Universidad Autónoma Metropolitana
Unidad Azcapotzalco
División de Ciencias Básicas e Ingeniería
Licenciatura en Ingeniería en Computación

Reporte final del Proyecto de Integración

Capa de servicios web para administrar la información de un espacio inteligente

Modalidad: Proyecto Tecnológico

Trimestre 2018 Invierno

Oscar Said Hernández Orozco

2123031838

oscaruxo@gmail.com

Dr. José Alejandro Reyes Ortiz

Departamento de Sistemas

jaro@correo.azc.uam.mx

Dra. Maricela Claudia Bravo Contreras

Departamento de Sistemas

mcbc@correo.azc.uam.mx

21 DE ABRIL DE 2018

Declaratoria

Yo, Dr. José Alejandro Reyes Ortiz, declaro que aprobé el contenido del presente Reporte de Proyecto de Integración y doy mi autorización para su publicación en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.

Dr. José Alejandro Reyes Ortiz

Yo, Dra. Maricela Claudia Bravo Contreras, declaro que aprobé el contenido del presente Reporte de Proyecto de Integración y doy mi autorización para su publicación en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.

Dra. Maricela Claudia Bravo Contreras

Yo, Oscar Said Hernández Orozco, doy mi autorización a la Coordinación de Servicios de Información de la Universidad Autónoma Metropolitana, Unidad Azcapotzalco, para publicar el presente documento en la Biblioteca Digital, así como en el Repositorio Institucional de la UAM Azcapotzalco.

Oscar Said Hernández Orozco

Resumen

En este proyecto se implementó una capa de servicios web cuya funcionalidad es administrar la información sobre eventos dentro de un sistema consciente del contexto basado en ontologías. Está capa está conformada por un servicio web con un conjunto de métodos para la escritura, eliminación y actualización de datos en la ontología de interés. La idea es que esta capa esté implementada como un componente de software adaptable a las arquitecturas de este tipo de sistemas con la finalidad de que sea reutilizable y pueda ser de utilidad en la creación de sistemas capaces de percibir detalles del medio físico en el que se encuentran, con la finalidad de modificarlo acorde a las necesidades de los usuarios o simplemente para recabar información del ambiente.

La ventaja de hacer uso de servicios web es el intercambio de datos entre aplicaciones a través de internet, además de que su implementación es estandarizada, lo que garantiza un alto grado de interoperabilidad. Esta interoperabilidad se puede abordar usando ontologías, las cuales, en el contexto de ciencias de la computación, son definiciones formales de tipos de datos, propiedades y relaciones entre entidades que representan elementos del mundo físico. Las ontologías se utilizan para lograr una mejor organización de la información y facilitar la comunicación entre aplicaciones de un dominio.

La capa de servicios resultante de este proyecto será utilizada para trabajos futuros que involucran la detección y representación de eventos es un ambiente académico inteligente, además de la obtención de la información recabada por sensores de dicho espacio (temperatura, presencia, humedad, luminosidad); con las herramientas proporcionadas por este servicio web, es posible realizar el enriquecimiento de un modelo ontológico que posteriormente será utilizado en diversos trabajos, como la predicción de eventos en un ambiente académico.

Tabla de contenido

Índice de figuras

1. Introducción

En computación, un sistema consciente del contexto hace referencia a sistemas que son capaces de percibir detalles del medio físico en el que se encuentran con la finalidad de modificarlo acorde a las necesidades de los usuarios o simplemente para recabar información del ambiente, tal como la luminosidad, humedad, temperatura, presencia de seres vivos, entre otra. El uso de este tipo de sistemas ha dado lugar a espacios que hoy se conocen como casas inteligentes, oficinas inteligentes, o cualquier entorno inteligente.

Por otro lado, la función de los servicios web es intercambiar datos entre aplicaciones a través de internet. La ventaja del uso de servicios es que su implementación está estandarizada, lo que garantiza un alto grado de interoperabilidad. Esta interoperabilidad se puede abordar usando ontologías, las cuales, en el contexto de ciencias de la computación, son definiciones formales de tipos de datos, propiedades y relaciones entre entidades que representan elementos del mundo físico. Las ontologías se utilizan para lograr una mejor organización de la información y facilitar la comunicación entre aplicaciones de un dominio.

En este proyecto se implementará una capa de servicios web cuya finalidad será administrar la información sobre personas, eventos y sensores de un sistema consciente del contexto basado en ontologías. La capa estará conformada por diversos servicios web para la escritura, eliminación y actualización de datos en la ontología de interés. La idea es que esta capa esté implementada como un componente de software adaptable a las arquitecturas de sistemas conscientes del contexto con la finalidad de que sea reutilizable y apoyar la creación de este tipo de sistemas.

2. Antecedentes

2.1 Proyectos de integración o terminales

1. Ontología genérica para la integración de servicios web semánticos [1]

   El objetivo principal es diseñar e implementar un modelo ontológico general para la representación de las características comunes de los lenguajes de descripción de servicios web WSDL 1.1, WSDL 2.0, OWL-S y SAWSDL y finalmente de implementar un sistema para evaluar el resultado de la integración y la correspondencia de la ontología. En este proyecto se desarrolla una ontología, a diferencia de este proyecto en donde el objetivo es administrar la información de una ontología ya existente, mediante una capa de servicios web.

2. Arquitectura orientada a servicios web para establecer grupos de colaboración entre investigadores [2]

   Se implementó un servicio web para escribir elementos y mostrar colaboraciones de los elementos de una ontología. En el presente proyecto se creó toda una capa de servicios que permite administrar completamente una ontología.

3. Extracción y representación de servicios web escritos en SAWSDL mediante una ontología [3]

   En este Proyecto se construye y almacena una ontología extrayendo los elementos representativos de servicios web escritos en *SAWSDL*. A diferencia de proyecto presentado en este documento, cuyo objetivo es diseñar e implementar una capa de servicios web para administrar la información de una ontología, no crear una.

2.2 Tesis

4. Espacios inteligentes con manipuladores móviles dotados de intuición artificial [4]

   El objetivo de este trabajo de tesis es diseñar un espacio inteligente donde un manipulador móvil y el espacio sean conscientes de las actividades que se desarrollan dentro de sí. En relación con el presente proyecto se tiene el uso de espacios inteligentes a los cuales está orientada la información de la ontología que se busca administrar con la capa de servicios.

2.3 Artículos

5. Hybrid Architecture to Support Context-Aware Systems [5]

   En este artículo se describe el sistema completo en donde se utilizará la capa de servicios web del presente proyecto, así como la descripción de la ontología que se usará. En general, la descripción de las distintas capas del sistema, sus distintos componentes tanto físicos como software.

6. An Ontology-based Context Model in Intelligent Environments [6]

   Este artículo describe el modelado de un sistema similar al que se analizará en este proyecto ya que utiliza ontologías representadas en *OWL* cuya información es accesible y manipulable. La ontología que se modela en este artículo, al igual que la ontología que se administrará mediante la capa de servicios representa también información de un entorno inteligente.

3. Justificación

El aumento en la implementación de espacios inteligentes ha propiciado un incremento en la cantidad de hardware y software utilizado en los sistemas en dichos espacios. Esto implica un importante crecimiento en la información que un sistema consciente del contexto recibe, procesa y almacena, lo que dificulta la administración y organización de la información, así como, la comunicación entre diferentes aplicaciones del espacio inteligente.

Las ontologías ofrecen una forma de comunicación estandarizada entre aplicaciones con un alto grado de interoperabilidad. Éstas son una herramienta clave para la representación de información y tienen como ventaja la posibilidad de ser manipuladas por servicios web. Por ello, en este proyecto se obtendrá una capa de servicios web que faciliten el manejo y organización de la información en un sistema consciente del contexto implementado bajo una arquitectura híbrida basada en ontologías.

Los componentes de software que formarán dicha capa podrán ser reutilizados por programadores de aplicaciones o sistemas conscientes del contexto, quienes serán los principales usuarios beneficiados. Por ello, este proyecto tiene una relevancia destacada en el área de la ingeniería en computación.

4. Objetivo general

Diseñar e implementar una capa de servicios web para administrar la información de un sistema consciente del contexto basado en ontologías.

4.1 Objetivos específicos

- Describir la ontología existente que contiene la representación de la información sobre personas, eventos y sensores correspondientes a un sistema consciente del contexto.

- Diseñar e implementar una capa de servicios web para la inserción, eliminación y actualización de información sobre personas, eventos y sensores en la ontología.

- Evaluar la funcionalidad de la capa de servicios mediante peticiones a través de clientes.

5. Marco teórico

El lenguaje *OWL*[1] está pensado para ser usado cuando la información contenida en los documentos necesita ser procesada por aplicaciones, al contrario que en las situaciones donde el contenido sólo necesita ser presentado a los humanos. *OWL* puede ser usado para representar explícitamente el significado de términos en vocabularios y las relaciones entre esos términos [7]. Esta representación de términos y sus interrelaciones se denomina ontología. OWL tiene mayor capacidad para expresar significado y semántica que *XML, RDF, RDF-S*, y, de este modo, OWL va más allá de estos lenguajes en su capacidad para representar contenido interpretable por un ordenador en la Web [8]. OWL es una revisión del Lenguaje de Ontologías Web *DAML+OIL* [2] incorporando lecciones aprendidas a partir del diseño y aplicación de *DAML+OIL* [9].

Las ontologías contemporáneas comparten muchas similitudes estructurales, indiferente al lenguaje en el cual fueron expresadas. La mayoría de las ontologías describen individuos (instancias), clases (conceptos), atributos y relaciones. A continuación, se describen cada uno de estos componentes [10]:

- Individuos: instancias u objetos
- Clases: conjuntos, colecciones, conceptos, clases en programación, tipos de objetos, o tipos de cosas
- Atributos: aspectos, propiedades, rasgos, características, o parámetros que objetos y clases pueden tener
- Relaciones: formas en las cuales clases e individuos se pueden relacionar unos con otros
- Restricciones: establecen descripciones formales de lo que deber ser verdad con el objetivo de que alguna aserción pueda ser aceptada como entrada
- Reglas: Declaraciones con forma de oraciones si-entonces que describen inferencias lógicas que pueden ser derivables de una aserción en una forma particular
- Axiomas: aserciones en una forma lógica que juntos incluyen toda la teoría que la ontología describe en su dominio de aplicación

Es posible manipular una ontología escrita en lenguaje *OWL* a través de aplicaciones escritas en Java median el uso de la *API*[3] *OWL API*. Esta *API* permite la creación, manipulación y serialización de ontologías *OWL*.

[1] Ontology Web Language (Lenguage de Ontologias Web)
[2] DARPA Agent Mrkup Language-Ontology Inference Layer
[3] Application Pogramming Interface (Interfaz de programación de aplicaciones)

Un servicio web es una tecnología que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma, pueden utilizar los servicios web para intercambiar datos en redes de ordenadores como Internet. La interoperabilidad se consigue mediante la adopción de estándares abiertos [11].

WSDL[4] es una notación *XML* [5]para describir un servicio web. Una definición *WSDL* indica a un cliente cómo componer una solicitud de servicio web y describe la interfaz que proporciona el proveedor del servicio web. Esta notación describe la forma de comunicación, es decir, los requisitos del protocolo y formatos de los mensajes necesarios para interactuar con los servicios listados en su catálogo. Las operaciones y mensajes que soporta se describen en abstracto y se ligan después al protocolo concreto de red y al formato del mensaje [12].

La estructura del *WSDL* tiene los siguientes elementos:

- Tipos de datos
  <*types*>: Esta sección define los tipos de datos usados en el mensaje. Se utilizan los tipos definidos en la especificación de esquemas XML
- Mensajes
  <*message*>: Aquí definimos los elementos de mensaje. Cada mensaje puede consistir en una serie de partes lógicas. Las partes pueden ser de cualquiera de los tipos definidos por XML
- Tipo de puerto
  <*portType*>: Con este apartado definimos las operaciones permitidas y los mensajes intercambiados en el servicio
- *Bindings*
  <*binding*>: Especifica los protocolos de comunicación usados
- Servicios
  <*service*>: Conjunto de puertos y dirección de estos. Esta parte final hace referencia a lo aportado por las secciones anteriores.

Apache Axis es un *framework* de código abierto, basado en XML para servicios web. Consiste en una implementación en Java y otra en C++ del servidor SOAP, así como diversos utilitarios y *APIs* para generar y desplegar aplicaciones de servicios web. Por medio de Apache Axis, los desarrolladores pueden crear aplicaciones computacionales interoperables y distribuidas [13].

Al emplear la variante Java de Axis, existen dos maneras de exponer código Java como servicio web. Lo más fácil es usar los archivos *JWS* (Java Web Service), nativos de Axis. La otra manera consiste en usar despliegues a la medida. Los

[4] Web Service Description Language (Lenguaje de descripción de servicios web)
[5] Extensible Markup Lanuguaje (Lenguaje de marcado extensible)

despliegues a la medida permiten adaptar los recursos que se desea exponer como servicios web.

Los archivos *JWS* contienen el código fuente de la clase Java que se desea exponer como servicio web. La principal diferencia entre un archivo Java corriente y un archivo *jws* consiste en su extensión. Otra diferencia es que los archivos *jws* se despliegan como código fuente y no como archivos .*class* compilados. Si se usa en conjunto con Apache *Tomcat*, basta con copiar el archivo *jws* en el directorio de Axis contenido en el directorio *WebApps* de *Tomcat*; en caso de usar otro servidor para el despliegue es necesario crear un archivo WAR a la medida.

Cuando un servicio web se expone por medio de Axis, un archivo WSDL se generará automáticamente al acceder a la URL del servicio web con el apéndice "?WSDL".

6. Desarrollo del proyecto

En esta sección se presenta el desarrollo del proyecto, el cual se ha dividido en cinco módulos para su análisis, diseño e implementación. Dichos módulos se describen a continuación.

6.1 Análisis de las ontologías

En el presente proyecto se manipulan las ontologías que representan un espacio inteligente, creadas por alumnos de la maestría en computación de la UAM Azcapotzalco. En las figuras 1 a 4 se presenta la estructura gráfica de dichas ontologías. En el apéndice A del presente documento se presentan fragmentos del código OWL de las ontologías.

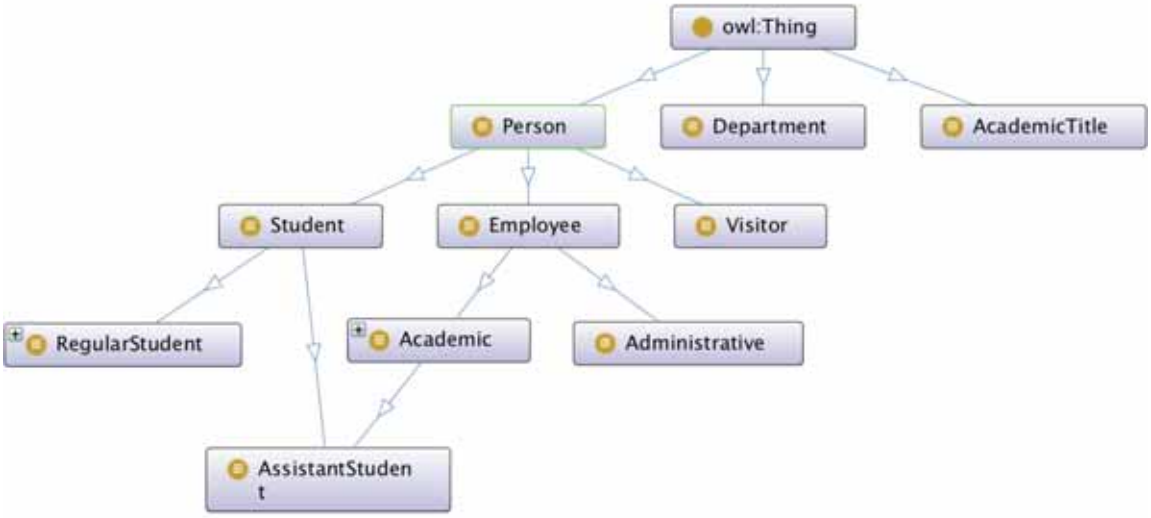

Figura 1 - Ontología Person

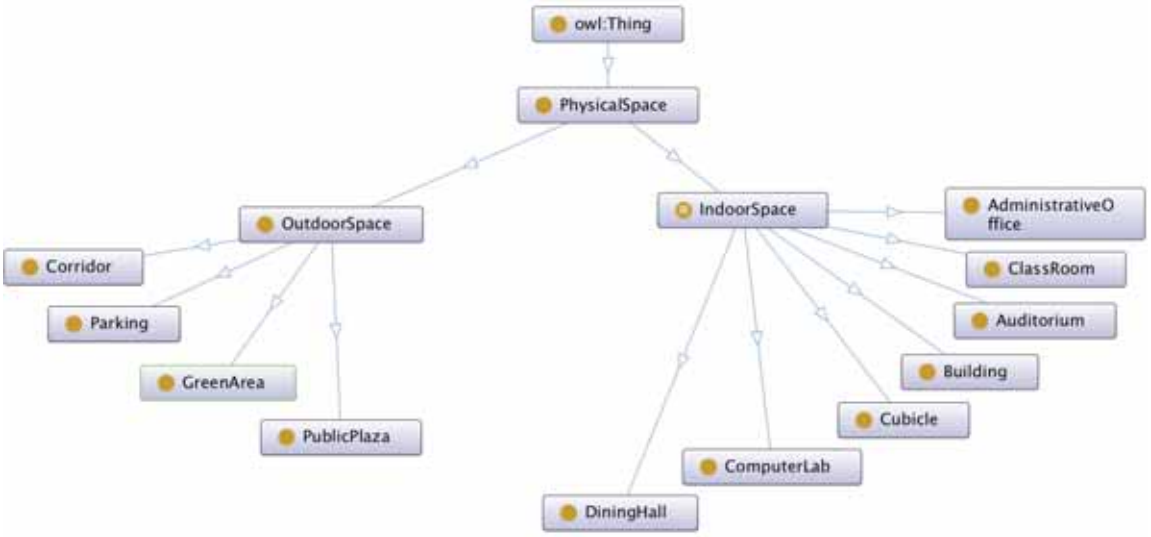

Figura 2 - Ontología PhysicalSpace

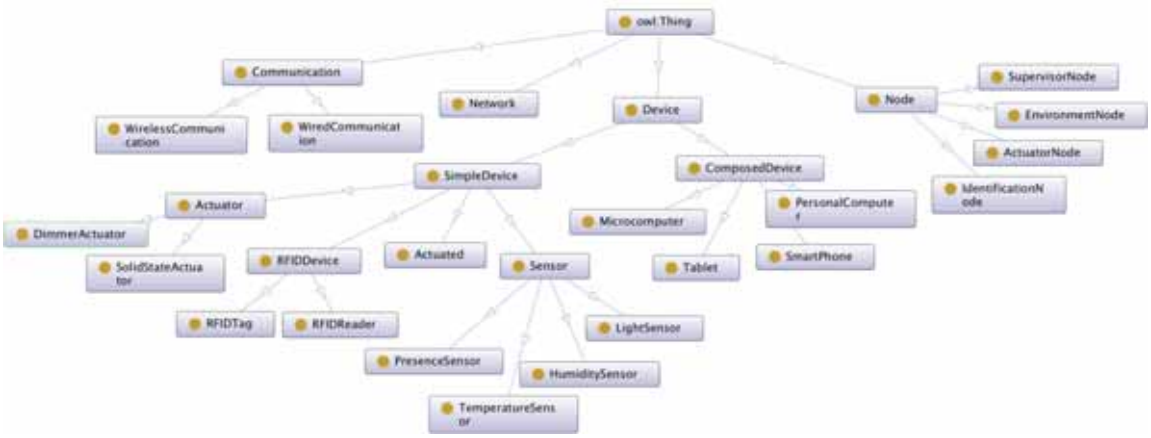

Figura 3 - Ontología SensorNetwork

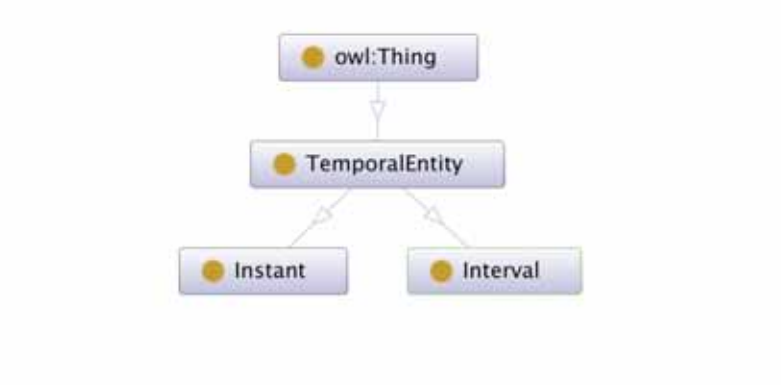

Figura 4 - Ontología Time

A continuación, en la Figura 5 se muestra de manera general la representación gráfica de la ontología **IntelligentEnivroment**, en la cual se agregan las otras cuatro.

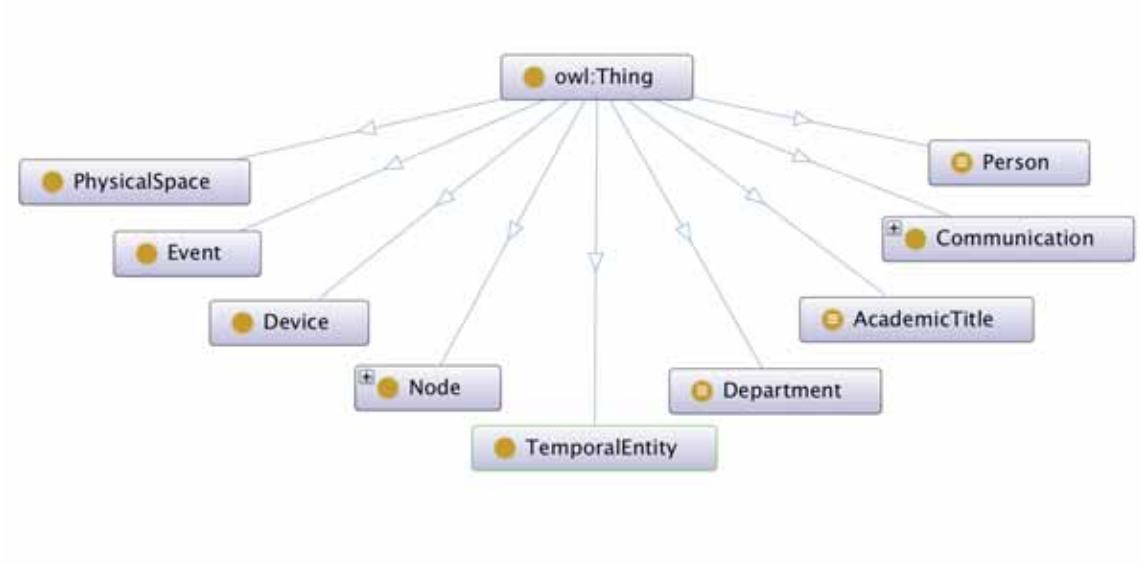

Figura 5 - Ontología IntelligentEnviroment

6.2 Diseño e implementación de clases

El proyecto en Java se desarrolla mediante el *IDE*[6] *NetBeans* y primero se crean las clases que modela la ontología con sus respectivos atributos. Estas clases deben estar construidas implementando las convenciones de JavaBean[7] para que, al momento del despliegue del servicio, sean accesibles desde un cliente.

En la Figura 6 se muestra el diagrama de clases UML[8] de la clase que modela la clase *Event* de la ontología. Esta clase contiene un constructor sin argumentos, atributos privados accesibles mediante métodos *get* y *set* que siguen una convención de nomenclatura estándar, e implementa la interfaz *serializable*; todo lo anterior de acuerdo con las convenciones JavaBeans. El código de la clase se presenta en el apéndice B de este documento.

[6] Integrated Development Environment (Entorno de desarrollo integrado)
[7] Especificación e componentes de software reutilizables creado por Sun Microsystems
[8] Unified Modeling Language (Lenguaje unificado de modelado)

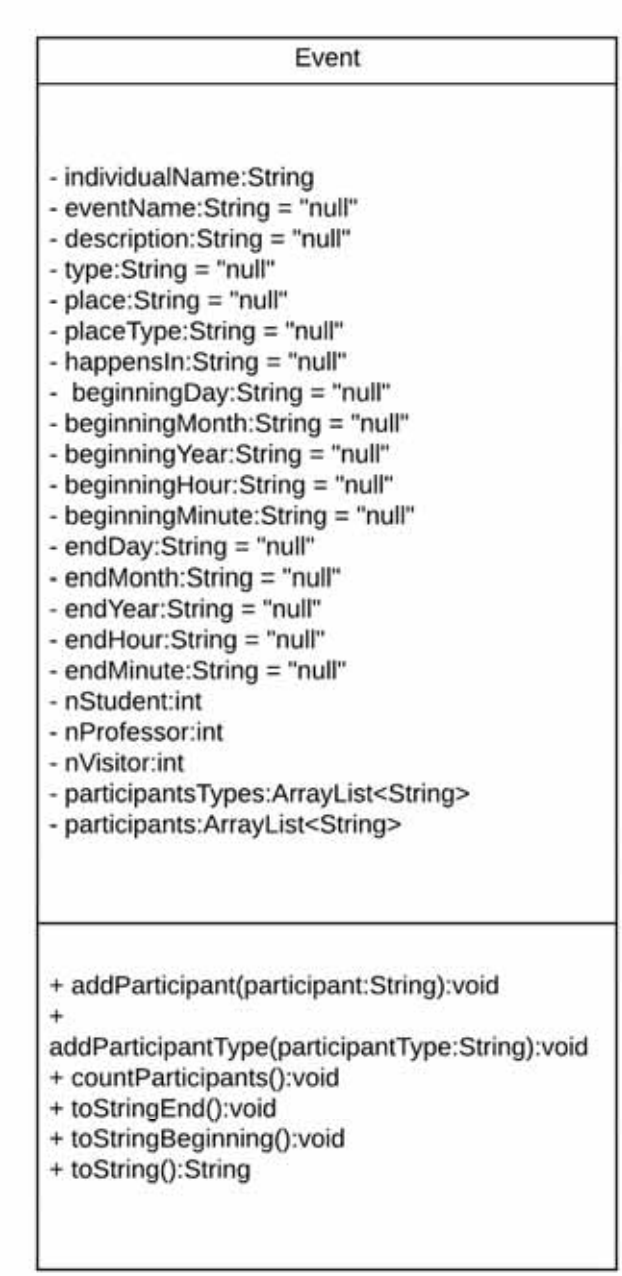

Figura 6 - Diagrama UML de la clase Event

Para utilizar las funciones que nos permitan manipular ontologías creadas en OWL desde aplicaciones Java, es necesario agregar la librería de la API al proyecto. Puede obtenerse la última versión de dicha librería desde alguno de sus repositorios oficiales[9]. Deberá descargarse el archivo *jar* como se muestra en la Figura 7.

[9] http://search.maven.org/#search%7Cga%7C1%7Ca%3A"owlapi-api"

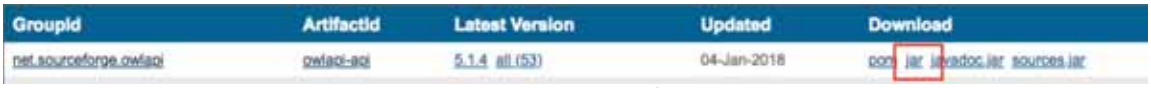

Figura 7 - Descarga de OWL API

Posteriormente se codifica la clase que contendrá los métodos que más tarde ofreceremos como un servicio web, esto haciendo uso de los métodos y clases proporcionados por la API OWL. En la Figura 8 se muestra el diagrama UML de dicha clase; en el apéndice B de este documento se presenta el código completo.

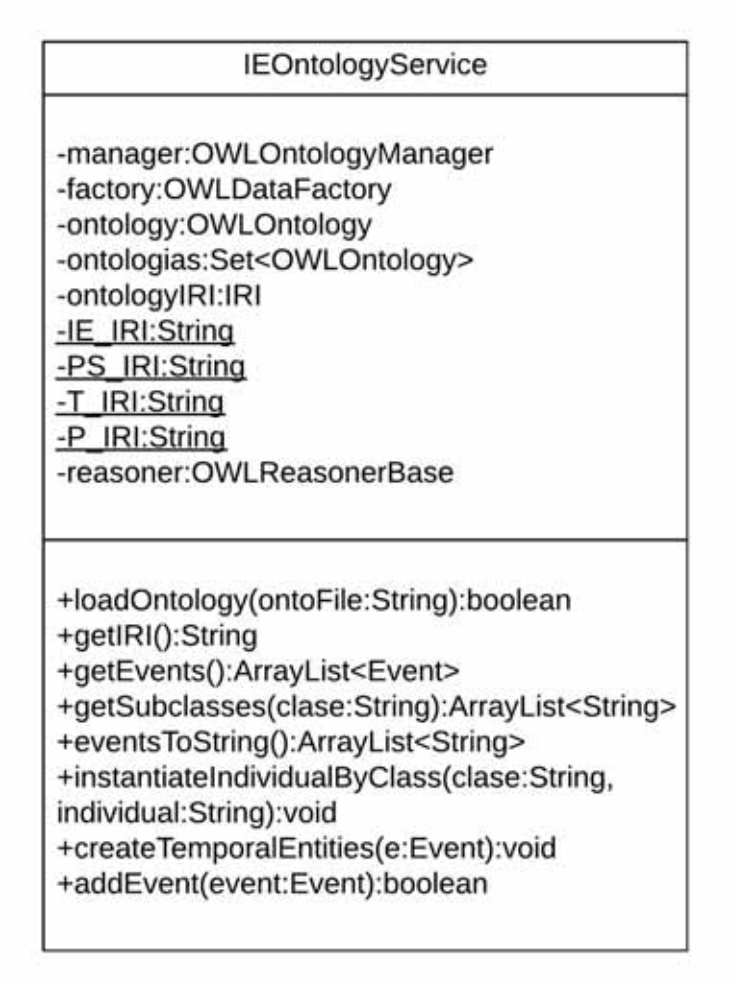

Figura 8 - Diagrama UML de la clase IEOntologyService

En colaboración con los usuarios finales de los servicios web realizados en este proyecto, se llegó a la conclusión de que, de manera general los únicos individuos en la ontología que deberían ser manipulados con frecuencia son los de la clase evento; los individuos del resto de clases son de una cantidad fija o será poco común agregar nuevos o quitarlos, por lo que en tales casos podrá realizarse manualmente sin mayor problema.

Por lo anterior, la clase **IEOntologyService**, únicamente contempla métodos para manipular los eventos de la ontología, tanto para escritura como para lectura, y otros métodos para obtener información general de la ontología que pueden resultar de utilidad o que se usan como herramienta dentro de los métodos de escritura y lectura.

Los dos métodos principales en esta clase son los siguientes:

- *getEvents():ArrayList<Event>*

  Este método recorre todas las subclases de la clase Evento en la ontología y recupera todos los individuos registrados, para cada uno de los individuos recupera todos sus atributos y relaciones con individuos de otras clases y se van asignando a los respectivos campos de un objeto de la clase *Event*; al terminar de recuperar todos los atributos de un individuo el objeto *Event* se agrega al *ArrayList* que será el objeto devuelto al terminar la ejecución del método.

  Durante la ejecución de este método se hacen uso de uno de los métodos de herramienta mencionados anteriormente, que es *getSubclasses (Figura 9),* en éste se especifica mediante el parámetro de tipo *String*, la clase de la cual deberá recuperar las subclases, en este caso se hace uso de él enviando como parámetro "*Event*".

```
public ArrayList<String> getSubclasses(String clase) {
    ArrayList<String> salida = new ArrayList<String>();
    Set<OWLClass> classes = ontology.getClassesInSignature(true);
    for (OWLClass c : classes) {
        if (c.toStringID().toLowerCase().contains(clase.toLowerCase())) {
            byte inicio = (byte) c.toStringID().indexOf("#");
            salida.add(c.toStringID().substring(inicio + 1));
        }
    }
    return salida;
}
```

Figura 9 - Método getSubclasses

En la Figura 10 se muestra parte del código de este método, específicamente la parte donde ser recorren las subclases recuperando listas de individuos y recorriendo dichas listas para ir asignando los valores al objeto *Event*.

```
for (IRI i : clases) {
    OWLClass classOWL = factory.getOWLClass(i);
    Set<OWLIndividual> individuos = classOWL.getIndividuals(ontology);
    if (!individuos.isEmpty()) {
        byte inicio = (byte) i.toString().indexOf("#");
        for (OWLIndividual ind : individuos) {
            System.out.println(i.toString().substring(i.toString().indexOf("#") + 1));
            Event e = new Event();
            e.setType(i.toString().substring(i.toString().indexOf("#") + 1));
```

Figura 10 - fragmento del método getEvents

- *addEvent(event:Event):boolean*

  La función de este método es agregar un nuevo método en la ontología, creando en el camino todas su propiedades y relaciones con otros individuos de la ontología, como son participantes y lugar en el que ocurre, que son parte de las ontologías **Person** y **PhysicalSpace**, respectivamente. Para su invocación debe construirse primero, un objeto *Event* y agregar todos sus atributos para poder usarlo como parámetro y recuperar todas las propiedades que se necesitan agregar a la ontología.

  Este método hace uso de dos métodos de herramienta, que son *instantiateIndividualByClass* y *createTemporalEntities*; el primero se usa para crear un individuo en una clase con el nombre de dicho individuo, ambos parámetros de tipo *String*(Figura 11). El otro, *cresateTemporalEntities* (Figura 12), usará el mismo objeto Event pasado al método principal y también hará uso de *instantieteIndividualByClass*, sólo que esta vez creará individuos de la clase *TemporalEntity* y tomará los valores de inicio y fin contenidos en el objeto *Event*.

```
public void instantiateIndividualByClass(String clase, String individual) {
    try {
        OWLClass classOWL = factory.getOWLClass(IRI.create(IE_IRI + "#" + clase));
        OWLNamedIndividual individualOWL = factory.getOWLNamedIndividual(IRI.create(ontologyIRI.toString() + "#" + individual));

        OWLClassAssertionAxiom classAssertionAxiom = factory.getOWLClassAssertionAxiom(classOWL, individualOWL);

        manager.addAxiom(ontology, classAssertionAxiom);

        manager.saveOntology(ontology);
    } catch (OWLOntologyStorageException e) {
        e.printStackTrace();
    }
}
```

Figura 11 - método instantiateIndividualByClass

```
public void createTemporalEntities(Event e){
    try {
        OWLClass classOWL = factory.getOWLClass(IRI.create(T_IRI + "#Instant"));

        //Para la entidad de inicio
        OWLNamedIndividual individualOWL = factory.getOWLNamedIndividual
        (IRI.create(ontologyIRI.toString() + "#begin" + e.getIndividualName().substring(0,1).toUpperCase()+e.getIndividualName().substring(1)));
        OWLClassAssertionAxiom classAssertionAxiom = factory.getOWLClassAssertionAxiom(classOWL, individualOWL);
        manager.addAxiom(ontology, classAssertionAxiom);

        OWLDataProperty dataPropertyOWL = factory.getOWLDataProperty(IRI.create(T_IRI + "#hasDay"));
        OWLDataPropertyAssertionAxiom dataPropertyAssertionAxiom = factory.getOWLDataPropertyAssertionAxiom(dataPropertyOWL, individualOWL,
                factory.getOWLTypedLiteral(e.getBeginningDay()));
        manager.addAxiom(ontology, dataPropertyAssertionAxiom);
```

Figura 12 - fragmento del método createTemporalEntities

6.3 Pruebas del funcionamiento de las clases

Para probar los métodos de la clase anterior se usa una clase Test, en ella se crea un evento y se comprueba en la ontología que sea hayan creado todos los individuos necesarios para las relaciones del evento y el evento con todas sus propiedades; también se prueba el correcto funcionamiento de la recuperación de eventos. En la Figura 13 se muestra el método *main* de la clase de pruebas.

```
public static void main(String[] args) {
    IEOntologyService IEservice = new IEOntologyService();
    if(IEservice.loadOntology("ontologias/IntelligentEnviroment.owl"))
        System.out.println("Ontología cargada");
    else
        System.out.println("No se cargó la ontologia");

    Event evento = new Event();
    evento.setIndividualName("seminar07");
    evento.setType("Seminar");
    evento.setEventName("Seminario de fisica 2");
    evento.setDescription("Nueva descripcion acerca del evento: Seminario de fisica 2");
    evento.setHappensIn("auditoriumW003");
    evento.addParticipant("37047");
    evento.addParticipant("204201210");
    evento.setBeginningDay("17"); evento.setBeginningMonth("04"); evento.setBeginningYear("2018");
    evento.setBeginningHour("12"); evento.setBeginningMinute("00");
    evento.setEndDay("17"); evento.setEndMonth("04"); evento.setEndYear("2018");
    evento.setEndHour("14"); evento.setEndMinute("00");
    IEservice.addEvent(evento);

    for (Event e : IEservice.getEvents()) System.out.println(e);
}
```

Figura 13 - método Main, clase de pruebas

En la Figura 14 se muestra el resultado de la ejecución, donde se pueden ver los eventos recuperados, incluyendo el creado y agregado por nuestro método, y desplegados en pantalla correctamente.

```
1       1       0       17-04-2018 12:00        17-04-2018 14:00        IndoorSpace     Auditorium      Seminar Semina
0       1       0       28-02-2018 13:00        null    IndoorSpace     Auditorium      Seminar Seminario de Fisica
1       1       0       13-04-2018 10:00        13-04-2018 12:30        IndoorSpace     ClassRoom       PostgraduateCo
0       0       0       30-03-2018 14:30        30-03-2018 15:15        IndoorSpace     Cubicle AcademicAdvising
10      1       0       null    null    IndoorSpace     ComputerLab     UndergraduateCourse     Curso Temas Selectos
0       1       0       14-04-2018 16:00        null    IndoorSpace     Auditorium      Congress        congreso de in
BUILD SUCCESSFUL (total time: 3 seconds)
```

Figura 14 - resultado de ejecución de pruebas

Una vez que se obtuvieron los resultados deseados con los métodos para la inserción y extracción de los eventos, se tendrán que hacer algunas modificaciones a los métodos para que puedan funcionar como servicio. Únicamente hay que agregar el método *loadOntology* al inicio de ambos métodos principales; esto debido a que al momento de invocar un método desde un cliente (por ejemplo, *loadOntology*) ser cargaran todos los valores de la clase, pero sólo persistirán mientras dure la invocación de dicho método, por lo que, si después intentamos invocar el método para extraer los eventos, por ejemplo, no habrá ninguna ontología cargada en la clase.

Por lo anterior se agrega el método *loadOntology* en los dos métodos principales y se les agrega el parámetro de entrada con la ruta de la ontología. La figura 15 muestra el diagrama UML final de la clase *IEOntologyService*.

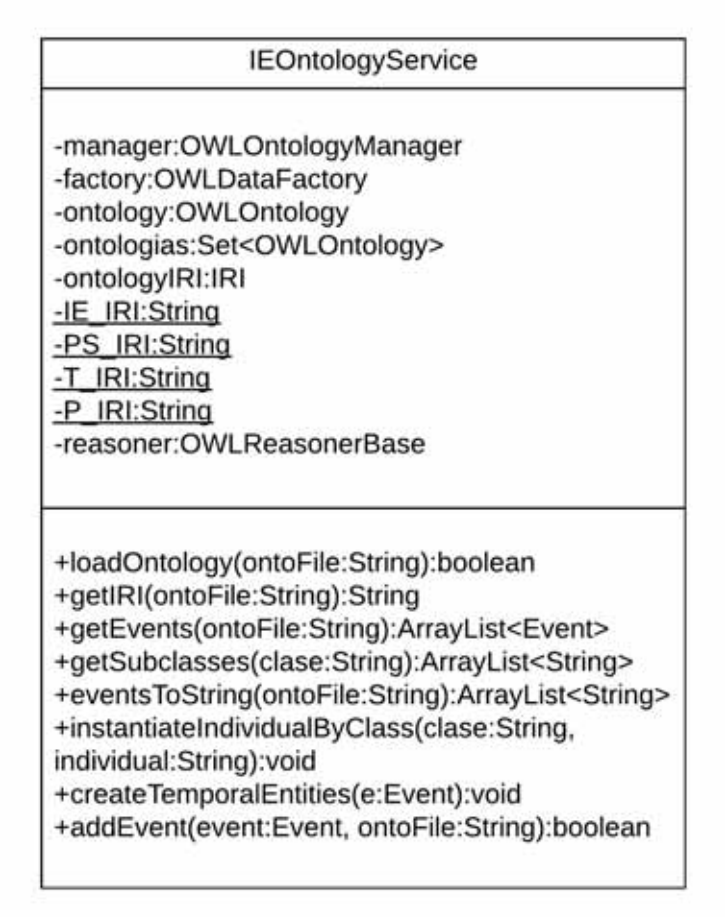

Figura 15 - Diagrama clase IEOntologyService modificada

6.4 Implementación del servicio web

Una vez hechas las modificaciones pertinentes para implementar la clase como un servicio, lo primero que se hace es importar la librería de Axis y sus dependencias al proyecto. La última versión de esta librería puede obtenerse desde el sitio web de Apache[10]. Una vez descargado el archivo *.zip*, se descomprime y se copian e importan al proyecto de *NetBeans*. En la Figura 16 se muestran los archivos *.jar* que deben ser importados.

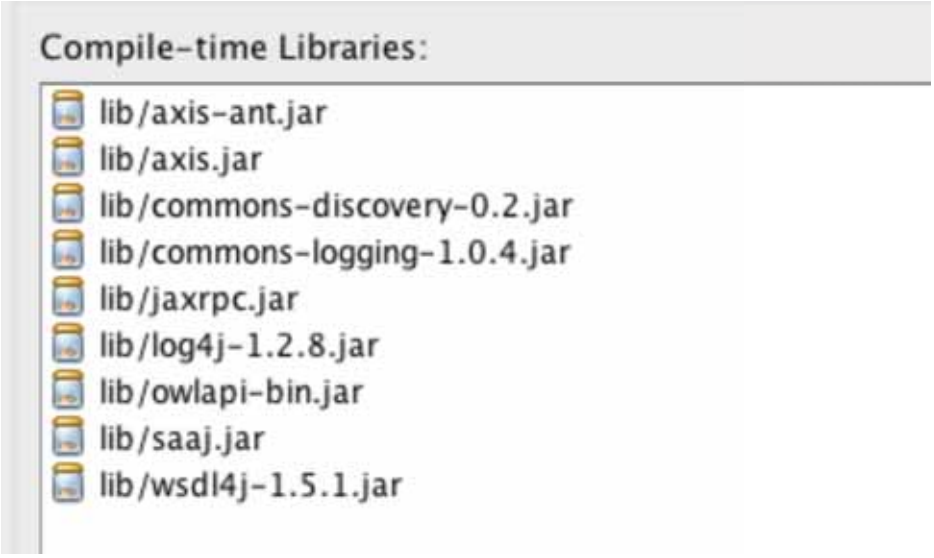

Figura 16 - librerias necesarias

Una vez importadas las librerías es necesario volver a construir el proyecto mediante la opción *Clean* and *Build* de *NetBeans* (Figura 17).

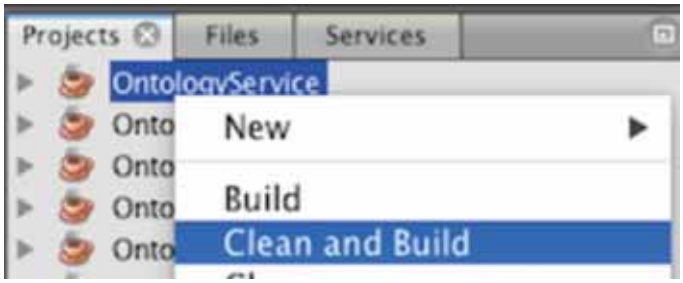

Figura 17 - Reconstrucción de proyecto con las nuevas librerías

[10] http://www.apache.org/dyn/closer.lua/axis/axis2/java/core/1.7.7/axis2-1.7.7-bin.zip

Posteriormente se procede a configurar el servidor web a utilizar, en este caso, para mantener la compatibilidad y facilidad de uso que ofrece el software de Apache se usará el servidor *Tomcat*. Lo primero es copiar toda la carpeta que se obtuvo del archivo .*zip* de la descarga de Axis en la carpeta *webapps* de *Tomcat* (Figura 18).

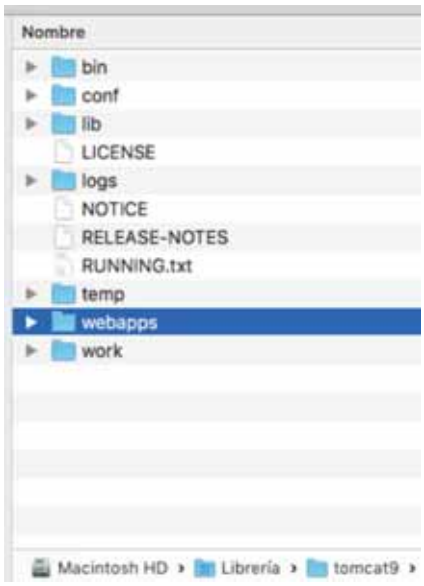

Figura 18 - carpeta webapps de Tomcat

Una vez copiada la carpeta Axis en la carpeta *webapps* de *Tomcat*, es necesario también agregar la librería de OWL API a las librerías de *Tomcat* (Figura 19).

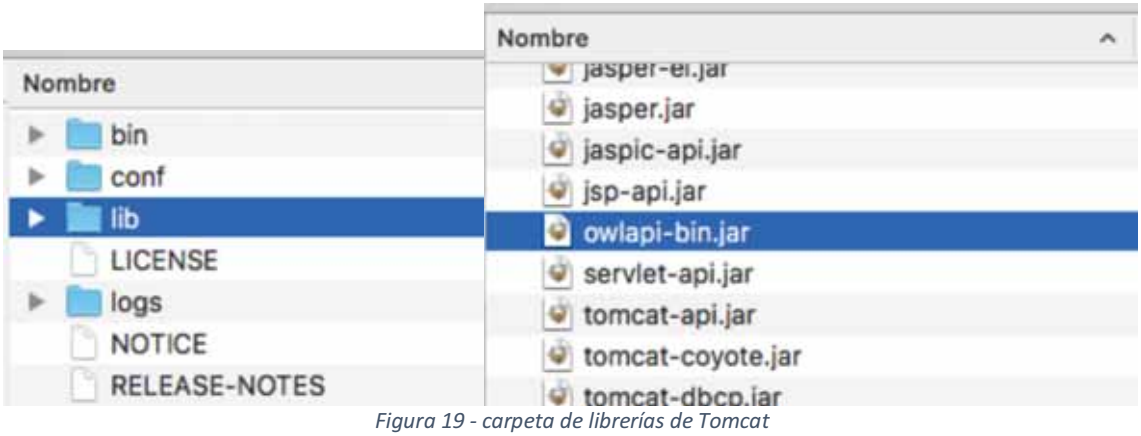

Figura 19 - carpeta de librerías de Tomcat

Ya configurado el servidor se procede a copiar las clases necesarias para el despliegue de estas como un servicio. Es necesario ir a la carpeta del proyecto de *NetBeans*, ir a la carpeta *build,* y dentro de la carpeta *classes,* copiar el archivo con los métodos a desplegar como servicio y las clases JavaBeans necesarias para su

funcionamiento; en este caso son la clase **IEOntologyService** y *Event*, respectivamente.

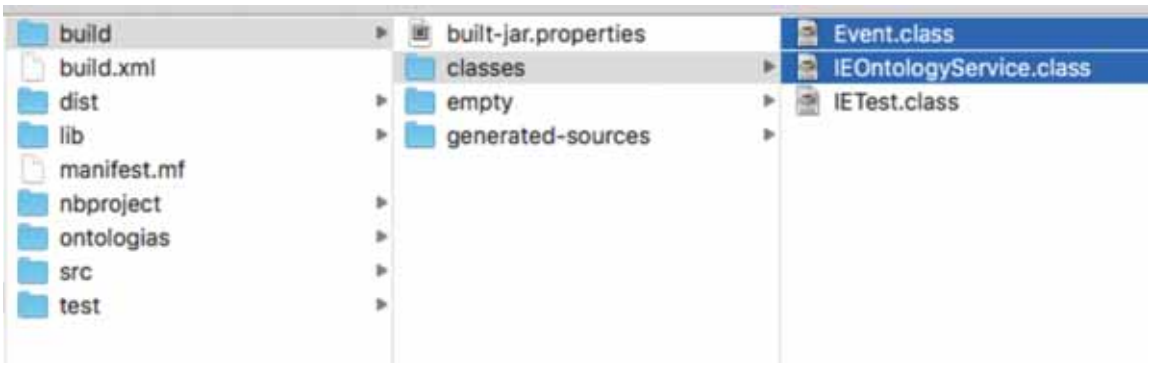

Figura 20 - archivos .class a copiar

Estos dos archivos se llevarán a diferentes directorios dentro de la carpeta axis que copiamos en la carpeta *webapps* de *Tomcat*; se pegarán del siguiente modo:

1. El archivo *.class* con los métodos a desplegar como servicio web se pegará en la raíz de la carpeta axis y se le cambiara manualmente la extensión a *.jws* (Figura 21).

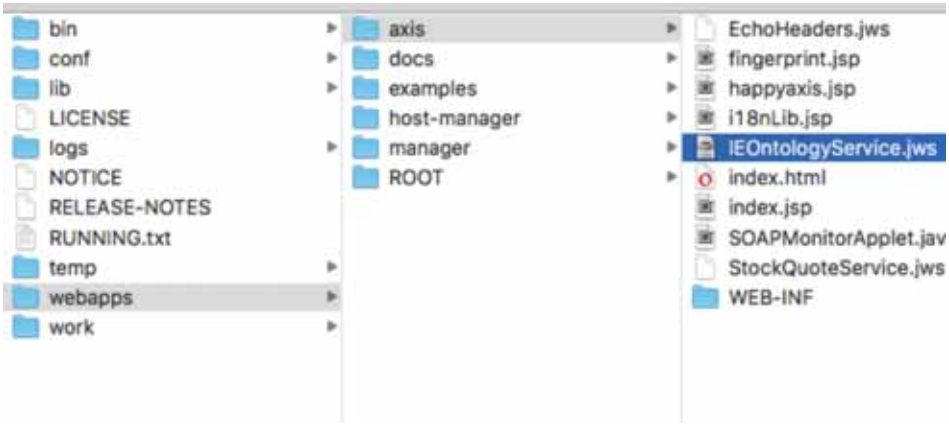

Figura 21 - Archivo renombrado

2. Ambos archivos se pegarán en las siguientes tres carpetas, esta vez sin cambiar la extensión de ninguno de los dos:

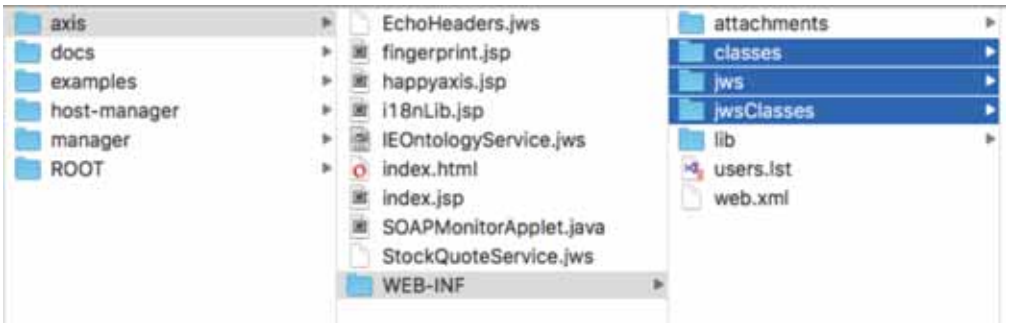

Figura 22 - Carpetas destino

Después de haber hecho todo lo anterior, se tiene listo *Tomcat* y Axis para el despliegue del servicio web. A continuación, lo único que se tiene que hacer es iniciar el servidor y se generará el servicio web de la clase *IEOntology* service.

Para comprobarlo accederemos desde un navegador de internet a la página local de nuestro servidor e intentaremos acceder a nuestro servicio mediante la URL de la clase, si el despliegue fue correcto se verá en pantalla un mensaje como el de la Figura 23.

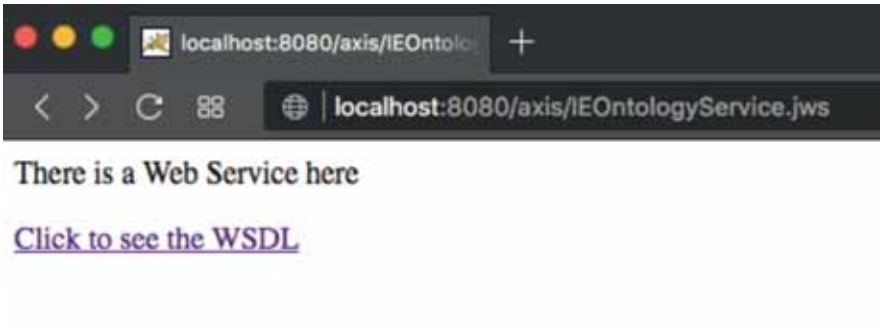

Figura 23 - Comprobación de despliegue correcto

Al hacer *click* en "*Click to see the WSDL"* nos direccionara al documento *WSDL* en sí, donde estarán listados todos los métodos, pero ya en forma de un esquema *XML*. También, después de dar *click* en el enlace en la barra de direcciones aparecerá la dirección de nuestro servicio web, que es la referencia que usaremos para poder crear clientes de este servicio (Figura 24). El código completo del archivo *WSDL* está reportado en el Apéndice B de este documento.

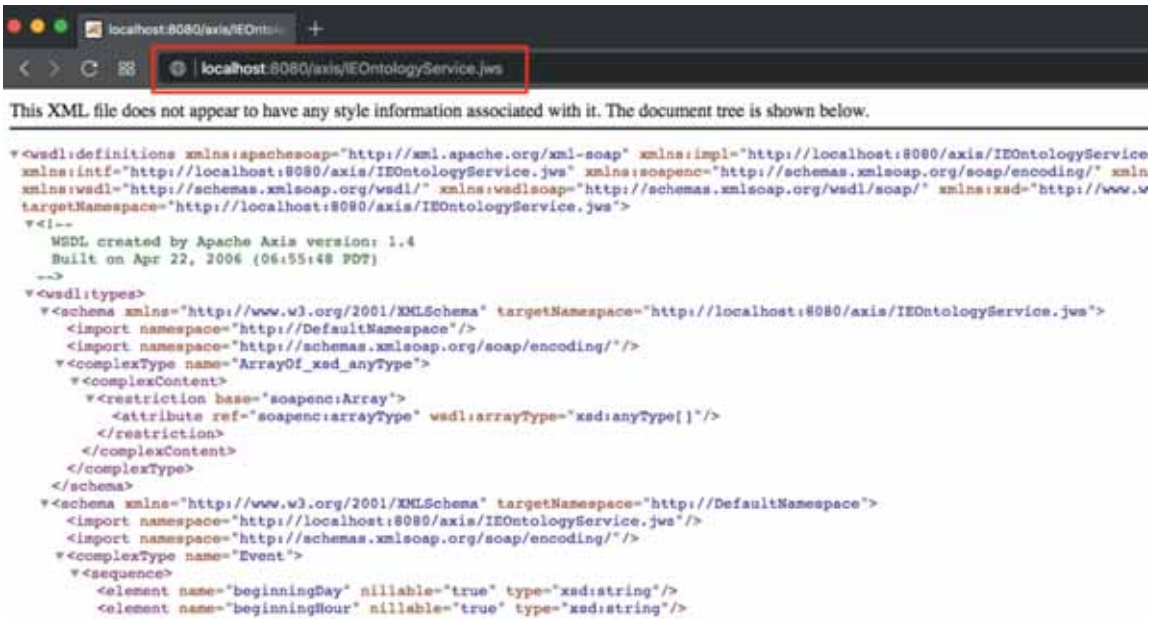

Figura 24 - URL del servicio y fragmento del código WSDL

6.5 Implementación de un cliente de evaluación

En esta parte del proyecto se diseñó e implementó un cliente para consumir el servicio web creado en la sección anterior; se decidió implementar este cliente en lenguaje de programación *C#* y el *IDE* Visual Studio, haciendo uso de la tecnología *WPF*[11] de *Microsoft*, la cual permite el desarrollo de interfaces de interacción en Windows tomando características de aplicaciones *Windows* y de aplicaciones web.

Se decidió implementar un cliente de pruebas en este lenguaje, en parte, para demostrar el nivel de interoperabilidad y estandarización que ofrece la tecnología de servicios web, ya que el servicio fue totalmente escrito en lenguaje *Java* y el cliente de invocación hace uso de él y lo manipula haciendo uso de lenguaje *C#.*

Una vez creado un proyecto *WPF* en Visual Studio lo primero que se hizo fue diseñar la vista de la ventana principal mediante el asistente visual de Visual Studio. El resultado fue una ventana sencilla con tres botones y una sección de texto donde se muestra retroalimentación a las acciones del usuario a modo de consola. El diseño de la ventana principal se muestra en la Figura 25.

[11] Windows Presentation Foundation

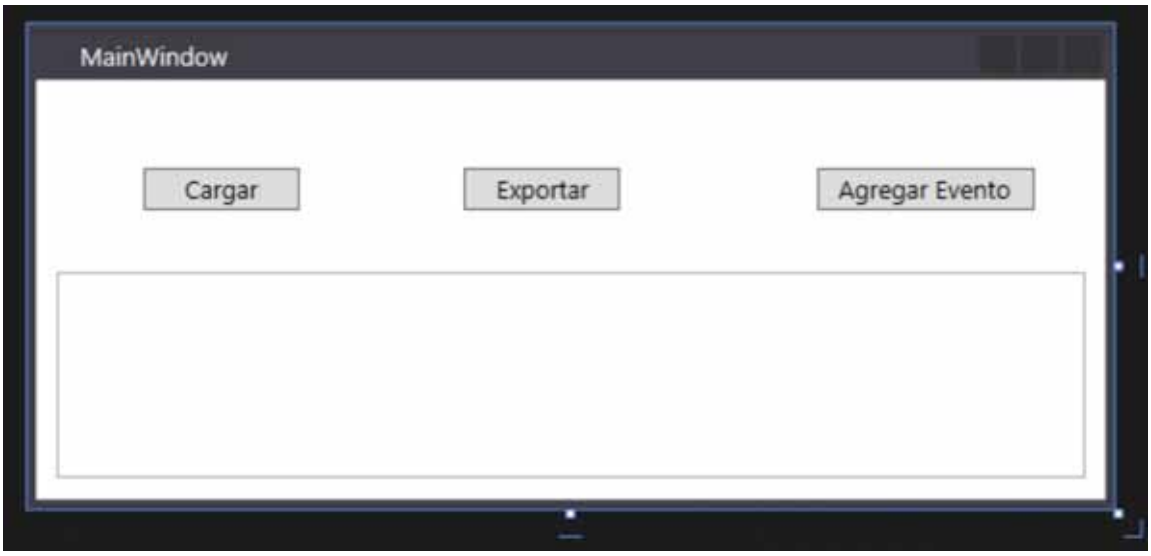

Figura 25 - Diseño de la ventana principal del cliente

El siguiente paso es agregar la referencia al servicio *WSDL* que habíamos obtenido anteriormente[12] y seguir el siguiente procedimiento:

1. En el explorador de soluciones, seleccionar el proyecto activo, en este caso *IEClient*, en la sección *Connected Services* hacer click derecho para desplegar las opciones y seleccionar agregar *referencia de servicio…*

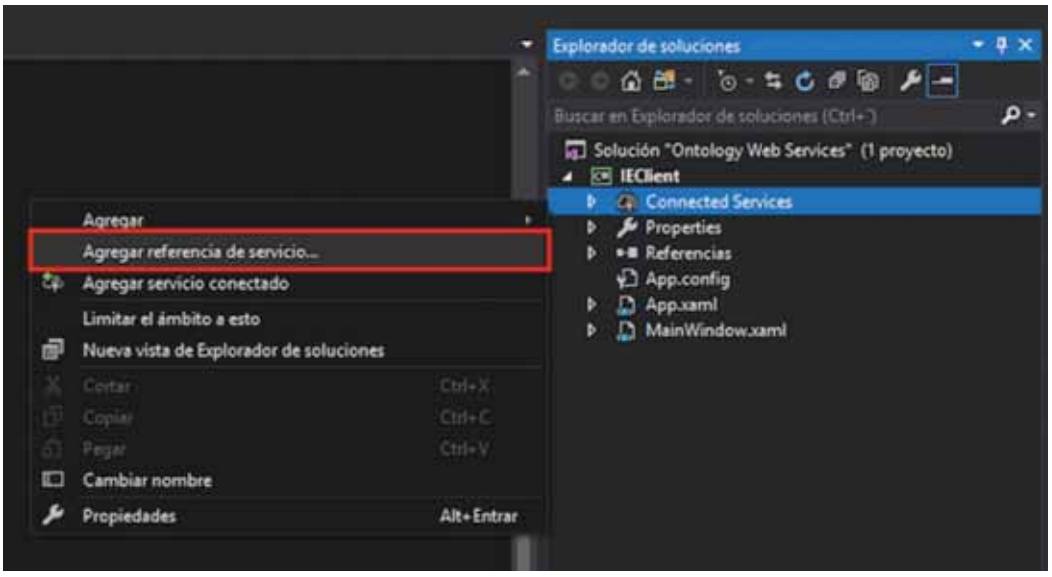

Figura 26 - Paso1, agregar referencia de servicio

[12] http://localhost:8080/axis/IEOntologyService.jws?wsdl

2. En la ventana emergente, en el campo de dirección, pegar la referencia del servicio obtenida anteriormente, asignar el espacio de nombres que se utilizará para el servicio dentro de la aplicación y seleccionar *Ir.*

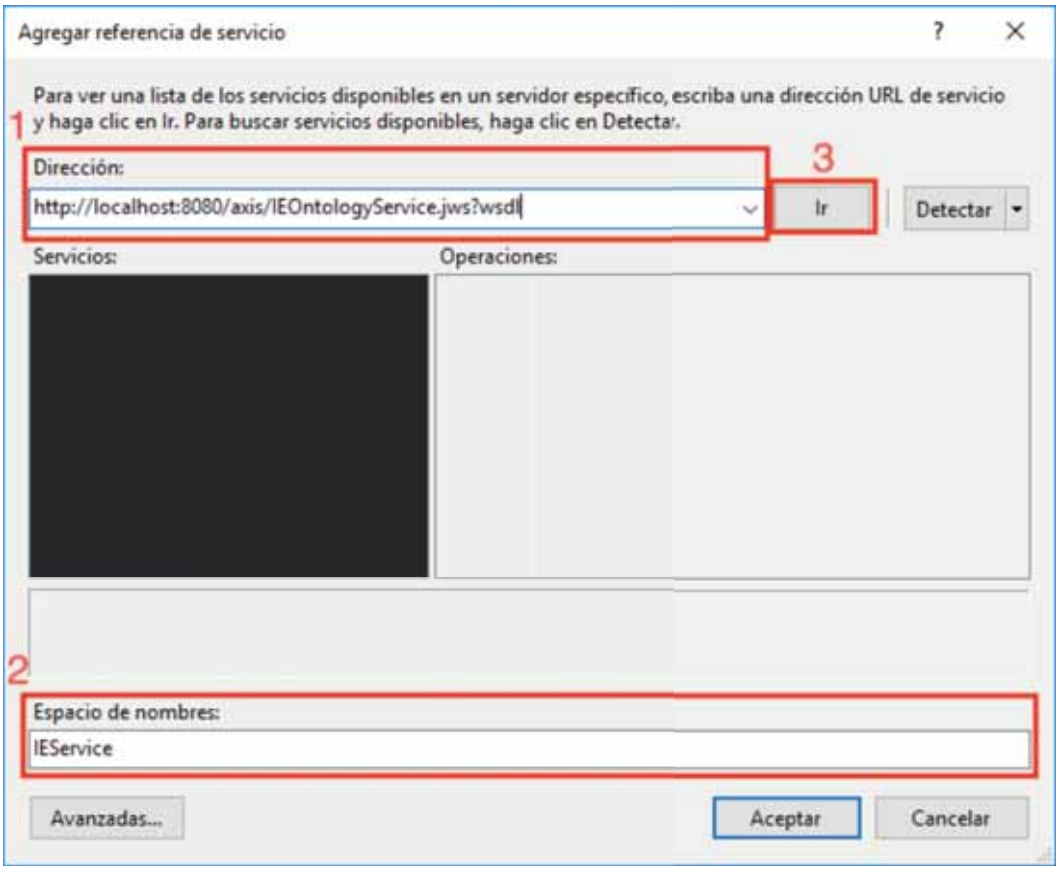

Figura 27 - Ventana agregar referencia de servicio

3. Después de seleccionar *Ir* se desplegarán los servicios encontrados en esa dirección, así como los métodos dichos servicios. Seleccionar *Aceptar*.

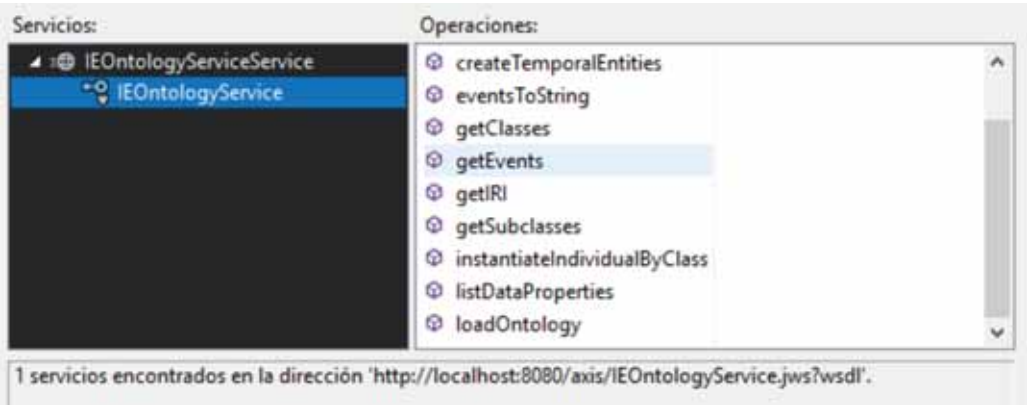

Figura 28 - Servicioes encontrados en la URL

4. Después basta con crear un objeto cliente dentro del código de ejecución de la aplicación para poder utilizar sus métodos.

```
public MainWindow()
{
    InitializeComponent();
}

string filename, export;

IEService.EOntologyServiceClient servicio = new IEService.EOntologyServiceClient();
```

Figura 29 - Creación de un objeto cliente

Una vez hecho el procedimiento anterior la aplicación está lista para utilizar el servicio web, por lo que se procede a darle la funcionalidad deseada a los botones de la ventana principal.

6.5.1 Funcionalidad de extracción

Lo primero que se tiene que hacer es cargar la ontología, en la Figura 29 puede verse arriba del texto marcado, dos variables de tipo *string*, estas variables se utilizaran para guardar la ruta de carga de la ontología y poder utilizarlo cada vez que se invoque un método; la otra contendrá la ruta seleccionada por el usuario para la exportación del archivo de salida.

El flujo de operación para la exportación de eventos es la siguiente:

1. Seleccionar la opción cargar, en la ventana emergente y buscar en el sistema el archivo de la ontología **IntelligentEnviroment**. Es importante mencionar que, para el correcto funcionamiento de la aplicación, todas las ontologías que importa deben estar almacenadas en el mismo directorio que la ontología general, ya que de otro modo tendrían que cargarse individualmente cada una.

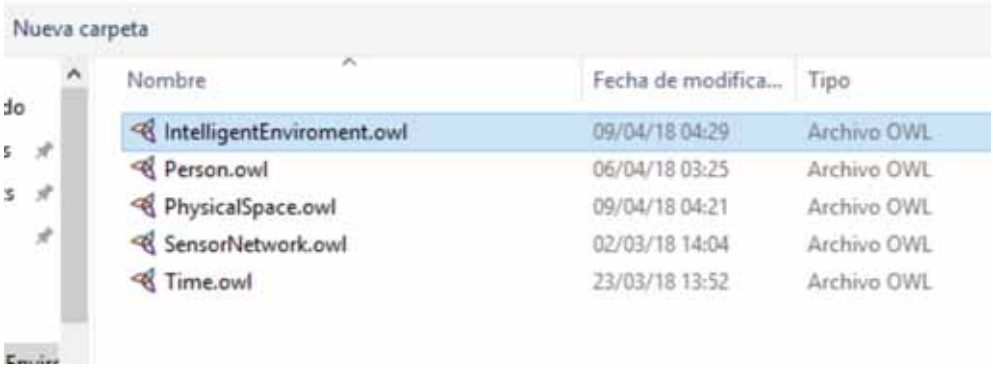

Figura 30 - Selección del archivo OWL de la ontología principal

Una vez seleccionado el archivo se selecciona abrir y en la consola de la ventana aparecerá un mensaje de éxito o fracaso, según sea el caso y mostrará también la ruta del sistema del archivo seleccionado.

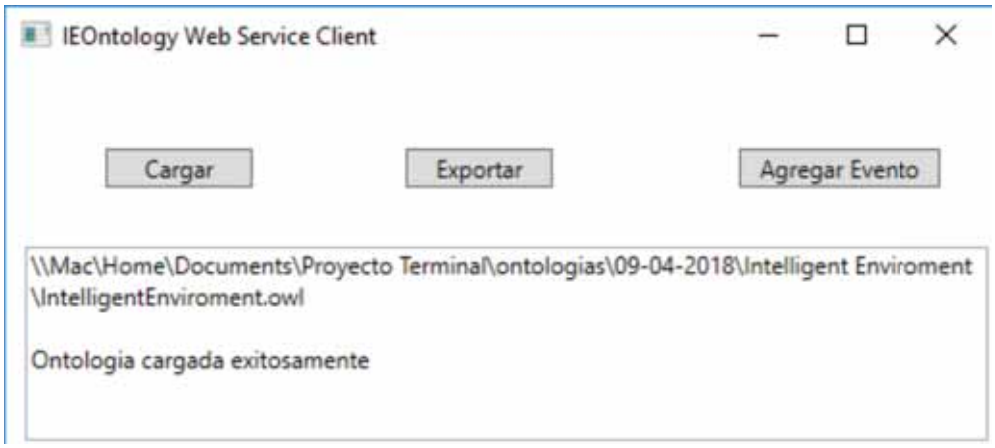

Figura 31 - Carga exitosa del archivo OWL

2. Al seleccionar la opción exportar, aparecerá una ventana emergente para seleccionar la ruta donde se desea guardar el archivo de salida. Al seleccionarla y guardar se creará el archivo y mostrará un mensaje de éxito y la ruta en donde se guardó el archivo.

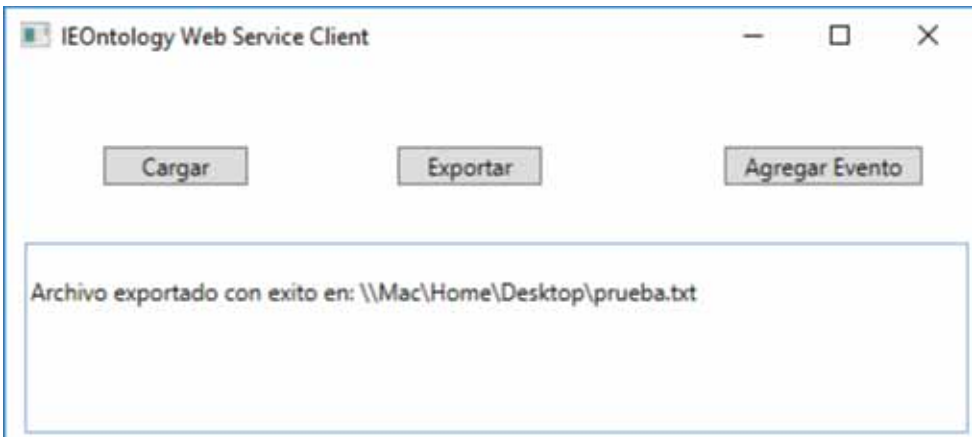

Figura 32 - Exportación exitosa del archivo

En la Figura 33 se muestra el fragmento de código perteneciente a las acciones realizadas al seleccionar la opción *cargar*.

```
private void Button_Click(object sender, RoutedEventArgs e)
{
    OpenFileDialog dlg = new OpenFileDialog();
    dlg.DefaultExt = ".owl";
    dlg.Filter = "Ontology (.owl)|*.owl";
    Nullable<bool> result = dlg.ShowDialog();

    if (result == true)
    {
        filename = dlg.FileName;
        textBox.Text = filename + "\n";
        if (servicio.loadOntology(filename)) textBox.Text += "\nOntología cargada exitosamente";
        else textBox.Text += "\nNo se pudo cargar la ontología";
    }
}
```

Figura 33 - Código de las acciones realizadas al hacer click en cargar

6.5.2 Funcionalidad de agregar evento

Para la funcionalidad de agregar un evento se diseñó otra ventana WPF, esta vez con un formulario para llenar todos los campos requeridos. Este formulario se despliega simultáneamente con la ventana principal, pero durante la ejecución de ésta, la principal es inaccesible hasta terminar la operación o cancelar. El diseño del formulario se presenta en la Figura 34.

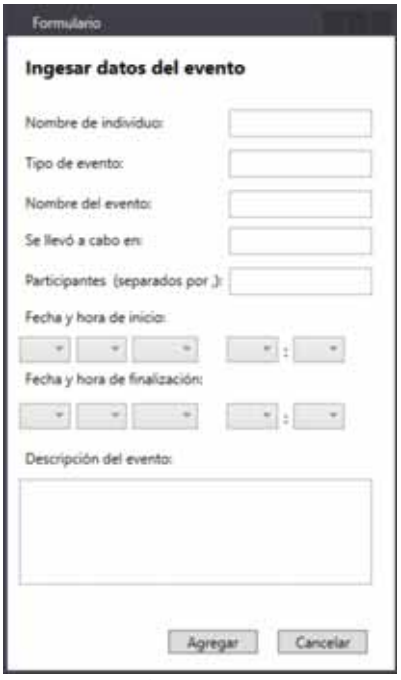

Figura 34 - Formulario para agregar un evento

El flujo de operación para agregar un evento es el siguiente:

1. Una vez cargada la ontología con éxito se habilita el botón para agregar, al seleccionarlo se abrirá una ventana emergente mostrando el formulario. Deberán llenarse todos los campos de texto, pudiendo dejar los campos de fecha y hora de finalización vacíos, en caso de que se desconozca la fecha y horas de finalización del evento a agregar.

2. Al dar click en el botón agregar se ejecutará el método *createEvent* del servicio web, recuperando los datos de todos los campos del formulario para construir el evento, y se devolverá un mensaje al usuario en caso de éxito o fracaso en forma de una pequeña ventana emergente, al cerrar dicha ventana se regresará el control a la ventana principal, pudiendo exportar los eventos o agregar un nuevo individuo.

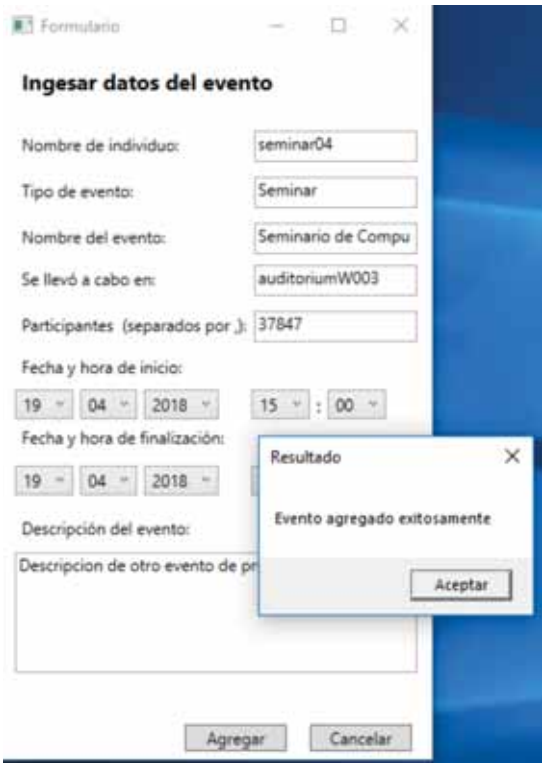

Figura 35 – Mensaje de éxito o fracaso en la operación

En la Figura 36 se muestra el fragmento de código perteneciente a la acción realizada al seleccionar la opción *agregar*, en la ventana de formulario.

```
private void Button_Click(object sender, RoutedEventArgs e)
{
    IEWebService.Event evento = new IEWebService.Event();
    evento.individualName = txtIndividuo.Text;
    evento.type = txtTipo.Text;
    evento.eventName = txtNombre.Text;
    evento.description = txtDescripcion.Text;
    evento.happensIn = txtLugar.Text;
    evento.participants = txtParticipantes.Text.Split(',');
    evento.beginningDay = comboDia.SelectedValue.ToString();
    evento.beginningMonth = comboMes.SelectedValue.ToString();
    evento.beginningYear = comboAnio.SelectedValue.ToString();
    evento.beginningHour = comboHora.SelectedValue.ToString();
    evento.beginningMinute = comboMinuto.SelectedValue.ToString();
    evento.endDay = comboDiaEnd.SelectedValue.ToString();
    evento.endMonth = comboMesEnd.SelectedValue.ToString();
    evento.endYear = comboAnioEnd.SelectedValue.ToString();
    evento.endHour = comboHoraEnd.SelectedValue.ToString();
    evento.endMinute = comboMinutoEnd.SelectedValue.ToString();

    if (servicio.addEvent(evento))
    {
        MessageBox.Show("Evento agregado exitosamente", "Resultado", MessageBoxButton.OK);
        this.Close();
    }
}
```

Figura 36 - Código de las acciones al hacer click en agregar

7. Resultados

Para las pruebas del producto final de este proyecto, se utilizó el cliente de invocación descrito en la sección anterior; se probaron los dos métodos principales y se analizó en ambos casos la validez de los resultados obtenidos. A continuación, se presentan los resultados obtenidos para cada una de las funciones principales.

7.1 Extracción de eventos

Después de haber realizado el procedimiento de operación para la función de exportar eventos, detallado en la sección 6.5.1, se obtiene como resultado de la exportación un archivo de texto plano de extensión *.txt,* en donde se representa la información de todos los eventos recuperados de la ontología, dispuesta en forma de columnas. El esquema de exportación en columnas se muestra en la Figura 37.

nP. Estu	nP. Prof	nP. Visi	Fecha Inicio	Fecha Fin	tLugar	tEspacio	Clase Evento	Nombre Evento	Descripción

Figura 37 - Cabecera del documento exportado

En la figura anterior se muestra los campos que conforman la cabecera del archivo resultante de la exportación de eventos, en el archivo de texto está separado cada campo por el carácter de retorno “\t”, tabulador. Se dispuso de esa manera para que los usuarios finales de esta capa de servicios puedan tratar estos archivos con facilidad para recuperar la información que ellos requieran desde otras aplicaciones.

A continuación, en la Figura 38 se expone el archivo resultado de la exportación. El archivo generado es consistente con los datos correspondientes a los cinco eventos que existen en la ontología y todos los campos resultan correctos. En la Figura 39, se muestra una imagen comparativa del software *Protégé*, en donde se presentan los individuos gráficamente los individuos que contiene la ontología.

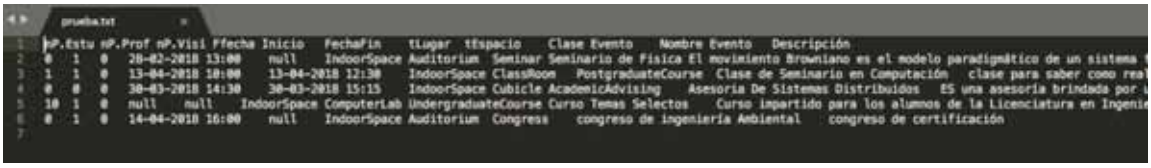

Figura 38 - Archivo resultante de exportación

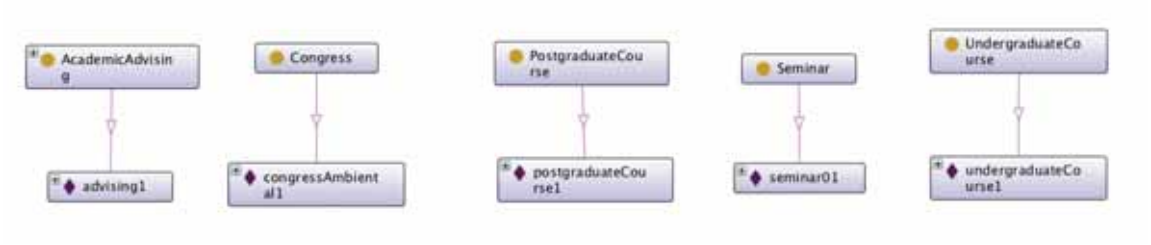

Figura 39 - Individuos exportados de la ontología

Respecto al documento resultante de la exportación se puede apreciar que resulta un poco difícil identificar a qué campo pertenece cada campo en los distintos renglones, esto debido a que algunos campos tienen más extensión que otros y sólo los separa un carácter de tabulador, por lo que se puede llegar a perder el orden; este detalle no representa ninguna dificultad a la hora de obtener información desde estos archivos ya que al hacer uso únicamente de tabuladores, se pueden obtener fácilmente los campos deseados desde código.

También se pueden apreciar algunos campos con el valor “*null*”; esto se decidió a manera de convención para homologar los campos en donde no se encontrará información dentro de la ontología, así, los usuarios finales se estos archivos pueden saber cuándo un campo no tiene un valor registrado en la ontología y tratarlo en la manera que mejor les convenga.

Al mismo tiempo, la separación que se escogió para cada campo en el archivo hace sencilla su visualización en software de hojas de cálculo, como Excel, ya que reconoce los separadores de tabulador como una columna y lo ordena de manera correcta. Un ejemplo de ello se muestra en la Figura 40.

A	B	C	D	E	F	G	H	I
nP.Estu	nP.Prof	nP.Visi	Ffecha Inicio	FechaFin	tLugar	tEspacio	Clase Evento	Nombre Evento
0	1	0	28/02/18 13:00	null	IndoorSpace	Auditorium	Seminar	Seminario de Fisica
1	1	0	13/04/18 10:00	13/04/18 12:30	IndoorSpace	ClassRoom	PostgraduateCourse	Clase de Seminario en ComputaciÝzn
0	0	0	30/03/18 14:30	30/03/18 15:15	IndoorSpace	Cubicle	AcademicAdvising	Asesoria De Sistemas Distribuidos
10	1	0	null	null	IndoorSpace	ComputerLab	UndergraduateCourse	Curso Temas Selectos
0	1	0	14/04/18 16:00	null	IndoorSpace	Auditorium	Congress	congreso de ingenierÝ≠a Ambiental

Figura 40 - Archivo exportado, abierto en Excel

7.2 Inserción de eventos

Para mostrar el resultado de esta funcionalidad se hace uso nuevamente del software *Protégé*, en donde se verán reflejados los cambios en forma de nuevos individuos agregados. Cada individuo tendrá todas las propiedades y relaciones que lo componen y se ajustarán a la nomenclatura establecida por los desarrolladores de estas ontologías.

Después realizar las operaciones presentadas en la sección 6.5.2 para agregar el evento presentado en la Figura 41, utilizando el formulario para agregar evento, se pueden observar los resultados reflejados en *Protégé*, presentados en la Figura 42.

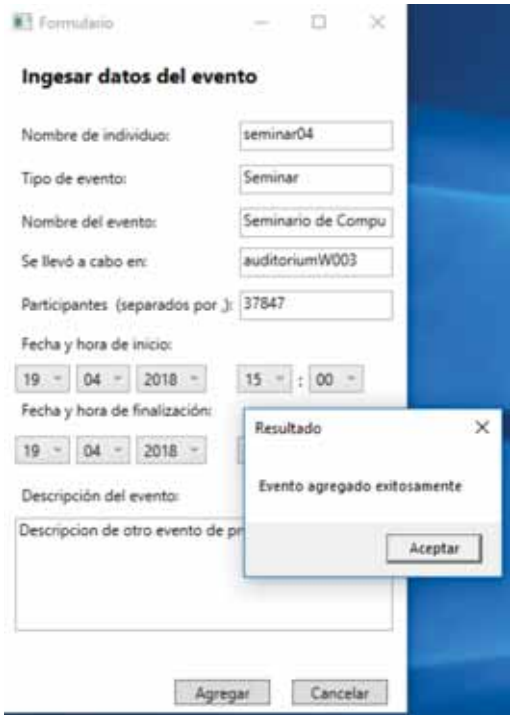

Figura 41 - Ejemplo de creación de un evento

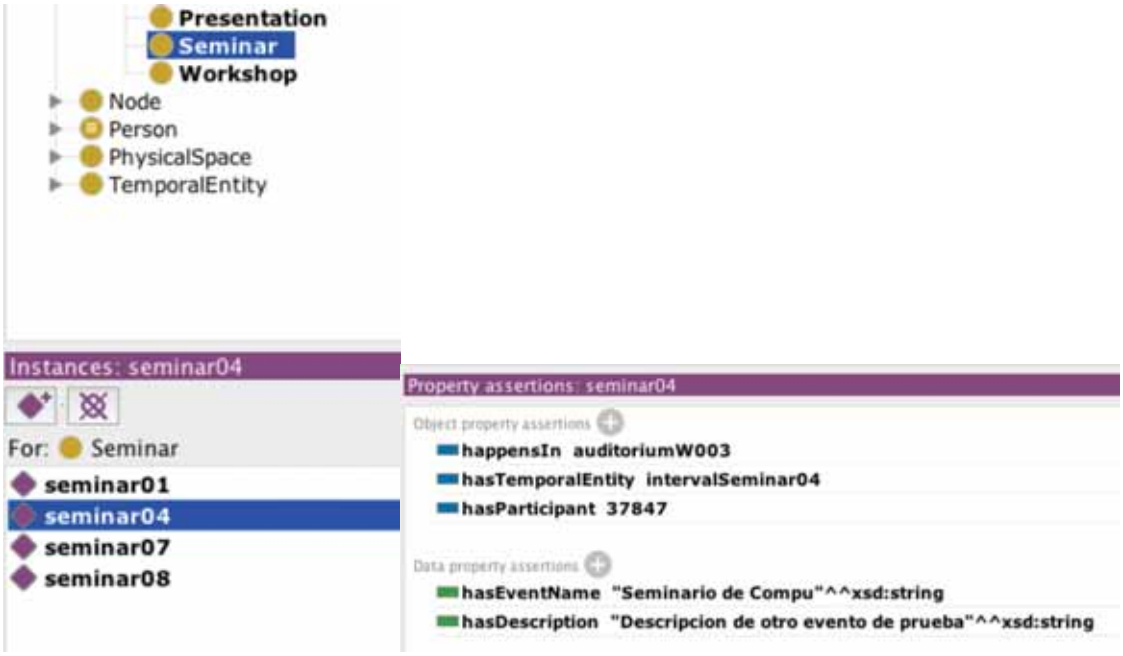

Figura 42 - Resultados de agregar evento

Como se puede ver en la Figura anterior, se creó el individuo junto con todos sus atributos y relaciones; también es importante corroborar que se hayan creado correctamente los individuos a los cuales hacen referencia sus relaciones. En este caso debe ser un individuo de la clase *Interval* que relaciona a dos individuos de la clase *Instant*, uno para almacenar la fecha y hora de inicio y otro para la de finalización. En la Figura 43 se muestra el individuo *Interval* creado correctamente y sus relaciones; en la Figura 44 se pueden ver ambos instantes creados y las propiedades de uno de ellos.

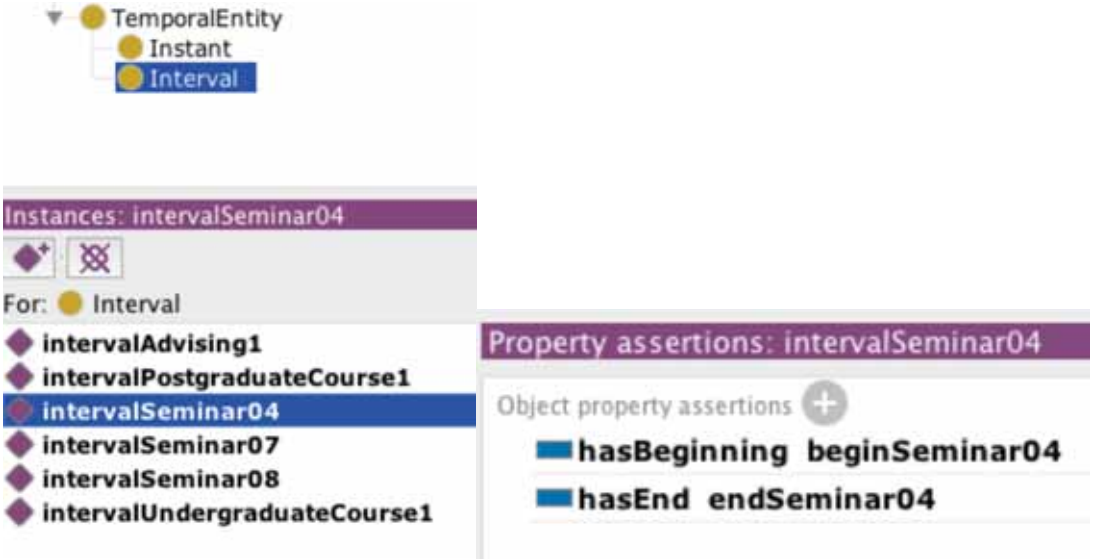

Figura 43 - Individuos de la clase Instant

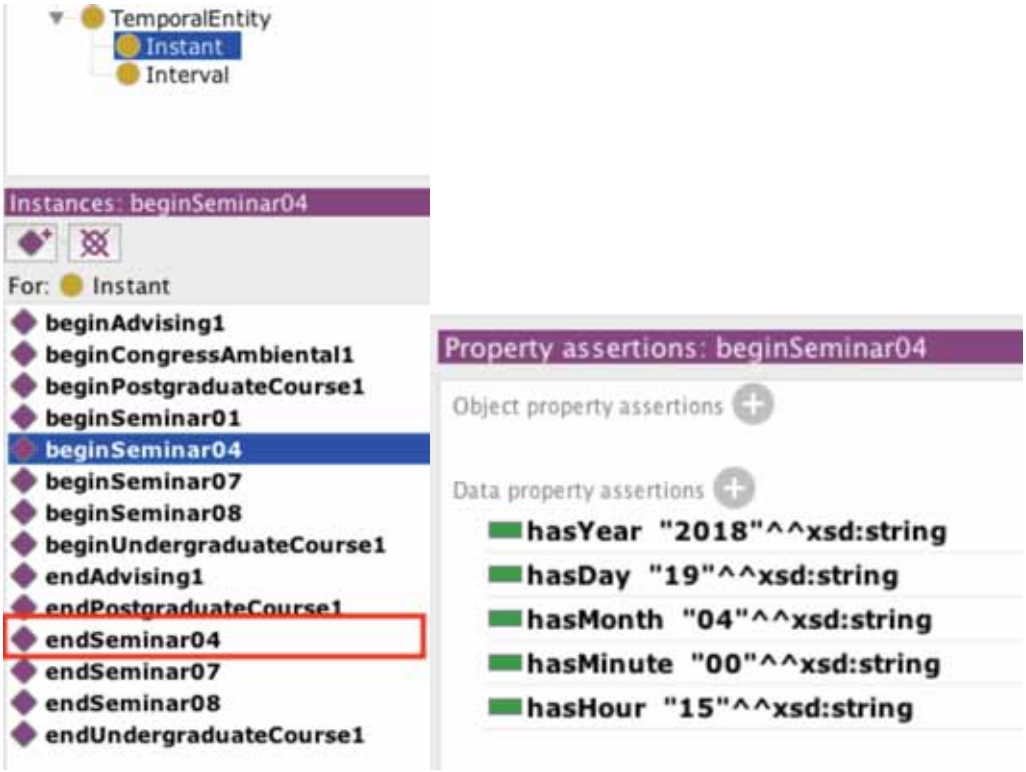

Figura 44 - Individuos de la clase Instant

8. Conclusiones

Con base en los resultados reportados en la sección anterior, se puede concluir que el objetivo general planteado para la realización de este proyecto fue cumplido satisfactoriamente, ya que se consiguió implementar y probar una capa de software que, mediante un servicio web administra la información de un sistema consciente del contexto basado en ontologías.

En cuanto a los objetivos específicos, algunos no fueron cubiertos en su totalidad como en el caso de la administración de información sobre personas o sensores debido a que, de acuerdo con los usuarios finales de este software, no resultaría de mucha utilidad poder administrar esa información ya que no es común agregar instancias de esos tipos, por lo que este servicio web se enfoca en la administración de eventos.

Como trabajo a futuro, el presente software podría ser modificado de modo que sea capaz de administrar información de modelos ontológicos más generalizados, ya que de momento se encuentra fuertemente acoplado con las ontologías para administrar un espacio inteligente presentadas en la sección 6.1 de este documento.

9. Bibliografía

[1] J. S. Revelo, «Ontología genérica para la integración de servicios web semánticos,» Universidad Autónoma Metropolitana, México, 2014.

[2] G. M. Torres, «Arquitectura orientada a servicios web para establecer grupos de colaboración entre investigadores,» Universidad Autónoma Metropolitana, México, 2017.

[3] D. H. Rubio, «Extracción y representación de servicios web escritos en SAWSDL mediante una ontología,» Universidad Autónoma Metropolitana, México, 2014.

[4] P. C. Ávila, «Espacios inteligentes con manipuladores móviles dotados de intuición artificial,» Universidad Uatónoma de México, México, 2015.

[5] J. A. R. O. Maricela Bravo, «Hybrid Architecture to suppport Context-Aware Systems,» de *Multi-agent Systems*, México, INTECH, 2017.

[6] X. H. W. H. K. P. D. Q. Z. T. Gu, «An Ontology-based Context Model in Intelligent Evironments,» Institute for Infocomm Research, Singapur, 2004.

[7] D. L. McGuinness, «OWL Web Ontology Language Overview,» 2004. [En línea]. Available: https://www.w3.org/TR/owl-features/. [Último acceso: 2018].

[8] M. K. Smith, «OWL Web Ontology Language Guide,» 2004. [En línea]. Available: https://www.w3.org/TR/2004/REC-owl-guide-20040210/. [Último acceso: 2018].

[9] D. L. M. Dan Connolly, «DAML+OIL (March 2001) Reference Description,» 2001. [En línea]. Available: https://www.w3.org/TR/daml+oil-reference. [Último acceso: 2018].

[10] P. H. Peter F. Patel-Schneider, «OWL Web Ontology Language Semantics and Abstract Syntax,» 2004. [En línea]. Available: https://www.w3.org/TR/2004/REC-owl-semantics-20040210/. [Último acceso: 2018].

[11] IBM Knowledge Center, «What is a web service?,» 2018. [En línea]. Available: https://www.ibm.com/support/knowledgecenter/en/SSGMCP_5.3.0/com.ibm.cics.ts.webservices.doc/concepts/dfhws_definition.html. [Último acceso: 2018].

[12] IBM Knowledge Center, «¿Qué es WSDL?,» 2014. [En línea]. Available: https://www.ibm.com/support/knowledgecenter/es/SSMKHH_9.0.0/com.ibm.etools.mft.doc/ac34640_.htm. [Último acceso: 2018].

[13] Apache Software Foundation, «Axis,» 2015. [En línea]. Available: http://axis.apache.org/axis/java/. [Último acceso: 2018].

[14] J. S. Revelo, «Ontología genértica para la integración de servicios web semánticps,» Universidad Autónoma Metropolitana, México, 2014.

10. Apéndices

10.1 Apéndice A

10.1.1 Fragmento de código de ontología Person

```
<?xml version="1.0"?>
<Ontology xmlns="http://www.w3.org/2002/07/owl#"
     xml:base="http://www.academia_uam.com/informacion_academica/ontologies/2016/Person"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:xml="http://www.w3.org/XML/1998/namespace"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     ontologyIRI="http://www.academia_uam.com/informacion_academica/ontologies/2016/Person">
    <Prefix name=""
IRI="http://www.academia_uam.com/informacion_academica/ontologies/2016/Person"/>
    <Prefix name="owl" IRI="http://www.w3.org/2002/07/owl#"/>
    <Prefix name="per"
IRI="http://www.academia_uam.com/informacion_academica/ontologies/2016/persona.owl"/>
    <Prefix name="rdf" IRI="http://www.w3.org/1999/02/22-rdf-syntax-ns#"/>
    <Prefix name="xml" IRI="http://www.w3.org/XML/1998/namespace"/>
    <Prefix name="xsd" IRI="http://www.w3.org/2001/XMLSchema#"/>
    <Prefix name="rdfs" IRI="http://www.w3.org/2000/01/rdf-schema#"/>
    <Declaration>
        <DataProperty IRI="#hasProject"/>
    </Declaration>
…
```

10.1.2 Fragmento de código de ontología PhysicalSpace

```
<?xml version="1.0"?>
<Ontology xmlns="http://www.w3.org/2002/07/owl#"
     xml:base="http://www.semanticweb.org/usuario/ontologies/2016/0/PhysicalSpace"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:xml="http://www.w3.org/XML/1998/namespace"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     ontologyIRI="http://www.semanticweb.org/usuario/ontologies/2016/0/PhysicalSpace">
    <Prefix name=""
IRI="http://www.semanticweb.org/usuario/ontologies/2016/0/PhysicalSpace"/>
    <Prefix name="owl" IRI="http://www.w3.org/2002/07/owl#"/>
    <Prefix name="rdf" IRI="http://www.w3.org/1999/02/22-rdf-syntax-ns#"/>
    <Prefix name="xml" IRI="http://www.w3.org/XML/1998/namespace"/>
    <Prefix name="xsd" IRI="http://www.w3.org/2001/XMLSchema#"/>
    <Prefix name="rdfs" IRI="http://www.w3.org/2000/01/rdf-schema#"/>
    <Declaration>
        <DataProperty IRI="#hasPeopleCapacity"/>
    </Declaration>
    <Declaration>
        <NamedIndividual IRI="#auditoriumW003"/>
    </Declaration>
…
```

10.1.3 Fragmento de código de ontología SensorNetwork

```
<?xml version="1.0"?>
<Ontology xmlns="http://www.w3.org/2002/07/owl#"
     xml:base="http://www.semanticweb.org/profesor/ontologies/2017/10/SensorNetwork2"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:xml="http://www.w3.org/XML/1998/namespace"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     ontologyIRI="http://www.semanticweb.org/profesor/ontologies/2017/10/SensorNetwork2">
```

```
    <Prefix name=""
IRI="http://www.semanticweb.org/profesor/ontologies/2017/10/SensorNetwork2"/>
    <Prefix name="owl" IRI="http://www.w3.org/2002/07/owl#"/>
    <Prefix name="rdf" IRI="http://www.w3.org/1999/02/22-rdf-syntax-ns#"/>
    <Prefix name="xml" IRI="http://www.w3.org/XML/1998/namespace"/>
    <Prefix name="xsd" IRI="http://www.w3.org/2001/XMLSchema#"/>
    <Prefix name="rdfs" IRI="http://www.w3.org/2000/01/rdf-schema#"/>
    <Declaration>
        <NamedIndividual IRI="#microArduinoMega1"/>
    </Declaration>
    <Declaration>
        <NamedIndividual IRI="#RFIDReader1"/>
    </Declaration>
    <Declaration>
        <DataProperty IRI="#hasIPAddress"/>
    </Declaration>
…
```

10.1.4 Fragmento de código de ontología Time

```
<?xml version="1.0"?>
<Ontology xmlns="http://www.w3.org/2002/07/owl#"
     xml:base="http://www.semanticweb.org/usuario/ontologies/2018/2/Time"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:xml="http://www.w3.org/XML/1998/namespace"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     ontologyIRI="http://www.semanticweb.org/usuario/ontologies/2018/2/Time">
    <Prefix name="" IRI="http://www.semanticweb.org/usuario/ontologies/2018/2/Time"/>
    <Prefix name="owl" IRI="http://www.w3.org/2002/07/owl#"/>
    <Prefix name="rdf" IRI="http://www.w3.org/1999/02/22-rdf-syntax-ns#"/>
    <Prefix name="xml" IRI="http://www.w3.org/XML/1998/namespace"/>
    <Prefix name="xsd" IRI="http://www.w3.org/2001/XMLSchema#"/>
    <Prefix name="rdfs" IRI="http://www.w3.org/2000/01/rdf-schema#"/>
    <Declaration>
        <DataProperty IRI="#hasYear"/>
    </Declaration>
    <Declaration>
        <Class IRI="#TemporalEntity"/>
    </Declaration>
    <Declaration>
        <Class IRI="#Interval"/>
    </Declaration>
    <Declaration>
        <DataProperty IRI="#hasMonth"/>
    </Declaration>
    <Declaration>
        <ObjectProperty IRI="#hasBeginning"/>
    </Declaration>
    <Declaration>
        <DataProperty IRI="#hasMinute"/>
    </Declaration>
…
```

10.1.5 Fragmento de código de ontología Intelligent Enviroment

```
<?xml version="1.0"?>
<Ontology xmlns="http://www.w3.org/2002/07/owl#"
     xml:base="http://www.semanticweb.org/profesor/ontologies/2018/2/IntelligentEnviroment"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:xml="http://www.w3.org/XML/1998/namespace"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"

ontologyIRI="http://www.semanticweb.org/profesor/ontologies/2018/2/IntelligentEnviroment">
    <Prefix name=""
IRI="http://www.semanticweb.org/profesor/ontologies/2018/2/IntelligentEnviroment#"/>
    <Prefix name="owl" IRI="http://www.w3.org/2002/07/owl#"/>
    <Prefix name="rdf" IRI="http://www.w3.org/1999/02/22-rdf-syntax-ns#"/>
```

```
    <Prefix name="xml" IRI="http://www.w3.org/XML/1998/namespace"/>
    <Prefix name="xsd" IRI="http://www.w3.org/2001/XMLSchema#"/>
    <Prefix name="rdfs" IRI="http://www.w3.org/2000/01/rdf-schema#"/>
    <Import>http://www.semanticweb.org/profesor/ontologies/2017/10/SensorNetwork2</Import>
    <Import>http://www.semanticweb.org/usuario/ontologies/2016/0/PhysicalSpace</Import>

<Import>http://www.academia_uam.com/informacion_academica/ontologies/2016/Person</Import>
    <Import>http://www.semanticweb.org/usuario/ontologies/2018/2/Time</Import>
    <Declaration>
        <Class
IRI="http://www.semanticweb.org/temporal/ontologies/2018/2/IntelligentEnviroment#Presentatio
n"/>
    </Declaration>
    <Declaration>
        <Class
IRI="http://www.semanticweb.org/temporal/ontologies/2018/2/IntelligentEnviroment#AcademicAdv
ising"/>
    </Declaration>
…
```

10.2 Apéndice B

10.2.1 Fragmento de Código de la clase Event en Java

Se omite la parte de los métodos de acceso

```
public class Event implements Serializable {
    private String individualName;
    private String eventName = "null";
    private String description = "null";
    private String type = "null";
    private String place = "null";
    private String placeType = "null";
    private String happensIn;
    private String beginningDay="null", beginningMonth = "null", beginningYear = "null";
    private String beginningHour="null", beginningMinute = "null";
    private String endDay="null", endMonth="null", endYear = "null";
    private String endHour="null", endMinute = "null";
    private int nStudent, nProfessor, nVisitor;
    private ArrayList<String> participantsTypes = new ArrayList();
    private ArrayList<String> participants = new ArrayList();

    public Event(){}
   …
public void countParticipants(){
        nStudent = 0; nProfessor=0; nVisitor=0;
        System.out.println(participantsTypes.size());
        if(!participantsTypes.isEmpty()){
            for(String s : participantsTypes){
                if(s.toLowerCase().contains("professor")) nProfessor++;
                else if(s.toLowerCase().contains("student")) nStudent++;
                else if(s.toLowerCase().contains("visitor")) nVisitor++;
            }
        }
    }

    public String toStringEnd(){
        String s = "";
        if(endDay=="null"||endMonth=="null"||endYear=="null") s = "null";
        else{
            s += this.endDay+"-"+this.endMonth+"-"+this.endYear+"
"+this.endHour+":"+this.endMinute;
        }
        return s;
    }

    public String toStringBeginning(){
```

```
        String s = "";
        if(beginningDay=="null"||beginningMonth=="null"||beginningYear=="null") s = "null";
        else{
            s += beginningDay+"-"+beginningMonth+"-"+beginningYear+"
"+beginningHour+":"+beginningMinute;
        }
        return s;
    }

    public String toString(){
        nStudent = 0; nProfessor=0; nVisitor=0;
        if(!participantsTypes.isEmpty()){
            for(String s : participantsTypes){
                if(s.toLowerCase().contains("professor")) nProfessor++;
                else if(s.toLowerCase().contains("student")) nStudent++;
                else if(s.toLowerCase().contains("visitor")) nVisitor++;
            }
        }
        String s = "";
        s +=  nStudent + "\t" + nProfessor + "\t" + nVisitor + "\t" + toStringBeginning() +
"\t" + toStringEnd()
                + "\t" + placeType + "\t" + place + "\t" + type + "\t" + eventName + "\t" +
description;
        return s;
    }
}
```

10.2.2 Fragmento de código del método getEvents

```
public ArrayList<Event> getEvents() {
        //loadOntology(ontofile);
        ArrayList<Event> salida = new ArrayList();
        ArrayList<IRI> clases = new ArrayList();

        for (String s : this.getSubclasses("IntelligentEnviroment")) {

clases.add(IRI.create("http://www.semanticweb.org/temporal/ontologies/2018/2/IntelligentEnvi
roment#" + s));
        }

        System.out.println("Eventos:\n");
        for (IRI i : clases) {
            OWLClass classOWL = factory.getOWLClass(i);
            Set<OWLIndividual> individuos = classOWL.getIndividuals(ontology);
            if (!individuos.isEmpty()) {
                byte inicio = (byte) i.toString().indexOf("#");
                for (OWLIndividual ind : individuos) {
                    //System.out.println(ind.toStringID());
                    System.out.println(i.toString().substring(i.toString().indexOf("#") +
1));
                    Event e = new Event();
                    e.setType(i.toString().substring(i.toString().indexOf("#") + 1));
                    OWLDataPropertyImpl nombre = new OWLDataPropertyImpl(factory,
IRI.create("http://www.semanticweb.org/temporal/ontologies/2018/2/IntelligentEnviroment#hasE
ventName"));
                    Set<OWLLiteral> valores = ind.getDataPropertyValues(nombre, ontology);
                    for (OWLLiteral valor : valores) {
                        String v = valor.getLiteral();
                        e.setEventName(v);
                        System.out.println(v);
                    }
                    OWLDataPropertyImpl descripcion = new OWLDataPropertyImpl(factory,
IRI.create("http://www.semanticweb.org/temporal/ontologies/2018/2/IntelligentEnviroment#hasD
escription"));
                    valores = ind.getDataPropertyValues(descripcion, ontology);
                    for (OWLLiteral valor : valores) {
                        String v = valor.getLiteral();
                        e.setDescription(v);
                        System.out.println(v);
```

```
                    }
                    OWLObjectPropertyImpl lugar = new OWLObjectPropertyImpl(factory,
IRI.create("http://www.semanticweb.org/profesor/ontologies/2018/2/IntelligentEnviroment#happ
ensIn"));
                    Set<OWLIndividual> opIndividuals = ind.getObjectPropertyValues(lugar,
ontology);
                    for (OWLIndividual opI : opIndividuals) {
                        for (OWLClassExpression oc : opI.getTypes(ontologias)) {
                            NodeSet<OWLClass> superclasses = reasoner.getSuperClasses(oc,
true);
                            for (Node<OWLClass> oc2 : superclasses) {
                                String superclass =
oc2.toString().substring(oc2.toString().indexOf("#") + 1, oc2.toString().indexOf(">"));
                                System.out.println(superclass);
                                e.setPlaceType(superclass);
                            }
                            inicio = (byte) oc.toString().indexOf("#");
                            String value = oc.toString().substring(inicio + 1);
                            value = value.replaceAll(">", "");
                            e.setPlace(value);
                            System.out.println(value);
                        }
                    }
                    OWLObjectPropertyImpl participant = new OWLObjectPropertyImpl(factory,
IRI.create("http://www.semanticweb.org/profesor/ontologies/2018/2/IntelligentEnviroment#hasP
articipant"));
                    opIndividuals = ind.getObjectPropertyValues(participant, ontology);
                    if (opIndividuals.isEmpty()) {
                        System.out.println("Sin participantes registrados");
                    }
                    for (OWLIndividual opI : opIndividuals) {
                        for (OWLClassExpression oc : opI.getTypes(ontologias)) {
                            inicio = (byte) oc.toString().indexOf("#");
                            String value = oc.toString().substring(inicio + 1);
                            value = value.replaceAll(">", "");
                            e.addParticipantType(value);
                            System.out.println(value);
                        }
                        inicio = (byte) opI.toString().indexOf("#");
                        e.addParticipant(opI.toStringID().substring(inicio));
                        System.out.println(opI.toStringID().substring(inicio));
                    }

                    OWLObjectPropertyImpl duration = new OWLObjectPropertyImpl(factory,
IRI.create("http://www.semanticweb.org/profesor/ontologies/2018/2/IntelligentEnviroment#hasT
emporalEntity"));
                    opIndividuals = ind.getObjectPropertyValues(duration, ontology);
                    if (opIndividuals.isEmpty()) {
                        System.out.println("Sin duracion registrada");
                    }
                    for (OWLIndividual opI : opIndividuals) {
                        if(opI.toStringID().contains("interval")){
                            OWLObjectPropertyImpl beginning = new
OWLObjectPropertyImpl(factory,
IRI.create("http://www.semanticweb.org/usuario/ontologies/2018/2/Time#hasBeginning"));
                            Set<OWLIndividual> opIndividuals2 =
opI.getObjectPropertyValues(beginning, ontology);
                            for (OWLIndividual opI2 : opIndividuals2) {
                                Set<OWLDataProperty> properties =
ontology.getDataPropertiesInSignature();
                                for (OWLDataProperty p : properties) {
                                    if (p.toStringID().contains("Time")) {
                                        if (p.toStringID().contains("Day")) {
                                            Set<OWLLiteral> values =
opI2.getDataPropertyValues(p, ontology);
                                            for (OWLLiteral value : values) {
                                                e.setBeginningDay(value.getLiteral());
                                            }
                                        } else if (p.toStringID().contains("Month")) {
                                            Set<OWLLiteral> values =
opI2.getDataPropertyValues(p, ontology);
```

```
                                        for (OWLLiteral value : values) {
                                            e.setBeginningMonth(value.getLiteral());
                                        }
                                    } else if (p.toStringID().contains("Year")) {
                                        Set<OWLLiteral> values =
opI2.getDataPropertyValues(p, ontology);
                                        for (OWLLiteral value : values) {
                                            e.setBeginningYear(value.getLiteral());
                                        }
                                    } else if (p.toStringID().contains("Hour")) {
                                        Set<OWLLiteral> values =
opI2.getDataPropertyValues(p, ontology);
                                        for (OWLLiteral value : values) {
                                            e.setBeginningHour(value.getLiteral());
                                        }
                                    } else if (p.toStringID().contains("Minute")) {
                                        Set<OWLLiteral> values =
opI2.getDataPropertyValues(p, ontology);
                                        for (OWLLiteral value : values) {
                                            e.setBeginningMinute(value.getLiteral());
                                        }
                                    }
                                }
                            }
                            System.out.println("Inicio : " + e.toStringBeginning());
                        }
                        OWLObjectPropertyImpl end = new OWLObjectPropertyImpl(factory,
IRI.create("http://www.semanticweb.org/usuario/ontologies/2018/2/Time#hasEnd"));
                        opIndividuals2 = opI.getObjectPropertyValues(end, ontology);
                        for (OWLIndividual opI2 : opIndividuals2) {
                            Set<OWLDataProperty> properties =
ontology.getDataPropertiesInSignature();
                            for (OWLDataProperty p : properties) {
                                if (p.toStringID().contains("Time")) {
                                    if (p.toStringID().contains("Day")) {
                                        Set<OWLLiteral> values =
opI2.getDataPropertyValues(p, ontology);
                                        for (OWLLiteral value : values) {
                                            e.setEndDay(value.getLiteral());
                                        }
                                    } else if (p.toStringID().contains("Month")) {
                                        Set<OWLLiteral> values =
opI2.getDataPropertyValues(p, ontology);
                                        for (OWLLiteral value : values) {
                                            e.setEndMonth(value.getLiteral());
                                        }
                                    } else if (p.toStringID().contains("Year")) {
                                        Set<OWLLiteral> values =
opI2.getDataPropertyValues(p, ontology);
                                        for (OWLLiteral value : values) {
                                            e.setEndYear(value.getLiteral());
                                        }
                                    } else if (p.toStringID().contains("Hour")) {
                                        Set<OWLLiteral> values =
opI2.getDataPropertyValues(p, ontology);
                                        for (OWLLiteral value : values) {
                                            e.setEndHour(value.getLiteral());
                                        }
                                    } else if (p.toStringID().contains("Minute")) {
                                        Set<OWLLiteral> values =
opI2.getDataPropertyValues(p, ontology);
                                        for (OWLLiteral value : values) {
                                            e.setEndMinute(value.getLiteral());
                                        }
                                    }
                                }
                            }
                            System.out.println("Fin: " + e.toStringEnd());
                        }
                    }
                    else {
```

```
                            Set<OWLDataProperty> properties =
ontology.getDataPropertiesInSignature();
                            for (OWLDataProperty p : properties) {
                                if (p.toStringID().contains("Time")) {
                                    if (p.toStringID().contains("Day")) {
                                        Set<OWLLiteral> values =
opI.getDataPropertyValues(p, ontology);
                                        for (OWLLiteral value : values) {
                                            e.setBeginningDay(value.getLiteral());
                                        }
                                    } else if (p.toStringID().contains("Month")) {
                                        Set<OWLLiteral> values =
opI.getDataPropertyValues(p, ontology);
                                        for (OWLLiteral value : values) {
                                            e.setBeginningMonth(value.getLiteral());
                                        }
                                    } else if (p.toStringID().contains("Year")) {
                                        Set<OWLLiteral> values =
opI.getDataPropertyValues(p, ontology);
                                        for (OWLLiteral value : values) {
                                            e.setBeginningYear(value.getLiteral());
                                        }
                                    } else if (p.toStringID().contains("Hour")) {
                                        Set<OWLLiteral> values =
opI.getDataPropertyValues(p, ontology);
                                        for (OWLLiteral value : values) {
                                            e.setBeginningHour(value.getLiteral());
                                        }
                                    } else if (p.toStringID().contains("Minute")) {
                                        Set<OWLLiteral> values =
opI.getDataPropertyValues(p, ontology);
                                        for (OWLLiteral value : values) {
                                            e.setBeginningMinute(value.getLiteral());
                                        }
                                    }
                                }
                            }
                            System.out.println("Inicio : " + e.toStringBeginning());
                            System.out.println("Fin: no registrado");
                        }
                    }
                    salida.add(e);
                    System.out.println();
                }
            }
        }
        return salida;
    }
```

10.2.3 Código del método addEvent

```
public boolean addEvent(Event event) {
        try{
            this.instantiateIndividualByClass(event.getType(), event.getIndividualName());
            OWLNamedIndividual individualOWL =
factory.getOWLNamedIndividual(IRI.create(ontologyIRI + "#" + event.getIndividualName()));

            //Se agrega la dataproperty hasEventName a la nueva instancia
            OWLDataProperty dataPropertyOWL = factory.getOWLDataProperty(IRI.create(IE_IRI +
"#hasEventName"));
            OWLDataPropertyAssertionAxiom dataPropertyAssertionAxiom =
factory.getOWLDataPropertyAssertionAxiom(dataPropertyOWL, individualOWL,
                    factory.getOWLTypedLiteral(event.getEventName()));
            manager.addAxiom(ontology, dataPropertyAssertionAxiom);

            //Se agrega la dataproperty hasDescription a la nueva instancia
            dataPropertyOWL = factory.getOWLDataProperty(IRI.create(IE_IRI +
"#hasDescription"));
```

```
            dataPropertyAssertionAxiom =
factory.getOWLDataPropertyAssertionAxiom(dataPropertyOWL, individualOWL,
                    factory.getOWLTypedLiteral(event.getDescription()));
            manager.addAxiom(ontology, dataPropertyAssertionAxiom);

            //Se crea relacion happensIn
            IRI objectPropertyIRI = IRI.create(ontologyIRI + "#happensIn");
            OWLObjectProperty objectPropertyOWL =
factory.getOWLObjectProperty(objectPropertyIRI);
            OWLNamedIndividual individual2OWL =
factory.getOWLNamedIndividual(IRI.create(PS_IRI + "#" + event.getHappensIn()));
            OWLObjectPropertyAssertionAxiom objectPropertyAssertionAxiom
                    = factory.getOWLObjectPropertyAssertionAxiom(objectPropertyOWL,
individualOWL, individual2OWL);
            manager.addAxiom(ontology, objectPropertyAssertionAxiom);

            //Se crean tantas relaciones hasParticipant como participantes tenga registrados
el evento
            for(String s : event.getParticipants()){
                objectPropertyIRI = IRI.create(ontologyIRI + "#hasParticipant");
                objectPropertyOWL = factory.getOWLObjectProperty(objectPropertyIRI);
                individual2OWL = factory.getOWLNamedIndividual(IRI.create(P_IRI + "#" + s));
                objectPropertyAssertionAxiom =
factory.getOWLObjectPropertyAssertionAxiom(objectPropertyOWL, individualOWL,
individual2OWL);
                manager.addAxiom(ontology, objectPropertyAssertionAxiom);
            }

            //Se crean los individuos de la clase TemporlEntity para este evento
            this.createTemporalEntities(event);

            //Se crea la relacion hasTemporalEntity con individuos de la clase Interval o
Instant segun sea el caso
            objectPropertyIRI = IRI.create(ontologyIRI + "#hasTemporalEntity");
            objectPropertyOWL = factory.getOWLObjectProperty(objectPropertyIRI);
            if(event.getBeginningDay().contains("null")){
                individual2OWL = factory.getOWLNamedIndividual
                (IRI.create(ontologyIRI + "#begin" +
event.getIndividualName().substring(0,1).toUpperCase()+event.getIndividualName().substring(1
)));
                objectPropertyAssertionAxiom
                        = factory.getOWLObjectPropertyAssertionAxiom(objectPropertyOWL,
individualOWL, individual2OWL);
                manager.addAxiom(ontology, objectPropertyAssertionAxiom);
            }
            else {
                individual2OWL = factory.getOWLNamedIndividual
                (IRI.create(ontologyIRI + "#interval" +
event.getIndividualName().substring(0,1).toUpperCase()+event.getIndividualName().substring(1
)));
                objectPropertyAssertionAxiom
                        = factory.getOWLObjectPropertyAssertionAxiom(objectPropertyOWL,
individualOWL, individual2OWL);
                manager.addAxiom(ontology, objectPropertyAssertionAxiom);
            }

            //Se guardan los cambios en la ontologia
            manager.saveOntology(ontology);

            return true;
        } catch (OWLOntologyStorageException e) {
            e.printStackTrace();
            return false;
        }
    }
```

10.2.4 Código del servicio WSDL

```
<wsdl:definitions xmlns:apachesoap="http://xml.apache.org/xml-
soap" xmlns:impl="http://localhost:8080/axis/IEOntologyService.jws"xmlns:intf="http://localh
ost:8080/axis/IEOntologyService.jws" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding
/" xmlns:tns1="http://DefaultNamespace"xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:w
sdlsoap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://localhost:8080/axis/IEOntologyService.jws">
<!--
WSDL created by Apache Axis version: 1.4
Built on Apr 22, 2006 (06:55:48 PDT)
-->
<wsdl:types>
<schema xmlns="http://www.w3.org/2001/XMLSchema" targetNamespace="http://localhost:8080/axis
/IEOntologyService.jws">
<import namespace="http://DefaultNamespace"/>
<import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>
<complexType name="ArrayOf_xsd_anyType">
<complexContent>
<restriction base="soapenc:Array">
<attribute ref="soapenc:arrayType" wsdl:arrayType="xsd:anyType[]"/>
</restriction>
</complexContent>
</complexType>
</schema>
<schema xmlns="http://www.w3.org/2001/XMLSchema" targetNamespace="http://DefaultNamespace">
<import namespace="http://localhost:8080/axis/IEOntologyService.jws"/>
<import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>
<complexType name="Event">
<sequence>
<element name="beginningDay" nillable="true" type="xsd:string"/>
<element name="beginningHour" nillable="true" type="xsd:string"/>
<element name="beginningMinute" nillable="true" type="xsd:string"/>
<element name="beginningMonth" nillable="true" type="xsd:string"/>
<element name="beginningYear" nillable="true" type="xsd:string"/>
<element name="description" nillable="true" type="xsd:string"/>
<element name="endDay" nillable="true" type="xsd:string"/>
<element name="endHour" nillable="true" type="xsd:string"/>
<element name="endMinute" nillable="true" type="xsd:string"/>
<element name="endMonth" nillable="true" type="xsd:string"/>
<element name="endYear" nillable="true" type="xsd:string"/>
<element name="eventName" nillable="true" type="xsd:string"/>
<element name="happensIn" nillable="true" type="xsd:string"/>
<element name="individualName" nillable="true" type="xsd:string"/>
<element name="participants" nillable="true" type="impl:ArrayOf_xsd_anyType"/>
<element name="participantsTypes" nillable="true" type="impl:ArrayOf_xsd_anyType"/>
<element name="place" nillable="true" type="xsd:string"/>
<element name="placeType" nillable="true" type="xsd:string"/>
<element name="type" nillable="true" type="xsd:string"/>
</sequence>
</complexType>
</schema>
</wsdl:types>
<wsdl:message name="createTemporalEntitiesResponse"></wsdl:message>
<wsdl:message name="getEventsRequest"></wsdl:message>
<wsdl:message name="listDataPropertiesResponse">
<wsdl:part name="listDataPropertiesReturn" type="impl:ArrayOf_xsd_anyType"/>
</wsdl:message>
<wsdl:message name="getIRIRequest">
<wsdl:part name="ontofile" type="xsd:string"/>
</wsdl:message>
<wsdl:message name="createTemporalEntitiesRequest">
<wsdl:part name="e" type="tns1:Event"/>
</wsdl:message>
<wsdl:message name="loadOntologyRequest">
<wsdl:part name="ontoFile" type="xsd:string"/>
</wsdl:message>
<wsdl:message name="getSubclassesRequest">
<wsdl:part name="clase" type="xsd:string"/>
```

```
</wsdl:message>
<wsdl:message name="loadOntologyResponse">
<wsdl:part name="loadOntologyReturn" type="xsd:boolean"/>
</wsdl:message>
<wsdl:message name="getClassesRequest">
<wsdl:part name="ontofile" type="xsd:string"/>
</wsdl:message>
<wsdl:message name="addEventRequest">
<wsdl:part name="event" type="tns1:Event"/>
</wsdl:message>
<wsdl:message name="instantiateIndividualByClassResponse"></wsdl:message>
<wsdl:message name="instantiateIndividualByClassRequest">
<wsdl:part name="clase" type="xsd:string"/>
<wsdl:part name="individual" type="xsd:string"/>
</wsdl:message>
<wsdl:message name="createEventRequest">
<wsdl:part name="individualName" type="xsd:string"/>
<wsdl:part name="eventType" type="xsd:string"/>
<wsdl:part name="eventName" type="xsd:string"/>
<wsdl:part name="eventDescription" type="xsd:string"/>
<wsdl:part name="happensIn" type="xsd:string"/>
<wsdl:part name="participants" type="xsd:string"/>
<wsdl:part name="beginningDay" type="xsd:string"/>
<wsdl:part name="beginningMonth" type="xsd:string"/>
<wsdl:part name="beginningYear" type="xsd:string"/>
<wsdl:part name="beginningHour" type="xsd:string"/>
<wsdl:part name="beginningMinute" type="xsd:string"/>
<wsdl:part name="endDay" type="xsd:string"/>
<wsdl:part name="endMonth" type="xsd:string"/>
<wsdl:part name="endYear" type="xsd:string"/>
<wsdl:part name="endHour" type="xsd:string"/>
<wsdl:part name="endMinute" type="xsd:string"/>
<wsdl:part name="ontofile" type="xsd:string"/>
</wsdl:message>
<wsdl:message name="getSubclassesResponse">
<wsdl:part name="getSubclassesReturn" type="impl:ArrayOf_xsd_anyType"/>
</wsdl:message>
<wsdl:message name="createEventResponse">
<wsdl:part name="createEventReturn" type="xsd:boolean"/>
</wsdl:message>
<wsdl:message name="getEventsResponse">
<wsdl:part name="getEventsReturn" type="impl:ArrayOf_xsd_anyType"/>
</wsdl:message>
<wsdl:message name="eventsToStringRequest">
<wsdl:part name="ontofile" type="xsd:string"/>
</wsdl:message>
<wsdl:message name="listDataPropertiesRequest">
<wsdl:part name="ontofile" type="xsd:string"/>
</wsdl:message>
<wsdl:message name="eventsToStringResponse">
<wsdl:part name="eventsToStringReturn" type="impl:ArrayOf_xsd_anyType"/>
</wsdl:message>
<wsdl:message name="getClassesResponse">
<wsdl:part name="getClassesReturn" type="impl:ArrayOf_xsd_anyType"/>
</wsdl:message>
<wsdl:message name="getIRIResponse">
<wsdl:part name="getIRIReturn" type="xsd:string"/>
</wsdl:message>
<wsdl:message name="addEventResponse">
<wsdl:part name="addEventReturn" type="xsd:boolean"/>
</wsdl:message>
<wsdl:portType name="IEOntologyService">
<wsdl:operation name="getClasses" parameterOrder="ontofile">
<wsdl:input message="impl:getClassesRequest" name="getClassesRequest"/>
<wsdl:output message="impl:getClassesResponse" name="getClassesResponse"/>
</wsdl:operation>
<wsdl:operation name="addEvent" parameterOrder="event">
<wsdl:input message="impl:addEventRequest" name="addEventRequest"/>
<wsdl:output message="impl:addEventResponse" name="addEventResponse"/>
</wsdl:operation>
<wsdl:operation name="getEvents">
```

```
<wsdl:input message="impl:getEventsRequest" name="getEventsRequest"/>
<wsdl:output message="impl:getEventsResponse" name="getEventsResponse"/>
</wsdl:operation>
<wsdl:operation name="listDataProperties" parameterOrder="ontofile">
<wsdl:input message="impl:listDataPropertiesRequest" name="listDataPropertiesRequest"/>
<wsdl:output message="impl:listDataPropertiesResponse" name="listDataPropertiesResponse"/>
</wsdl:operation>
<wsdl:operation name="createTemporalEntities" parameterOrder="e">
<wsdl:input message="impl:createTemporalEntitiesRequest" name="createTemporalEntitiesRequest
"/>
<wsdl:output message="impl:createTemporalEntitiesResponse" name="createTemporalEntitiesRespo
nse"/>
</wsdl:operation>
<wsdl:operation name="instantiateIndividualByClass" parameterOrder="clase individual">
<wsdl:input message="impl:instantiateIndividualByClassRequest" name="instantiateIndividualBy
ClassRequest"/>
<wsdl:output message="impl:instantiateIndividualByClassResponse" name="instantiateIndividual
ByClassResponse"/>
</wsdl:operation>
<wsdl:operation name="createEvent" parameterOrder="individualName eventType eventName
eventDescription happensIn participants beginningDay beginningMonth beginningYear
beginningHour beginningMinute endDay endMonth endYear endHour endMinute ontofile">
<wsdl:input message="impl:createEventRequest" name="createEventRequest"/>
<wsdl:output message="impl:createEventResponse" name="createEventResponse"/>
</wsdl:operation>
<wsdl:operation name="getIRI" parameterOrder="ontofile">
<wsdl:input message="impl:getIRIRequest" name="getIRIRequest"/>
<wsdl:output message="impl:getIRIResponse" name="getIRIResponse"/>
</wsdl:operation>
<wsdl:operation name="eventsToString" parameterOrder="ontofile">
<wsdl:input message="impl:eventsToStringRequest" name="eventsToStringRequest"/>
<wsdl:output message="impl:eventsToStringResponse" name="eventsToStringResponse"/>
</wsdl:operation>
<wsdl:operation name="getSubclasses" parameterOrder="clase">
<wsdl:input message="impl:getSubclassesRequest" name="getSubclassesRequest"/>
<wsdl:output message="impl:getSubclassesResponse" name="getSubclassesResponse"/>
</wsdl:operation>
<wsdl:operation name="loadOntology" parameterOrder="ontoFile">
<wsdl:input message="impl:loadOntologyRequest" name="loadOntologyRequest"/>
<wsdl:output message="impl:loadOntologyResponse" name="loadOntologyResponse"/>
</wsdl:operation>
</wsdl:portType>
<wsdl:binding name="IEOntologyServiceSoapBinding" type="impl:IEOntologyService">
<wsdlsoap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
<wsdl:operation name="getClasses">
<wsdlsoap:operation soapAction=""/>
<wsdl:input name="getClassesRequest">
<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://D
efaultNamespace" use="encoded"/>
</wsdl:input>
<wsdl:output name="getClassesResponse">
<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://l
ocalhost:8080/axis/IEOntologyService.jws" use="encoded"/>
</wsdl:output>
</wsdl:operation>
<wsdl:operation name="addEvent">
<wsdlsoap:operation soapAction=""/>
<wsdl:input name="addEventRequest">
<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://D
efaultNamespace" use="encoded"/>
</wsdl:input>
<wsdl:output name="addEventResponse">
<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://l
ocalhost:8080/axis/IEOntologyService.jws" use="encoded"/>
</wsdl:output>
</wsdl:operation>
<wsdl:operation name="getEvents">
<wsdlsoap:operation soapAction=""/>
<wsdl:input name="getEventsRequest">
<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://D
efaultNamespace" use="encoded"/>
```

```
</wsdl:input>
<wsdl:output name="getEventsResponse">
<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://l
ocalhost:8080/axis/IEOntologyService.jws" use="encoded"/>
</wsdl:output>
</wsdl:operation>
<wsdl:operation name="listDataProperties">
<wsdlsoap:operation soapAction=""/>
<wsdl:input name="listDataPropertiesRequest">
<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://D
efaultNamespace" use="encoded"/>
</wsdl:input>
<wsdl:output name="listDataPropertiesResponse">
<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://l
ocalhost:8080/axis/IEOntologyService.jws" use="encoded"/>
</wsdl:output>
</wsdl:operation>
<wsdl:operation name="createTemporalEntities">
<wsdlsoap:operation soapAction=""/>
<wsdl:input name="createTemporalEntitiesRequest">
<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://D
efaultNamespace" use="encoded"/>
</wsdl:input>
<wsdl:output name="createTemporalEntitiesResponse">
<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://l
ocalhost:8080/axis/IEOntologyService.jws" use="encoded"/>
</wsdl:output>
</wsdl:operation>
<wsdl:operation name="instantiateIndividualByClass">
<wsdlsoap:operation soapAction=""/>
<wsdl:input name="instantiateIndividualByClassRequest">
<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://D
efaultNamespace" use="encoded"/>
</wsdl:input>
<wsdl:output name="instantiateIndividualByClassResponse">
<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://l
ocalhost:8080/axis/IEOntologyService.jws" use="encoded"/>
</wsdl:output>
</wsdl:operation>
<wsdl:operation name="createEvent">
<wsdlsoap:operation soapAction=""/>
<wsdl:input name="createEventRequest">
<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://D
efaultNamespace" use="encoded"/>
</wsdl:input>
<wsdl:output name="createEventResponse">
<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://l
ocalhost:8080/axis/IEOntologyService.jws" use="encoded"/>
</wsdl:output>
</wsdl:operation>
<wsdl:operation name="getIRI">
<wsdlsoap:operation soapAction=""/>
<wsdl:input name="getIRIRequest">
<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://D
efaultNamespace" use="encoded"/>
</wsdl:input>
<wsdl:output name="getIRIResponse">
<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://l
ocalhost:8080/axis/IEOntologyService.jws" use="encoded"/>
</wsdl:output>
</wsdl:operation>
<wsdl:operation name="eventsToString">
<wsdlsoap:operation soapAction=""/>
<wsdl:input name="eventsToStringRequest">
<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://D
efaultNamespace" use="encoded"/>
</wsdl:input>
<wsdl:output name="eventsToStringResponse">
<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://l
ocalhost:8080/axis/IEOntologyService.jws" use="encoded"/>
</wsdl:output>
```

```
</wsdl:operation>
<wsdl:operation name="getSubclasses">
<wsdlsoap:operation soapAction=""/>
<wsdl:input name="getSubclassesRequest">
<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://D
efaultNamespace" use="encoded"/>
</wsdl:input>
<wsdl:output name="getSubclassesResponse">
<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://l
ocalhost:8080/axis/IEOntologyService.jws" use="encoded"/>
</wsdl:output>
</wsdl:operation>
<wsdl:operation name="loadOntology">
<wsdlsoap:operation soapAction=""/>
<wsdl:input name="loadOntologyRequest">
<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://D
efaultNamespace" use="encoded"/>
</wsdl:input>
<wsdl:output name="loadOntologyResponse">
<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://l
ocalhost:8080/axis/IEOntologyService.jws" use="encoded"/>
</wsdl:output>
</wsdl:operation>
</wsdl:binding>
<wsdl:service name="IEOntologyServiceService">
<wsdl:port binding="impl:IEOntologyServiceSoapBinding" name="IEOntologyService">
<wsdlsoap:address location="http://aisii.azc.uam.mx:8080/axis/IEOntologyService.jws"/>
</wsdl:port>
</wsdl:service>
</wsdl:definitions>
```