OWL e incluye un motor de inferencia basado en reglas [17].

A.3.3 Sesame

Es una arquitectura que permite almacenar y consultar información en RDF y RDF-Schema desarrollada por Administrator Nederland bv. Esta ofrece la funcionalidad de añadir y eliminar información escrita en RDF en los repositorios, que puede almacenarse en varias bases de datos (MySQL, Oracle etc.) gracias a una interfaz que realiza la traducción de las solicitudes en lenguajes de consulta como RQL, RDQL, SeRQL a las respectivas sentencias del motor de base de datos [18].

A.3.4 KAON Tool

Desarrollado por la infraestructura KAON (Karlsruhe Ontology) Semantic Web, implementa un interfaz independiente del sistema en el que se almacenarán las ontologías, ya sean cualquier base de datos o un fichero texto. Implementa un API para leer las descripciones de los recursos, emplea RQL para realizar consultas y soporta tanto ontologías DAML + OIL como RDF [19].

A.4 EDITORES DE ONTOLOGÍAS

Para la construcción de una ontología la principal herramienta son los editores. Muchos editores han sido desarrollados para un lenguaje específico, sin embargo, han venido incorporando nuevos módulos para soportar otros lenguajes de especificación diferentes.

Los principales editores de ontología hasta el momento han sido:

A.4.1 WebOnto

Fue desarrollado por el KMI *(Knowledge Media Institute, Instituto de Conocimiento Multimedia)* en 1997 como editor de ontologías OCML, cuya característica principal es que permite la participación de varios usuarios en el desarrollo de la ontología por medio de un applet java.

A.4.2 OntoEdit

Es un editor que soporta F-Logic RDF y OIL aunque después almacena el conocimiento en XML. La última versión es OntoEdit v0.6 [20]

A.4.3 OilEd

Este herramienta inicialmente se basaba en el desarrollo de ontologías OIL y DAML + OIL, pero se han ido actualizando para soportar la mayoría de los lenguajes de especificación actuales. Es ampliamente utilizado en el área de la investigación debido a que permite interactuar con un razonador como FACT o RACER con el cual se puede verificar la consistencia de una ontología. Algunas de sus desventajas son el no soportar ontologías grandes, no contar con un control de versiones y no el permitir la migración e integración con otras ontologías [21].

A.4.4 Protégé

Protégé fue desarrollado por el SMI (*Stanford Medical Informatic,* Centro de Informatica Médica de Stanford) en la Universidad de Stanford. Actualmente es uno de los editores de ontologías más usado por investigadores para desarrollar sus ontologías, ya que es una herramienta que se actualiza con bastante regularidad y a la que se le pueden añadir módulos y plugins con nuevas funcionalidades, que permiten que la ontología desarrollada se exporte a los diferentes lenguajes de especificación más empleados actualmente (RDF, DAML, OWL, etc.). La versión estable para Julio de 2009 es la v3.3.1 y en fase beta la v. 4.0 [22].

Protégé cuenta con un API de código abierto desarrollado en el lenguaje JAVA, que provee de métodos y clases para cargar y almacenar ontologías, así como para manipular los modelos y realizar consultas [22].

Protégé soporta dos maneras de modelar ontologías:

La primera opción, la brinda el editor Protégé-Frames que permite modelar ontologías basadas en *frames*, de acuerdo al protocolo OKBC (*Open Knowledge Base Connectivity*, Conectividad de Bases de Conocimiento. Abiertas). En este modelo, una ontología está integrada por un conjunto de clases organizadas en una jerarquía para representar los conceptos de un dominio, un conjunto de slots asociados a las clases para describir sus propiedades y relaciones, y un conjunto de instancias de aquellas clases, las cuales adquieren unos valores específicos en sus propiedades [22].

Como segunda opción, la brinda el editor Protégé-OWL, que permite construir ontologías para la Web Semántica, basadas en el lenguaje OWL, en cuyo caso la ontología puede incluir las descripciones de clases, propiedades y sus instancias. El uso OWL permite la descripción de la semántica asociada a un dominio, con cual se pueden hacer deducciones de consecuencias lógicas, por ejemplo, datos que literalmente no están presentes en la ontología, pero que hacen parte de la semántica del dominio [22].

Protégé OWL-siempre ha tenido una estrecha relación con Jena, el cual es uno de los API de Java más utilizado para RDF y OWL, la representación de modelos, el análisis sintáctico, la persistencia en bases de datos, consultas y algunas herramientas de visualización. El intérprete Jena ARP (An RDF Parser, Un Interprete de RDF) aún se utiliza como intérprete de Protégé-OWL, y otros servicios tales como la validación del lenguaje y los tipos de datos han sido reutilizados de Jena [22].

Desde el agosto de 2005, el Protégé-OWL se integró más estrechamente con Jena. Esta integración permite a los programadores usar ciertas funciones de Jena en tiempo real, sin necesidad un largo proceso de recarga. La arquitectura de esta integración se nuestra en la Figura 1:

La clave de esta integración radica en que ambos sistemas funcionan sobre un modelo de representación "triple" de bajo nivel. Protégé tiene su propio mecanismo nativo de almacenamiento basado en *frames* (*Protege Frame Store*), el cual ha sido encapsulado por Protégé-OWL con las clases TripleStore. En el entorno de Jena, a las interfaces correspondientes se denominan Grafo y Modelo. El Protégé TripleStore ha sido encapsulado en el grafo Jena (ProtegeGraph), de modo que cualquier acceso desde el API Jena opere sobre la tripleta de Protégé. Para modificar la tripleta, el API convencional de Protégé-OWL debe ser usado. Sin embargo, este mecanismo trae consigo la ventaja de que permite el empleo de métodos de Jena para la consulta, mientras la ontología es editada desde Protégé [22].

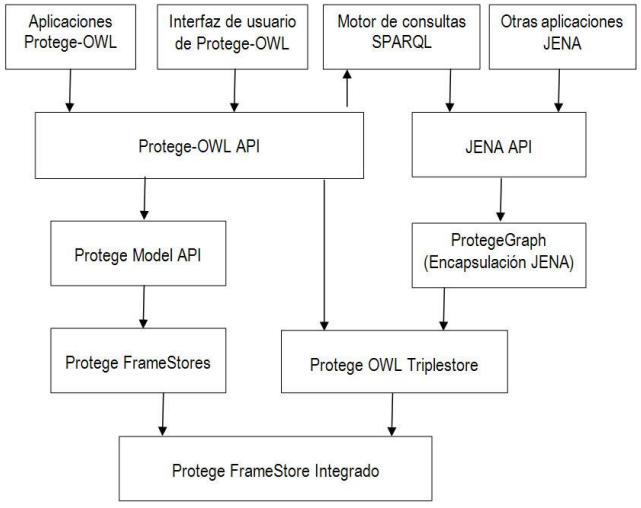

Figura 1 Arquitectura de integración entre JENA y Protege-OWL API

ANEXO B CASO DE ESTUDIO

Este anexo incluye el modelo de casos de uso, los diagramas de paquetes y de clases relacionadas con la implementación del gestor de perfiles de usuario, basado en la arquitectura de referencia que se desempeña dentro del entorno de IMS (*IP Multimedia Subsystem***,** Subsistema Multimedia IP).

B.1 MODELO DE CASOS DE USO EXTENDIDOS

B.1.1 Caso de uso Crear Instancia

Iniciador	Aplicación IMS
Propósito	Crear instancias de cualquiera de las clases establecidas en la ontología de perfiles de usuario.
Resumen	La aplicación de IMS determina que es necesario crear una nueva instancia de una clase, como por ejemplo de una Persona (*Person*), y realiza una solicitud al gestor de perfiles de usuario utilizando la interfaz de comunicación que éste ofrece. Dicha solicitud se procesa para generar la correspondiente respuesta y almacenar los cambios en la ontología.

Precondiciones: Ninguna.

Flujo principal:

- La aplicación IMS solicita la creación de una nueva instancia
- El gestor crea la nueva instancia en la ontología y guarda los cambios
- El sistema retorna un mensaje con el identificador asignado en la ontología para la nueva instancia.

Flujo de excepción:

E1: Si la clase invocada para crear la nueva instancia no existe en la ontología, entonces el sistema retorna un error reportando que el recurso invocado no existe.

E2: Si los datos enviados como parámetros por la aplicación al gestor, no tienen el formato establecido para cada una de las propiedades que hacen parte de la clase, entonces el sistema retorna un mensaje notificando que los parámetros son inválidos.

B.1.2 Caso de uso Actualizar instancia

Iniciador	Aplicación IMS
Propósito	Cambiar los datos almacenados en la ontología de perfiles de usuario.
Resumen	La aplicación IMS determina que uno de los valores del perfil de usuario debe ser actualizado, entonces envía una solicitud para su actualización haciendo uso de los métodos ofrecidos por el gestor de perfiles de usuario. El gestor procesa la solicitud y modifica los datos de la instancia requerida, enviando una respuesta para notificar que la acción se realizó con éxito.

Precondiciones: Existe una instancia en la ontología de perfiles de usuario cuyo identificador es conocido por la aplicación IMS.

Flujo principal:

- La aplicación IMS realiza una solicitud de actualización, agregándole el identificador y los nuevos datos de la instancia que será actualizada.
- El gestor actualiza la instancia en la ontología y guarda los cambios.
- El sistema retorna un mensaje reportando que la operación fue exitosa.

Flujo de excepción:

E1: En el caso de que el identificador de la instancia a actualizar enviado por la aplicación de IMS no exista en la ontología, el sistema retorna un mensaje reportando dicho error.

E2: Si los datos enviados como parámetros por la aplicación al gestor, no tienen el formato establecido para las propiedades de la instancia, entonces el sistema retorna un mensaje notificando que surgió un error en la operación.

B.1.3 Caso de uso Eliminar Instancia

Iniciador	Aplicación IMS
Propósito	Eliminar de la ontología de perfiles de usuario instancia de una Clase.
Resumen	Cuando la aplicación IMS establece que alguna instancia de una clase ya no es de utilidad en el perfil de usuario, envía una solicitud al gestor de perfiles de usuario, el cual la procesa y elimina la instancia solicitada, modificando la ontología, y envía una notificación de éxito en la operación.

Precondiciones: Existe una instancia en la ontología de perfiles de usuario cuyo identificador es conocido por la aplicación IMS.

Flujo principal:

- La aplicación IMS solicita que se elimine una instancia.
- El gestor elimina la instancia en la ontología y guarda los cambios.
- El sistema retorna un mensaje reportando que la operación fue exitosa.

Flujo de excepción:

E1: En el caso de que el identificador enviado por la aplicación de IMS no exista en la ontología, el sistema retorna un mensaje reportando que el recurso no fue encontrado.

B.1.4 Caso de uso Consultar Instancia

Iniciador	Aplicación IMS
Propósito	Consultar la información de una instancia de la ontología de perfiles de usuario.
Resumen	La aplicación IMS puede consultar toda la información de una instancia de cualquier clase que hace parte del perfil de usuario, recuperando ya sea información que ella misma ha generado o generada por otras aplicaciones. En dicho caso esto envía una solicitud, invocando los métodos ofrecidos por el gestor de perfiles de usuario para realizar la consulta de una instancia. El gestor procesa la solicitud y tras consultar en la ontología los datos requeridos de la instancia, retorna dicha información.

Precondiciones: Existe una instancia en la ontología de perfiles de usuario cuyo identificador es conocido por la aplicación IMS.

Flujo principal:

- La aplicación IMS solicita la información de una instancia, valiéndose de su identificador.
- El gestor de perfiles de usuario realiza una consulta en la ontología para recuperar la información relacionada con la instancia solicitada.
- El servicio retorna un mensaje que contiene toda información que fue consultada.

Flujo de excepción:

E1: En el caso de que el identificador enviado por la aplicación de IMS no exista en la ontología, el sistema retorna un mensaje de error reportando que los parámetros enviados no son correctos.

B.1.5 Caso de uso Remover Instancia de Propiedad

Iniciador	Aplicación IMS
Propósito	Permitir a la aplicación IMS remover una instancia de la colección de instancias asociadas a una propiedad.
Resumen	La aplicación de IMS tras haber seleccionado la instancia que desea remover de las agrupadas en una propiedad, invoca a los métodos establecidos por el gestor de perfiles de usuario para removerla de dicho grupo. El gestor de perfiles de usuario procesa la solicitud y realiza los cambios correspondientes en la ontología para remover la instancia de la propiedad, proceso que al ser exitoso, retorna una confirmación a la aplicación IMS

Precondiciones:

- Existe una instancia en la ontología de perfiles de usuario cuyo identificador es conocido por la aplicación IMS.
- La aplicación IMS tiene el conocimiento de una instancia, cuya clase posee propiedades con valores multiples, por ejemplo la propiedad intereses de la persona, que tiene un conjunto de instancias de la clase Interés.

Flujo principal:

- La aplicación IMS envía una solicitud para remover una instancia que hace parte de la colección de valores de una propiedad.
- El gestor realiza los cambios solicitados en la ontología y los almacena.
- El sistema retorno un mensaje de éxito en la operación

Flujo de excepción:

E1: Si la instancia cuya propiedad se desea modificar no existe, entonces el sistema envía un mensaje reportando que el recurso no fue encontrado.

E2: Si el identificador de la instancia a remover de la colección de valores de la propiedad no existe, entonces el sistema envía un mensaje reportando un error en los parámetros enviados.

B.1.6 Caso de uso Consultar Instancias de Propiedad

Iniciador	Aplicación IMS
Propósito	Permitir a la aplicación IMS acceder al conjunto de instancias que están asociadas a una propiedad.
Resumen	La aplicación IMS realiza una consulta del conjunto de instancias que hacen parte de la propiedad de un usuario, valiéndose de los métodos ofrecidos por el gestor. El gestor de perfiles de usuario procesa la solicitud y luego de consultar el conjunto de instancias de la propiedad, le envía a la aplicación IMS dichos datos.

Precondiciones:

- Debe existir una instancia en la ontología de perfiles de usuario, cuyo identificador es conocido por la aplicación IMS.
- La aplicación IMS tiene el conocimiento de una instancia, cuya clase posee propiedades con múltiples valores, por ejemplo la propiedad intereses de la persona, que tiene un conjunto de instancias de la clase Interés.

Flujo principal:

- La aplicación IMS envía una solicitud para consultar todos los valores que hacen parte de la colección de una propiedad.
- El gestor realiza una consulta en la ontología.
- El sistema retorno un mensaje con todos los valores asignados a la propiedad

Flujo de excepción:

E1: Si la instancia cuya propiedad se desea consultar no existe, entonces el sistema envía un mensaje reportando que el recurso no fue encontrado.

B.1.7 Caso de uso Adicionar Instancia a Propiedad

Iniciador	Aplicación IMS
Propósito	Permitir a la aplicación IMS la adición de una instancia al valor de una propiedad
Resumen	La aplicación de IMS luego de haber creado instancias, puede asociarlas como el valor de una propiedad de otra instancia. Por ejemplo, en el caso de una instancia de la clase "*Person*" (persona), esta cuenta con la propiedad llamada *prefereces* (preferencias), la cual es un conjunto de instancias de la clase *Preference* (preferencia), que será incrementado cuando se adicionan nuevas instancias a la propiedad. Para esto la aplicación IMS envía al gestor de perfiles de usuario la solicitud de adición de instancia junto con la información necesaria para realizar la operación. El gestor de perfiles de usuario procesa la solicitud y realiza los cambios correspondientes en la ontología, que al ser exitosos, éste retorna una respuesta de confirmación a la aplicación IMS..

Precondiciones:

- Debe existir una instancia en la ontología de perfiles de usuario cuyo identificador es conocido por la aplicación IMS.
- La aplicación IMS tiene el conocimiento de una instancia, cuya clase posee propiedades con múltiples valores.

Flujo principal:

- La aplicación IMS solicita la adición de una instancia
- El servicio adiciona la instancia a la colección de valores de una propiedad en la ontología.
- El servicio notifica a la aplicación IMS que la operación fue exitosa.

Flujo de excepción:

E1: Si la instancia a la que se le quiere adicionar un valor no existe, entonces el sistema envía un mensaje de error reportando que el recurso no fue encontrado.

E2: Si la identificación de la instancia enviada en la solicitud para ser agregada al conjunto de valores de una propiedad, no corresponde a ninguna de las presentes en la ontología, el sistema retorna un mensaje de error reportando que el parámetro enviado no es válido.

B.1.8 Caso de uso Actualizar Instancias de Propiedad

Iniciador	Aplicación IMS
Propósito	Permitir a la aplicación IMS modificar en una sola acción todo el conjunto de instancias asociadas a una propiedad
Resumen	Dado que una propiedad puede tener como valores un conjunto de instancias, el gestor de perfiles de usuario permite modificarlas realizando una solicitud de actualización, con el nuevo conjunto de instancias que serán asociadas a la propiedad. Como respuesta a la solicitud, el gestor realiza las correspondientes asociaciones dentro de la ontología y envía una notificación de éxito a la aplicación de IMS

Precondiciones:

- Debe existir una instancia en la ontología de perfiles de usuario con una propiedad a la que puede asignársele múltiples valores, y cuyo identificador es conocido por la aplicación IMS
- La aplicación IMS tiene el conocimiento de los identificadores de las instancias que desea adicionar como valores de una propiedad.

Flujo principal:

- La aplicación IMS realiza la solicitud para actualizar los valores de una propiedad.
- El gestor de perfiles de usuario actualiza los datos en la ontología y almacena los cambios.
- El sistema retorna un mensaje de éxito en la operación.

Flujo de excepción:

E1: Si la instancia cuya propiedad se desea actualizar no existe, entonces el sistema envía un mensaje reportando que el recurso no fue encontrado.

E2: Si la identificación de las instancias enviadas en la solicitud no corresponden a ninguna de las presentes en la ontología, el sistema retorna un mensaje de error reportando los parámetros enviados en la solicitud no solo validos.

B.1.9 Caso de uso Gestionar Servicio

Iniciador	Operador IMS
Propósito	Permitir a la entidad responsable del dominio de IMS establecer la configuración del servicio ofrecido por el gestor de perfiles de usuario.
Resumen	El Operador de IMS que es el responsable del servidor de aplicaciones donde se encuentra alojado el gestor de perfiles de usuario y la ontología, se encarga de la configuración inicial del entorno así como los de los procesos de iniciar y parar el servicio.

Precondiciones: Ninguna

Flujo principal:

- El administrador de la red IMS configura el entorno inicial de aplicación
- El administrador publica el servicio
- El servicio muestra un mensaje de bienvenida para reportar que está disponible

Flujo de excepción:

E1: Si los archivos de descripción de la ontología no fueron encontrados en la ubicación especificada en la configuración, entonces se muestra un error notificando al administrador de dicha situación.

B.2 DIAGRAMA DE PAQUETES DEL CASO DE ESTUDIO

El diagrama de paquetes de diseño del gestor de perfiles de usuario se muestra en la Figura 2, cuyos componentes se describen a continuación.

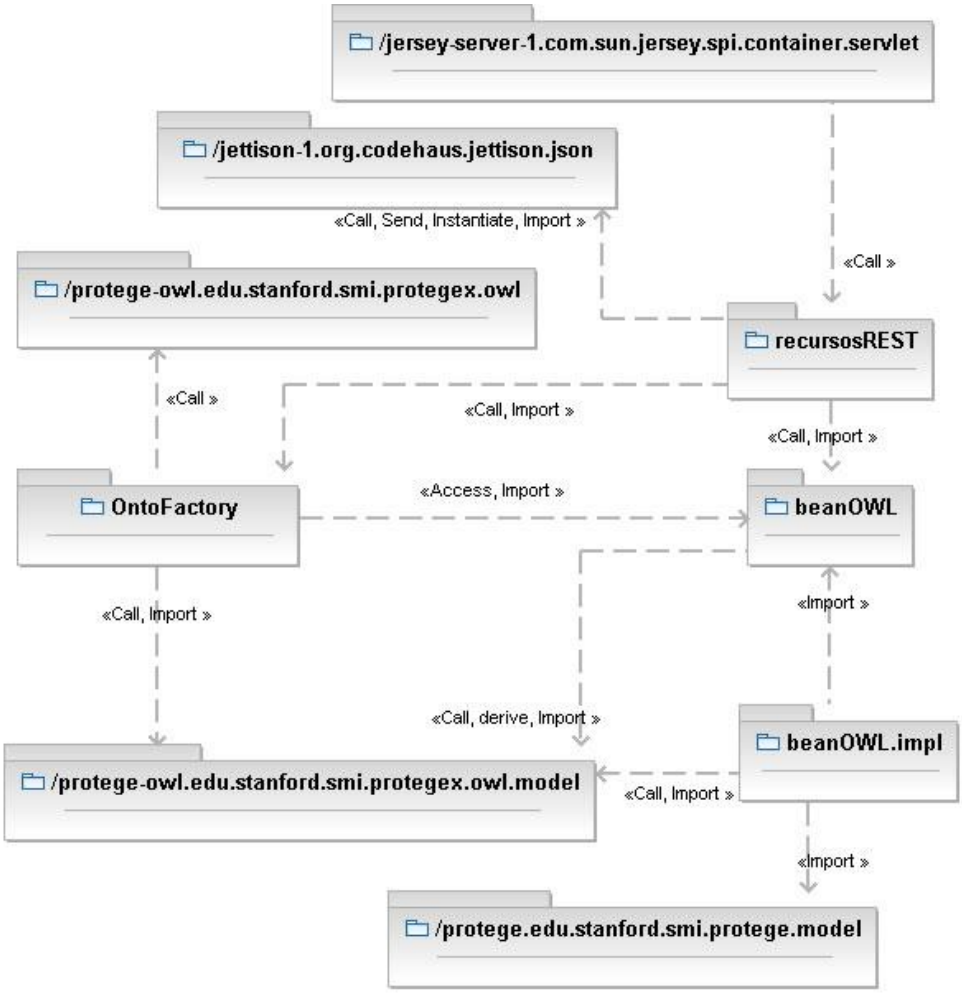

Figura 2 Diagrama de paquetes del gestor de perfiles de usuario.

com.sun.jersey.spl.container.servlet: Es un API de código abierto que basándose en el JAX-RS (*The Java API for RESTful Web Services, API Java para Servicios Web RESTful*), conocido como el JSR 311 (*Java Specification Request,* Solicitud de Especificación Java) que ofrece una implementación de referencia para la construcción de servicios Web RESTful. Este API permite crear una interfaz de comunicación que recibe las peticiones HTTP (*HyperText Transfer Protocol,* Protocolo de transferencia de hipertexto) provenientes de las aplicaciones IMS, las cuales son enviadas al correspondiente recurso REST (*REpresentation State Transfer*, Transferencia de Estado Representacional) para darle respuesta.

jettison-1.org.codehaus.jettison.json: Es una colección de APIs Java el cual permite leer y escribir en JSON (*JavaScript Object Notation,* Notación de Objetos de JavaScript), que es un formato ligero para el intercambio de información. Este permite de manera casi transparente para el desarrollador, la implementación de servicios web basados en JSON.

recursosREST: Contiene todas las clases que procesan las peticiones recibidas por medio de la interfaz de comunicación ofrecida por el servicio web RESTful. Estos recursos se valen usan el paquete OntoFactory y las interfaces definidas en el paquete beansOWL, para manipular la ontología.

protege-owl.edu.standford.sml.protegex.owl Es un librería de código abierto Java para los lenguajes ontológicos OWL y RDF(s). Este API provee las clases y los métodos para cargar, guardar, consultar, manipular y realizar razonamiento sobre modelos de datos en el lenguaje OWL. Adicionalmente, el API esta optimizado para la implementación de interfaces gráficas de usuario.

protege-owl.edu.standford.sml.protegex.owl.model: Es una librería de código abierto Java en el cual se especifican los modelos básicos para la manipulación de las clases, propiedades e instancias de una ontología en lenguaje OWL que la convierten en indispensable a la hora de utilizar el API Protégé OWL.

protege.edu.standford.sml.protege.model: Es una librería de código abierto Java en el cual se especifican los modelos básicos para la manipulación de las clases, propiedades e instancias que sirve de base para los APIS de Protégé OWL.

ontoFactory: Este paquete contiene la clase encargada de la construcción de objetos para el manejo de la ontología, basándose en las interfaces definidas en el paquete BeansOWL, lo cual hace innecesario el conocimiento del modelo ontológico que soporta, dado que se encarga de realizar todas las operaciones sobre objetos, aun desconociendo por completo de que tipo son.

beanOWL: Es el paquete que alberga todo el conjunto de interfaces definidas para la manipulación de cada una de las clases y propiedades con que cuenta la ontología de perfiles de usuario. Las interfaces aquí definidas obedecen las convenciones de nomenclatura de métodos, construcción, y comportamiento de la notación de JavaBean, lo cual facilita la manipulación de la ontología, haciendo innecesario el total conocimiento del modelo ontológico que se está soportando.

Implementación beanOWL (beanOWL.impl): En este paquete se realiza la implementación específica para protégé de las interfaces definidas en el paquete BeansOWL. En sí este paquete es quien usa las librerías del API de Protégé para el manejo de la ontología de perfiles de usuario.

Los paquetes de ontofactory, beansOWL, y la implementación de beansOWL fueron generados a partir de la ontología de perfiles de usuario, haciendo uso de las funcionalidades que ofrece Protégé para la generación automática de código para la manipulación de ontologías en el leguaje OWL.

B.3 DIAGRAMA DE CLASES DEL CASO DE ESTUDIO

En la Figura 3 se muestra el diagrama de clases del gestor de perfiles de usuario. En este diagrama se pretende mostrar las principales relaciones entre las diferentes clases implementadas y su relación con los paquetes que las contienen.

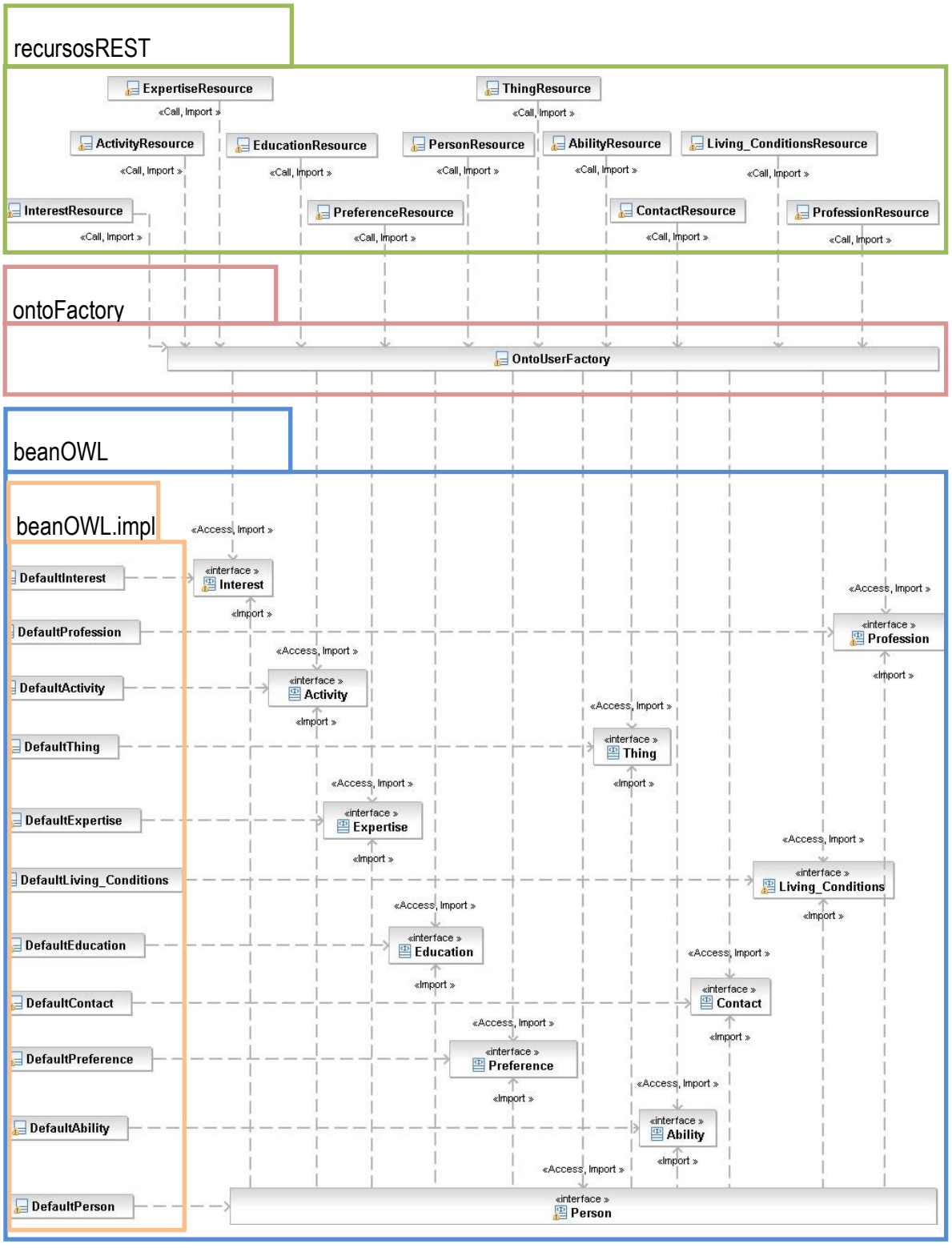

Figura 3 Diagrama de clases del gestor de perfiles de usuario

B.4 DESCRIPCIÓN DE LA ONTOLOGIA DE PERFILES DE USUARIO

Las ontologías se componen de instancias, clases, propiedades, restricciones y las relaciones entre cada uno de estos componentes. Las clases son el centro de la mayoría de las ontologías, dado que estas describen conceptos de un dominio y poseen subclases que representan conceptos que son más específicos que la superclase. Las propiedades llamadas también *slots* describen las características de las clases e instancias de las mismas. Las restricciones también denominadas facetas, o restricciones de rol son aplicadas a los *slots* [23]

La ontología de perfiles de usuario utilizada en este proyecto, se basa en el trabajo expuesto en [24], la cual fue adaptada para que fuese coherente con la información manejada en el perfil de usuario en el entorno de IMS, a la vez, que se modificó el lenguaje en el que estaba desarrollada a OWL.

En este apartado se explica cada una de los componentes que hacen parte de la ontología haciendo énfasis en sus clases y propiedades:

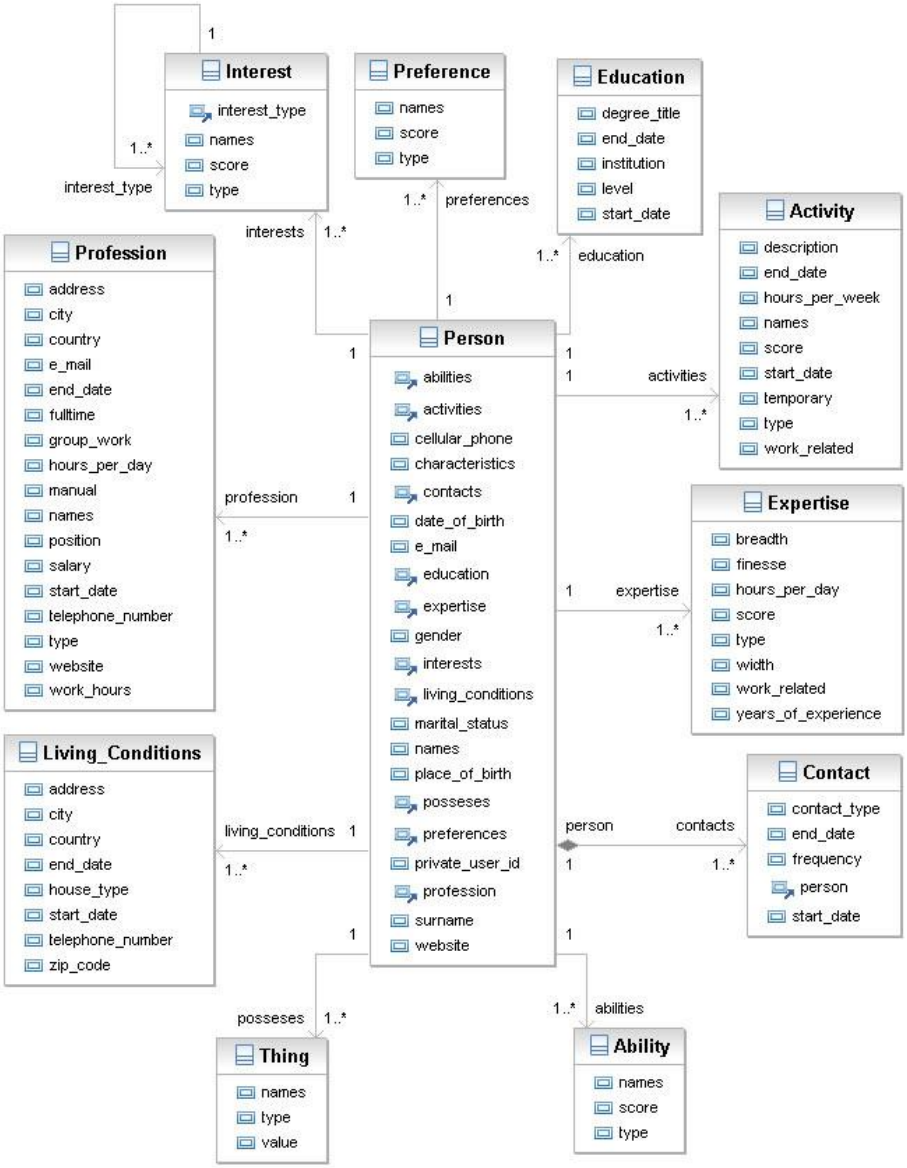

Figura 4 Diagrama de clases de la ontología de perfiles de usuario.

Nombre de la clase	Descripción de la clase
Person	Contiene la Información del Usuario básica como su nombre, fecha de nacimiento, el correo electrónico.
Ability	Habilidades y discapacidades mentales y físicas del usuario.
Living Conditions	Información relevante acerca del lugar de residencia y tipo de vivienda.
Contact	Otra persona, con la cual la persona esté relacionada, incluyendo parientes, amigos y colaboradores
Preference	Preferencias del usuarios, por ejemplo "le gustan los gatos", "le gusta el color azul", o "le desagrada la música clásica"
Interest	Intereses en pasatiempos o trabajos relacionados-Por ejemplo "interés en los deportes" o "interés en la cocina"
Activity	Actividades, o pasatiempo o trabajo relacionados. Por ejemplo "coleccionar estampillas" o "investigar artículos históricos"
Education Level	Nivel educativo, incluyendo por ejemplo los diplomas universitarios y lenguajes.
Profesion	La ocupación del usuario
Expertise	Incluye toda clase de experiencias, como la experiencia en el uso de computadoras.
Thing	Cosas del hogar o no que el usuario posee o relacionadas a estas, como un automóvil, una casa, un libro o una mascota.

Tabla 1 Clases de la ontología de perfil de usuario

Las propiedades de tipo objeto son aquellas que relacionan clases u objetos entre sí.

Propiedad	Descripción
abilities	Colección de habilidades de un usuario
activities	Colección de actividades realizadas por un usuario
contacts	Colección de contactos
education	Colección de educación de persona
expertise	Colección de experiencia de una persona
interest_type	Colección de tipo de interés de un Interés
interests	Colección de intereses de la persona
living_conditions	Colección de Condiciones de vida de una persona
person	Propiedad que hace alusión a una instancia de la clase persona
posseses	Colección de posesiones de la persona
preferences	Colección de preferencias de una persona
profession	Colección de profesiones de una persona

Tabla 2 Propiedades de tipo objeto de la ontología de perfiles de usuario

Las propiedades de tipo dato, son aquellas que describen datos simples dentro de la ontología.

Propiedad	Descripción
address	Dirección
breadth	Amplitud de conocimiento
cellular_phone	Teléfonos celulares
characteristics	Características
city	Ciudad
contact_type	Tipo de contacto
country	País
date_of_birth	Fecha de nacimiento
degree_title	Título de grado
description	Descripción
e-mail	Correos electrónicos
end_date	Fecha final en un intervalo de tiempo
finesse	Refinamiento
frequency	Frecuencia
fulltime	Para evaluar si un trabajo es de tiempo completo o medio tiempo.
gender	Género Masculino / Femenino
group_work	Para evaluar si es un trabajo en grupo o no
hours_per_day	Horas por día que se desarrolla una actividad
hours_per_week	Horas por semana que se desarrolla una actividad
house_type	Tipo que casa por ejemplo granja, apartamento.
institution	Nombre de la una institución
level	Hacer referencia al nivel de estudios: primaria, secundaria y universitaria
manual	Hace referencia a si es un trabajo manual o no.
marital_status	Estado civil de una persona
names	Nombre
place_of_birth	Lugar de nacimiento
position	Posición
private_user_id	Identidad privada del usuario
public_user_id	Colección de propiedades públicas de un usuario
salary	Salario
score	Puntuación
start_date	Fecha de inicio
surname	Apellido
telephone_numbe	Teléfonos fijos
type	Tipo
uses_of_the_computer	Uso del computador
value	Costo de un artículo
website	Sitios web
width	Ancho
work_hours	Horas de trabajo
work_related	Determinar si una actividad está relacionada con el trabajo o un pasatiempo.
years_of_experience	Años de experiencia
zip_code	Código zip

Tabla 3 Propiedades de los datos simples de la ontología de perfiles de usuario

ANEXO C CONFIGURACION DEL SDS PARA LA PROVICION DE SERVICIOS

Este anexo describe los pasos necesarios para la configuración de la plataforma IMS en la provisión de servicios. La información relacionada con la instalación y configuración del SDS se encuentran en [25] .

C.1 DESCRIPCIÓN DE LA PERSPECTIVA *PROVISIONING* DE ECLIPSE

Una perspectiva de Eclipse es una agrupación de vistas y editores que dan apoyo a una actividad completa del proceso de desarrollo software. La perspectiva *Provisioning* permite configurar la simulación del CSCF que provee el SDS, con el fin de administrar el trafico entre los clientes y los contenedores SIP. Esta perspectiva puede seleccionarse haciendo clic en los iconos de perspectiva del lateral izquierdo, eligiendo el menú "*Window>Open Perspective*" o seleccionado el menú *SDS>Server>Provisioning* como se muestra en la Figura 5.

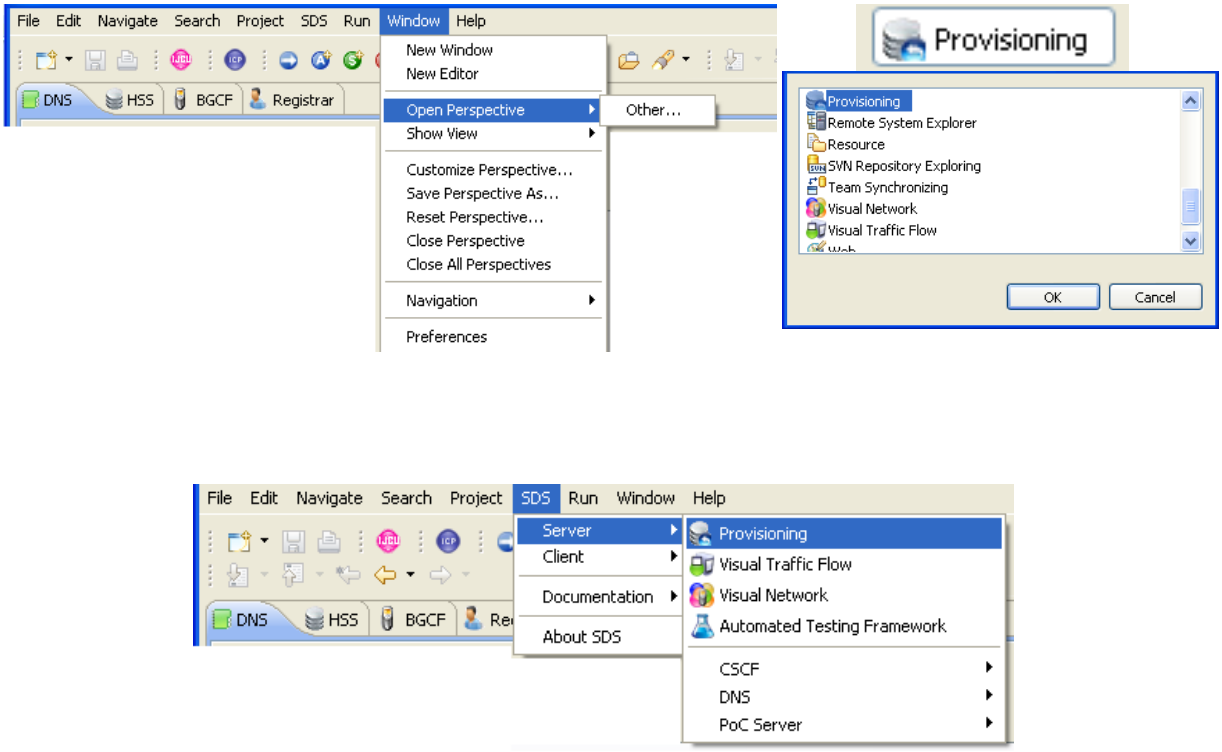

Figura 5 Selección de perspectiva Provisioning.

Esta perspectiva cuenta con cuatro pestañas DNS, HSS, BGCF y Registrar como se muestra en la Figura 6. El anexo C se centra en la configuración del DNS y el HSS. La funcionalidad BGCF es utilizada en el establecimiento de llamadas desde IMS hacia una red de circuitos conmutada, su descripción está por fuera del alcance de este documento, para más detalles remitirse a [26]. Por último la funcionalidad Registrar será descrita en el anexo E.

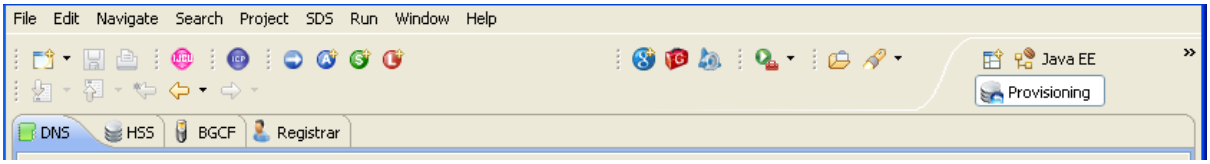

Figura 6 Perspectiva *Provisioning*.

C.1.1 Configuración del DNS

La pestaña DNS de la perspectiva *Provisioning* es utilizada en la definición de dominios virtuales para los iFC (*Initial Filter Criteria*, Criterios de Filtrado Inicial). Permite asignar a cada servidor de aplicación un nombre de dominio, de esta forma se pueden enviar todas las peticiones a un servidor de aplicación distinto simplemente con el cambio de la dirección IP para un nombre de dominio definido.

El SDS viene pre configurado con un DNS (*Domain Name Server*,Servidor de Nombres de Dominio) para mensajería IMS (IMS-M), Presencia y PGM (*Group List Management*,Gestion de Listas de Grupos), y PoC (*Push-to-Talk over Cellular*). Dentro del caso de estudio se creó un dominio llamado sip: sgpu.com. Todos los mensajes SIP serán enviados a este host. Los pasos para crear un nuevo dominio son los siguientes:

1. Seleccionar la pestaña SIP/SIPS Keys.
2. Oprimir el botón Add.
3. Ingresar los datos para los campos de la URI.
4. Ingresar los datos del record DNS.
5. Oprimir el botón Save para guardar los cambios.

La Figura 7 identifica los elementos necesarios asociados a los pasos anteriores para crear el nuevo dominio.

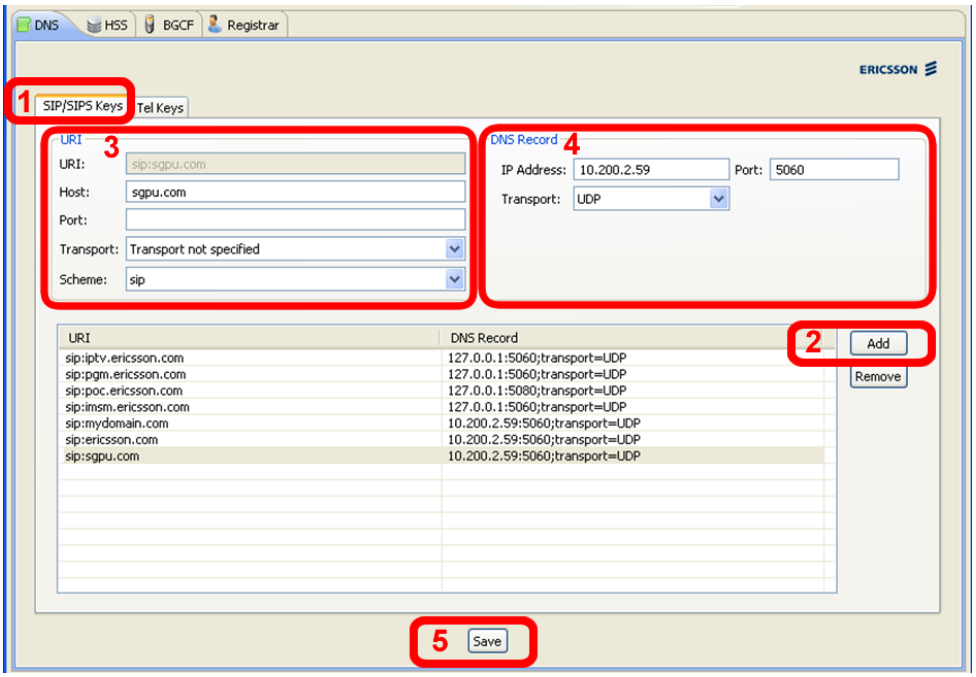

Figura 7 DNS Provisioning – adicionar un Nuevo registro DNS

C.1.2 Configuración del HSS

La configuración del HSS (*Home Subscriber Server*, Servidor de Suscriptor Local) involucra:

- La configuración del iFC y los SPT (*Service Point Triggers* , Activadores de Punto de Servicio).
- La configuración del Perfil de Servicio.
- La configuración del Perfil de Usuario.
- La configuración del PSI *(Public Service Identity,* Identidad de Servicio Público*)* (la cual no es abordada en este documento.)

La Figura 8 muestra cada una de las pestañas que conforman la funcionalidad ofrecida para el HSS.

Figura 8 Herramienta para la configuración del HSS.

C.1.2.1 La configuración del iFC

Un iFC puede ser descrito como un punto de entrada a un servidor de aplicaciones. El iFC puede tener uno o varios activadores (*triggers*), a su vez estos activadores hacen parte de un grupo. Los pasos para crear un nuevo iFC son los siguientes:

1. Seleccionar la pestaña HSS
2. Seleccionar la pestaña *Initial Filter Criteria*
3. Seleccionar la pestaña *Definition.* El SDS viene pre configurado para IMS-M, PGM, y PoC. Para el caso de estudio se creó un iFC llamado SGPU.
4. Presionar el botón *Add* para crear un nuevo iFC.
5. En la sección Initial *Filter Criteria*, establecer un nombre para el iFC.
6. En la sección *Aplication Server*, establecer la dirección del servidor utilizando el dominio creado anteriormente.
7. Presionar el botón *Save* para guardar los cambios.

La Figura 9 identifica los elementos necesarios asociados a los pasos anteriores para crear el nuevo iFC.

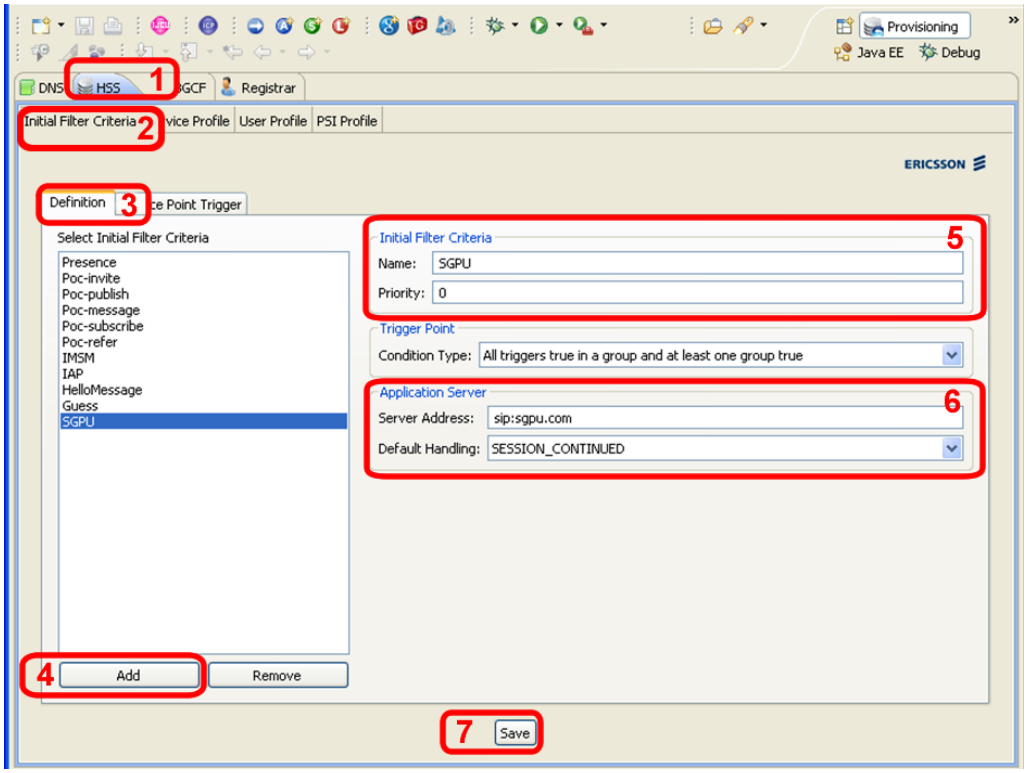

Figura 9 HSS Provisioning – Crear un nuevo iFC.

C.1.2.2 Activadores de Punto de Servicio

Los SPT son condiciones lógicas basadas en características de los mensajes SIP, como:

- El valor del *Request URI.*
- El método de petición SIP (ej.: INVITE, OPTIONS, SUBSCRIBE, etc.).
- La presencia o ausencia de alguna cabecera SIP.
- La concordancia parcial o completa entre el contenido de cualquier cabecera SIP.
- El *sesión case* (ej.: si la petición SIP es originada por el usuario servido, direccionada al usuario registrado servido, o direccionada al usuario no registrado servido).
- Descripción de la sesión (ej.: cualquier concordancia parcial o completa)

Los pasos para crear un nuevo SPT son los siguientes:

1. Seleccionar la pestaña *Service Point Trigger*.
2. Asegurarse de escoger el IFC adecuado.
3. Presionar el botón *Add*.
4. Ingresar el nombre del nuevo SPT.
5. En el campo *Trigger Definition* , hacer la selección del tipo de *trigger* , su valor y sus condiciones.
6. Presionar el botón *Save* para guardar los cambios.

La Figura 10 identifica los elementos necesarios asociados a los pasos anteriores para crear el nuevo SPT.

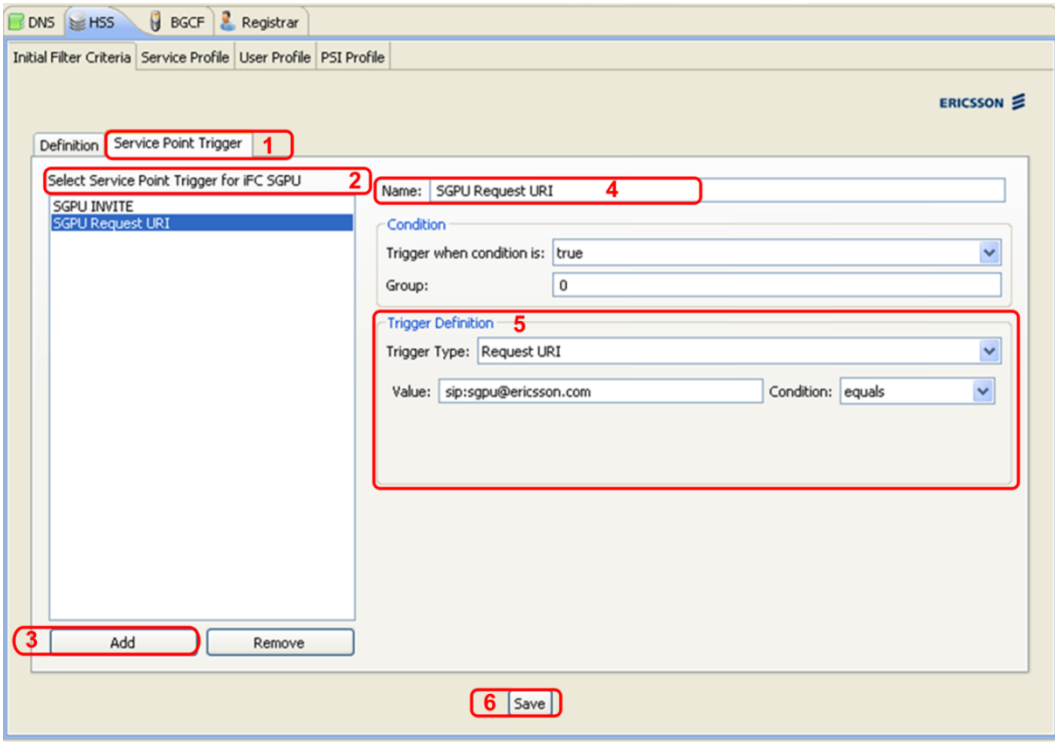

Figura 10 HSS Provisioning – Crear un nuevo SPT.

C.1.2.3 Configuración del Perfil del Servicio

Cada perfil de servicio es asignado a un usuario particular, este perfil puede estar conformado por varios iFCs. Por ejemplo, cuando un mensaje llega al CSCF desde un usuario, el CSCF busca el perfil de servicio del usuario e invoca a los servidores de aplicación basados en este perfil.

Los pasos para crear un nuevo Perfil de Servicio son los siguientes:

1. Seleccionar la pestaña *Service Profile* dentro del HSS. El SDS viene pre configurado para IMS-M, PGM y *PoC*. Para el servicio de prueba se creó un perfil de servicio llamado SGPU profile.
2. Presionar el botón Add. Esto crea un nuevo Perfil de Servicio.
3. En la sección *Service Profile* establecer el nombre del perfil de servicio y seleccionar de una lista los iFCs que tenga asociados.
4. Presionar el botón Save para guardar los cambios.

La Figura 11 identifica los elementos necesarios asociados a los pasos anteriores para crear el nuevo Perfil de Servicio.

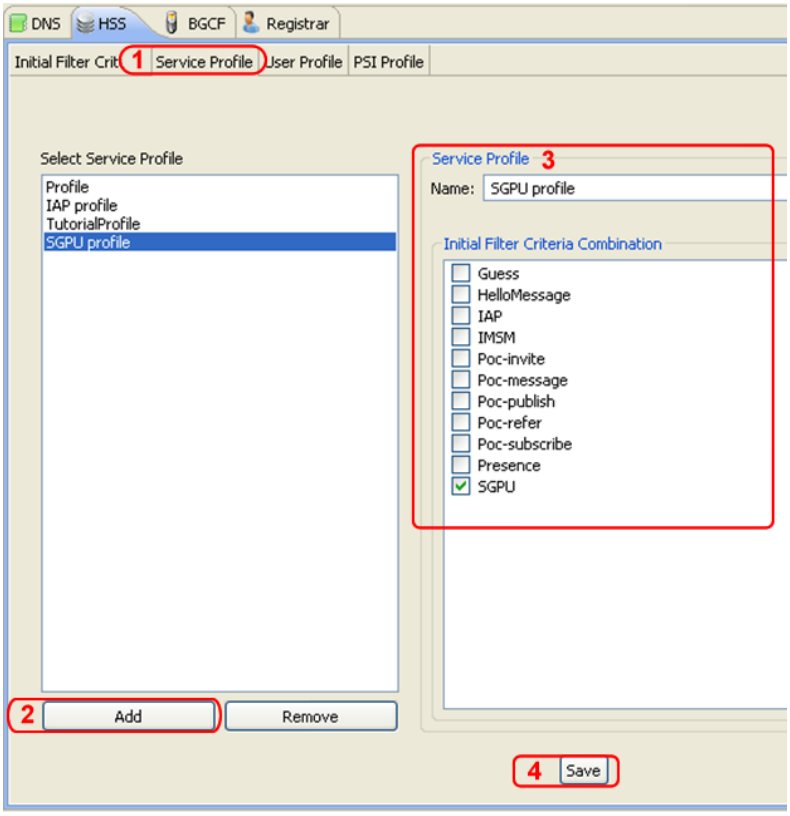

Figura 11 HSS Provisioning – Crear un nuevo Perfil de Servicio.

C.1.2.4 Configuración del Perfil de Usuario

Antes de que el servidor de aplicación pueda ser activado, el usuario debe estar almacenado en el HSS. La funcionalidad Perfil de Servicio permite hacer esto. El SDS viene pre configurado por defecto con los usuarios Alice y Bob. Para el servicio de prueba se creó un nuevo usuario siguiendo los siguientes pasos.

1. Seleccionar la pestaña *User Profile* dentro del HSS.
2. Presionar el botón *Add.* De esta forma se crea un nuevo perfil de usuario con los datos obligatorios por defecto.
3. Establecer los datos para el campo *Public User ID*, *Private User ID*, *Password* y el *Service Profile* al que está asociado. Los otros datos no son requeridos para el propósito de este anexo. Para más información sobre estos datos referirse al [26].
4. Presionar el botón *Save*, para guardar todos los cambios.

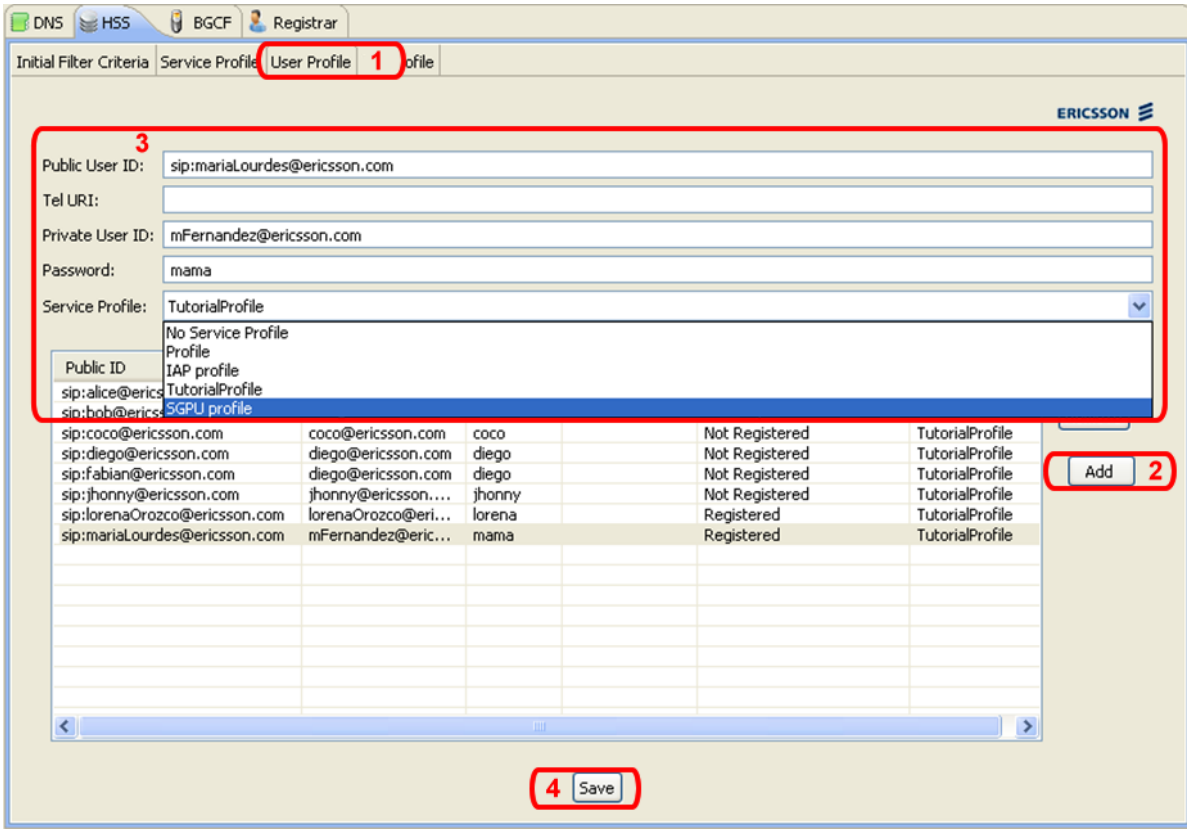

Figura 12 HSS Provisioning – Crear un nuevo Perfil de Usuario.

C.1.3 Configuración del CSCF

Para la configuración del CSCF el SDS proporciona una perspectiva llamada Visual Network, la cual permite configurar y poner en marcha el servidor DNS el CSCF, el emulador para un dispositivo móvil y el servidor PoC.

Esta perspectiva puede seleccionarse haciendo clic en los iconos de perspectiva del lateral izquierdo, eligiendo el menú "*Window>Open Perspective*" o seleccionado el menú *"SDS>Server>Visual Network"* como se muestra en la Figura 13.

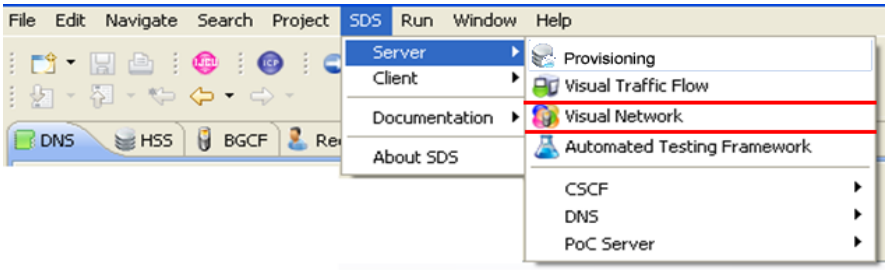

Figura 13 Selección de la perspectiva Visual Network.

Esta perspectiva contiene una paleta que permite agregar o eliminar entidades de red en un panel, con solo arrastrar y soltar. La Figura 14 muestra los elementos de esta perspectiva.

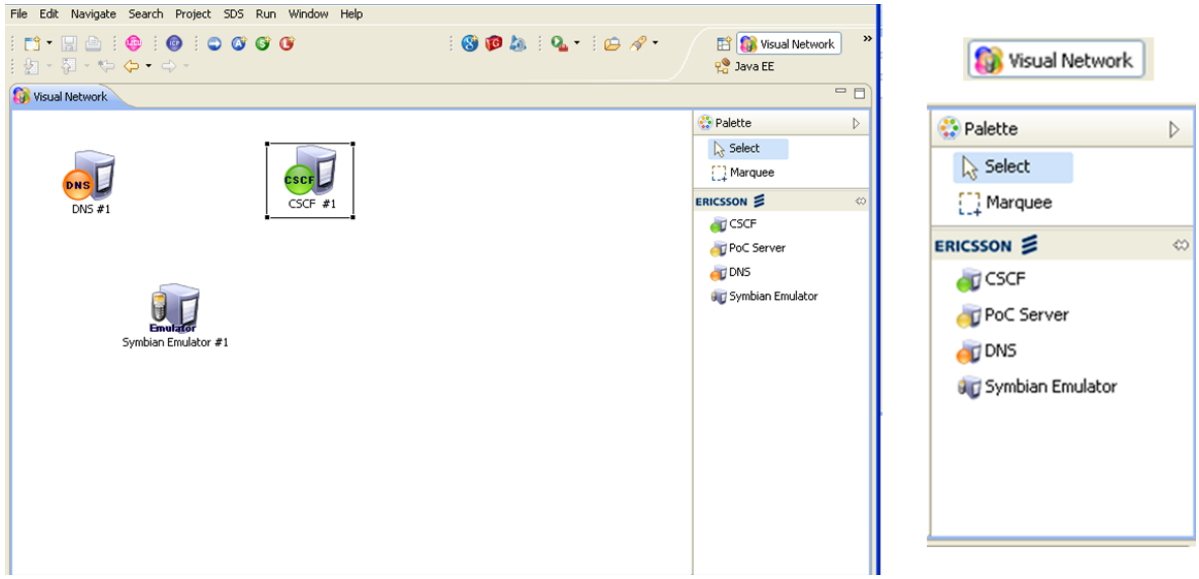

Figura 14 Perspectiva Visual Network.

Dentro de las opciones configurables del CSCF se encuentran: Transporte, *Roaming*, Autenticación y Registro. Para acceder a las mismas se debe seleccionar el CSCF dentro del panel de la perspectiva, dar clic derecho y seleccionar la opción *Properties* como se muestra en la Figura 15. De esta misma forma se puede seleccionar la opción *Star* que pone en marcha el servidor.

Figura 15 Opciones para la entidad CSCF.

La Figura 16 muestra el panel preferencias y la opción *Transport* seleccionada, la cual permite establecer la dirección IP y los puertos UDP y TCP para el S-CSCF, P-CSCF y el I-CSCF. Las opciones *Roaming*, *Authentication* y Registrar están por fuera del los propósitos de este anexo, para mayor información ver [26].

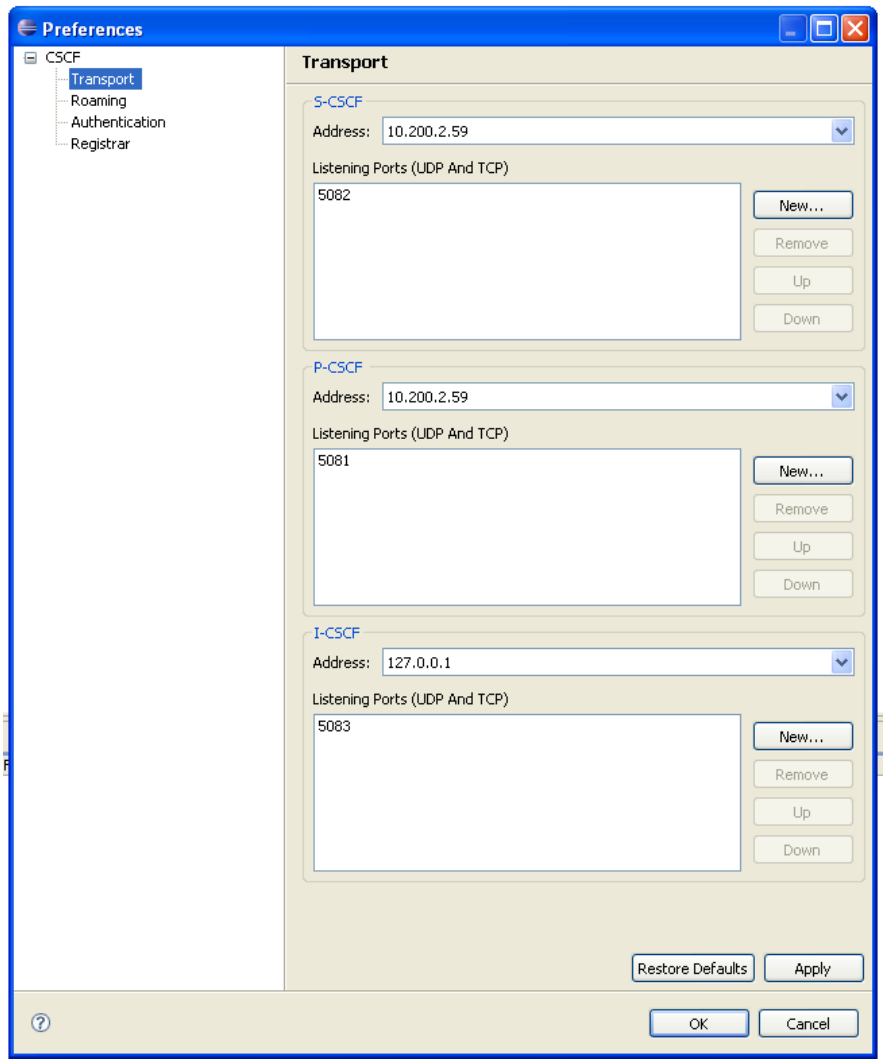

Figura 16 Preferencias de configuración del CSCF

ANEXO D IMPLEMENTACIÓN DEL SERVICIO DE PRUEBA

Este anexo incluye el modelo de casos de uso, los diagramas de paquetes y de clases, relacionadas con la implementación del servicio desplegado sobre el servidor de aplicación Saifin en el dominio IMS, el cual interactúa con la arquitectura de referencia.

D.1 MODELO DE CASOS DE USO EXTENDIDOS

D.1.1 Caso de uso Aceptar invitación

Iniciador	Cliente IMS
Propósito	Aceptar o rechazar el inicio de sesión.
Resumen	Para iniciar una sesión el cliente realiza una petición con el método INVITE en donde indica con qué usuario (o recurso) quiere establecer la sesión. El servidor responde ya sea rechazando o aceptado esa petición en una serie de respuestas. Las respuestas llevan un código de estado que brindan información acerca de si las peticiones que fueron resueltas con éxito o si se produjo un error. La petición inicial y todas sus respuestas constituyen una transacción.

Precondiciones: Debe estar creada en la plataforma IMS el perfil de usuario y su perfil de servicio.

Flujo principal:

- El cliente IMS se registra en la plataforma IMS
- El cliente envía la petición de inicio de sesión
- El servicio acepta la petición
- Se establece la sesión entre las dos entidades.

Flujo de excepción:

E1: En caso de que el usuario no se encuentra registrado en la plataforma IMS (no tiene un perfil de usuario), el sistema muestra un error indicando lo sucedido.

E2: En el caso de que el servicio no se encuentre desplegado el sistema muestra un error indicando lo sucedido.

E3: En caso de no ser establecida la sesión el sistema muestra el error sucedido.

D.1.2 Caso de uso Confirmar recepción de mensaje

Iniciador	Cliente IMS
Propósito	Confirmar que un mensaje o un conjunto de mensajes han llegado correctamente.
Resumen	El servidor de aplicación responde de manera afirmativa con un ACK informando al cliente sobre la recepción de un mensaje.

Precondiciones: Debe estar establecida la sesión entre el cliente y el servicio.

Flujo principal:

- El cliente envía un mensaje al servicio con una solicitud
- El servicio atiende el mensaje y responde con un ACK.
- El cliente es enterado de la recepción exitosa de mensaje.

Flujo de excepción:

E1: En caso de que el servicio no responda positivamente después de un periodo de tiempo, el sistema muestra un error indicando lo sucedido.

D.1.3 Caso de uso Finalizar Sesión

Iniciador	Cliente IMS
Propósito	Terminar la sesión establecida entre el cliente y el servidor de aplicación.
Resumen	Al terminar la sesión, el agente de usuario de la parte que terminó la sesión, envía hacia la otra parte una petición con el método BYE. Cuando es recibido se genera una respuesta con el código de estado correspondiente.

Precondiciones: el usuario debe estar registrado y debe existir una sesión.

Flujo principal:

- El usuario sale de la aplicación
- El cliente envía una petición al servicio con el método BYE.
- El servicio responde con el código correspondiente.

D.1.4 Caso de uso Gestionar Mensajes.

Iniciador	Cliente IMS
Propósito	Analizar las peticiones realizada por el cliente y solicitar los recursos pertinentes al gestor de perfiles de usuario.
Resumen	El cliente envía diferentes solicitudes (crear, eliminar, actualizar, recuperar) al servicio para que sean realizadas sobre la información de perfil de usuario, el servicio las analiza y establece un canal de comunicación para reenviar las solicitudes al Gestor y obtener una respuesta que reenvía al cliente en formato JSON.

Precondiciones: el usuario debe estar registrado y debe existir una sesión.

Flujo principal:

- El cliente envía una solicitud al servicio.
- El servicio analiza el mensaje y obtiene los datos del mismo.
- El servicio establece el canal de comunicación.
- El servicio invoca la operación solicitada al gestor.
- El servicio retorna un mensaje al cliente.

Flujo de excepción:

E1: Si el gestor retorna un fallo, error o excepción el servicio retorna un mensaje informando sobre lo sucedido.

D.1.5 Caso de uso Crear Cliente REST

Iniciador	Servicio IMS
Propósito	Acceder a los recursos web RESTful que permiten la interacción entre el cliente y el perfil de usuario.
Resumen	Una vez una solicitud se ha establecido, el servicio utiliza el cliente REST para acceder a los recursos web, de esta forma se reenvían la solicitudes realizadas por el cliente al Gestor de perfiles el cual gestiona los datos y entrega una respuesta.

Precondiciones: El servicio ha recibido una solicitud para gestionar información de perfil de usuario.

Flujo principal:

- EL servicio crea un cliente REST por defecto.
- El servicio establece la URL base del recurso web.
- El servicio crear una instancia del recuso web utilizando la URL base.
- El servicio invoca la operación solicitad al gestor.

Flujo de excepción:

E1: Si el canal no puede ser creado por un fallo, error o excepción el servicio retorna un mensaje informando sobre lo sucedido.

D.1.6 Caso de uso Crear Recurso

Iniciador	Servicio IMS
Propósito	Gestionar la solicitud del cliente para crear un recurso en la base de conocimiento.
Resumen	El servicio analiza la petición del cliente, obtiene la información necesaria para crea el recurso, obtiene la identidad pública de usuario, establece un canal de comunicación y utiliza el cliente REST para acceder a los recursos web.

Precondiciones: el usuario debe estar registrado y debe existir una sesión.

Flujo principal:

- El servicio obtiene la identidad pública de usuario.
- El servicio le da formato JSON a los datos que se envían al Gestor de perfiles.
- El servicio establece el canal.
- El servicio invoca al recurso y envía los datos de la solicitud junto con la operación que se desea realizar.

Flujo de excepción:

E1: Si el gestor retorna un fallo, error o excepción, el servicio retorna un mensaje informando sobre lo sucedido.

D.1.7 Caso de uso Eliminar Recurso

Iniciador	Servicio IMS
Propósito	Gestionar la solicitud del cliente para eliminar un recurso en la base de conocimiento.
Resumen	El servicio analiza la petición del cliente, obtiene la identificación del recurso que se desea eliminar, obtiene la identidad pública de usuario, establece un canal de comunicación y utiliza el cliente REST para acceder al recurso que se desea eliminar.

Precondiciones: el usuario debe estar registrado y debe existir una sesión.

Flujo principal:

- El servicio obtiene la identidad pública de usuario.
- El servicio establece el canal.
- El servicio invoca al recurso y envía los datos de la solicitud junto con la operación que se desea realizar.

Flujo de excepción:

E1: Si el gestor retorna un fallo, error o excepción el servicio retorna un mensaje informando sobre lo sucedido.

D.1.8 Caso de uso Actualizar Recurso

Iniciador	Servicio IMS
Propósito	Gestionar la solicitud del cliente para actualizar un recurso en la base de conocimiento.
Resumen	El servicio analiza la petición del cliente, obtiene la identificación del recurso que se desea, obtiene la identidad pública de usuario, actualiza y establece un canal de comunicación y utiliza el cliente REST para acceder al recurso que se desea actualizar.

Precondiciones: el usuario debe estar registrado y debe existir una sesión.

Flujo principal:

- El servicio obtiene la identidad pública de usuario.
- El servicio establece el canal.
- El servicio invoca al recurso y envía los datos de la solicitud junto con la operación que se desea realizar.

Flujo de excepción:

E1: Si el gestor retorna un fallo, error o excepción el servicio retorna un mensaje informando sobre lo sucedido.

D.1.9 Caso de uso Recuperar Recurso

Iniciador	Servicio IMS
Propósito	Gestionar la solicitud del cliente para recuperar un recurso de la base de conocimiento
Resumen	El servicio analiza la petición del cliente, obtiene la identificación del recurso que se desea recuperar, obtiene la identidad pública de usuario, establece un canal de comunicación y utiliza el cliente REST para acceder al recurso que se desea actualizar.

Precondiciones: el usuario debe estar registrado y debe existir una sesión.

Flujo principal:

- El servicio obtiene la identidad pública de usuario.
- El servicio establece el canal.
- El servicio invoca al recurso y envía los datos de la solicitud junto con la operación que se desea realizar.

Flujo de excepción:

E1: Si el gestor retorna un fallo, error o excepción el servicio retorna un mensaje informando sobre lo sucedido.

D.2 DIAGRAMA DE PAQUETES DEL SERVICIO

El diagrama de paquetes de diseño del servicio se muestra en la Figura 17, los componentes del mismo se describen a continuación.

El paquete Jsr311.api.1.javax.ws.rs.core: Contiene todas las librerías que permiten crear el canal de comunicación para el envió y recepción de datos entre el dominio IMS y Gestor de perfiles de usuario.

El paquete Ssa.api.javax.servlet.sip: Contiene las librerías estándar utilizado para implementar aplicaciones en Java que puedan procesar peticiones SIP.

El paquete restCom: Este paquete contiene las clases encargadas de implementar el canal para el intercambio de recursos web y el cliente RESTfull.

El paquete service : Contiene las clases encargadas de implementar el Servlet SIP que atiende a las solicitudes del cliente IMS.

El paquete jersey.client-1.com.sun.jersey.api.client contiene las librerías utilizadas en la implementación del un cliente REST, el cual puede acceder a un servicio RESTful utilizando el canal creado con el paquete jsr311.api.1.javax.ws.rs.core .

El paquete jettison-1.org.codehaus.jettison.json: Es una colección de APIs Java la cual permite leer y escribir en JSON (*JavaScript Object Notation,* Notación de Objetos de JavaScript), que es un formato ligero para el intercambio de información.

El paquete javaee.javax.servlet: Contiene las clases e interfaces estándar para el desarrollo de Servlets.

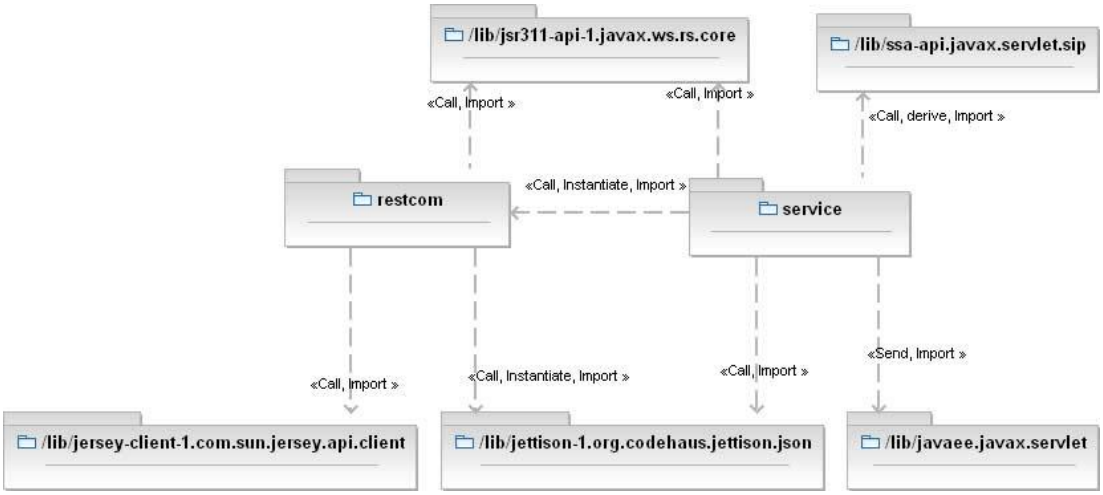

Figura 17 Diagrama de paquetes del Servicio de Prueba.

D.3 DIAGRAMA DE CLASES DEL SERVICIO.

La Figura 18 muestra las clases que fueron utilizadas para el desarrollo del servicio.

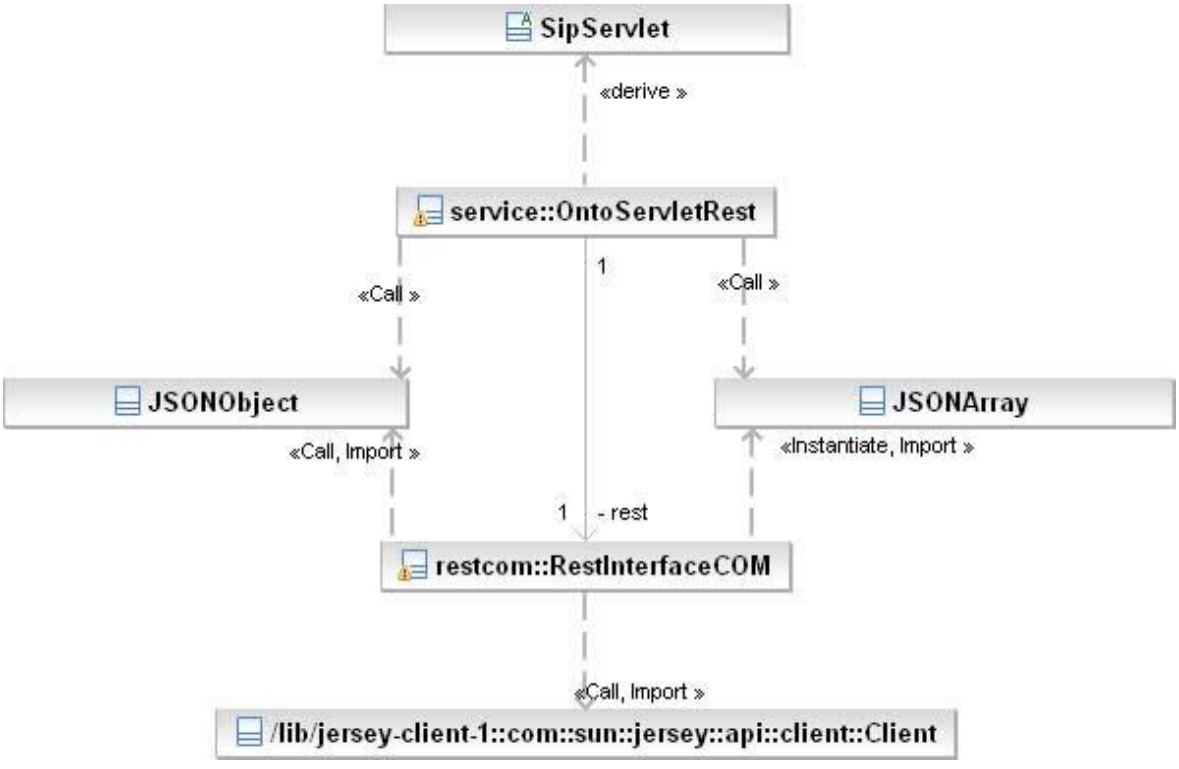

Figura 18 Diagrama de clases del Servicio de Prueba.

En el diagrama anterior, la clase OntoServletRest del paquete service, extiende de la clase SipServlet, es la encargada del establecimiento de la sesión SIP, el análisis de los mensajes, la invocación de recursos y el envió de respuestas al cliente. Por otro lado, la clase RestInterfaceCOM del paquete restCom es la encargada de la implementación del cliente REST para el establecimiento de un canal HTTP que permita la comunicación entre el dominio IMS y el Gestor de perfiles de usuario. Las clases JSONArray y JSONObject son empleadas para dar formato JSON a la información enviada. La clase Client de Jersey es la implementacion del cliente REST utilizada por la clase RestInterfaceCOM.

ANEXO E DESARROLLO DE CLIENTES IMS CON SDS

El objetivo de este anexo es mostrar los detalles de configuración y desarrollo de la aplicación cliente IMS que interactúa con servicio de prueba. Este anexo hace una descripción de la plataforma ofrecida por el SDS para la generación de clientes SIP y muestra el funcionamiento de las clases que fueron utilizadas.

E.1 PLATAFORMA PARA EL DESARROLLO DE CLIENTES IMS

El SDS proporciona la herramienta ICP (*IMS Client Platform*, Plataforma Cliente IMS) la cual permite el desarrollo de aplicaciones cliente SIP tanto de escritorio como móviles (con soporte para dispositivos con sistema operativo Symbian), proporcionado un API de alto nivel. El ICP corre como un servicio de fondo (*background*), sobre los dispositivos y es transparente para las aplicaciones cliente.

Los pasos para la creación, despliegue y pruebas de una aplicación cliente utilizando el ICP se encuentran en [26].

E.1.1 Configuración del Perfil de usuario ICP

El ICP permite establecer la información del perfil de usuario para el registro en el CSCF, mediante una aplicación de configuración. Esta aplicación se encuentra por defecto en la carpeta del panel de control de Windows como se muestra en la Figura 19. El proceso que se describe a continuación se debe realizar para que un usuario pueda acceder aun servicio. Sin estos pasos es imposible que la aplicación del cliente pueda funcionar.

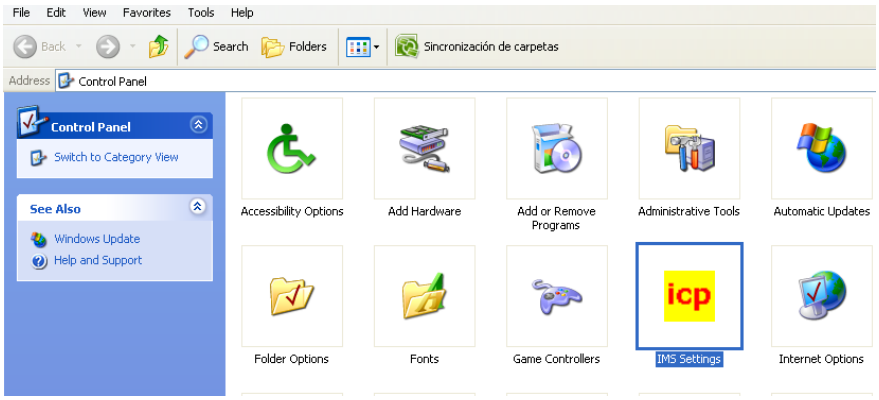

Figura 19 Panel de control – ICP Settings

La interfaz grafica del ICP *Settings* se muestra en la Figura 20, a través de esta se puede realizar el registro del usuario, siguiendo los siguientes pasos:

1. Seleccionar la pestaña *Profile Manager.*
2. Seleccionar en el campo *Management* la opción IMSSettings.
3. Presionar el botón *Configuration*.

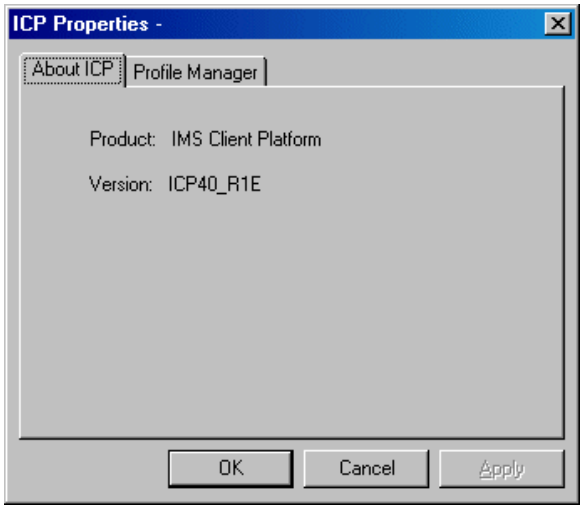

Figura 20 Interfaz gráfica del ICP *Settings*

Los elementos necesarios para los pasos anteriores se pueden observar en la Figura 21.

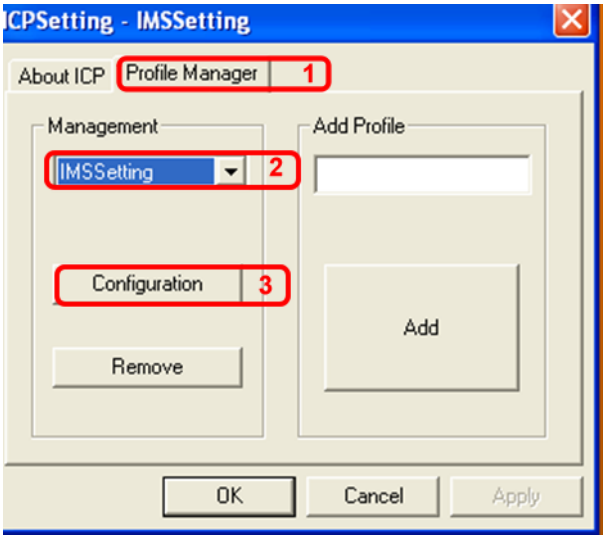

Figura 21 Selección del perfil IMS.

A continuación se establecen los valores para la dirección IP y el puerto del P-CSCF al que se desea acceder, seleccionado la pestaña Server addresses, como se muestra en la Figura 22. Los valores que vienen por defecto son: la dirección IP del localhost y el puerto 5081.

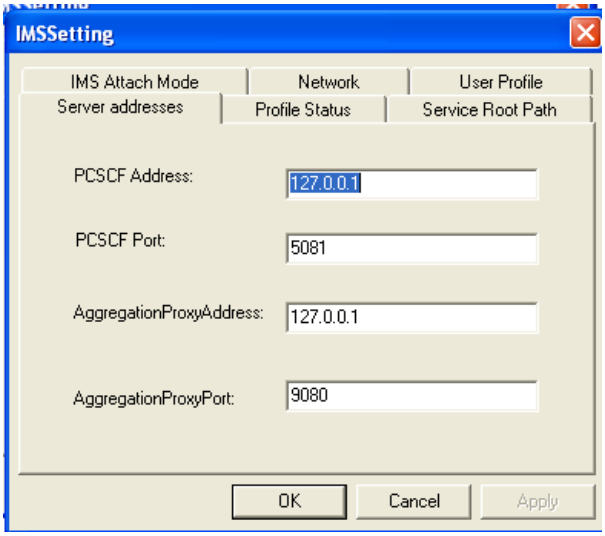

Figura 22 Configuración de dirección del P-CSCF.

El siguiente paso es el ingreso de los datos del perfil de usuario para el registro en la plataforma IMS, esto se hace a través de la pestaña User Profile, la cual pide la información del Public User ID, el Private User ID y el Password. Para cargar estos datos se debe presionar el botón Apply como se muestra en la Figura 23.

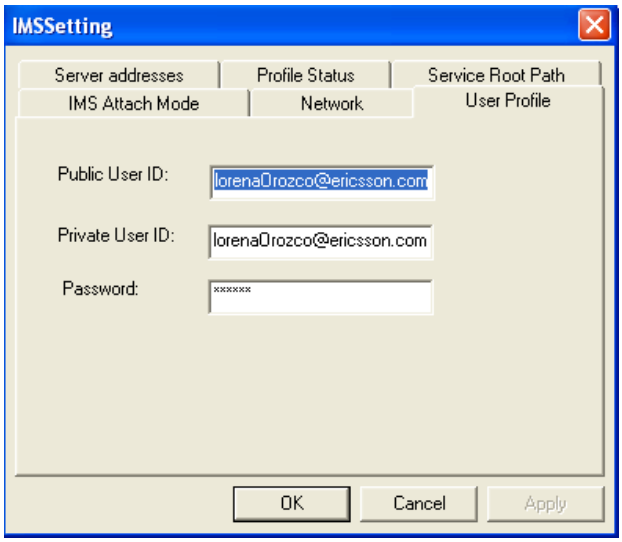

Figura 23 Datos para el registro del Usuario.

El último paso es realizar el registro del perfil de usuario, para esto se selecciona la pestaña Profile Status, se presiona en primera instancia el botón Stop Profile y luego el botón Star Profile, como se muestra en la Figura 24. Estos pasos permiten registrar un usuario para luego establecer una sesión con los servicios que tenga permitidos.

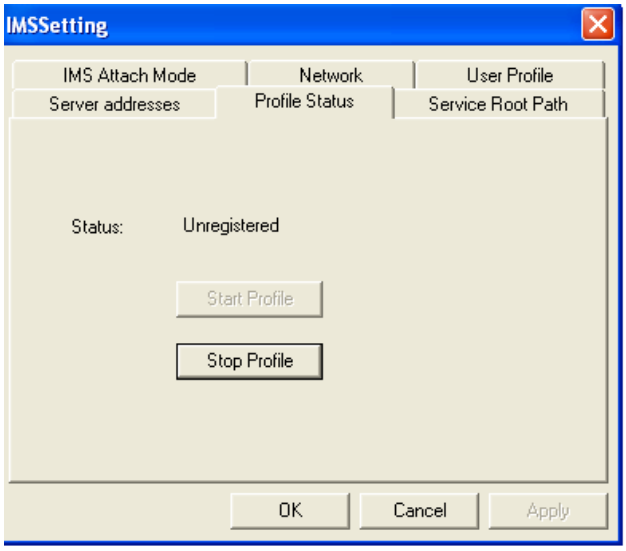

Figura 24 Registro del Usuario.

Al terminar se oprime el botón OK.

E.1.2 Adaptadores ICP

Las aplicaciones cliente ICP requieren de adaptadores que permiten el registro, inicio de sesión, envío de mensajes, entre otros. Uno o varios de estos adaptadores son necesarios para que una aplicación cliente funcione efectivamente. En el caso de la aplicación cliente del servicio de prueba se utilizaron las siguientes clases adaptadoras:

- *PlatformAdapter.java*
  Implementa la interfaz ICP *IPlatformListener,* la cual es requerida para crear un IProfile.
- *ProfileAdapter.java*
  Implementa la interfaz ICP *IProfileListener,* la cual es requerida para crear un IService.
- ServiceAdaper.java
  Implementa la interfaz ICP *IServiceListener,* la cual es requerida para crear un ISession.
- SessionAdapter.java
  Implementa la interfaz ICP *ISessionListener,* para capturar los siguientes eventos implementados por la aplicación cliente del servicio de prueba.
  - processError
  - processSessionStartFailed
  - processSessionMessage

Las interfaces IProfile, IService y ISession son descritas en las siguientes secciones.

E.1.3 Creación de un IProfile y supervisión de estado del ICP

IProfile es la interfaz principal de la función ICP. Esta proporciona los métodos para las distintas operaciones ICP. Una aplicación cliente IMS debe utilizar esta interfaz para registrarse en el ICP, antes de que cualquier sesión pueda ser creada. La aplicación cliente también puede supervisar el estado del ICP utilizando el método getState(). Sin embargo primero se debe crear un objeto IPlatform ya que sobre este se basa el IProfile.

Algo para tener en cuenta es que sólo un objeto IPlatform debe ser creado en la aplicación cliente. Por otro lado, múltiples IProfile pueden ser creados basados en un mismo objeto IPlatform. La Figura 25 muestra el diagrama de secuencias de la creación del IProfile y la supervisión del estado del ICP y a continuación se hace una breve explicación.

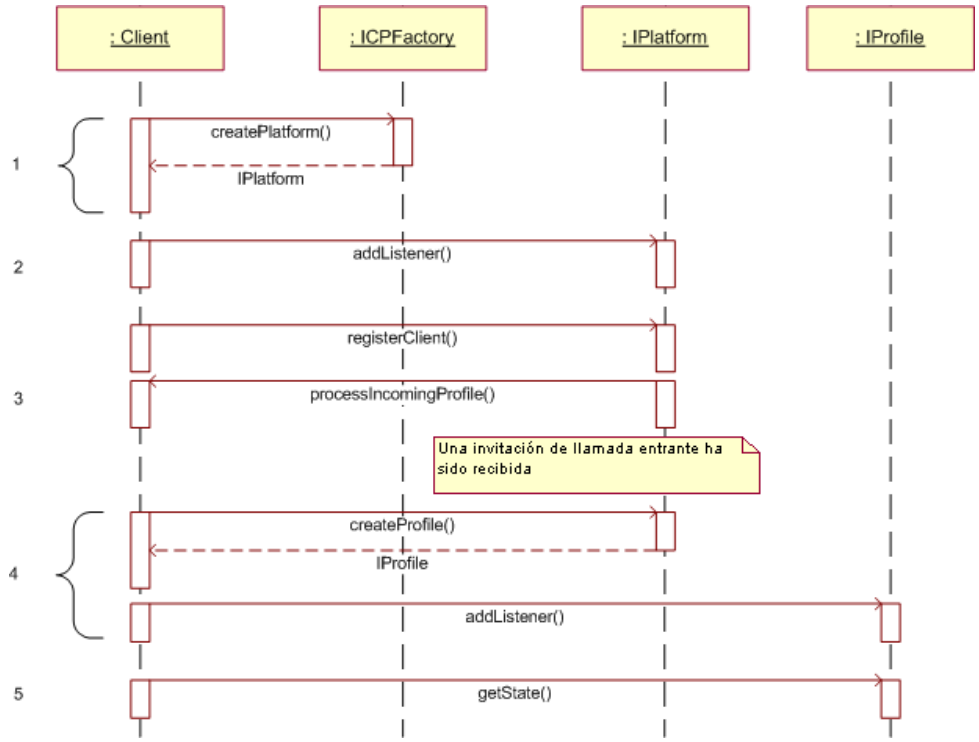

Figura 25 Creación de un IProfile y supervisión de estado del ICP.

1. El Cliente crea un objeto de la clase iPlatform llamando al método iPlatform e implementa la interfaz IPlatformListener.

2. El cliente se registra en el ICP llamando al método registerClient(String aApplicationName)

3. Dado que varios perfiles pueden haber sido creados anteriormente, se hace una llamada al método processIncomingProfile () el cual permite determinar que perfil está asociado con el Cliente A.

4. El Cliente crea un objeto de iProfile llamando al método createProfile (String aProfileName), e implementa la interfaz IProfileListener.

5. El cliente monitorea el estado de ICP llamando al método getState ().

E.1.4 Creación de un IService.

IService es una interfaz utilizada para:

- manipular el comportamiento paging-mode,
- el envío de opciones para descubrir otras capacidades,

- la suscripción a ciertos eventos,
- la publicación del estado de un evento,
- el envío de peticiones REFER,
- el envío de mensajes, entre otros.

Para poder realizar estas acciones se debe crear un objeto IService. Una aplicación cliente puede crear y controlar más de un IService simultáneamente.

En la Figura 26 se muestra el diagrama de secuencias de la creación de un IService, el cual se describe a continuación.

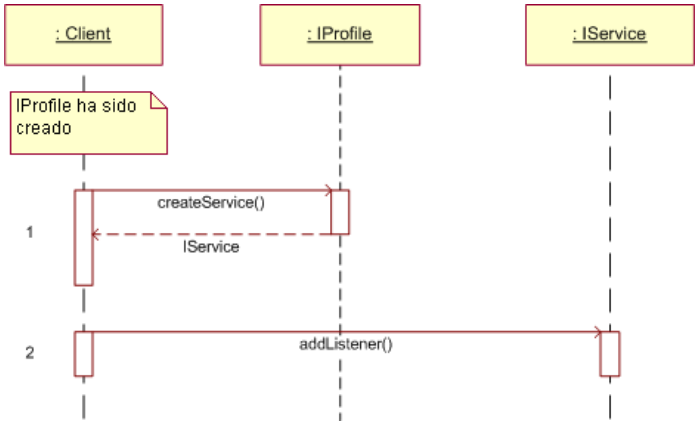

Figura 26 Creando un IService

1. El cliente crea un objeto de la clase IProfile y luego crea un IService llamando al método CreateService (String aServiceId, aSubfeature String) del IProfile.
2. El cliente implementa el IServiceListener para monitorear los eventos de *callback* del IService.

E.1.5 Creación un ISession

ISession representa una sesión genérica que ayuda a implementar las funciones estándar de sesión. Una aplicación cliente puede crear y controlar más de un ISession simultáneamente. En la Figura 27 se muestra el diagrama de secuencia para la creación de una ISession, el cual se describe a continuación.

1. Tanto el Cliente A como el cliente B han creado un IService. El IService del Cliente A crea un ISession. El Cliente A inicia una sesión genérica mediante una llamada al método start () y envía una invitación al Cliente B.

2. El Cliente B recibe el evento de *callback* proceessIncomingSession() con un objeto ISession, Cliente B implementa un ISessionListener para monitorear los eventos de *callback* del ISession. Entonces Cliente B recibe el evento de *callback* processInvitation() que contiene la descripción de la sesión. Este evento de *callback* notifica al cliente B que una invitación a iniciar sesión ha sido recibida. El Cliente A puede aceptar o rechazar la invitación.

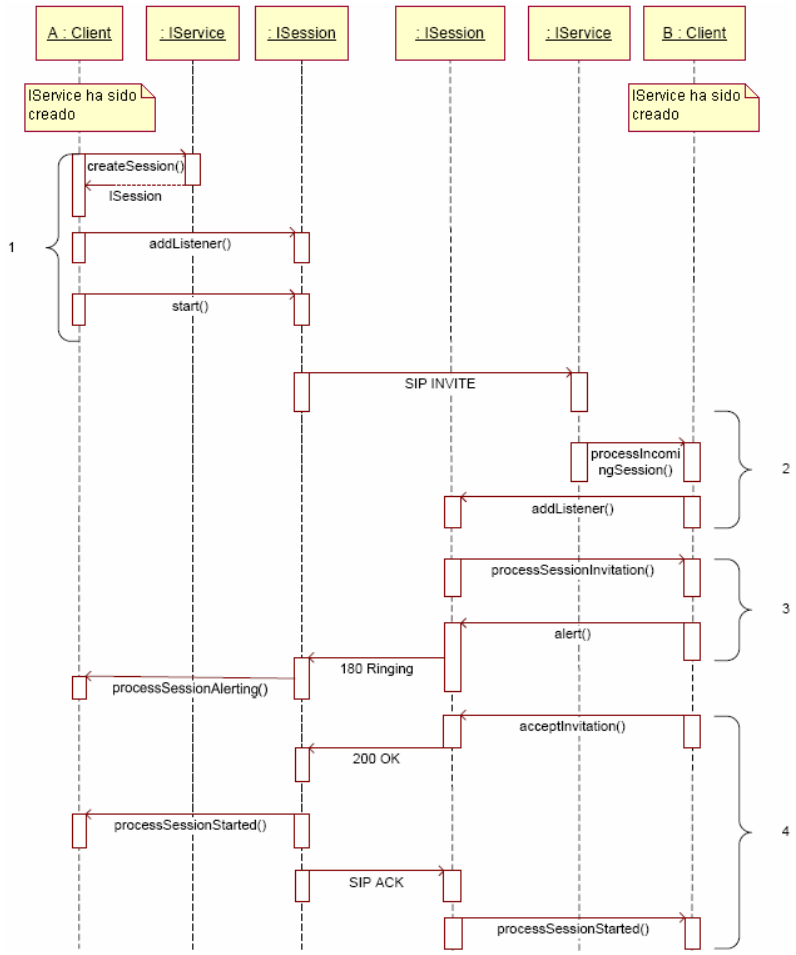

Figura 27 Diagrama de secuencia de creación de una ISession.

3. De manera opcional, el Cliente B alerta al Cliente A con la señal de llamada SIP 180 donde la invitación a una sesión ha sido recibida llamando al método de alerta ().

4. El Cliente B acepta la invitación llamando al método acceptInvitation(), y el Cliente A recibe el evento de *callback* processSessionStarted(). Luego el Cliente B recibe también el evento de callback processSessionStarted () que indica que la sesión ha sido establecida con éxito. De esta forma se establece una sesión genérica.

E.1.6 Envío de Mensajes

Un cliente envía un mensaje instantáneo a un cliente B. Cliente B recibe el mensaje y responde del cliente A con un mensaje nuevo.

En la Figura 28 se muestra un diagrama de secuencias para el envió de mensajes

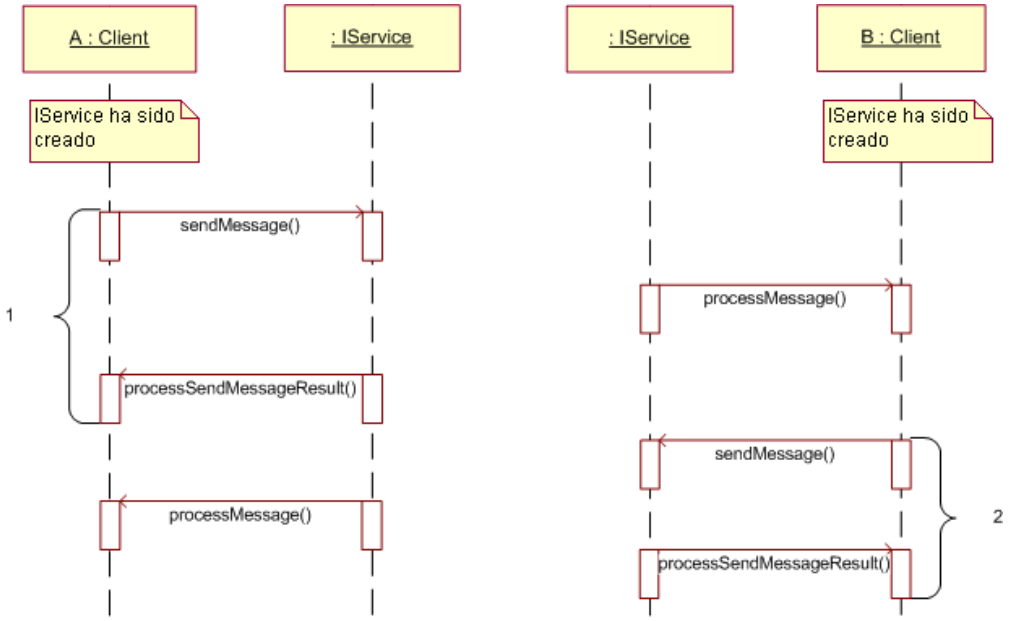

Figura 28 Diagrama de secuencia para el envío de mensajes.

1. Tanto el cliente A como él cliente B han creado un IService. El Cliente A envía un mensaje instantáneo llamando al método SendMessage(), el Cliente B lo recibe con un evento de callback processMessage() y retorna el mensaje. El cliente A recibe un evento de *callback* processSendMessageResult() que indica que el cliente B ha recibido correctamente el mensaje.

2. Cliente B envía un nuevo mensaje llamando al método SendMessage(), y el cliente A recibe el evento de callback processMessage() con el mensaje retornado, el Cliente B recibe el evento de *callback* processSendMessageResult().

La Figura 29 muestra el diagrama de clases para el cliente ICP, creado para el servicio de prueba.

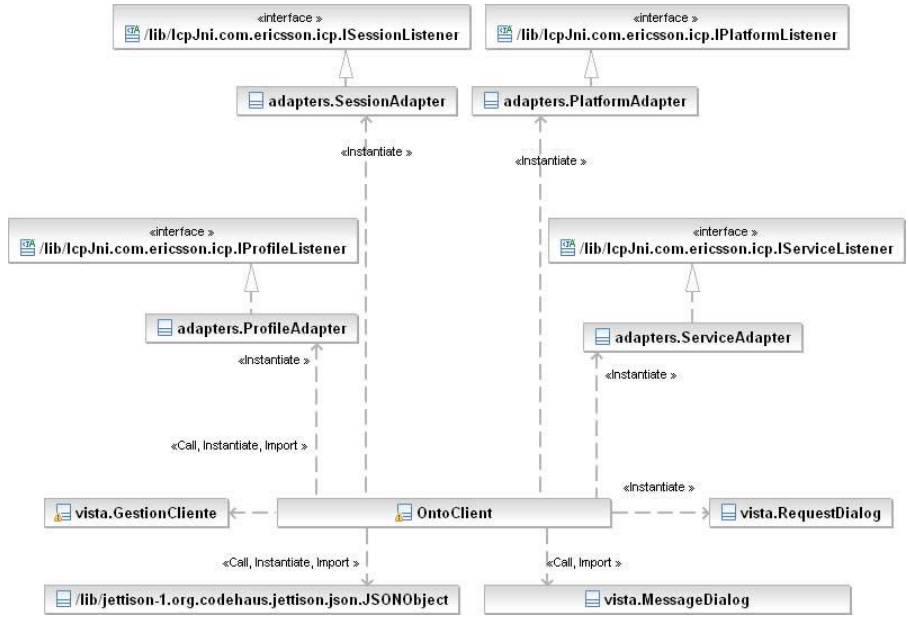

Figura 29 Diagrama de Clases del Cliente.

La clase principal OntoClient utiliza los cuatro adaptadores nombrados anteriormente, que le proporcionan las capacidades básicas de un cliente SIP. La clase JSONObject es utilizada para dar formato a la información enviada hacia el servidor de aplicación. Las clases de paquete vista permiten la captura y despliegue de datos asociados al perfil de usuario.

La Figura 30 muestra el diagrama de paquetes para el cliente ICP.

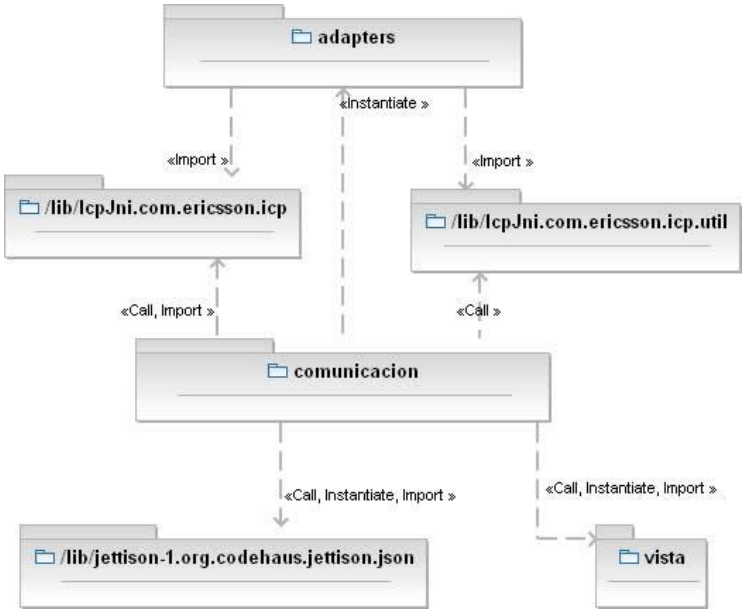

Figura 30 Diagrama de Paquetes del Cliente.

E.2 Instalación de clientes ICP sobre móviles con sistema operativo Symbian.

Los pasos para correr un cliente en el emulador Symbian se muestran en detalle en [26]. Para este anexo es importante tener en cuenta los siguientes puntos:

- Las interfaces graficas desarrolladas con Swing no son compatibles con el sistema operativo Symbian, de esta forma, el hecho de que un cliente Windows funcione correctamente no implica que deba funcionar en Symbian. Para solucionar este inconveniente se recomienda realizar un cliente totalmente optimizado para Symbian con interfaces graficas desarrolladas con AWT.
- El cliente desarrollado para Symbian debe estar soportado sobre el JDK 1.4, el emulador no soporta librerías de JDKs superiores.
- El paquete JETTISON que da soporte para JSON, no es compatible con las librerías que soporta el cliente Symbian.

REFERENCIAS

[1] Ontolingua. Disponible en : http://www.ksl.stanford.edu/software/ontolingua/

[2] Vinay K. Chaudhri, Adam Farquhar, Richard Fikes, Peter D. Karp, "*Open Knowledge Base Connectivity*", disponible en: http://www.ai.sri.com/~okbc/spec/okbc2/okbc2.html

[3] Michael Kifer, Georg Lausen, James Wu. "*Logical Foundations of Object-Oriented and Frame-Base Languages*"

[4] John Dominguez., Enrico Motta, Oscar Corcho Garcia. "*Knowledge Modelling in WebOnto and OCML: A User Guide*". Instituto de conocimiento Multimedia. (1999)

[5] Enrique Herrera, Juan Carlos Herrera, Eduardo Peis, Yusef Hassan. "Ontologías, metadatos y agentes: recuperación semántica de la información". Universidad de Granada. España. 2003

[6] Jeff Hefling, James Hendler, Sean Luke. "*SHOE: A Knowledge Representation Language for Internet Applications*" Instituto de Estudios Computacionales Avanzados. Universidad de Maryland.1999

[7] Francisco Javier Honrubia López. "Introducción a las Ontologías". Escuela Universitaria Politécnica de Albacete. España. 2002

[8] Frank van Harmelen."*The Complexity of the Web Ontology Language*". IEEE Inteligent Systems. 2002, marzo-abril, pp.71-72

[9] Deborah McGuinness , Frank Van Harmelen ."*OWL Web Ontology Language Overview*". Recomendación de la W3C. Febrero 2004. Disponible en :http://www.w3.org/TR/2004/REC-owl-features-20040210/

[10] Gregory Karvounaraki, Sofia Alexaki, Vassilis Christophides, Dimitris Plexousakis. "RQL: A Declarative Query Language for RDF". Instituto de Ciencias Computacionales. FORTH, Heraklion, Grecia

[11] José Javier Samper Zapater. "Ontologías para servicios web semánticos de Información de tráfico". Departamento de Informática. Universidad de Valencia. Junio 2005

[12] Aduna B.V., Sirma AI Ltd."Chapter 6. The SeRQL query language". Desde "User Guide for Sesame rev. 1.2.6". Copyright © 2002-2006. Disponible en http://www.openrdf.org/doc/sesame/users/ch06.html

[13] Richard Fikes, Pat Hayes, Ian Horrocks. "DAML Query Language (DQL) abstract specification". Estados Unidos y la Uniión europea Programa DAML (DARPA Agent Markup Language). Abril 2003. Disponible en http://www.daml.org/2003/04/dql/dql

[14] Richard Fikes, Pat Hayes, Ian Horrocks, "OWL-QL, A Language for Deductive Query Answering on the Semantic Web". Universidad de Stanford. California. 2004

[15] SPARQL Query Language for RDF. Recomendación de la W3C. Enero 2008. Disponible en: http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/

[16] Stefan Decker, Michael Erdmann, Dieter Fensel, Rudi Studer "Ontobroker: Ontology based Access to Distributed and Semi-Structured Information". Universidad de Karlsruhe. Karlsruhe. Alemania. 1998

[17] Jena. "A Semantic Web Framework for java" Disponible en http://jena.sourceforge.net/, Última visita febrero 2009.

[18] Jeen Broekstra, Arjohn Kampman, Frank Van Harmelen. "Sesame: a Generic Architecture for Storing and Querying RDF and RDF Schema". Primera conferencia Internacional de la Web Semántica. Cerdeña, Italia .Junio 2002.

[19] Thomas Gabel, York Sure, Johanna Voelker. "KAON An Overview.(Karlsruhe Ontology Management Infrastructure)" Instituto de ciencias de la informática Aplicada y métodos formales de descripción. Universidad de Karlsruhe. Abril 2004

[20] York Sure, Rudi Studer. "On-To-Knowledge OntoEdit". Instituto de ciencias de la informática Aplicada y métodos formales de descripción, Universidad de Karlsruhe.

[21] Sean Bechhofer, Ian Horrocks, Carole Goble and Robert Stevens."OilEd: a Reason-able Ontology Editor for the Semantic Web". Departamento de Ciencias Computacionales. Universidad de Manchester. Reino Unido. 2001

[22] The Protégé Ontology Editor and Knowledge Acquisition System. Universidad de Stanford. Disponible en http://protege.stanford.edu/

[23] Natalya Fridman Noy, Deborah McGuinnes. "Desarrollo de Ontologías-101: Guía para crear tu Primera Ontología". Reporte técnico KSL-01-05. Laboratorio de Sistemas de Conocimiento de Stanford, Universidad de Stanford. Septiembre 2005

[24] Maria Golemati, Akrivi Katifori, Costas Vassilakis, George Lepouras, Constantin Halatsis "*Creating an Ontology for the User Profile: Method and Applications*". Primera Conferencia Internacional IEEE sobre los desafíos en la Investigación en Ciencias de la Información (RCIS -*Research Challenges in Information Science*), Morocco 2007.

[25] Service Development Studio (SDS) 4.1 FD1 Installation Instructions. Disponible en : http://www.ericsson.com/developer/sub/open/technologies/ims_poc/docs/sds_40_install_inst

[26] Ericsson Service Development Studio (SDS) 4.1 FD1 Developer's Guide. Disponible en: http://www.ericsson.com/developer/sub/open/technologies/ims_poc/docs/sds_40_dev_guide