DEPARTAMENTO DE INGENIERIA TELEMÁTICA

UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

PROYECTO FINAL DE CARRERA

SERVICIOS WEB EN LA WEB SEMÁNTICA PARA SISTEMAS DE ENTREGAS EN E-LEARNING

Autor: Carlos Villarroya Bes
Tutor: Pedro José Muñoz Merino

Mayo de 2012

Título: Servicios Web en la Web Semántica para sistemas de entregas en e-learning.

Autor: Carlos Villarroya Bes

Director: Pedro José Muñoz Merino

EL TRIBUNAL

Presidente: _____

Vocal: _____

Secretario: _____

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día 28 de Mayo de 2012 en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

Agradecimientos

En este apartado podrían ser escritos muchos nombres de gente que ha pasado por mi vida a lo largo de la aventura que ha significado mi paso por la Universidad. Sé que no es necesario que los escriba puesto que ya saben perfectamente quiénes son. Sin embargo, sí me gustaría dedicar unas palabras:

A mis padres, José y Ana María, ya que sin ellos llegar a este punto hubiera sido totalmente imposible. Simplemente, gracias por todo. Os quiero.

A Joseba. Sin ti, esto seguiría siendo una hoja en blanco. Muchas gracias. Te quiero.

Fer, qué diferentes somos y qué bien nos compenetramos. Gracias por estar ahí para lo bueno y para lo malo durante todos estos años. Gracias Nando.

A todos los sonimágicos que me han ayudado… Eva, Lara, Ana, Anita, Alvarito, Marian, Dani, Lucía, Camarma, Iru, Clara… y un largo etcétera. Gracias chicos.

A mis Quintanos, Vero, Anica, Gaby, Javi, Meli, Kiko, Morales,…

A mis mañicos Kike y Víctor, aunque a veces no te lo merezcas.

Y a mi tutor, Pedro, por haber tenido la paciencia de ayudarme con este proyecto.

Resumen

Por un lado, la enseñanza sigue siendo hoy en día una de las prioridades más importantes para las personas y, sin embargo, no todo el mundo cuenta con la posibilidad tener acceso a ella de la forma en la que se conocía hasta hace pocos años.

Por otro lado, Internet ha supuesto una revolución en la información en todos sentidos y más concretamente en las posibilidades que éste es capaz de ofrecer al usuario.

Si unimos enseñanza con Internet, somos capaces de agregar muchas de las propiedades de Internet a tal valorado bien como es la enseñanza.

Partiendo de lo anterior, y especificando hacia el objetivo de este proyecto, un campo mucho más concreto de esta fusión es la entrega de todos los documentos que pueda realizar un usuario a un sistema centralizado para así recogerlos. Dichos documentos serán las prácticas que realizan los usuarios para una asignatura.

En este proyecto se proponen un conjunto de servicios Web utilizando técnicas de Web Semántica para la manipulación de datos relacionados con los sistemas software de entrega en e-learning.

También se ha implementado una aplicación que hace uso de los servicios Web definidos. Para ello, se han creado las ontologías correspondientes (utilizando la herramienta Protégé), se han definido e implementado los servicios Web que utilizan datos anotados en dichas ontologías y se ha realizado una aplicación que invoca algunos de dichos servicios Web.

Entre las tecnologías utilizadas se encuentran el entorno Jena y SDB.

Abstract

On the one hand, nowadays learning is still one of the most important priorities for people and yet, not everyone has the opportunity to gain access to it the way it was known until recent years.

On the other hand, Internet has revolutionized information in many ways and especially in the possibilities it can offer the user.

If we combine teaching with the Internet, we are able to add many of the Internet possibilities to the teaching, we add value to teaching.

Based on the above, and specifying toward the goal of this project, a much more specific for this merger is the delivery of all documents that a user can send to a centralized system in order to collect them. These documents may be the practices of a subject.

In this project, we propose a set of Web Services using Semantic Web techniques for handling data related to delivery software systems in e-learning.

An application that uses the Web services defined has also been implemented . To do this, the corresponding ontology has been created (using the Protégé tool), the Web services have been defined and implemented using data recorded in this ontology and an application that invokes some of these Web services has been made as well.

Technologies such as Jena and SDB were used.

Índice General

Lista de figuras

Acrónimos

AJAX Asynchronous JavaScript And XML (traducido del Inglés JavaScript Asíncrono y lenguaje XML)…

API Application Programming Interface (traducido del Inglés Interfaz de Programación de Aplicaciones)…

AWT Abstract Window Toolkit (traducido del Inglés Kit de Herramientas de Ventana Abstracta)…

CVS Concurrent Versions System (traducido del Inglés Sistema de Versiones Concurrentes).

DAWG Data Access Working Group (traducido del Inglés Grupo de Trabajo de Acceso a Datos)…

DAML DARPA Agent Markup Language (traducido del Inglés Lenguaje de Marcado del Agente DAML)…

DARPA Defense Advanced Research Projects Agency (traducido del Inglés Agencia de Investigación de Proyectos Avanzados de Defensa)…

FTP File Transfer Protocol (traducido del Inglés Protocolo de Transferencia de Archivos)…

HTML HyperText Markup Language (traducido del Inglés Lenguajes de Marcado de Hipertexto)…

HTTP HyperText Transfer Protocol (traducido del Inglés Protocolo de Transferencia de Hiper Texto)…

IP Internet Protocol (traducido del Inglés Protocolo de Internet)…

JDK Java Development Kit (traducido del Inglés Kit de Desarrollo para Java)…

JFC Java Foundation Classes (traducido del Inglés Clases Base Java)…

JSP JavaServer Pages (traducido del Inglés Servidor de Páginas Java)…

LMS Learning Management System (traducido del Inglés Sistema de Gestión del Aprendizaje)…

LRN Learning Resource iNterchange (traducido del Inglés Intercambio de Recursos de Aprendizaje)…

OWL Web Ontology Language (traducido del Inglés Lenguaje de Ontologías

Web.

PHP PHP Hypertext Pre-processor(traducido del Inglés Preprocesador de Hiper Texto PHP)…

RMI Java Remote Method Invocation (traducido del Inglés Invocation de Métodos Remotos en Java.

RPC Remote Procedure Call (traducido del Inglés Llamada a Procedimiento Remoto)…

RDF Resource Description Framework (traducido del Inglés Marco de Descripción de Recursos)…

SDK Software Development Kit (traducido del Inglés Kit de Desarrollo Software).

SMTP Simple Mail Transfer Protocol (traducido del Inglés Protocolo Simple de Transferencia de Correo)…

SOAP Simple Object Access Protocol (traducido del Inglésn Protocolo de Acceso a Objetos Simple).

SPARQL SPARQL Protocol and RDF Query Language (traducido del Inglés Lenguaje de Consulta RDF y Protocolo de Acceso a Dato).

SQL Structure Query Language (traducido del Inglés Lenguaje de Consulta Estructurado).

TDB Task DataBase (traducido del Inglés Base de datos de Tareas).

TCP Transport Control Protocol (traducido del Inglés Protocolo de Control de Transporte).

UDDI Universal Description, Discovery and Integration (traducido del Inglés Integración, Descubrimiento y Descripción Universal).

URI Uniform Resource Identifier (traducido del Inglés Identificador Uniforme de Recurso).

URL Uniform Resource Locator (traducido del Inglés Localizador Uniforme de Recurso).

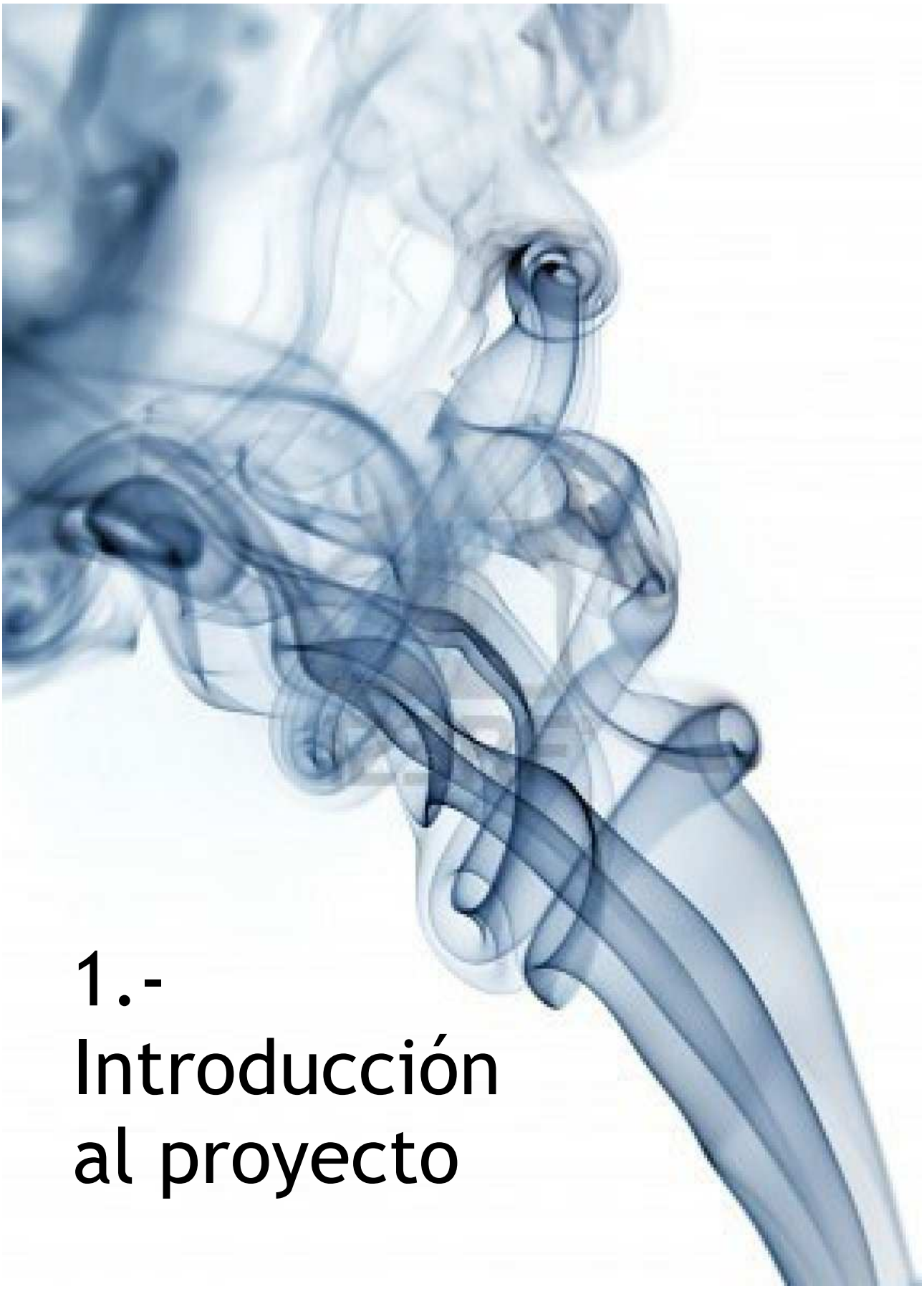
WebCT Web Course Tools (traducido del Inglés Herramientas para Cursos Web)….

WSDL Web Services Description Language (traducido del Inglés Lenguaje de Descripción de los Servicios Web)…

WWW Worl Wide Web (traducido del Inglés Red de Amplitud Mundial)…

W3C World Wide Web Consistorium (traducido del Inglés Consistorio de la Red de Amplitud Global)……

XML Extensible Markup Language (traducido del Inglés Lenguaje de Marcas extensible).



1.- Introducción al proyecto

1.1 MOTIVACIONES

La enseñanza, según la R.A.E., es 'la acción o efecto de enseñar, sistema y método de dar instrucción'.

La primera motivación del proyecto fue seguir con un tema relacionado con la Universidad, es decir, con la enseñanza y, a la vez, trabajar con el e-learning y la Web semántica ya que, en mi caso, era la primera vez que entraba en contacto con todo ello. La definición del e-learning que mejor expresa mi motivación es definir el e-learning como una modalidad de enseñanza a distancia en la que se utiliza Internet como vía de comunicación y que se apoya en las nuevas tecnologías de la información para conseguir el mayor recurso de contenidos, interacción entre los usuarios y el mayor nivel posible de conocimientos adquiridos por los alumnos.

Con el e-learning se ha conseguido eliminar la barrera-espacio temporal existente en la educación ya que los cursos son accesibles desde cualquier lugar y en cualquier momento, brindando así la posibilidad de la enseñanza a un grupo de personas mucho más elevado.

La figura del profesor sigue presente realizando funciones como pueden ser el seguimiento de los estudiantes para comprobar su progreso durante el curso o como la resolución de cualquier duda que puedan tener. Sin embargo, los alumnos podrán también interactuar entre ellos gracias a plataformas como son los chats, foros de discusión, etc.

Una parte fundamental en un curso es la realización de trabajos y prácticas por parte del alumno. Hasta ahora, el profesor tenía la función de controlar y supervisar la realización y entrega de las prácticas. Un cambio muy significativo producido con las prácticas gracias al e-learning es que la coordinación de las prácticas deja de estar a cargo del profesor para estar dirigido por un sistema de entrega de prácticas. Estos sistemas se encuentran dentro de los sistemas de gestión LMS.

El profesor continuará corrigiendo las prácticas pero el propio sistema se encargará de realizar acciones como son calcular datos estadísticos como el número de alumnos que no ha realizado cierta práctica, controlar la entrega de prácticas mandando avisos si son necesarios, mandar un aviso para recomendar una práctica adicional si se

tiene suspendido el laboratorio, etc.

Otra de las motivaciones a la hora de realizar este PFC sobre los sistemas de entrega, fue conocer en detalle estas herramientas y las tecnologías que utilizan, ya que poseen una gran relevancia para el aprendizaje junto con un fuerte impacto y múltiples usos, como son facilitar el aprendizaje, su productividad, publicar cursos y contenidos, etc. Los sistemas de entrega realizan un procesado automático por parte de las máquinas para dar información relevante para el proceso del aprendizaje.

Así mismo, me resultó muy motivador el hecho de utilizar servicios Web en el proyecto porque considero muy interesante que a través de la creación de éstos, es decir, creando un API, se pueda acceder a una aplicación o un programa en un sistema hosting remoto a través de la red. Con un servicio Web podemos acceder desde donde queramos y sin la necesidad de tener todo el código en nuestro host ya que únicamente tenemos que realizar la llamada a éste, pasándole como parámetros aquellos que hayan sido definidos anteriormente para su servicio, y recibiremos así lo que deseamos sin preocuparnos de nada más, únicamente de que la comunicación se realice exitosamente. Es decir, nos están permitiendo comunicarnos entre distintas máquinas con plataformas y programas distintos.

Por último, cabe destacar la motivación que es enfrentarse a un conjunto de tecnologías desconocidas de las que has oído hablar pero no sabes muy bien ni qué son ni para qué se utilizan con el fin de conseguir sacar adelante un problema al que te enfrentas desde cero. Con esto hago referencia a tecnologías como SDB, Jena, la herramienta Protégé para realizar las ontologías, servicios Web con AXIS accediendo a diferentes bases de datos, la creación de clientes stubs para los servicios Web, etc.

1.2 OBJETIVOS

Una vez leído el planteamiento del PFC y ver que sí poseía las suficientes motivaciones para continuar con su desarrollo, hubo que definir los objetivos que se querían alcanzar, siendo éstos los siguientes.

En primer lugar, fue primordial la realización de una ontología que modelase diferentes aspectos de los sistemas de entrega para el aprendizaje, y realizar un análisis

de la mayoría de ellos, así como ver qué cosas tienen en común. Se trataba de realizar un estudio general de los sistemas de entregas de prácticas sin ceñirse a ningún patrón en concreto.

En segundo lugar, fue necesaria la creación de una serie de servicios Web para características comunes de sistema de entrega de prácticas creando para, una vez devueltas, conseguir realizar una serie de funciones que pudieran ser muy diferentes entre sí y con fines distintos. Haciendo hincapié en el objetivo del proyecto, se está hablando de tareas como pueden ser mandar email recordatorios a aquellos grupos del laboratorio que no hayan realizado una práctica, siendo configurable el número de días respecto a la fecha final de entrega en el que se quiere mandar, realizar estadísticas sobre grupos, por ejemplo, que han entregado la práctica dentro del intervalo de fechas de entrega, que tienen suspensa una asignatura en concreto, que de todas las prácticas del laboratorio han entregado sólo una cierta cantidad de ellas, etc.

Sin embargo, todo ello no tenía sentido si no se podía poner en práctica de una manera rápida, útil y sencilla, por lo que también fue imprescindible la creación de una aplicación en un entorno real con una parte gráfica. Para ello, se realizó una interfaz gráfica con JSwing y una parte lógica integrando los citados servicios Web. Por supuesto, se manipularon los datos según la ontología definida.

1.3 ORGANIZACIÓN DE LA MEMORIA

A la hora de realizar la memoria intenté que todos sus puntos claves quedasen lo más diferenciados posible para entender claramente todos los pasos realizados en el desarrollo del proyecto. Para conseguirlo, dividí la memoria en siete partes claramente diferenciadas, eligiendo en primer lugar un capítulo de introducción al proyecto que nos acerca en detalle las motivaciones que han dado lugar a la realización del presente proyecto, así como los objetivos que se han marcado al comienzo del mismo, finalizando con una primera aproximación a las distintas partes y capítulos que componen esta memoria.

A continuación, realicé un apartado para el estado del arte que nos acerca la actualidad del contexto en el que se enmarca el presente proyecto, definiendo las diferentes tecnologías utilizadas en su desarrollo.

En tercer lugar, está el apartado donde elaboré la propuesta y realización de la Ontología que definiese cualquier sistema de entrega de prácticas, para llevar a cabo posteriormente, la realización de los servicios Web. Éstos serán desarrollados en el apartado siguiente. No sólo son definidas sus creaciones sino que también el porqué de su elección frente a otros.

En antepenúltimo lugar, se encuentra el apartado donde se explica la implementación de los servicios Web para el sistema de entregas real con una interfaz gráfica para ayudar en la utilización de éstos.

Para finalizar tenemos los apartados donde se definen las conclusiones tras haber realizado el proyecto completamente así como toda la biografía utilizada en todo el proceso.

2.- Estado del arte

INTRODUCCIÓN

En el presente capítulo se hará una breve descripción del estado actual de las diferentes tecnologías que confluyen en este Proyecto y que son necesarias para su realización:

- Estándares Web

Comenzaremos introduciendo diferentes aspectos de los estándares Web que usaremos para llevar a cabo esta aplicación. Posteriormente, profundizaremos en las técnicas de la Web semántica, ya que son en las que se basará primordialmente este proyecto. Estas tecnologías son RDF, SPARQL, OWL o Jena, por ejemplo.

- Educación con la Web semántica

A continuación, nos centraremos en los cambios y nuevas posibilidades que nos brinda la Web semántica respecto a la educación así como en las ventajas y desventajas de su integración.

- Sistema de entregas en e-learning

En este apartado comentaremos los diferentes sistemas de entrega de prácticas que existen, las características que tienen y cuáles son los más destacados y por qué.

- Servicios Web y sus tecnologías.

Se analizarán qué son los servicios Web, cómo se realizan, cuál es su ciclo de despliegue y su funcionamiento, así como las tecnologías y lenguajes que utilizan, como por ejemplo, WSDL, Tomat, Axis, etc.

- Herramienta Protégé

Para finalizar este capítulo describiremos la herramienta Protégé utilizada para crear ontologías. Así mismo, comentaremos qué son las ontologías, qué tipos existen, qué componentes y propiedades poseen, etc.

2.1 ESTÁNDARES WEB

Los estándares Web son un conjunto de especificaciones y recomendaciones técnicas generadas por el World Wide Web Consistorium (W3C) que definen mejores prácticas para la construcción de sitios en Internet . Entre los estándares Web que existen, ver Figura 2.3, tres de ellos son esenciales para el funcionamiento de WWW: Uniform Resource Identifier (URI), HTTP e HyperText Markup Language (HTML). Para realizar transacciones en Internet podemos utilizar múltiples protocolos, entre todos ellos, destacamos HTTP, el cuál define la sintaxis y la semántica de los elementos de esta arquitectura. [1]

Para poder acceder a un recurso, es necesario que éste sea identificado de forma unívoca. El estándar URI realiza esta labor a través de una cadena corta de caracteres que puede estar formada por el protocolo de acceso al recurso, el dominio, la ruta y consulta en caso de ser necesaria.

Una vez se ha accedido a los datos y se ha transportado la información hasta el equipo cliente, necesitamos presentarlos al usuario final, para ello se utiliza un lenguaje para que el navegador interprete y presente la información, ese lenguaje de programación es HTML

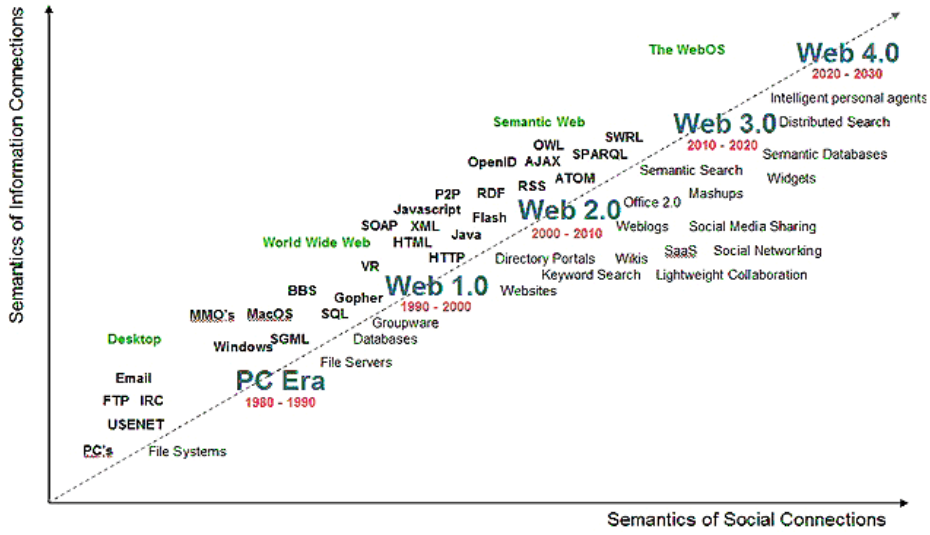


Figura 2.1 Semántica de la Web y lenguajes de programación

2.1.1 Término Web 1.0

La Web 1.0 hace referencia al estado más básico de la WWW, un estado estático en cuyas páginas no le era permitido al usuario interactuar con el contenido de la página debido a que, como el propio nombre del estado indica, eran estáticas en lugar de dinámicas.

En el lado del servidor, las opciones eran poco potentes y a los usuarios únicamente se les permitía comunicarse a través de formularios HTML enviados vía email, correo electrónico o enlaces.

Más adelante, surgió la necesidad de brindar una nueva forma de publicar los contenidos en WWW en la que la información existente en la página no fuese estática sino que fuese capaz de ser actualizada. Para que fuese posible, fue necesario poseer una base de datos actualizada con la información a introducir y nuevas tecnologías como SQL o PHP. Este modelo se conoce como Web dinámica o Web 1.5. Un inconveniente que no se había conseguido solucionar seguía siendo que el único capaz de modificar el contenido de la página Web era el webmaster, el creador de la Web y responsable de su mantenimiento. El usuario únicamente era capaz de leer la información, no interactuar con ella. [2]

2.1.2 Término Web 2.0

Su principal objetivo era conseguir aplicaciones Web orientadas a los usuarios en lugar de al webmaster como era en el estándar anterior.

La Web 2.0 se fundamenta en dejar atrás el modo pasivo de un usuario que accede a una página Web para dar paso a un modo activo en el crecimiento de los contenidos de ésta para proporcionar servicios interactivos en red, dando al usuario el control de sus datos. Por lo tanto, un usuario puede interactuar con otros usuarios y modificar el contenido del sitio Web. El webmaster ocupará las funciones de mantenimiento, supervisión y resolución de problemas derivados de su funcionamiento. [3]

A su vez, permite que los usuarios de la red obtengan información desde una variedad de sitios simultáneamente y puedan suministrarla en su propio sitio para lograr nuevos propósitos.

Algunos ejemplos que han canalizado la inteligencia colectiva son:

- Wikipedia: enciclopedia gratuita y online, escrita y editada por voluntarios.
- Blogs: espacio Web personal donde el usuario puede escribir cronológicamente cualquier tipo de contenido como pueden ser los noticias, enlaces a otras páginas, opiniones, etc.

Otros ejemplos de la Web 2.0 son los servicios de las redes sociales, servicios de alojamiento de videos como el portal youtube, las wikis, blogs, mashups y las folksonomías (clasificación por medio de etiquetas en lugar de por medio de directorios propios de las taxonomías de la Web 1.0).

2.1.3 Término Web 3.0

La información existente en una página Web tiene significado para los usuarios pero resulta difícilmente inteligible para los ordenadores. La idea de la Web 3.0 es darle significado a las páginas Web añadiendo información adicional con una estructura tal que pueda ser entendida por los ordenadores. Dicha información tiene que describir el contenido, el significado y la relación de los datos para que los ordenadores, por medio de técnicas de inteligencia artificial, sean capaces de emular y mejorar la obtención de conocimiento, hasta el momento, reservada a los usuarios, todo ello sin la necesidad de usar operadores humanos. De ahí el nombre de Web Semántica. [4]

La Web Semántica busca corregir los dos problemas principales de la Web, la sobrecarga de información y la heterogeneidad de las fuentes de información con el consiguiente problema de interoperabilidad. Se pretende resolver acciones o problemas cotidianos de forma automática.

La información estará como contenidos accesibles a aplicaciones sin buscador, transformando la red en una base de datos. Esto permite un nuevo nivel de integración de datos y de aplicación interoperable, haciendo los datos tan accesibles y enlazables

como las páginas Web. Para que sea posible, los datos tienen que ser descriptos por las tecnologías RDF y OWL, además de XML. Estas tecnologías se combinan para aportar descripciones explícitas de los recursos de la Web, de tal forma que el contenido queda desvelado como los datos de una base de datos accesibles por Web, o las etiquetas inmersas en el documento que permiten que los gestores de contenido interpreten los documentos y realicen procesos inteligentes de captura y tratamiento de información.

Se está permitiendo que el software sea capaz de procesar el contenido, razonar sobre éste, realizar las operaciones de combinaciones necesarias y ejecutar deducciones lógicas.

Por todo ello, la Web Semántica es una Web extendida, dotada de mayor significado en la que cualquier usuario en Internet puede encontrar respuestas a sus preguntas de forma más rápida y sencilla gracias a la información mejor definida.

Las tecnologías de la Web 3.0, como programas inteligentes, utilizan por tanto datos semánticos para conseguir una manipulación de datos más eficiente.

El resumen de la evolución de los estándares Web y su acceso se muestra en la siguiente figura:

Tipo	Web 1.0	Web 2.0	Web 3.0
Internet	Expositiva de lectura	Social de lectura-escritura	Global de lectura-escritura-multimedia
Acceso	Indexación simple	Web Semántica Manual	Web Semántica Automática

Figura 2.2 Evolución de Internet y su acceso.

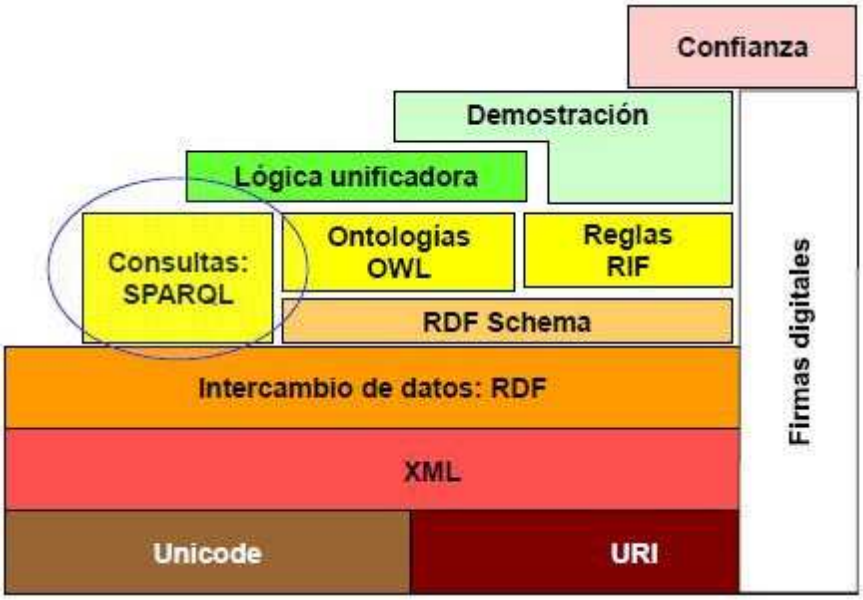


Figura 2.3 Tecnologías propias de la Web 3.0

La anterior figura 2.3 muestra el esquema de todas las tecnologías involucradas en la Web 3.0. Serán comentadas a continuación de manera independiente.

2.1.3.1 RDF

El RDF (Resource Description Framework) es un estándar escrito en XML del grupo W3C creado para proporcionar un marco estándar, entendible por las aplicaciones informáticas, para la interoperabilidad en la descripción de los contenidos de una página Web como pueden ser el título, la información del copyright etc. Está basado en la codificación, reutilización e intercambio de metadatos estructurados y forma parte de la actividad de la semántica Web del grupo W3C. [5]

Algunos ejemplos del uso de RDF son:

- Descripción del contenido e información de las imágenes presentes en una página Web.
- Descripción del contenido de los motores de búsqueda.
- Descripción de las bibliotecas electrónicas.

2.1.3.1.1 Modelo de datos, sintaxis y esquema RDF.

1. El modelo de datos RDF es el siguiente:

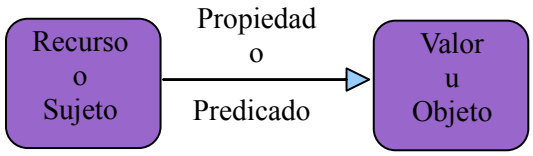


Figura 2.4 Representación del funcionamiento de RDF.

En RDF se identifican o representan todos los elementos con identificadores de Web (URIs).

Como se puede apreciar en la figura 2.4, RDF se basa es un modelo formal de declaraciones con sujeto, predicado y objeto. Un recurso (sujeto) se define o describe a través de un conjunto de propiedades (predicado) y unos valores fijados de éstas (objetos). Se puede entender como un par recurso-valor. Sin embargo, con estas asociaciones recurso-propiedad-valor no se asigna un único valor individual al recurso sino que se representan a su vez las relaciones que existen entre todos los recursos.

En resumen:

- Recurso: cualquier objeto que se puede identificar unívocamente por una URI.
- Propiedades: son aspectos específicos, características, atributos o relaciones utilizadas para describir recursos. Cada tipo de propiedad tiene sus valores específicos, define los valores permitidos, los tipos de recursos que puede describir y las relaciones que existen entre las distintas propiedades.
- Valor: valor de esa propiedad para ese recurso.

Existen validadores RDF online como el http://www.w3.org/RDF/Validator/ [6] que te permiten comprobar la sintaxis de cualquier documento RDF de manera gratuita.

2.- La sintaxis básica, como se ha apuntado anteriormente, es la de XML1.0. Además se pueden distinguir dos tipos de construcciones sintácticas para codificar RDF:

por un lado la serializada que expresa, de una forma muy regular, todas las capacidades de un modelo de datos RDF; y por otro la sintaxis abreviada que incluye construcciones adicionales.

3.- El esquema RDF es un conjunto de informaciones relativas a las clases de recursos que sirve para explicitar las relaciones jerárquicas que establecen entre ellos, o bien para matizar el carácter obligatorio u opcional de las propiedades y otras restricciones como el número de ocurrencias, etc. Son utilizados para corregir problemas causados por la disparidad terminológica ya que permiten expresar un espacio o esquema inequívoco al consignar el recurso que define la semántica correspondiente al principio de un registro de metadatos.

Un ejemplo RDF basado en un rdf creado para mi proyecto es el siguiente:

```
<?xml version="1.0"?>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  mlns:pf="http://www.owl-ontologies.com/ProyectoFinal.owl#">

<pf:Delivery rdf:ID="Delivery3">
<pf:practiceDelivered
        rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
        Practice3
</pf:practiceDelivered>
   …
</pf:Delivery>
```

2.1.3.2 SPARQL

El SPARQL (Sparql Protocol And RDF Query Language) es un lenguaje de consulta estándar para realizar consultas de diversas fuentes de datos almacenadas en formato RDF o visto como RDF vía el software de conectividad middleware. Se trata de una recomendación para crear un lenguaje de consulta dentro de la Web semántica que está ya implementada en muchos lenguajes y bases de datos. [7]

Se encuentra normalizado por el el grupo DAWG (RDF Data Access Working Group) del W3C (Word Wide Web Consortium).

Con SPARQL los desarrolladores y usuarios finales pueden representar y utilizar los resultados obtenidos en las búsquedas a través de una gran variedad de información, como son datos personales, redes sociales y metadatos sobre recursos digitales como música e imágenes.

Para poder llevarse a cabo las consultas, el protocolo SPARQL describe el proceso involucrado entre un cliente SPARQL y un procesador de consultas SPARQL (SPARQL EndPoint). Para hacerlo de manera visual, la W3C proporciona una interfaz genérica y luego define vinculaciones de dicha interfaz sobre dos medios distintos:

- sobre HTTP (HyperText Transfer Protocol)
- sobre SOAP (Simple Object Access Protocol)

Por ello, la funcionalidad de SPARQL está dividida en tres especificaciones: un lenguaje de consulta, un formato para las respuestas dirigidas al cliente SPARQL y un medio para el transporte de consultas y respuestas entre ambos puntos:

- SPARQL Query Language: Núcleo de SPARQL. Explica la sintaxis para la composición de sentencias y su concordancia.

- SPARQL Protocol: Formato utilizado para la devolución de los resultados de las búsquedas (consultas SELECT o ASK), a partir de un esquema de XML.

- SPARQL Query XML Results Format: Describe el acceso remoto de datos y la transmisión de las consultas de los clientes a los procesadores. Utiliza WSDL para definir protocolos remotos para la consulta de bases de datos basadas en RDF.

SPARQL permite:

- Realizar consultas de los datos obligatorios y opcionales presentes en un documento RDF con sus conjunciones y disyunciones.
- Realizar consultas para comprobar y constatar valores.
- Realizar operaciones de creación, modificación y borrado de los datos.

SPARQL se divide en los bloques:

- El lenguaje de consulta
- El motor para el almacenamiento y recuperación de los datos.

2.1.3.2.1 Consultas SPARQL.

La mayoría de las consultas SPARL contienen un conjunto de patrones triples llamados “patrones de gráficos simples” que representarán patrones mediante los cuales se buscarán tripletes coincidentes en los datos. Son como el modelo de datos RDF con el triplete Recurso-Propiedad-valor pero en SPARQL cada uno de ellos puede ser una variable distinta.

El patrón de SPARQL coincide con el de RDF si los términos de ese subgrafo RDF pueden ser sustituido por las variables de SPARQL.

Una consulta SPARQL se divide en dos partes:

- La cláusula que identifica el tipo de consulta que se va a realizar. Sirve para definir los datos que van a ser devueltos en la respuesta. (P.e. SELECT)
- La cláusula que identifica el patrón sobre el que se filtrarán los tripletes. (WHERE)

Existen 4 tipos de consulta:

- SELECT: devuelve el resultado en forma tabular sobre XML, mostrando las vinculaciones encontradas de las variables
- CONSTRUCT: genera y devuelve el grafo RDF siguiendo el patrón definido en la consulta.
- ASK: devuelve un resultado booleano dependiente de si existe una solución a la consulta.
- DESCRIBE: devuelve información sobre un recurso específico. Depende de la configuración del procesador.

Existe una cláusula adicional FILTER que permite filtrar las consultas utilizando para ello operadores con el fin de obtener resultados más exactos. Existen 4 tipos de operadores:

- Operadores lógicos.
  - A||B, A&&B, ¡A, (A)
  - De comparación: =, !=, <, >, <=, >=
- Operadores aritméticos.
- Unarios +A, -A
- Binarios (A op B)
- Operadores y funciones RDF.
  - Boleanos. BOUND(A), isURI(A), isBLANK(A), isLITERAL(A)
  - String. STR(A), LANG(A), DATATYPE(A)
- Operadores para realizar STRINGS

Ejemplo de una consulta SPARQL.

Este ejemplo ha sido sacado de un Servicio Web realizado para el proyecto y su objetivo es devolver las prácticas que envía el alumno “Joseba”. Esta consulta está almacenada en el archivo 'q21.rq' y contiene la siguiente información:

```
PREFIX proyect: <http://www.owl-ontologies.com/ProyectoFinal.owl#>

SELECT ?x
WHERE { proyect:Joseba proyect:sendsPractice ?x . }
```

Como se puede apreciar en el ejemplo anterior, no ha sido declarada la cláusula “FROM” para definir dónde vamos a realizar la búsqueda. Esto se debe a que la consulta se realiza con SDB de Jena y es en el archivo de configuración “sdb.ttl” donde se ha indicado que la base de datos dónde se va a realizar la búsqueda está almacenada en el ordenador y es de tipo MySQL.

En la primera línea encontramos la palabra clave PREFIX para declarar el namespace “proyect”, que asocia la URI contenida entre los símbolos <> y la etiqueta

"proyect", que se usará en adelante para describir el namespace. Pueden ser incluidos varios PREFIX en una misma consulta.

A continuación tenemos la palabra SELECT para indicar qué resultados queremos obtener en la respuesta, que en ese caso serán todos.

Por último tenemos la palabra clave WHERE para indicar el patrón sobre el que se filtrarán los tripletes de la base de datos. En este caso devolverá todos objetos cuyo recurso sea "Joseba" y el predicado sea "sendsPractice".

El resultado obtenido de esta consulta es el siguiente:

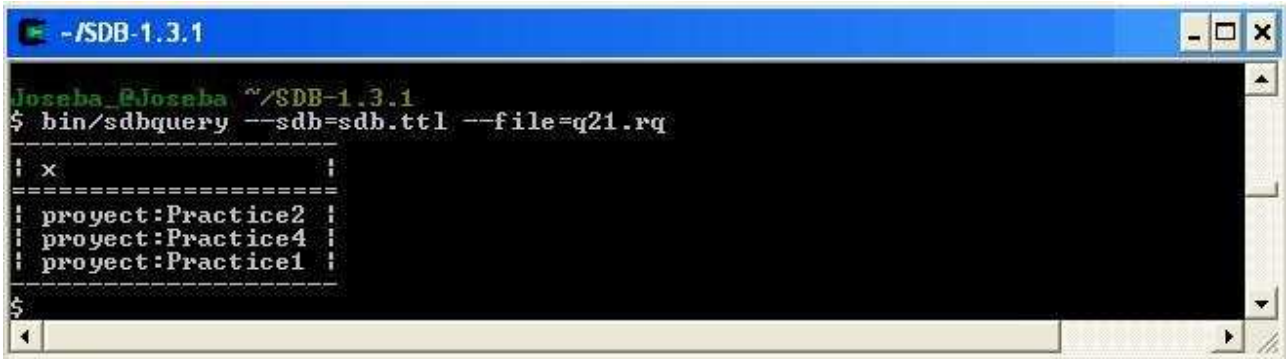


Figura 2.5 Resultado obtenido de una consulta SPARQL

2.1.3.3 OWL

El lenguaje OWL ha de ser usado cuando los documentos presentan una información que, en lugar de ser presentada a los seres humanos, tiene que ser procesada por diversas aplicaciones. Con su uso, se representa el significado de los términos en el vocabulario y las relaciones que existen entre éstos, lo que se conoce como ontología.

Es un lenguaje para la definición de nuevos vocabularios que ha llegado a ser capaz de realizar con mayor optimización la representación de los términos y sus relaciones que las tecnologías XML, RDF y RDF-Schema, por lo que se consigue una mejor representación del contenido interpretable por una máquina en la Web.

Existen 3 tipos de sublenguajes de OWL [8]:

- OWL-Lite: es el lenguaje menos expresivo de todos ellos. Está destinado a ser usado en aquellas situaciones donde sólo existan una jerarquía de clase y

limitaciones simples, es decir, construcciones simples. A la jerarquía sólo aceptará la posibilidad de clasificación.

- OWL-DL (Descripción Lógica): es un lenguaje con una grado de expresividad intermedia. Puede ser considerado como una extensión del lenguaje anterior. Está basado en descripciones lógicas de primer orden y, por ello, son sensibles a razonadores automáticos, capaces de comprobar la clasificación jerárquica y las inconsistencias de la ontología. Son las indicadas para usuarios que requieren el máximo de expresividad pero exigiendo completitud computacional (se garantiza que todas las conclusiones son computables) y decibilidad (todos los cálculos acaban en un tiempo finito) .

- OWL-Full: es el lenguaje más expresivo de los 3 sublenguajes. Como en el caso del OWL-DL, este lenguaje puede ser considerado como una extensión del lenguaje anterior. El uso de este lenguaje está destinado a ser usado cuando el uso de expresiones muy complejas tiene mayor peso frente a que sea capaz de garantizarse la garantía computacional de la jerarquía. Por ello, no es posible utilizar razonadores automatizados.

OWL posee numerosas propiedades que sirven para definir los vocabularios que se van a utilizar. Serán explicadas posteriormente en el apartado 2.6.1.1 para explicar de forma conjunta la ontología realiza y el tipo de propiedades que se han utilizado y el por qué.

En resumen, a la hora de determinar que sublenguaje va a ser utilizado habrá que barajar entre estas 3 posibilidades:

- Si son suficientes construcciones simples, se utilizará OWL-Lite.
- Si es necesario máxima expresividad pero exigiendo completitud computacional, se utilizará OWL-DL.
- Si es necesario el uso de instalaciones de modelado potentes y de muy alta expresividad sin posibilidad de utilizar razonadores, se utilizará OWL-Full.

2.1.3.4 JENA

Jena es un framework desarrollado por los laboratorios Hewlett Parckard como parte del "HP Labs Semantic Web Programme", con el objetivo de manipular metadatos para programar aplicaciones de la Web Semántica desde aplicaciones escritas en java. Se trata de un código abierto. [9]

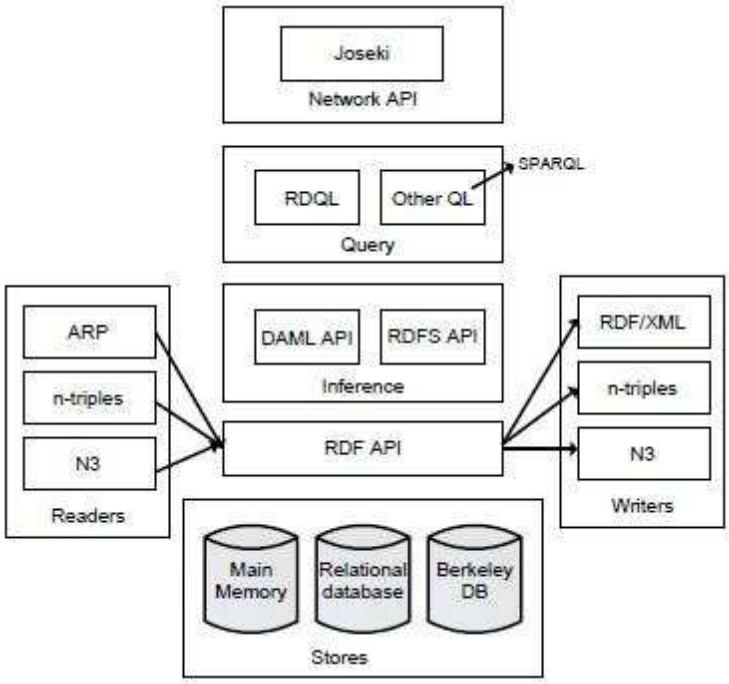


Figura 2.6 Arquitectura de Jena

Como se puede apreciar en la figura 2.6, para realizar la creación, manipulación y consulta de grafos RDF se utiliza la 'RDF API', que constituye el núcleo principal de JENA y da soporte para las tecnologías utilizadas para el almacenamiento de los datos, 'stores'. A la hora de realizar las consultas a los stores, se utiliza el protocolo y lenguaje de consulta SPARQL, para lo que se utilizará Joseki ya que implementa un servidor HTTP necesario para ello, y con el uso de los writters se ejecutarán consultas más compactas y elegantes.

JENA incluye:

- Analizador sintáctico para RDF
- API de RDF que permite:

    - crear grafos RDF programando en Java:
    - leer los grafos y cargarlos en memoria.
    - serializar los grafos en formatos como RDF/XML, N-Triples o N3.
- API para trabajar con ontologías en lenguajes como:
    - OWL
    - RDF-Schema
    - DAML
- Capacidad de razonar con el uso de razonadores OWN o RDFS.
- Capacidad para procesar consultas SPARQL tanto a repositorios locales como remotos.
- Motores de inferencia y conectores a motores externos.
- Mecanismos de almacenamiento desarrollados en Java para RDF. Los RDF almacenados se accederán a través de consultas SPARQL.
- Servidor HTTP de RDF, Joseki: es un servidor para publicar modelos RDF en la Web. Los modelos tienen URLs y pueden ser accedidos por queries usando HTTP GET.

Los enunciados del modelo se almacenan en un grafo base. El razonador o motor de inferencia puede utilizar el contenido de dicho grafo junto con las reglas semánticas del lenguaje para mostrar un conjunto completo de enunciados. Esto implica incluir también aquellos enunciados que se pueden inferir a partir del grafo base.

Es muy importante diferenciar 3 partes fundamentales para hacer una consulta de Jena:

- El lenguaje de consulta (SPARQL).
- El motor de búsqueda. (ARQ).
- El motor de almacenamiento (SDB o TDB).

A continuación se presenta un ejemplo de una consulta SPARQL de Jena. Éste ha sido resumido del uso de las consultas en el proyecto:

```
 StoreDesc storeDesc = new

StoreDesc(LayoutType.LayoutTripleNodesHash,DatabaseType.MySQL) ;
Dataset ds=null;
try{
        Class.forame("com.mysql.jdbc.Driver");
        String jdbcURL = "jdbc:mysql://localhost:3306/proyecto";
        SDBConnection conn = new SDBConnection(jdbcURL, "root",
                "passroot") ;
        Store store = SDBFactory.connectStore(conn, storeDesc);
        ds = DatasetStore.create(store) ;

}catch( ClassNotFoundException e ){
        e.printStackTrace();
String stringQuery =
      "PREFIX base: <http://www.owl-ontologies.com/ProyectoFinal.owl#> "+
       "PREFIX rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#>"+
       "PREFIX xsd:  <http://www.w3.org/2001/XMLSchema#>"+
       "SELECT ?x "+
       "WHERE { base:"+Reminder1+" base:"+deliveryReminderMadeBy+" ?x . }";


//Convierte la String que contiene la consulta a un objeto Query y la traduce a
   expresión de álgebra SPARQL. La variable 'ds' define el almacenamiento de
   los datos RDF cargados en memoria.
    QueryExecution qe = QueryExecutionFactory.create(stringQuery, ds) ;

 //Se selecciona el tipo de consulta a realizar, se ejecuta y se almacenan los
                    resultados en un bucle.
  ResultSet rs = qe.execSelect();

  while(rs.hasNext()){
        QuerySolution soln = rs.nextSolution() ;
        String link=soln.getLiteral("x").getString();
         resultados.add(link);
   }
```

2.1.3.4.1 ARQ

ARQ es un motor de consulta para Jena que soporta el lenguaje de consulta de RDF SPARQL y permite:

- ejecutar consultas SPARQL con filtrado si es necesario y realizar actualizaciones

en la base de datos.

- realizar funciones extendidas de SPARQL: agregación, group-by, subselect, negación,...

Para llevar a cabo las consultas se utiliza el API de implementación donde la mayor parte de las aplicaciones que se utilizarán se encuentran en el paquete "com.hp.hpl.jena.query". Solo en el caso de crear aplicaciones que construyan consultas automáticamente o modifiquen el comportamiento del motor de consulta será necesario utilizar otro paquete del API [10].

Las clases del paquete "com.hp.hpl.jena.query" son las siguientes:

- Query : representa la solicitud de consulta. Es un contenedor para todos los detalles de la consulta. Los objetos de ésta clase son normalmente creados utilizando los métodos de la clase QueryFactory que proporcionan acceso a varios analizadores.
- QueryExecution: representa la ejecución de la consulta.
- QueryExecutionFactory: se trata de un lugar para obtener las instancias de la clase QueryExecution.
- DatasetFactory: se trata de un lugar para realizar los conjuntos de datos, incluidos los DataSource (un DataSet actualizable).

Añadiendo para las consultas del tipo SELECT:

- QuerySolution : una solución simple a la consulta.
- ResultSet: se trata de un iterador que almacena todos los QuerySolution obtenidos.
- ResultSetFormatter: convierte un ResultSet en distintos formatos: texto, un grafo RDF o XML.

Los 4 tipos de consulta se corresponden con los 4 tipos de consulta propios de SPARQL.

- Consulta SELECT:

Los pasos básicos para hacer una consulta del tipo SELECT son:

- En primer lugar se crea la consulta a partir de un string utilizando para ello la clase QueryFactory.
- La consulta y el modelo o el conjunto de datos RDF que son requeridos se pasan a la clase QueryExecutionFactory para producir una instancia de la ejecución de la consulta.
- Los resultados son manejados en un bucle y finalmente la ejecución de la consulta es cerrada.

- Las consultas CONSTRUCT, ASK y DESCRIBE realizan el mismo esquema de pasos que para el caso de las consultas de tipo SELECT con la diferencia que una vez creada la instancia de la ejecución de la consulta con la clase QueryExecutionFactory no es necesario manejar los resultados con un bucle sino que es directo al crear el modelo con la clase Model, seleccionando el método específico para cada uno de las consultas. Estos métodos serán execConstruct(), execDescribe() y execAsk() para CONSTRUCT, ASK Y DESCRIBE respectivamente. Un ejemplo de una consulta de tipo SELECT es el siguiente:

```
String stringQuery =
"PREFIX base: <http://www.owl-ontologies.com/ProyectoFinal.owl#> "+
"PREFIX rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#>"+
"PREFIX xsd:  <http://www.w3.org/2001/XMLSchema#>"+
"SELECT ?x "+
"WHERE { base:"+Joseba+" base:"+sendsPractice+" ?x . }";
//Paso 1
QueryExecution qe = QueryExecutionFactory.create(stringQuery, ds) ;
// Paso 2
ResultSet rs = qe.execSelect();
//Paso 3
while(rs.hasNext()){
    QuerySolution soln = rs.nextSolution() ;
    String link=soln.getLiteral("x").getString();
    resultados.add(link);
 }
```

El ejemplo de la consulta anterior es el siguiente:

Figura 2.7 Resultado de un ejemplo de una consulta SPARQL

2.1.3.4.2 SDB

Es un motor de almacenamiento de consultas y documentos RDF, ambos casos para SPARQL, que funciona sobre una base de datos relacional. El almacenamiento es proporcionado por unas base de datos SQL siendo compatibles otros modelos si fuese necesario su uso: Microsoft SQL Server 2005, Oracle 10gR2, IBM DB2, PostgreSQL v8, MySQL 5.0, HSQLDB 1.8, H2 1.0.73. Estas bases de datos han de ser instaladas de forma independiente para realizar posteriormente su uso. [11]

Para acceder y gestionar el almacenamiento SDB es necesario el uso del API de Jena y una secuencia de comandos por consola que permiten trabajar con el almacenamiento de tripletes SDB.

- Los comandos por consola utilizados son:
  - sdbconfig: sirve para inicializar los datos creando un diseño inicial básico sin añadir todos los índices a la tabla 'triples'.
  - Sdbload: carga los datos en la base de datos.
  - Sdbconfig: añade todos los índices a la tabla 'triples'. Es necesario para que las consultas se ejecuten posteriormente en un tiempo razonable.
  - sdbquery: invoca una consulta SPARQL sobre la base de datos.

- Para utilizar SDB, es necesaria la descripción del almacenamiento realizado, para lo cual se utilizada un archivo de descripción del almacenamiento llamado "sdb.ttl". Este documento describe:

  - La configuración del depósito de los datos como son el diseño, el tipo de conexión y el tipo de la base de datos a utilizar.
  - Los detalles de la conexión a la base de datos como los datos de acceso necesarios para la conexión (nombre de la base de datos, host a utilizar y del usuario junto con su contraseña) y del driver utilizado para su correcto funcionamiento.

Un ejemplo del documento sdb.ttl es el siguiente:

```
@prefix rdfs:    <http://www.w3.org/2000/01/rdf-schema#> .
@prefix rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix ja:      <http://jena.hpl.hp.com/2005/11/Assembler#> .
@prefix sdb:     <http://jena.hpl.hp.com/2007/sdb#> .

# Store only - no connection details

<#mySstore> rdf:type sdb:Store ;
                    sdb:layout        "layout2" ;
                    sdb:connection    <#conn> ;
.
<#conn> rdf:type sdb:SDBConnection ;
                    sdb:sdbType         "mysql" ;
                    sdb:sdbName         "proyecto" ;
                    sdb:sdbHost         "localhost" ;
                    sdb:sdbUser         "root" ;
                    sdb:sdbPassword     "passroot" ;
                    sdb:driver          "com.mysql.jdbc.Driver" ;
```

- Diseño de la base de datos:

En SDB se realiza el almacenamiento de un conjunto de datos RDF en una base de datos SQL. Sin embargo, dicha base de datos no posee únicamente un único diseño. El más común a la hora de usar SDB en el diseño 2 "layout2".

Posee 4 tablas:

- 'Triples': para los grafos por defecto.
- 'Quads': para los grafos nombrados.
- 'Nodes': datos introducidos en la ontología realizada.
- 'Prefixes: prefijos utilizados para nombrar los namespaces.

El resultado de una consulta utilizando los comandos por consola es el siguiente:

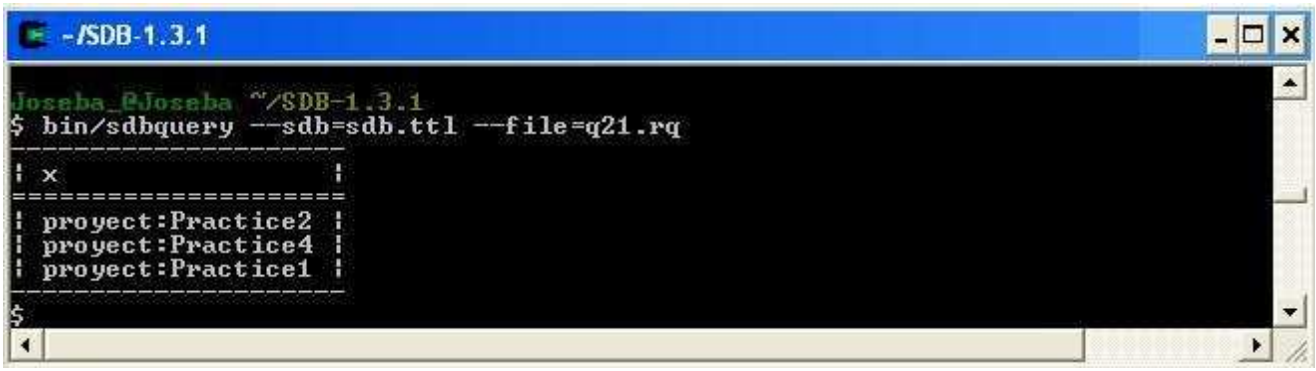


Figura 2.8 Resultado de un ejemplo de una consulta SDB

Si se quiere utilizar SDB desde Java en lugar de utilizarlo a través de los comandos por consola, tendrá que ser utilizado el API de Jena.

Las 3 clases principales a utilizar del API de Jena son:

- **Store**: cada base de datos es un objeto de la clase Store. Posee toda la información para formatear, cargar y acceder a una base de datos SDB.
- **SDBFactory**: a partir de esta clase y sus métodos se crearán los objetos de la clase Store.
- **DatasetStore**: representa un conjunto de datos RDF respaldado por un objeto de la clase 'Store'.
    - **GraphSDB**: proporciona el adaptador entre en API de Jena y la clase Store.
    - **SDBConnection**: elvuelve toda la conexión fundamental a la base de datos realizando al mismo tiempo todas las acciones relacionadas con el logueo

(identificación).

- **StoreDesc**: describe en menor detalle el almacenamiento que la clase Store.

Un ejemplo de una consulta SDB desde Java es la siguiente:

```
StoreDesc storeDesc = new
            StoreDesc(LayoutType.LayoutTripleNodesHash,DatabaseType.MySQL) ;

Class.forName("com.mysql.jdbc.Driver");
String jdbcURL = "jdbc:mysql://localhost:3306/proyecto";
SDBConnection conn = new SDBConnection(jdbcURL, "root", "passroot") ;
Store store = SDBFactory.connectStore(conn, storeDesc);
Dataset ds = DatasetStore.create(store) ;
```

2.1.3.4.3 TDB

TDB, al igual que SDB, es un motor de almacenamiento desarrollado en Java que permite almacenar gratos RDF y recuperarlos a través de consultas SPARQL. Se trata de un componente de Jena que permite ser usado como un almacenamiento RD no transaccional y de alto rendimiento sobre una sola máquina. [12]

Para acceder y gestionar el almacenamiento TDB es necesario el uso del API de Jena y una secuencia de comandos por consola.

Los comandos por consola utilizados son:

- **tdbloader**: es el gestor general y constructor del índice. Realiza las operaciones de carga más eficientemente que la simple lectura del documento RDF.
- **tdbquery**: invoca una consulta SPARQL sobre la base de datos. El almacenamiento está ligado a cada ejecución de este comando así que es importante medir el tiempo de la consulta usando –time.
- **tdbdump**: vuelca el almacenamiento con el formato N-Quads.
- **tdbstats**: realiza estadísticas para el conjunto de datos almacenado.

Descripción del almacenamiento.

Los comandos TDB usan una descripción de un ensamblador para el almacenamiento persistente o una referencia directa al directorio con el índice y los archivos de los nodos.

Los ensambladores de Jena son mecanismos generales para describir objetos mayoritariamente modelos o conjuntos de datos. Su utilidad más importante es que los datos de la aplicación sobre la que se va a trabajar pueden ser modificados sin cambiar el código del programa ya que la descripción se encuentra almacenada en un archivo independiente.

Se pueden definir 3 situaciones diferentes: para un conjunto de datos, un grafo o la mezcla de los dos. El esquema a seguir es el mismo para los 3 casos:

- Declarar los prefijos utilizados.
- Introducir el estado para poder cargar el TDB.
- Finalmente, la descripción del conjunto de datos.

Un ejemplo de la descripción del almacenamiento es el siguiente:

```
@prefix tdb:     <http://jena.hpl.hp.com/2008/tdb#> .
@prefix rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix ja:      <http://jena.hpl.hp.com/2005/11/Assembler#> .

[] ja:loadClass "com.hp.hpl.jena.tdb.TDB" .

<#dataset> rdf:type        tdb:DatasetTDB ;
 tdb:location "DB" ;
```

Si se quiere utilizar TDB desde Java en lugar de utilizarlo a través de los comandos por consola, tendrá que ser utilizado el API de Jena.

Para crear y conectar con un grafo TDB o un conjunto de datos RDF se utilizarán los métodos de la clase TDBFactory. Para especificar la conexión es posible dar el nombre de un directorio o dar el archivo del ensamblador descrito anteriormente.

Un ejemplo es el siguiente:

```
//Modo directo: introducir un resguardo de un TDB del directorio nombrado
String directory = "MyDatabases/DB1" ;
Model model = TDBFactory.createModel(directory) ;
...
model.close() ;
```

```
// Introducir un resguardo de un conjunto de datos TDB

String directory = "MyDatabases/Dataset1" ;
Dataset dataset = TDBFactory.createDataset(directory) ;
...
dataset.close() ;
```

2.2 WEB SEMÁNTICA Y EDUCACIÓN

La Web Semántica se entiende como una extensión a la Web que tenemos actualmente a la que se le proporciona un significado a la información con el objetivo de que exista una 'cooperación' entre ordenadores y usuarios. Por esto, la relación entre la educación y la Web semántica es *"lograr agentes software que interpreten el significado de los contenidos de la Web, para ayudar a los usuarios a desarrollar sus tareas" (Koper, 2004, p.16)*. [14].

Basándonos en esta idea, la Web Semántica en la educación tiene el objetivo de desarrollar e implementar meta programas de información concebidos como sistemas de información lógica acoplable, con una estructura computacional completa y ejecutable individualmente, aunque a su vez, tienen que posibilitar que otros sistemas puedan unirse para ampliar las funciones para las que fueron inicialmente creados.

La idea principal es favorecer la actividad formativa, haciendo hincapié en los entornos de enseñanza virtual. Lo que se pretende optimizar es el uso de unidades de aprendizaje, el rol del estudiante y las funcionalidades de búsqueda y navegación a través de la información. Por ejemplo, ayudar al estudiante a elegir las unidades de aprendizaje más apropiadas a sus intereses, objetivos y estilo de estudio. Sin embargo,

existen más casos a destacar como son permitir a los profesores y estudiantes identificar qué recursos tienen relación con otros, cuáles tienes propiedades similares, visualizar sus relaciones, etc.

En definitiva, se trata de poner a disposición de los usuarios herramientas que ayuden con su aprendizaje y que no se den por hecho sus utilidades o para qué van a ser utilizadas.

2.2.1 E-learning

Con la llegada de Internet, las posibilidades de acceso a la información incrementaron muy rápidamente y, a medida que éste iba evolucionando a mejor calidad con mayor de velocidad de acceso y transmisión de datos, el contenido que se podía consultar se vio modificado de simples textos a cualquier tipo de información mucho más elaborada y con mayor tamaño de almacenamiento como pueden ser animaciones, audio, video, etc. [15]

Gracias a Internet y al origen y desarrollo de las nuevas tecnologías, las formas de enseñar se han visto incrementadas y el modo presencial entre el docente y el discente ya no es el único modo de aprendizaje.

La interacción virtual entre los sujetos permitió la aparición de la educación a distancia utilizando las tecnologías de la comunicación, lo que se conoce con el nombre de e-learning.

El e-learning es una modalidad de aprendizaje que desarrolla el conocimiento mediante el uso de las nuevas tecnologías, centrándose en el uso de Internet. Se pasa de un contacto físico entre profesor y alumno a una interacción vía Internet utilizando para ello herramientas o aplicaciones de hipertexto como el correo electrónico, páginas Web, foros de discusión, chats o plataformas de formación, como soporte de los procesos de enseñanza.

La principal ventaja del e-learning es que ha eliminado la barrera espacio-temporal. La enseñanza hasta ahora exigía asistir presencialmente a un centro formativo localizado en un lugar específico y siguiendo unos horarios determinados. Con el uso del

e-learning el acceso a la información se puede realizar desde cualquier lugar remoto, eliminando así la barrera espacial. A su vez, la flexibilidad horaria es total ya que el alumno, en cualquier momento, puede acceder a la información de manera individual o interactuar con los alumnos de la misma "clase" o curso que se esté impartiendo, eliminando así la barrera temporal. Esta interacción es en tiempo real mediante los chats pero gracias a los foros de discusión no es necesario.

Otras ventajas que ha proporcionado son:

- El aumento de las personas con acceso a la formación.
- La interacción entre los propios alumnos, éstos con el profesor y con los recursos online.
- Prácticas en entorno de simulación virtual con menor inversión que en una modalidad de educación presencial.
- Actualización de los contenidos: los contenidos del curso pueden ser actualizados en cualquier momento con bajo coste.
- Reducción general de coste a nivel metodológico.

2.3 SISTEMA DE ENTREGAS DE E-LEARNING

2.3.1 Definición y funcionalidades

Un sistema de entregas nos permite, como bien su nombre indica, entregar el contenido de cualquier tipo de información para llevar a cabo una prestación rápida y fácil de éste. Trata de recopilar en un único destino toda la información necesaria de distintas fuentes, es decir, centraliza el envío de información en un punto.

Una de las tareas fundamentales que pueden proporcionar los sistemas software para el aprendizaje, son los módulos para entrega de tareas. La misión fundamental de estos módulos es la de posibilitar a los alumnos la entrega de prácticas, ejercicios, documentos, etc. a través de Internet , mientras que a los profesores les ofrece una serie de facilidades para la gestión de dichas entregas, su corrección, publicación, etc. Trogovov [16] divide el proceso de los sistemas de entregas en 4 partes: entrega, recolección, calificación, y devolución.

2.3.2 Características comunes de los sistemas de entregas on-line.

- Accesibilidad desde cualquier lugar con conectividad banda ancha.
- Disponibilidad 24 horas x 365 días.
- Usabilidad y sencillez de cara al usuario final: profesor o alumno. Ideal para usuarios que no tienen una experiencia previa de formación online.
- Aprovechamiento las ventajas de la formación a distancia, eliminando costes añadidos y desplazamiento de profesores y alumnos, eliminando la necesidad de compartir horarios, ya que pueden conectarse en momentos distintos
- Permiten al profesor generar sus propios cursos y contenidos mediante un asistente de contenido formativo incorporado, para el que no se requiere de formación informática previa.

2.3.3 Análisis de las ventajas de los sistemas de entregas de e-learning.

Las ventajas de la utilización de sistemas software de e-learning para la gestión de entregas son muchas. Por ejemplo, en [17] se citan los aspectos tecnológicos que soportan dichas ventajas de estos sistemas: la transmisión rápida de las entregas con independencia de la localización de alumnos y profesores, el formato digital de las entregas que permite guardarlas en un almacén central con datos asociados, y la capacidad de procesado de los ordenadores que permite realizar diferentes análisis y cálculos eficientes.

Con el uso de sistemas software de e-learning se ha mejorado tanto el envío de la información por parte de los alumnos como la recepción de sus feedbacks. No sólo se ha visto reducido su tiempo de recepción sino que ha mejorado la eficacia y valoración de los alumnos a cerca de éstos.

Respecto al uso de las tecnologías, no sólo se ha conseguido mejorar su almacenamiento, sino que se han eliminado problemas como pueden ser la poca inteligibilidad debido a la caligrafía de los alumnos, los alumnos han ganado confianza ya que queda demostrado que se han conseguido eliminar posibles plagios en sus trabajos.

Los sistemas de entregas de e-learning han ido evolucionando con el tiempo. En [17] se expone un modelo y se comenta sobre la evolución desde entregas con email a sistemas de entregas complejos. Esta evolución es mayor aún hoy en día con sistemas de entregas más depurados o la adaptación y utilización de sistemas como entornos de desarrollo como subversión.

2.3.4 Clasificación de los sistemas de entregas de e-learning.

Los sistemas de entregas de hoy en día se dividen en dos grandes grupos, según estén integrados como un módulo más de una herramienta de e-learning como puede ser un LMS, o si son aplicaciones exclusivamente dedicadas a la entrega para la docencia.

Por un lado, entre el grupo de módulos integrados en otras herramientas de e-learning nos encontramos:

2.3.4.1 LRN

LRN es actualmente el software libre más ampliamente utilizado en todo el mundo para la enseñanza a distancia. Se trata de una plataforma caracterizada por la presentación categorizada de la información y los recursos ofrecidos. Es un poderoso entorno para soportar la gestión del aprendizaje en cursos y comunidades. El acceso a dichos materiales se estructura mediante portales, haciendo su uso más sencillo y por consiguiente, asequible a todos los usuarios, principiantes, medios y avanzados. [18]

Fue diseñado como un sistema que transfiere tanto control y flexibilidad como sea posible para usuarios finales y los administradores del grupo de los cuales tanto su necesidad por la entrega de contenido y elección difieren.

Características comunes propias de este tipo de sistemas son la administración de grupos, el almacenamiento de archivos, la utilización de foros o el uso del calendario. También poseen un apartado de noticias y un buzón de sugerencias, autentificación para su uso, uso de chats, etc.

2.3.4.2 WebCT

Se trata de un sistema virtual de aprendizaje a través de Internet. Una de sus principales características es su amplia flexibilidad para poder llevar a cabo el diseño de los cursos, característica que queda respaldada por su gran uso por parte de numerosas instituciones educativas para el aprendizaje. Las herramientas que se utilizan en este tipo de sistemas son los foros de discusión o chats, sistemas de envío de emails, etc. Con ello se busca que los alumnos no únicamente aprendan sino que también compartan conocimientos y experiencias con el resto. [19]

2.3.4.3 Dokeos

Como las dos anteriores, es una aplicación de software libre con una interfaz de usuario realizada en varios idiomas para llevar a cabo cursos online, administrando para ello todos sus contenidos. Por ello, incluye distribución de contenidos, calendario, proceso de entrenamiento, chat en texto, audio y video, administración de pruebas y guardado de registros. Su principal característica es la sencillez de su uso, tanto para expertos como principiantes. Además del enfoque de e-learning puro, soporta formación mixta (online y presencial) ya que aporta muchas funcionalidades de trabajo en grupo que sirven para apoyar a una formación presencial previamente ofrecida. Se trata además de una herramienta reutilizable (una vez preparado el curso, puede seguirse tantas veces como se desee) y trazable, de cara a asegurar y certificar el seguimiento que el alumno ha hecho del curso ante instituciones y terceros (ayudas, subvenciones, etc.). [20]

2.3.4.4 Moodle

Moodle es una aplicación de software libre para dar consistencia a la educación a distancia proporcionando al profesor todas las herramientas necesarias para crear un curso al que el alumno podrá acceder fácilmente desde cualquier ordenador, por lo que elimina también las barreras espacio temporales. Con Moodle se permite favorecer el aprendizaje a través de la relación informal entre personas ya que el eje central es el alumno, no los contenidos o las herramientas, permitir crear un espacio personal de

aprendizaje, compartir conocimientos y experiencias con otros, favorecer el networking, fidelizar y retener a los participantes respecto a la institución, introducir herramientas, contenidos y elementos comunes a todos ellos. [21]

Por otro lado, en el grupo de aplicaciones que están exclusivamente dedicadas a la entrega se encuentran:

2.3.4.5 ASAP.

ASAP es un sistema de entregas realizado en y para la Universidad Carlos III. Éste permite llevar a cabo las características comunes a los sistemas de entrega como son la propia entrega de las prácticas por parte de los alumnos desde cualquier lugar, permite controlar por parte del profesor todas aquellas prácticas que no han sido entregadas, etc.

2.3.4.6 SUBVERSION

Los sistemas de control de versiones son herramientas software que permiten la automatización de tareas comunes sobre archivos (guardar, recuperar, registrar, borrar, identificar, mezclar) manteniendo una correcta gestión sobre las versiones de la información almacenada.

→SVN: Subversion es un sistema de control de versiones libre desarrollado por CollabNet como mejora sobre CVS (Concurrent Versions System). SVN permite gestionar ficheros y directorios, y los cambios realizados sobre éstos, recuperar las distintas versiones de los datos o examinar el histórico de cambios, permite ser llevado a cabo funcionamiento en local o en red, lo que permite compartir/colaborar en los proyectos desde distintos equipos y con diferentes usuarios. Todo esto está permitido ya que se trata de un depósito centralizado para almacenar y mantener información cuya base fundamental es el repositorio que está organizado de forma jerárquica en directorios y archivos. [22]

→ GIT: es otro sistema de control de versiones libre diseñado para manejar tanto proyectos de gran amplitud como pequeños repositorios personales, ambos casos de una manera sencilla y rápida. Cada uno de los directorios de trabajo de GIT es un repositorio

con un historial y una revisión completa, independientes del acceso por red o de un servidor central. GIT permite llevar a cabo un desarrollo distribuido, un desarrollo no lineal con un soporte robusto, te permite diseñar un kit de herramientas para su manejo, etc. [23]

2.4 SERVICIOS WEB

2.4.1 Introducción

Coloquialmente hablando, un Servicio Web es un sitio Web que nos permite realizar ciertas tareas a través de él, es decir, no tenemos que realizar todas nuestras tareas en nuestro equipo sino que le damos las instrucciones para realizarla y obtenemos un resultado.

Un servicio Web es un conjunto de funcionalidades contenidas en un mismo paquete que se encuentra publicado en la red para ser utilizado por otros programas, permitiendo que aplicaciones remotas puedan comunicarse. Son considerados como "ladrillos" para construir sistemas distribuidos o como un tipo de middleware (software de conectividad) donde los mensajes que se envían y reciben son de tipo SOAP.

Están basados en la tecnología XML:

- SOAP (Simple Object Access Protocol).
- WSDL (Web Service Description Language).
- UDDI (Universal Description Discovery and Integration)

El modo de funcionamiento no es únicamente individual sino que se pueden agrupar varios Servicios Web para proporcionar funcionalidades más avanzadas.

La idea de un Servicio Web es ofrecer servicios que son accedidos mediante diferentes mensajes, que pueden ser creados a partir de XML o un metalenguaje usado para especificar los lenguajes y protocolos necesarios. Por ejemplo, aquellos Servicios Web que utilizan XML para crear los mensajes, siguen el siguiente flujo:

- las peticiones y respuestas a un método serán documentos XML.
- las estructuras de datos se expresarán en XML.

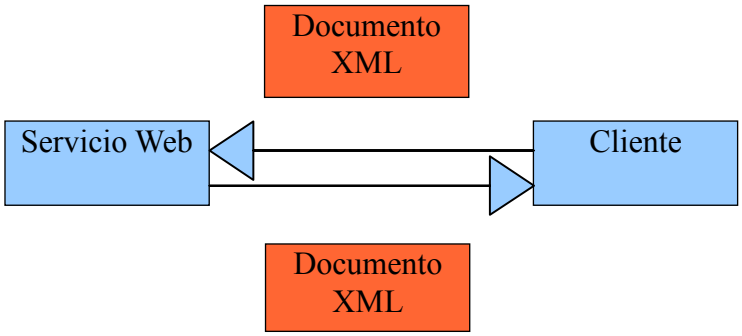


Figura 2.9 Representación de la idea fundamental de un Servicio Web a través de mensajes XML

Los Servicios Web poseen una amplia variedad de características siendo las más destacables las siguientes:

- Interoperabilidad: los Servicios Web pueden ser escritos en diferentes lenguajes de programación gracias al protocolo SOAP (XML).
- Ubicuidad: la comunicación se basa en HTTP y XML.
- Facilidad de uso: su utilización es sencilla ya que se usan entornos de desarrollo que facilitan el desarrollo y adaptación.
- El modelo de desarrollo se basa en "módulos" de bajo acoplamiento.
- Estandarización: los Servicios Web están basados en estándares cuyas funciones son:
  - evitar desarrollos basados en estándares cerrados.
  - reutilizar de servicios.
  - proporcionar amplio soporte de los fabricantes.

Los Servicios Web actualmente están intentando proporcionar otras cualidades como la accesibilidad, confianza en los proveedores, seguridad, posibilidad de realizar transacciones, escalabilidad, manejabilidad, cobro y pruebas y, para ello, las soluciones que ha propuesto son tolerancia a fallos, paralelismo, distribución, auto-corrección, un diseño basado en capas y en componentes simples y la utilización de los estándares comentados anteriormente.

Un Servicio Web puede ser comparado con un “modelo de componentes”.

Modelo de acoplamiento	Modelo de los Servicios Web
Alto acoplamiento	Bajo acoplamiento
Aplicaciones de Intranet	Aplicaciones de Internet
Desarrollos dentro de la empresa	Desarrollos entre empresas
Transporte basado en TCP/IP	Transporte basado en HTTP
Granularidad fina	Granularidad gruesa.

Figura 2.10 Modelo de acoplamiento vs Modelo de Servicios Web

Tras un Servicio Web existe un servicio o componente en un lenguaje tradicional (Java, C++, etc.) que puede accederse tanto con los mecanismos proporcionados por el lenguaje de programación empleado como de manera local o remota.

Los requisitos de un Servicio Web son:

- Descubrimiento para que se encuentren.
- Descripción del servicio.
- Transporte para la comunicación entre el servicio y quien lo usa.
- Entorno para ejecutar el servicio.

2.4.2 Arquitectura de un Servicio Web.

La arquitectura de un Servicio Web define un envoltorio para acceder a un código preexistente de manera independiente del lenguaje y de la plataforma.

Se conforma de tres partes:

- **Proveedores** del servicio que implementan el servicio y publican la descripción del servicio tanto de manera local como en un registrador. La

descripción de realiza con WSDL, donde se detalla el interfaz y la implementación del servicio (tipos de datos, operaciones y localización en la red).

- **Registros** del servicio para mantener una base de datos sobre los servicios registrados. Esto es opcional ya que podría darse el caso en el que la descripción del servicio estuviese en el proveedor directamente.

- **Solicitantes** del servicio que primero buscan* el servicio en un registrador y posteriormente se enlazan con el proveedor para acceder al servicio. En realidad se trata de una aplicación que realiza las peticiones y procesa las respuestas, ambas según el estándar SOAP. Existe la posibilidad de ofrecer una interfaz al usuario final para facilitar su uso.

    - *Búsqueda: El proceso de búsqueda engloba dos fases, una estática en tiempo de diseño en la que se obtiene la interfaz para el desarrollo del programa, y una dinámica en tiempo de ejecución en la que se recupera la información de localización para poder posteriormente invocar al servicio.

Figura 2.11 Arquitectura 1 de un Servicio Web

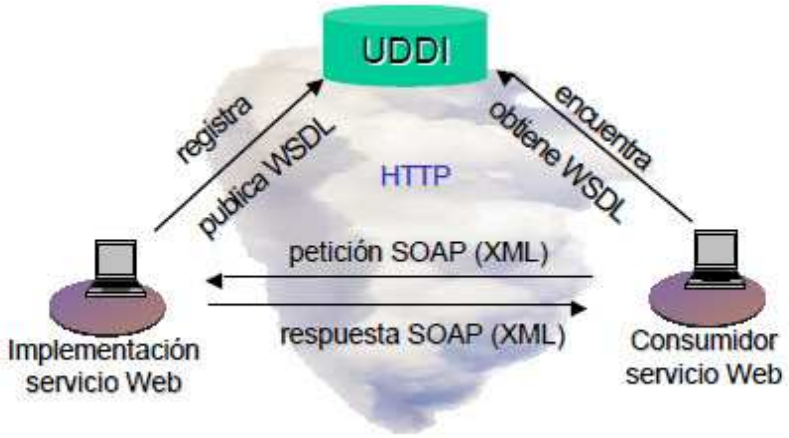


Figura 2.12 Arquitectura 2 de un Servicio Web.

Cuando se habla de 'servicio' se hace referencia a la implementación software proporcionada por el proveedor que al mismo tiempo puede actuar como solicitante si usa otros Servicios Web.

2.5.2.1 Ciclo de vida de un Servicio Web

El **ciclo de vida** de un Servicio Web se completa con 4 pasos:

- Construcción: realización del desarrollo y las pruebas del Servicio Web, y la descripción del interfaz e implementación del servicio.
- Desplegado: publicación del interfaz y definición del servicio en un registro y disposición en el entorno de ejecución del proveedor.
- Ejecución: disponibilidad por si algún cliente lo invoca.
- Gestión: es necesario llevar a cabo tareas de seguridad, disponibilidad, rendimiento, calidad, etc.

Sin embargo, un Servicio Web no tiene definido un único escenario para el ciclo de vida descrito anteriormente sino que existen 4 escenarios diferentes:

- **Green-Field:** Es el escenario más extenso ya que se crea el Servicio Web desde cero, creando en primer lugar una nueva clase que implementará la funcionalidad que se le quiera proporcionar al Servicio Web y posteriormente se utilizan las

herramientas para convertir la clase en el Servicio Web. Finalmente se publica.

- **Bottom-Up:** Es similar al tipo anterior pero con la diferencia de que la funcionalidad ya está implementada

- **Top-Down:** se parte de la descripción del interfaz (el archivo wsdl), y se implementa la funcionalidad del Servicio Web creando un objeto o clase que corresponda según la especificación.

- **Meet-in-the-middle:** es una combinación de los escenarios Bottom-Up y Top-Down. En el punto de partida se dispone de la especificación del Servicio Web y de la implementación de su funcionalidad. Para realizar el Servicio Web habrá que "cuadrar" ambos elementos.

2.4.2.2 La Pila conceptual de un Servicio Web.

Figura 2.13 Pila conceptual de un Servicio Web

La base de la pila de los Servicios Web, figura 2.13, es la red, ya que los Servicios Web deben ser accesibles por la red para poder ser invocados. Debido a su gran expansión, HTTP es el estándar de facto aunque se podrían utilizan otros como SMTP, FTP, etc. En todo caso, la capa de red se plantea como algo completamente transparente para el desarrollo de Servicios Web gracias a su independencia e interoperabilidad.

Un nivel superior se encuentra XML, encargado de la comunicación ya que se usa como formato para el protocolo de mensajes. El protocolo seleccionado que cumple los requisitos necesarios es SOAP (Simple Object Access Protocol), mecanismo estándar para el envío de mensajes y llamadas remotas usando XML que define en este lenguaje tanto el formato de las peticiones/respuestas como el de los errores. Puede utilizar múltiples protocolos de red (HTTP, SMTP, RMI, etc.). Es posible mantenerse aislado de los detalles de implementación SOAP para la utilización de los Servicios Web.

La capa de descripción de servicios tiene el objetivo de facilitar el uso y creación de los Servicios Web. Para ello se utiliza WSDL que define el interfaz y los mecanismos de interacción del servicio para automatizar la creación de clientes de acceso al Servicio Web.

Por último están la capa de publicación/descubrimiento ya que un Servicio Web debe ser publicado para poder ser invocado por los clientes. La forma más simple consiste en la publicación estática del fichero WSDL, que se recupera a un fichero local. También es posible localizarlo de forma dinámica usando un registro UDDI.

2.4.3 Protocolos y lenguajes empleados.

XML eXtensible Markup Language	Un Servicio Web es una aplicación web creada en XML.
WSDL Web Services Definition Service	Este protocolo se encarga de describir el Servicio Web cuando es publicado. Es el lenguaje XML que los proveedores emplean para describir sus Servicios Web.
SOAP Simple Object Access Protocol	Permite que programas que corren en diferentes sistemas operativos se comuniquen. La comunicación entre las diferentes entidades se realiza mediante mensajes que son rutados en un sobre SOAP.
UDDI Universal Description Discovery and Integration	Este protocolo permite la publicación y localización de los servicios. Los directorios UDDI actúan como una guía telefónica de Servicios Web.

Figura 2.14 Protocolos y lenguajes utilizados en un Servicio Web

2.4.3.1 SOAP (Simple Object Access Protocol)

Es un estándar del W3C desarrollado inicialmente por IBM, Microsoft, DevelopMentor y UserLand Software que especifica todas las reglas necesarias para ubicar Servicios Web XML, integrarlos en aplicaciones y establecer la comunicación entre ellos.

Es un protocolo que define la sintaxis XML para el intercambio de mensajes y la codificación de estructuras de datos con independencia del protocolo de transporte (HTTP).

Contempla dos modelos de comunicación de los cuales el proveedor del Servicio Web seleccionará el adecuado a usar:

- RPC donde los mensajes SOAP son más estrictos, se realiza la llamada a un método con argumentos y los tipos de datos son codificados en XML.
- Orientado a un documento donde se envía un documento XML como datos. En este modelo hay mayor libertad para elegir la estructura xml del mensaje enviado.

Hasta ahora, las peticiones Web se automatizaban mediante mecanismos propios de entornos controlados (intranet) ya que los servicios estaban pensados para usuarios fijos y éstos obtenían la respuesta en formato HTML directamente. Además era necesario crear parseadores HTML para poder obtener toda la información necesaria de las respuestas. Al estar establecidos el servicio para usuarios fijos, un pequeño cambio realizado en cualquier capa del servicio afectaba a todas las capas superiores.

Debido a todos los problemas existentes con la automatización de las peticiones Web, se llevó a cabo un nuevo proceso de automatización con el intercambio de información XML. Para hacerlo posible son necesarios formatos XML específicos para la comunicación entre los dos extremos que serán Servicios Web. Adicionalmente se puede agregar un API para facilitar la construcción y procesado de los contenidos XML.

2.4.3.1.1 Funcionamiento de SOAP

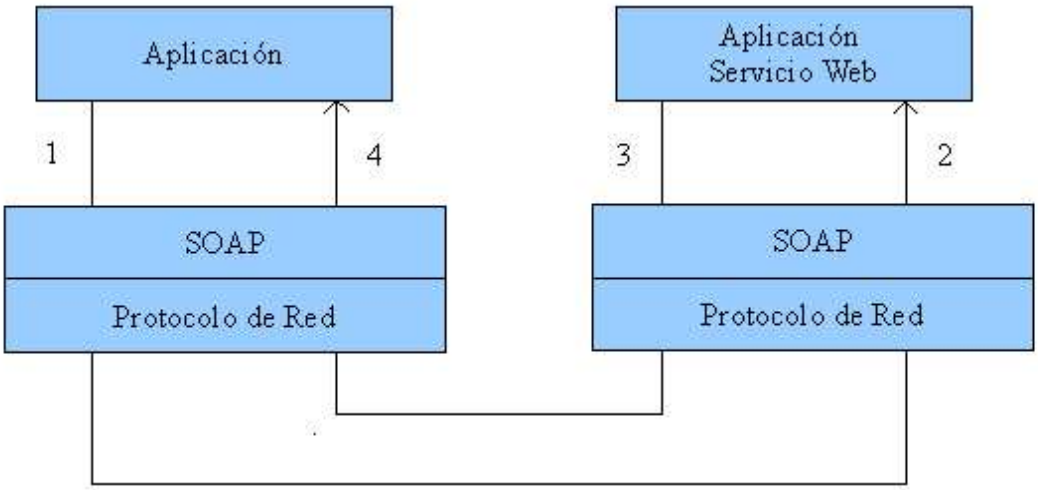


Figura 2.15 Comunicación SOAP

1. El cliente crea un mensaje SOAP, la petición que invoca al Servicio Web. El cliente SOAP interactúa con el protocolo de red para enviar el mensaje a través de la red.
2. El mensaje llega hasta el intérprete SOAP del servidor, que encamina la petición al Servicio Web. El intérprete SOAP se encarga de realizar las conversiones necesarias en objetos específicos del lenguaje de programación (marshall/unmarshall). La conversión viene definida por los esquemas incluidos dentro del mensaje.
3. El Servicio Web procesa la petición y genera una respuesta, que genera un mensaje SOAP, que se envía de vuelta por la red.
4. El mensaje es recibido, y convertido a los tipos específicos de la aplicación.

2.4.3.1.2 Estructura de un mensaje SOAP.

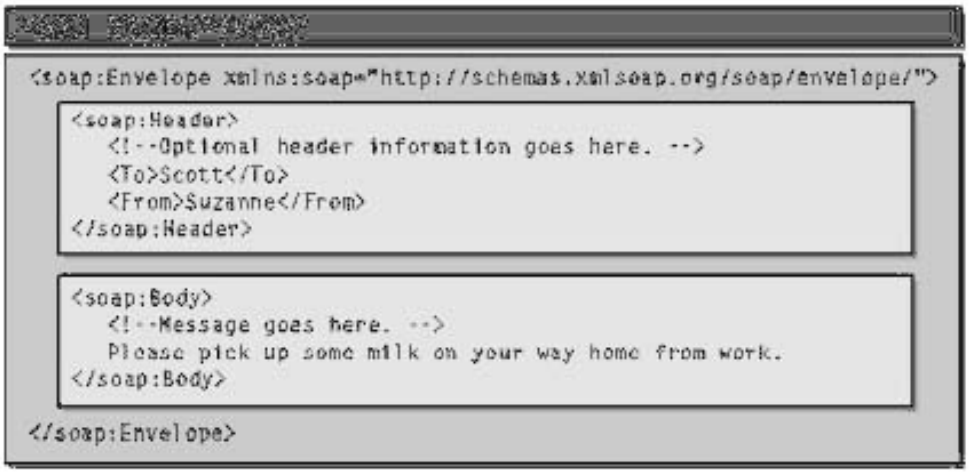


Figura 2.16 Estructura de un mensaje SOAP

- Header (cabecera): es un elemento opcional (en el caso de existir, debe ser el primer hijo del envelope). Es extensible y está pensado para incluir información auxiliar para intermediarios o para el usuario final.
- Body (cuerpo): es la parte principal del mensaje, Cuando se hace una petición, el cuerpo contiene un único elemento con el espacio de nombres asociados al nombre completo del servicio invocado, el nombre del método invocado y los argumentos del método con el tipo necesario.

2.4.3.2. WSDL

Es un formato XML para describir los Servicios Web ofrecidos por un proveedor. Se creará un wsdl diferente para describir cada uno de los servicios, qué operaciones se soportan y cómo se utilizan. También permite describir el formato que el cliente debe seguir para solicitar cada una de las operaciones. Es equivalente, conceptualmente, a los IDL (Interfaz Definition Languaje) de CORBA.

Un WSDL es como un contrato entre el servidor y el cliente. El servidor se compromete a proporcionar ciertos servicios siempre y cuando el cliente mande una petición SOAP con el formato adecuado.

Un archivo WSDL permite definir el interfaz abstracto (tipos de datos, mensajes y operaciones), los enlaces a formatos de mensajes (SOAP) y los enlaces a protocolos (HTTP).

2.4.3.2.1 Estructura de un fichero WSDL

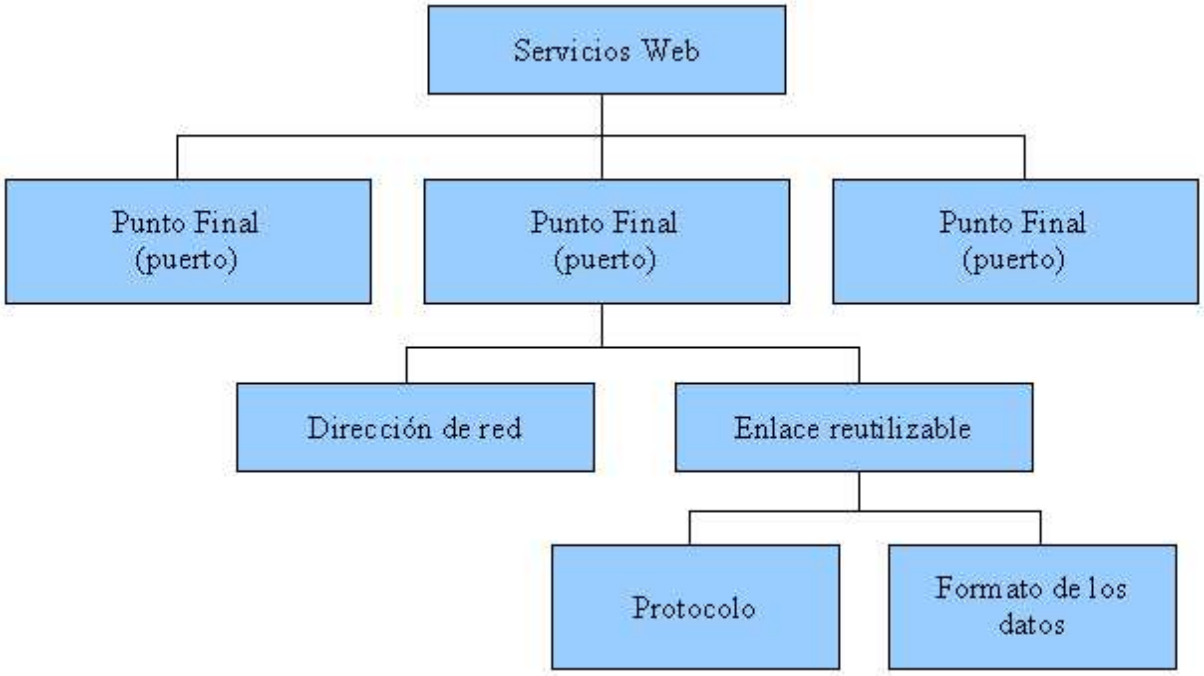


Figura 2.17 Estructura de un fichero WSDL

Se pueden tener distintos puertos para varios servicios.

El enlace reutilizable tiene un determinado interfaz de manera abstracta.

Un archivo WSDL define:

- Los tipos de datos, basados en XML Schema.
- Los mensajes: descripciones abstractas de los datos que se intercambian. Son los elementos básicos de la información.
- Las operaciones: como flujos de entrada y salida de mensajes.
- Los tipos de puertos: colecciones de operaciones que se van a agrupar.

- Enlace reutilizable con un protocolo concreto y con la especificación de un formato de datos.
- Puerto: punto de acceso al servicio.
- Servicio.

2.4.3.2.2 Elementos que aparecen en un documento WSDL.

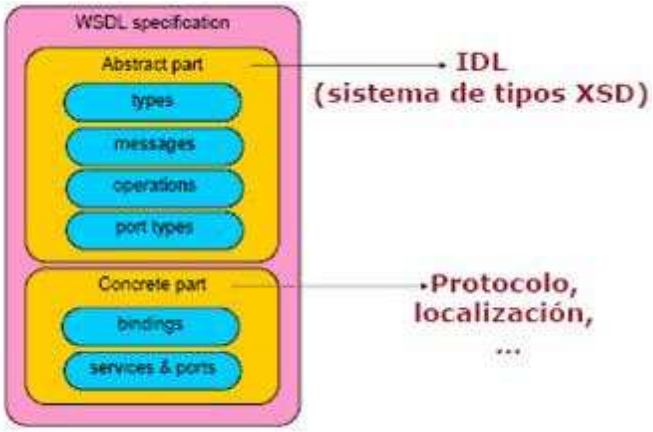


Figura 2.18 Elementos de un fichero WSDL

- Types: contenedor de las definiciones de los tipos de datos.
- Message: definición abstracta de los datos que se están comunicando.
- Operation: descripción abstracta de una acción admitida por el servicio.
- Port Type: es el conjunto abstracto de operaciones admitidas por uno o más puntos finales.
- Binding: es la especificación del protocolo y del formato de datos para un tipo de puerto determinado.
- Port: es el punto final único que se define como la combinación de un enlace y una dirección de red.
- Service: es la colección de puntos finales relacionados.

2.4.3.3 UDDI (Universal Description Discovery and Integration)

Es una norma para describir los componentes disponibles de Servicios Web. Este estándar permite a las empresas registrarse en un tipo de directorio, sección amarilla,

de Internet, que les ayuda a anunciar sus servicios y ofrece una clasificación de éstos tanto en función de los servicios como de las empresas que los ofrecen, de tal forma que las compañías puedan encontrarse unas a otras y realizar transacciones en la Web. El proceso de registro y consultas se realiza utilizando mecanismos basados en XML y HTTP(S). En el proyecto UDDI se trabaja para proveer un método de acceso común a los meta datos necesarios para determinar si un elemento de código previamente elaborado es suficiente, y si lo es, cómo accederlo.

Una vez implementado un Servicio Web y definido un documento WSDL, es posible confiar en que todos los posibles clientes encuentren directamente esta especificación, o bien se puede hacer uso de un repositorio UDDI para publicar dicho servicio.

UDDI es un modelo de registro público que permite a los desarrolladores de servicios publicar sus Servicios Web, y a los consumidores encontrar los que más les interesen. Evidentemente, una organización o particular puede desempeñar ambos roles simultáneamente.

Las especificaciones UDDI consisten en un modelo de repositorio y un API para el registro y consulta de los Servicios Web, disponible sobre SOAP. Dentro de este repositorio se contempla la existencia de información sobre:

1. Organizaciones:

    - Páginas blancas: direcciones, contactos e identificadores conocidos.
    - Páginas amarillas: clasificaciones basadas en taxonomías estándar (localización y tipo de productos/servicios).
    - Páginas verdes: información técnica sobre los servicios que se exponen.

2. Servicios Web:
    - Enlace a la definición técnica del servicio (tModel), referenciando su documento de especificación formal (WSDL), mediante una URL externa.
    - Colección de tModel, como puntos de acceso al Servicio (Binding Template), las ubicaciones donde conectarse para usar el servicio.
    - El modelo de registros UDDI contempla la existencia de:
        - Registros públicos, empresas que ofrecen un punto de acceso público,

donde se registran los servicios ofertados a terceros.

- Registros privados, internos a las organizaciones, donde se registran los servicios de ámbito local.

No existe una relación formal entre UDDI y WSDL, ya que la especificación de los registros UDDI se realizó de forma que sea independiente del formalismo utilizado para la descripción final del Servicio. Aún así, la información para describir un servicio, proporcionada por un documento WSDL, complementa a la información que se puede encontrar en un registro UDDI.

Un error habitual es considerar que las descripciones completas de los Servicios se encuentran dentro del repositorio UDDI y lo que se almacena es un enlace con la localización de dicha descripción, así como una clasificación de dicho Servicio.

2.4.4 Ciclo de implementación y uso de un Servicio Web

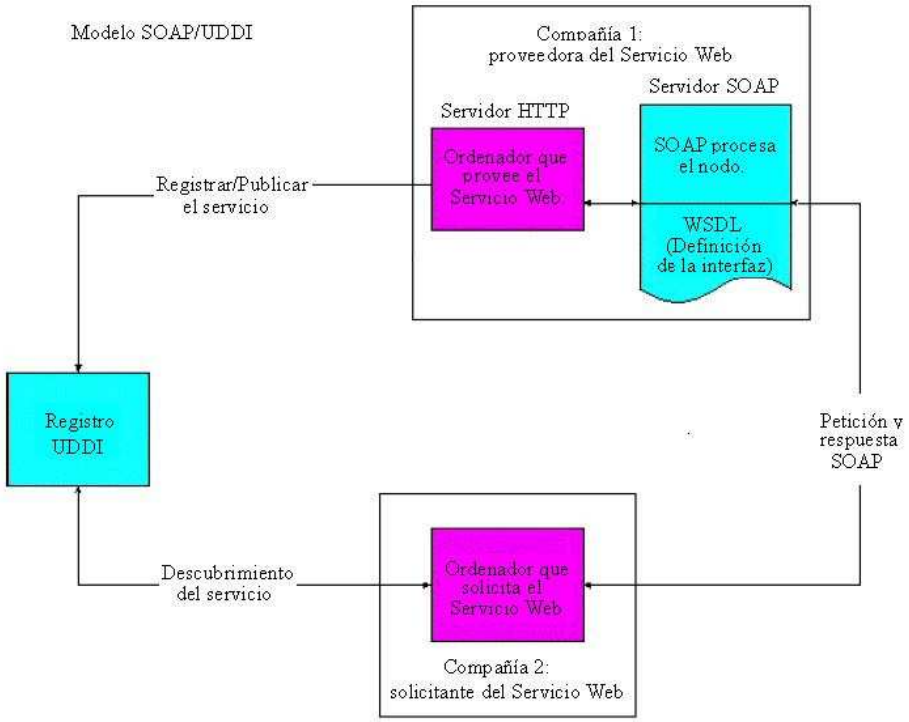


Figura 2.19 Ciclo de implementación y uso de un Servicio Web.

En ciclo de despliegue representado en la figura 2.19 sigue los siguientes pasos:

- El proveedor del servicio crea el Servicio Web.
- El proveedor describe el servicio con el archivo WSDL.
- El proveedor registra el servicio en el registro UDDI.
- Otro servicio o usuario localiza y solicita el servicio registrado al consultar los registros UDDI almacenados.
- El servicio o usuario solicitante escribe una aplicación que liga el servicio registrado utilizando SOAP.
- Se intercambian datos y mensajes XML sobre HTTP.

2.4.5 Tecnologías utilizadas

2.4.5.1 AXIS

Apache Axis es una implementación de SOAP que proporciona librerías necesarias para la generación de Servicios Web, un compilador de Java a WSDL y un compilador de WSDL a Java. Genera el documento WSDL correspondiente a un interfaz Java, dicha interfaz está sujeta a ciertas restricciones. El documento WSDL permite generar ficheros stubs (proxies) y skeletons para invocar e implementar Servicios Web. De esta manera un cliente escrito sobre cualquier plataforma pueda invocar el Servicio Web.

A continuación se visualiza como emplear Axis para generar el Servicio Web:

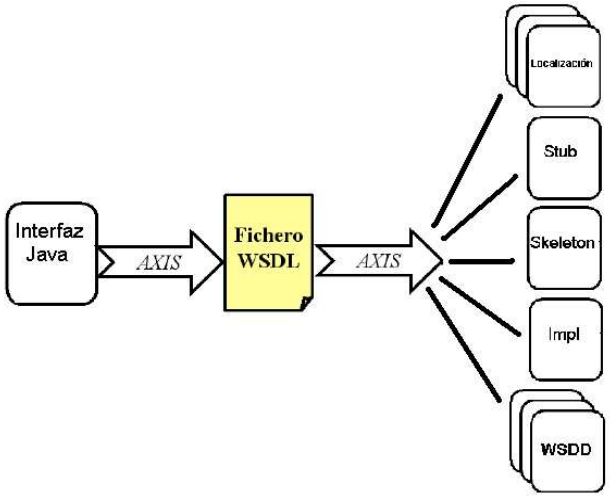


Figura 2.20 Uso de AXIS en la generación de un Servicio Web

Pasos de generación de ficheros:

1. Crear la interfaz del Servicio Web.
2. A partir de esta interfaz, generar el archivo .wsdl.
3. Mediante este archivo, generar los ficheros del Servicio Web:

- Interfaz del Servicio Web que contiene los métodos publicados por este. Esta interfaz se empleará tanto en el lado del cliente como en el lado del Servidor.

- Archivos de localización: una interfaz y una clase que la implementa. Ofrecen métodos para la localización del Servicio Web a partir del punto de enlace indicado en el fichero de descripción .wsdl, u otros indicados cómo parámetros. Actúan del lado del cliente como factoría para objetos que representan al Servicio Web.

- Archivo Stub para la codificación-decodificación entre los métodos y objetos Java y las peticiones-respuesta SOAP. Se emplea de forma transparente en clientes del Servicio Web siguiendo el patrón proxy.

- Archivo Impl que ha de albergar la implementación de las operaciones ofrecidas por el servicio y por tanto el archivo generado automáticamente no contiene más que la cabecera de los métodos. Además de esto en este fichero tiene lugar la codificación-decodificación de objetos Java y peticiones-respuesta SOAP del lado del servidor.

- Archivo Skeleton para la conexión del Servicio Web con el motor de Axis. Contiene la definición de cada método publicado. Actúa del lado del servidor realizando una llamada al método correspondiente del archivo Impl cada vez que se invoca una operación del Servicio Web.

- Archivos .wsdd (deploy.wsdd y undeploy.wsdd), uno para publicar el Servicio Web y otro para eliminar el Servicio Web.

4. Publicar el Servicio Web mediante la ejecución del archivo .wsdd correspondiente.

2.4.5.2 Tomcat

Tomcat (llamado también Jakarta Tomcat o Apache Tomcat) funciona como un contenedor de Servlets desarrollado bajo el proyecto Jakarta de la Apache Software Foundation.

Implementa las especificaciones de los servlets y de las JSPs (JavaServer Pages) de Sun Microsystems.

Es considerado un servidor de aplicaciones

La jerarquía de directorios de Tomcat es la siguiente:

- bin - arranque, cierre, y otros scripts y ejecutables
- common - clases comunes que pueden utilizar Catalina y las aplicaciones Web
- conf - ficheros XML y los correspondientes DTDs para la configuración de Tomcat
- logs - logs de Catalina y de las aplicaciones
- server - clases utilizadas solamente por Catalina
- shared - clases compartidas por todas las aplicaciones Web
- webapps - directorio que contiene las aplicaciones Web
- work - almacenamiento temporal de ficheros y directorios

2.4.6 Ciclo de despliegue

Para llevar a cabo el despliegue de un Servicio Web se puede hacer de dos maneras diferentes, utilizando JWS o usando otro método que nos da la herramienta AXIS.

Se usa un WSDD o descriptor de despliegue que se generará de manera automática, por lo que se dice que es un fichero codificado en XML que contiene una descripción de lo que se despliega de un Servicio Web (tipo de despliegue, clases y métodos que contiene, etc.

Los pasos que conforman el ciclo de despliegue son los siguientes:

1. Escribir el servidor

Si se quieres evitar errores de nomenclatura e invocación de las clases, se forzará a que todas las clases del servicio formen parte de un mismo paquete. Se realizará una clase Java principal que implemente las funcionalidades necesarias y una interfaz que será usada para generar el dichero WSDL.

2. Compilar y desplegar el Servicio Web.

El primer paso es compilar la clase Java del Servicio Web y su interfaz. A continuación, generar el fichero WSDL con la librería Java2WSDl incluida en AXIS.

Un ejemplo del comando a utilizar es:

java org.apache.axis.wsdl.Java2WSDL -o nombreWsdl.wsdl -l "http://localhost:puerto:axis/services/nombreServicioWeb" -n urn:nombreEspacio -p"nombrePaquete" urn:nombreEspacio nombreParque.claseJava.

Posteriormente, se generará el resto de ficheros necesarios para el despliegue del Servicio Web. Para ello, se utilizará el WSDL generado en el paso anterior y el comando WSDL2Java de AXIS.

Un ejemplo del comando a utilizar es:

java org.apache.axis.wsdl.WSDL2Java -o . -d Session -s -p nombrePaquete.paquete2 nombreWsdl.wsdl

Con este comando le estamos diciendo que genere, a partir del fichero actual donde se encuentra el paquete que contiene la clase Java y la interfaz, una nueva clase servidora para cada sesión, generando también los ficheros para el despliegue del servicio. Los ficheros creados son los enumerados en el punto 3 del apartado anterior de AXIS.

El siguiente paso a realizar es rellenar el envoltorio BindingImpl ya que no es mecánico como los anteriores. Se debe realizar una clase que realice las mismas funciones que la clase inicial Java. Por ello, lo más fácil es crear un objeto de esa clase y utilizar todos sus métodos.

Para seguir desplegando el Servicio Web es necesario compilar todas las clases proporcionadas por WSDL2Java y poner nuestro código disponible en AXIS indicándole cuál es la implementación del Servicio. La manera más eficaz de realizar esta tarea es empaquetar todo el código en un archivo jar y ponerlo en el lugar indicado, que es la carpeta /lib de axis que a su vez se encuentra dentro de tomcat.

Por último sólo queda desplegar el Servicio Web utilizando el archivo deploy.wsdd generado con WSDL2Java.

Un ejemplo del comando a utilizar es:

java org.apache.axis.client.AdminClient nombrePaquete/paquete2/deploy.ws

3. Escribir el cliente con los stubs generados.

Debido a la forma en la que ha sido realizado el Servicio Web, para generar el cliente se utilizarán los stubs generados por el WSDL2Java de AXIS en lugar de usar interfaces de invocación dinámica para el cliente. Se usarán por tanto la clase Service, ServiceLocator y la interfaz de la clase.

El stub contiene el endpoint, el servicio y la llamada, obteniendo en el cliente un objeto remoto en lugar de una llamada. Esto es conveniente cuando se quieren realizar varias invocaciones port-types.

4. Compilar y ejecutar el cliente

Este último paso consiste en compilar la clase cliente como cualquier otra clase Java y ejecutarla, teniéndose que obtener el resultado buscado en el método de la clase Java inicial.

2.5 LA HERRAMIENTA PROTÉGÉ

Para la realización de la ontología del Sistema de Entrega de Prácticas se utilizó Protégé, un editor de código abierto con el OWL-Plugin, que es una herramienta integrada de software para desarrollar sistemas basados en el conocimiento, es decir, ontologías. Es una aplicación escrita en Java con el uso de Swing para crear su interfaz gráfica.

Los requisitos para poder utilizar la aplicación son:

- Protégé 2.1 o superior.
- El plugin Protégé-OWL: permite describir los conceptos y nuevas facilidades con un mayor abanico de operaciones (por ejemplo, los operadores lógicos y,o y negación). Está basado en un modelo lógico diferente que hace posible que los conceptos sean definidos así como descritos. Gracias a ello, conceptos completos puedes ser definidos por conceptos simples. Además de esa simplicidad añadida a la hora de elaborar la ontología, se puede usar un razonador para mantener la jerarquía de los conceptos bien estructurada, es decir, comprobar que todos los estados y definiciones de la ontología sean coherentes entre sí.
- El plugin Protégé-Wizards.

Protégé te permite:

- Modelar una ontología de las clases que describes un tema particular.
- Crear una herramienta de adquisición de conocimiento para recoger conocimiento.
- Entrar en casos específicos de datos y de la creación de una base de conocimiento.
- La ejecución de los casos.

2.5.1 Ontologías OWL

Las ontologías son usadas por las personas, las bases de datos y aplicaciones que necesiten compartir un dominio de información para capturar el conocimiento sobre

ciertos dominios de interés. Describen los conceptos del dominio y las relaciones que existen entre todos éstos pudiendo ser interpretadas por una máquina. En otras palabras, definen vocabularios que las máquinas pueden entender y que son especificados con la suficiente precisión como para permitir diferenciar términos y referenciarlos de manera precisa.

Una ontología es posible tanto para grandes taxonomías que caracterizan sitios Web (p.e. yahoo), como para categorizadores para vender y sus características (p.e. Amazon).

OWL (Web Ontology Language) es el desarrollo más reciente sobre los lenguajes de ontologías estándar del W3C (World Wide Web). Es el lenguaje de Ontologías Web, es decir, un lenguaje compatible con el W3C y con la Web Semántica en particular.

Las novedades introducidas por OWL sobre las ontologías son:

- Compatibilidad con los estándares Web de accesibilidad e internacionalización.
- Abiertas y extensibles.
- Capacidad de ser distribuidas a través de varios sistemas.
- Son escalables según las necesidades de la Web.

Los principales sistemas de uso de ontologías en la Web son:

- Portales Web
  - Reglas de categorización utilizadas para mejorar la búsqueda
- Colecciones Multimedia
  - Búsquedas basadas en contenido para medios no textuales
- Administración de Sitios Web Corporativos
  - Organización taxonómica automatizada de datos y documentos
  - Asignación entre Sectores Corporativos (¡fusiones!)
- Documentación de Diseño
  - Explicación de partes "derivadas" (p.ej. el ala de un avión)
  - Administración explícita de Restricciones

- Agentes Inteligentes
  - Expresión de las Preferencias y/o Intereses de los usuarios
  - Mapeo de contenidos entre sitios Web
- Servicios Web y Computación Ubicua
  - Composición y Descubrimiento de Servicios Web
  - Administración de Derechos y Control de Acceso

2.5.1.1 Componentes de las ontologías OWL.

Las ontologías OWL pueden ser de tres modelos distintos según el sublenguaje utilizado (OWL-Lite, OWL-DL o OWL-Full) pero siempre poseen los mismos componentes independientemente del sublenguaje seleccionado: Individuos, Propiedades y Clases que en términos de Protégé corresponderían a Instances, Slots y Classes.

- Individuos: representan los objetos del dominio en el que estamos interesados. A la hora de crear los individuos hay que declarar qué nombre referencia a cada individuo ya que se puede dar el caso de que dos nombres distintos hagan referencia al mismo, caso que tiene que ser declarado explícitamente para no producirse errores.

- Propiedades: relaciones binarias que relacionan o enlazan individuos.

  - Existen tres tipos de propiedades:

    - Propiedades del tipo de datos: enlazan un objeto con valor del tipo de dato del XML-Schema o con un literal RDF.

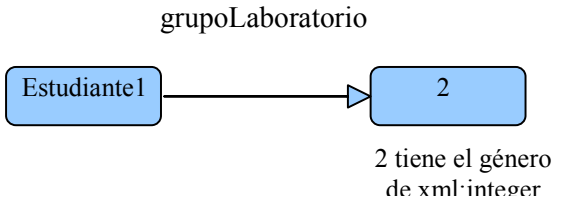


Figura 2.21 Representación de la propiedad tipo de dato

- Propiedades del objeto: enlazan un objeto con otro objeto.

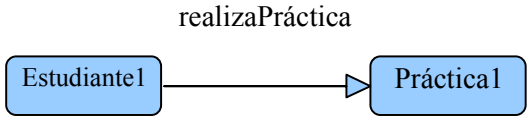


Figura 2.22 Representación de la propiedad del objeto

- Propiedades de comentario: pueden ser utilizadas para añadir información a la clase, al individuo o a cualquiera de las dos propiedades anteriores.

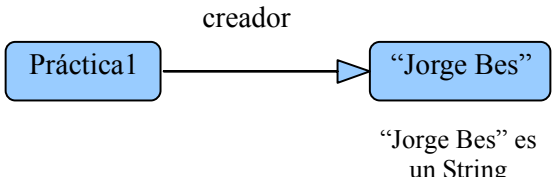


Figura 2.23 Representación de la propiedad de comentario.

◦ Las características de las propiedades:

- Las Propiedades funcionales: si una propiedad es funcional está restringiendo a una única relación de un individuo con otro vía esa propiedad, es decir, un individuo nunca podrá estar relacionado con dos o más individuos con esa misma propiedad. Para enlazar con otro individuo tendrá que utilizar una propiedad diferente o que no sea funcional.

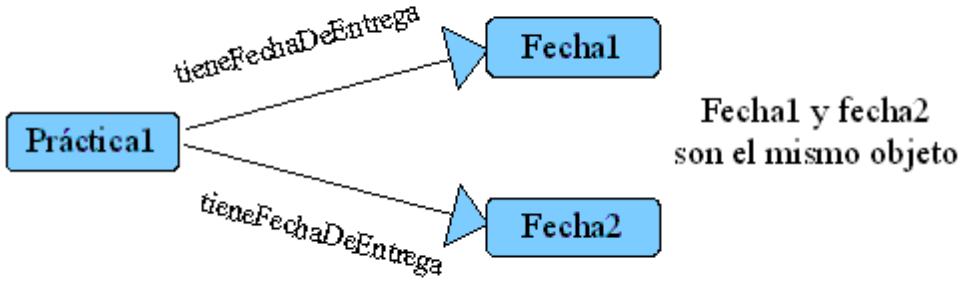


Figura 2.24 Representación de un ejemplo de una propiedad funcional.

Si la propiedad "tieneFechaDeEntrega" es funcional implica que los objetos Fecha1 y Fecha2 de la clase Fecha son el mismo puesto que no puede haber dos conexiones a objetos diferentes.

- Propiedades funcionales inversas: si una propiedad es funciona inversa quiere decir que la propiedad inversa de una propiedad es funcional. Posee las mismas características que la propiedad funcional pero para el caso de las propiedades inversas.

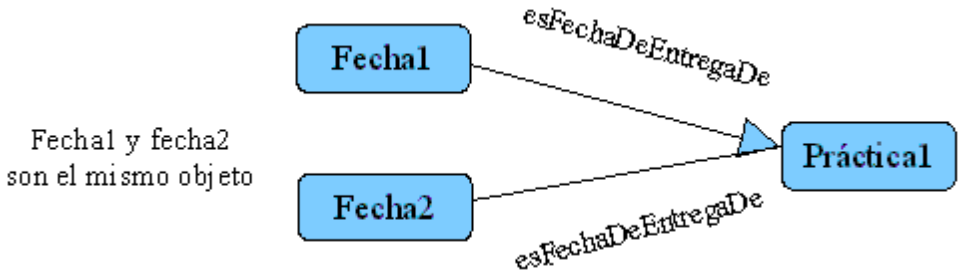


Figura 2.25 Representación de un ejemplo de una propiedad funcional inversa.

- Propiedades transitivas: si un individuo está relacionado con otro por media de una propiedad transitiva y éste último a su vez con un tercero con la misma propiedad, el primer y el tercer individuo están relacionados con la misma propiedad.

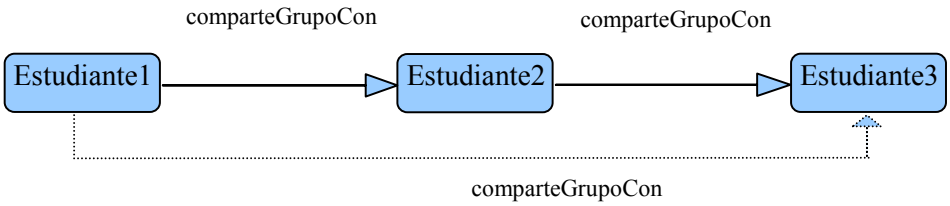


Figura 2.26 Representación de un ejemplo de una propiedad transitiva.

- Propiedades simétricas: si dos individuos están enlazados con una propiedad simétrica, ésta se cumple en los dos sentidos del enlace.

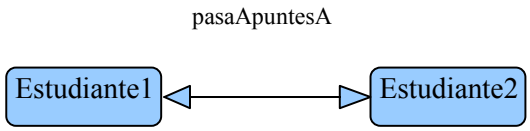


Figura 2.27 Representación de un ejemplo de una propiedad simétrica.

Con esta propiedad estamos declarando que Estudiante1 le pasa los apuntes a Estudiante2 y viceversa.

A la hora de crear una propiedad, independientemente del tipo que sea, es obligatorio especificar el dominio y en rango. Para definirlos, únicamente hay qué seleccionar qué clases pertenecen al dominio y al rango. El rango variará relativamente en función del tipo de propiedad mientras que el dominio será similar para ambos casos..

- Para el caso de propiedades del objeto: definen y limitan todas las posibilidades de los enlaces entre los individuos a través de las propiedades, ya que una vez definidos, una propiedad enlazará individuos del dominio con individuos del rango.

- Para el caso de propiedades del tipo de datos: definen y limitan las posibilidades de los enlaces, vía las propiedades, entre los individuos y los tipos de datos. El rango en este caso no será un individuo de una clase definida sino que será un tipo de dato (p.e.: Int, Float, Boolean, Date, etc). Junto con

esta pequeña variación, existe la posibilidad de definir los valores permitidos para los tipos de datos. Estos valores son fijos y establecidos por el usuario. (P.e., para el caso de Int, es posible definir como valores fijos: 1,3 y 8).

Para describir y definir una ontología son necesarias las propiedades descritas anteriormente debido a que son usadas para crear restricciones.

2.6 JSWING

El paquete Swing es un paquete gráfico que apareció en la versión 1.2 de Java como parte de la JFC, Java Foundation Classes. Está formado por un extenso conjunto de componentes de interfaces de usuario abarcando entre ellas botones, tablas, marcos, etc...

Swing existe desde la JDK 1.1 aunque antes de la existencia del paquete Swing, las interfaces gráficas se realizaban a través de AWT (Abstract Window Toolkit), de quien Swing hereda todo el manejo de eventos. Eso por ello que, normalmente, para toda componente AWT existe una componente Swing que la reemplaza, por ejemplo, la clase Button de AWT es reemplazada por la clase JButton de Swing.

Las componentes de Swing utilizan la infraestructura de AWT, incluyendo el modelo de eventos AWT, el cual se rige como una componente que reacciona a eventos tales como, eventos del ratón, teclado etc... Es por esto, que la mayoría de los programas Swing necesitan importar los paquetes de AWT *java.awt* y *java.awt.event*.

Las componentes propios de Swing se identifican porque pertenecen al paquete *javax.swing*.

La arquitectura Swing presenta una serie de ventajas respecto a su antecedente AWT:

- Amplia variedad de componentes.
- Aspecto modificable (*look and feel*): Se puede personalizar el aspecto de las interfaces o utilizar varios aspectos que existen por defecto (Metal Max, Basic Motif, Window Win32).
- Arquitectura Modelo-Vista-Controlador: Esta arquitectura da lugar a todo un enfoque de desarrollo muy arraigado en los entornos gráficos de usuario realizados con técnicas orientadas a objetos. Cada componente tiene asociado una clase de modelo de datos y una interfaz que utiliza. Se puede crear un modelo de datos personalizado para cada componente, con sólo heredar de la clase *Model*.
- Gestión mejorada de la entrada del usuario: Se pueden gestionar combinaciones de teclas en un objeto k*eyStrok* y registrarlo como componente. El evento se activará cuando se pulse dicha combinación si está siendo utilizado el componente, la ventana en que se encuentra o algún hijo del componente.
- Objetos de acción (*action objects*): Estos objetos cuando están activados (*enabled*) controlan las acciones de varios objetos componentes de la interfaz. Son hijos de *ActionListener.*
- Contenedores anidados: Cualquier componente puede estar anidado en otro. Por ejemplo, un gráfico se puede anidar en una lista.
- Escritorios virtuales: Se pueden crear escritorios virtuales o "interfaz de múltiples documentos" mediante las clases *JdesktopPane* y *JInternalFrame*.
- Bordes complejos: Los componentes pueden presentar nuevos tipos de bordes. Además el usuario puede crear tipos de bordes personalizados.
- Componentes para tablas y árboles de datos: Mediante las clases *Jtable* y *JTree*.

- Potentes manipuladores de texto: Además de campos y áreas de texto, se presentan campos de sintaxis oculta *JPassword*, y texto con múltiples fuentes *JTextPane*. Además hay paquetes para utilizar ficheros en formato HTML o RTF.

- Capacidad para "deshacer": En gran variedad de situaciones se pueden deshacer las modificaciones que se realizaron.

Swing incorpora su nuevo conjunto de eventos para sus componentes. Lo más importantes a destacar son los siguientes:

- *AncestorEvent*: Antecesor añadido desplazado o eliminado.

- *CaretEvent*: El signo de intercalación del texto ha cambiado.

- *ChangeEvent*: Un componente ha sufrido un cambio de estado.

- *DocumentEvent*: Un documento ha sufrido un cambio de estado.

- *HyperlinkEvent*: Algo relacionado con un vínculo hipermedia ha cambiado.

- *ListDataEvent*: El contenido de una lista ha cambiado o se ha añadido o eliminado un intervalo.

- *ListSelectionEvent*: La selección de una lista ha cambiado.

- *MenuEvent*: Un elemento de menú ha sido seleccionado o mostrado o bien no seleccionado o cancelado.

- *PopupMenuEvent*: Algo ha cambiado en un *JPopupMenu*.

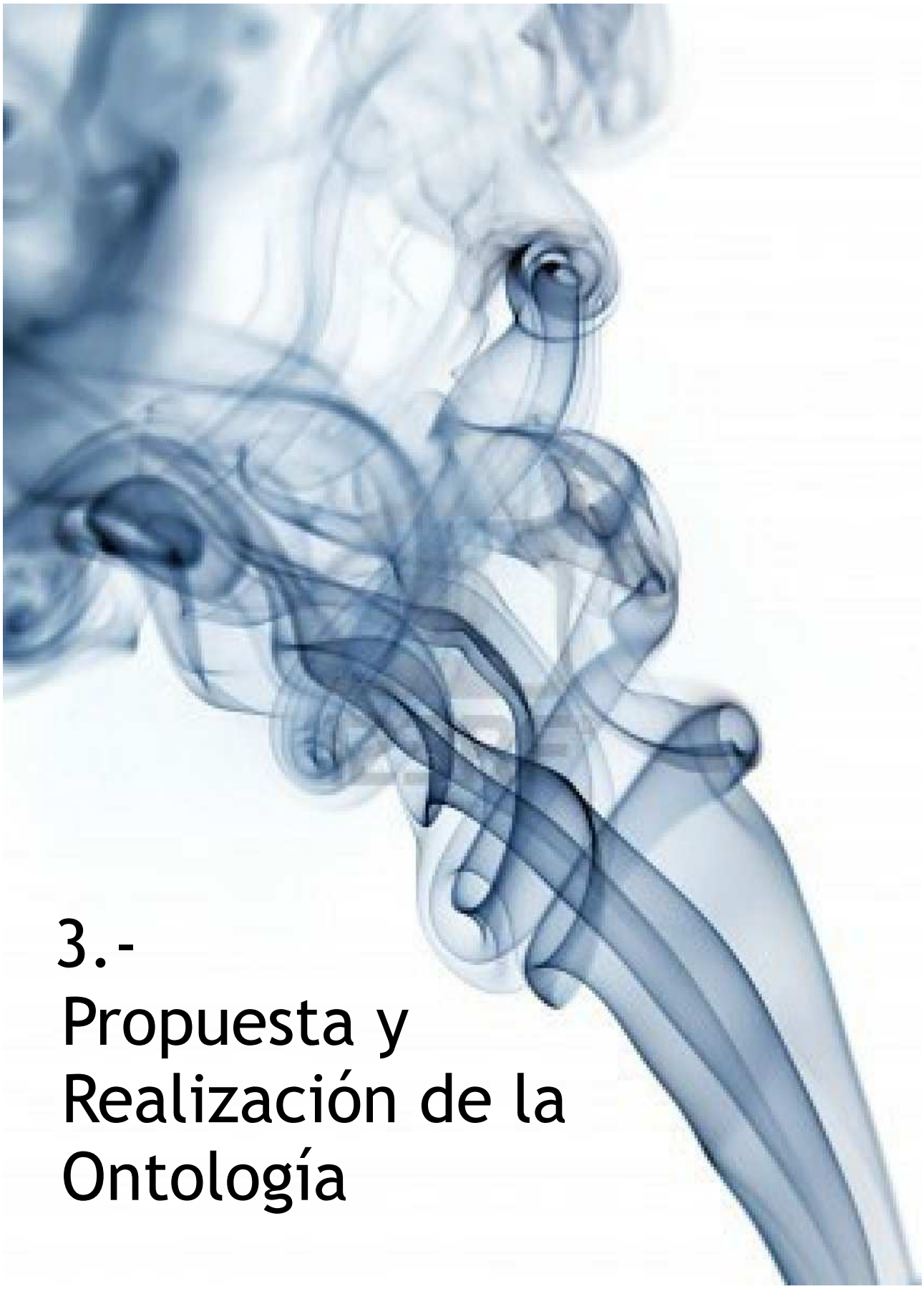
- *TableColumnModelEvent*: El modelo para una columna de tabla ha cambiando.

- *TableModelEvent*: El modelo de una tabla ha cambiado.

- *TreeExpansionEvent*: El nodo de un árbol se ha extendido o se ha colapsado.

- *TreeModelEvent*: El modelo de un árbol ha cambiado.

- *TreeSelectionEvent*: La selección de un árbol ha cambiado de estado.



3.- Propuesta y Realización de la Ontología

3.1 REALIZACIÓN DE LA ONTOLOGÍA DEL SISTEMA DE ENTREGA DE PRÁCTICAS CON LA HERRAMIENTA PROTÉGÉ.

Para realizar la ontología del sistema de entrega de prácticas con la herramienta Protégé, se seleccionó el sublenguaje OWL-DL ya que era necesario el máximo de expresividad a la vez que se exigía completitud computacional y capacidad de decidir.

La ontología se compone de las siguientes clases:

- Comment: Representa cada uno de los comentarios (*Student Comment* o *Teacher Comment*), que pueden ser hechos por el profesor o por un alumno.
- Student Comment: comentario hecho por un *Student*.
- Teacher Comment: comentario hecho por un *Teacher*.
- Delivery: Objeto que representa una entrega hecha por un grupo.
- Evaluation: Objeto que representa la evaluación hecha por un profesor a cerca de un *Delivery* para un grupo en concreto.
- Practice: cada objeto de esta clase corresponde a cada una de las prácticas que conforman en laboratorio de una asignatura.
- Reminder: esta clase representa a cada uno de los recordatorios que tiene que realizar el profesor para un grupo específico y una práctica en concreto.
- Student: representa cada uno de los estudiantes.
- Teacher: representa a un profesor.

1.- Las propiedades que tienen tipos de datos básicos, realizadas para cada uno de las clases son las siguientes:

PRACTICE

Nombre de la propiedad	Tipo de dato	Explicación.
difficultyLevel	Integer	Nivel de dificultad de la práctica.
exercisesNumber	Integer	Número de ejercicios que la componen.
startDateOfDelivery	Date	Fecha inicial para poder ser

		entregada.
finalDateOfDelivery	Date	Fecha límite para ser entregada.
maximumLengthOfLeaves	Integer	Número máximo de hojas que puede tener la memoria de la práctica.
numberOfDeliveriesCompleted	Integer	Número de entregas realizadas.
subjectName	String	Nombre de la asignatura.
weightPracticeInTheOverallEvaluation	Float	Peso de la práctica dentro de la evaluación total del laboratorio.

Figura 3.1 Propiedades de la clase *Practice*

Una práctica (*Práctice*), como se representa en la figura 3.1, contiene diferentes propiedades que la definen por completo, siendo las merecedoras de una explicación más exhaustiva las siguientes:

- *'difficultyLevel':* En una práctica puede estar definido el nivel de ésta, que será utilizado para calcular, junto con el peso total de la práctica 'weightPracticeInTheOverallEvaluation' y el roll del estudiante '*studentFunction*', la suma de todas las entregas realizadas.
- *'numberOfDeliveriesCompleted'*: esta propiedad corresponde al número total de entregas 'Delivery' que corresponden a la práctica en concreto que se está definiendo. En otras palabras, contabiliza el número de grupos que han realizado la entrega de una práctica.

TEACHER

nameAndSurname	String	Nombre y apellidos
subjectName	String	Nombre de la asignatura que imparte.

Figura 3.2 Propiedades de la clase *Teacher*

STUDENT

nameAndSurname	String	Nombre y apellidos
practiceGroup	Integer	Grupo al que pertenece.
subjectName	String	Nombre de la asignatura.
studentFunction	String	Roll del estudiante.

studentFunctionWeight	Integer	Peso del roll del estudiante
numberOfDeliveriesCompleted	Integer	Número total de entregas realizadas por el estudiante.

Figura 3.3 Propiedades de la clase *Student*

Las características a explicar de la figura 3.3 son:

- *studentFunction:* Esta propiedad representa el papel que tiene el alumno a la hora de realizar la práctica. Esté roll puede ser por ejemplo: jefe de proyecto (sólo uno), analista, gestión de configuración, calidad, pruebas, etc. Esta propiedad será tomada en cuenta a la hora de calcular la nota individual del alumno en la asignatura (comentada en la figura 3.1).

COMMENT

writer	String	Nombre del creador del comentario.
practiceGroup	Interger	Número del grupo al que pertenece.
subjectName	String	Nombre de la asignatura.
practiceCommented	String	Práctica comentada.

Figura 3.4 Propiedades de la clase Comment

DELIVERY

datetOfDelivery	Date	Fecha de entrega.
practiceDelivered	String	Práctica entregada.
practiceGroup	Integer	Grupo que realiza la entrega.
subjectName	String	Nombre de la asignatura.
lengthOfLeavesDelivered	Integer	Número de hojas de la meroria.

Figura 3.5 Propiedades de la clase Delivery

EVALUATION

practiceDelivered	String	Práctica entregada.
practiceGroup	Integer	Grupo que realiza la entrega.
subjectName	String	Nombre de la asignatura.
qualificationObtained	Float	Cualificación total en la práctica.
difficultyLevel	Integer	Dificultad de la práctica.
exercisesNumber	Integer	Número total de ejercicios.
weightPracticeInTheOverallEvaluation	Float	Peso total de la asignatura en el laboratorio.

Figura 3.6 Propiedades de la clase Evaluation

Respecto a la figura 3.6, es necesario explicar las siguientes propiedades:

- *qualificationObtained:* se trata de la calificación obtenida por el grupo entero en la práctica entregada.
- *weightPracticeInTheOverallEvaluation*: peso que tiene una asignatura dentro del conjunto global de todas las prácticas, ya que no necesariamente todas las prácticas tienen que tener el mismo peso ya que tienen objetivos o alcances diferentes, nivel de implicación, etc.

REMINDER

practiceGroup	Integer	Grupo
subjectName	String	Nombre de la asignatura
finalDateOfDelivery	Date	Fecha final de entrega
subjectToRemind	String	Asignatura a recordar.

Figura 3.7 Propiedades de la clase Reminder

2.- Las propiedades que tienen como objeto otro objeto de una clase definida anteriormente, son las siguientes:

PRACTICE

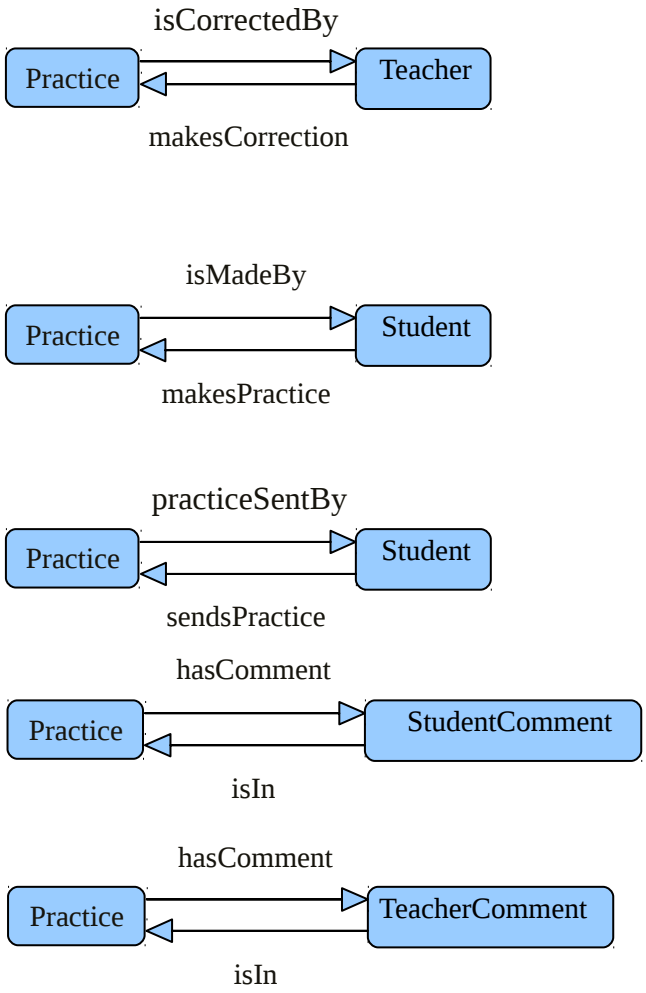


Figura 3.8 Conjunto de las propiedades de la clase *Practice*.

Como se aprecia en la figura 3.8, para una práctica se define lo siguiente:

- Quién la realiza: isMadeBy / makesPractice. Enlaza la práctica con el alumno.
- Quién la envía: practiceSentBy / sendsPractice: Enlaza la práctica con el alumno.
- Quién la corrige: isCorrectedBy / makesCorrection. Enlaza la práctica con el profesor.
- Qué comentarios del alumno o profesor tiene: hasComment / isIn. Enlaza la práctica con los comentarios, tanto del alumno como del profesor.

TEACHER

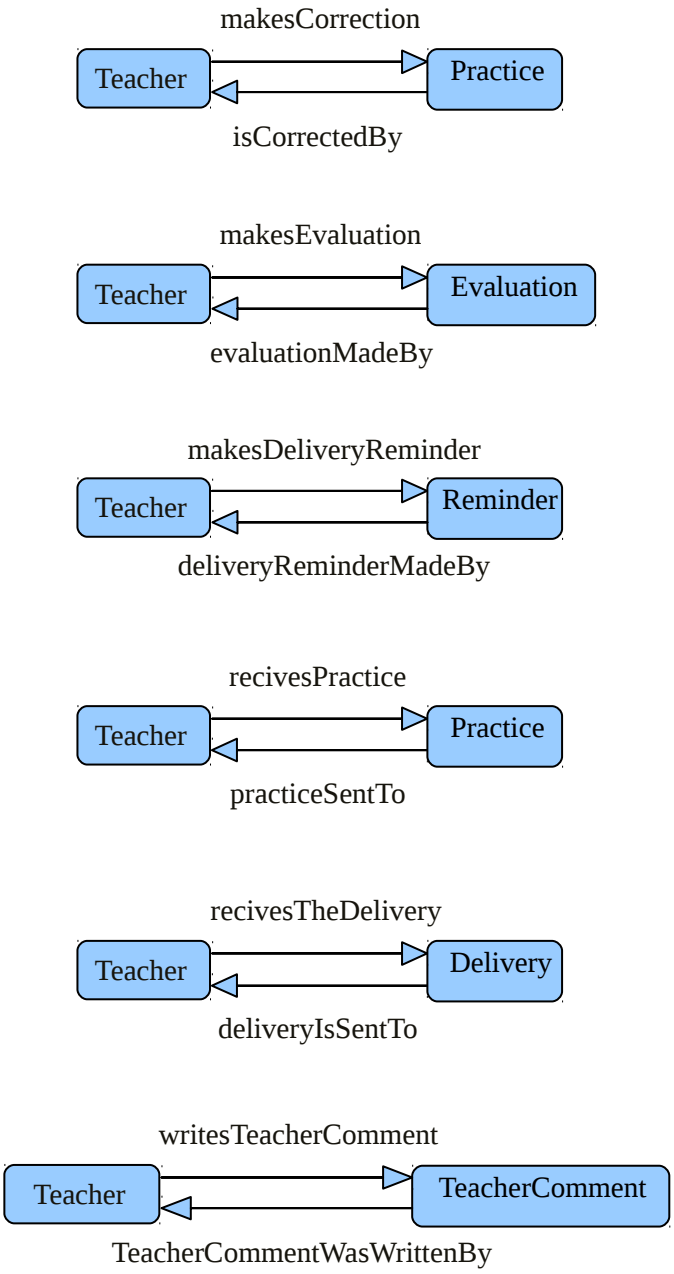


Figura 3.9 Conjunto de las propiedades de la clase Teacher

La figura 3.9 refleja las siguientes propiedades de la clase Teacher:

- Qué práctica corrige: makesCorrection / isCorrectedBy. Enlaza el profesor con la práctica.
- Qué evaluación realiza: makesEvaluation / evaluationMadeBy. Enlaza la evaluación con el profesor.
- Qué recordatorio hace: makesDeliveryReminder / deliveryReminderMadeBy. Enlaza la clase Reminder con la clase Teacher.
- Qué prácticas recibe el profesor: recivesPractice / practiceSentTo: Enlaza

al profesor con la Práctica.

- Qué entrega recibe: recivesTheDelivery / deliveryIsSentTo. Enlaza un objeto Delivery con el professor.
- Qué comentarios escribe: writesTeacherComment / TeacherCommentWasWrittenBy. Enlaza al professor con un comentario.

STUDENT

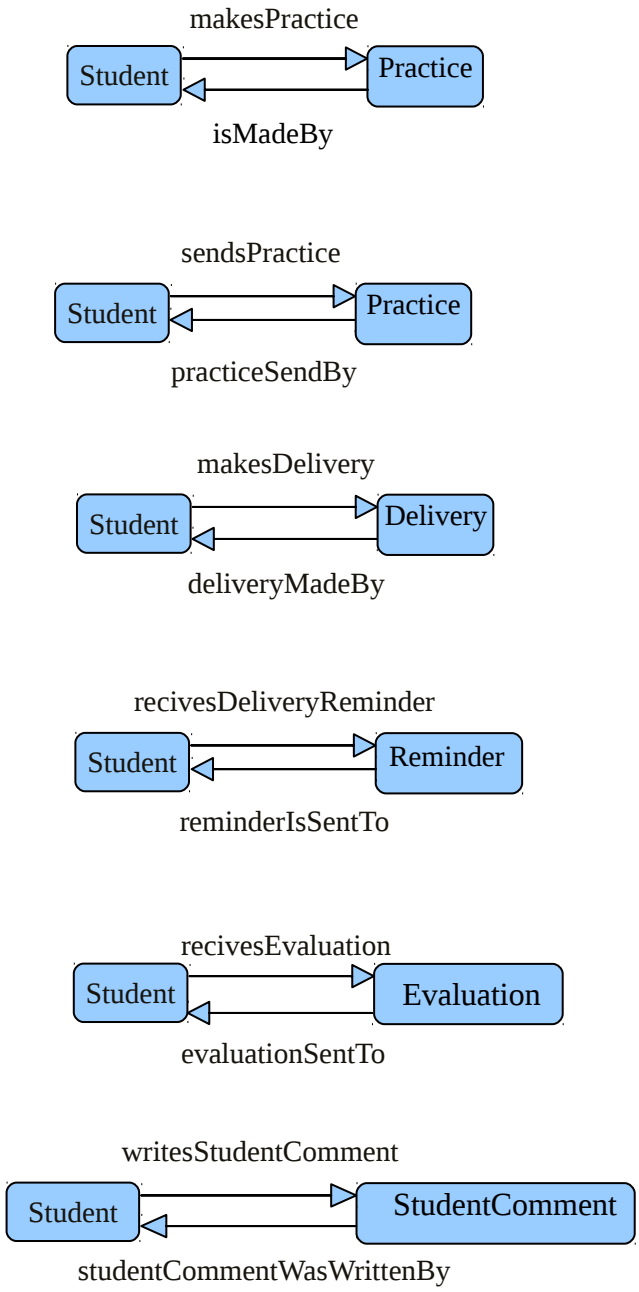


Figura 3.10 Conjunto de las propiedades de la clase Student.

En la figura 3.10 vemos todas las propiedades definidas para un estudiante, es decir, la clase Student:

- Qué práctica realiza: makesPractice / isMadeBy. Enlaza al alumno con una práctica.
- Qué práctica envía: sendsPractice / practiceSendBy. Enlaza a un alumno con una práctica.
- Qué entrega hace: makesDelivery / deliveryMadeBy. Enlaza a un alumno con una entrega (Delivery).
- Qué recordatorio recibe: recivesDeliveryReminder / reminderIsSentTo. Enlaza a un alumno con un recordatorio.
- Qué evaluación recibe: recivesEvaluation / evaluationSentTo. Enlaza al alumno con una evaluación.
- Qué comentario escribe: writesStudentComment / studentCommentWasWrittenBy. Enlaza a un alumno con un comentario.

DELIVERY

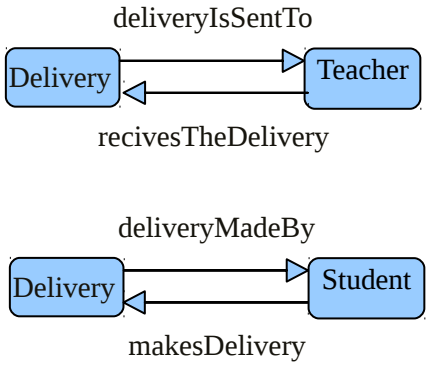


Figura 3.11 Conjunto de las propiedades de la clase Delivery.

La figura 3.11 representa las 2 propiedades de objeto de posee la clase Delivery:

- Quién la realiza: deliveryMadeBy / makesDelivery. Enlaza la entrega con un alumno.
- De quién la recibe: deliveryIsSentTo / recivesTheDelivery. Enlaza una entrega con un profesor.

EVALUATION

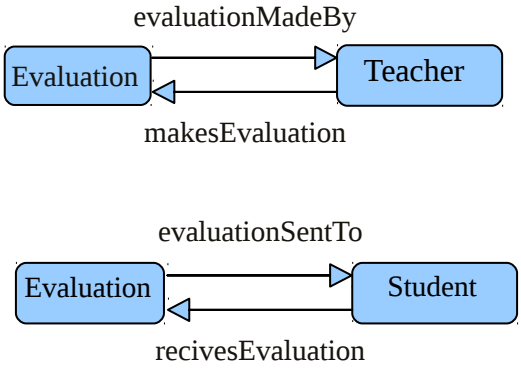


Figura 3.12 Conjunto de las propiedades de la clase Evaluation.

La clase Evaluation posee dos propiedades de objetos como se representa en la figura 3.12:

- Quién la realiza: evaluationMadeBy / makesEvaluation. Enlaza a una evaluación con un profesor.
- A quién es enviada: evaluationSentTo / recivesEvaluation. Enlaza la evaluación con un estudiante.

STUDENT COMMENT

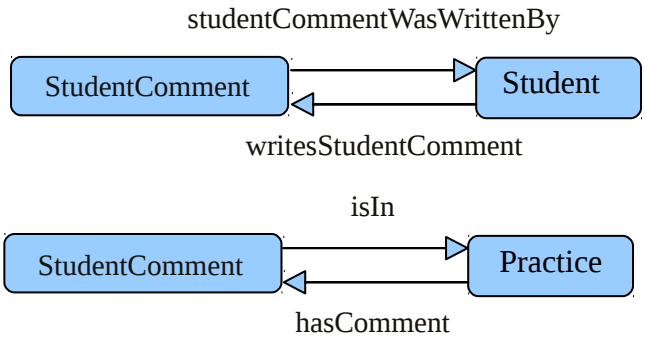


Figura 3.13 Conjunto de las propiedades de la clase StudentComment.

La figura 3.13 muestra las propiedades que posee un objeto de la clase StudentComment:

- Quién lo escribe: studentCommentWasWrittenBy / writesStudentComment. Enlaza un comentario con un estudiante.

- A qué práctica corresponde: isIn / hasComment. Enlaza un comentario con su práctica correspondiente.

TEACHER COMMENT

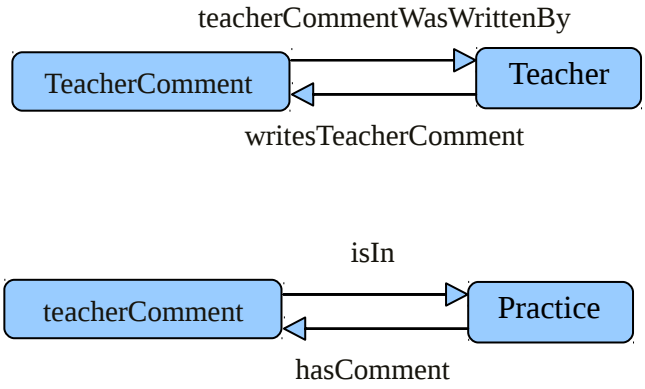


Figura 3.14 Conjunto de las propiedades de la clase TeacherComment.

En la figura 3.14 podemos observar las dos propiedades para la clase TeacherComment:

- Quién lo escribe: teacherCommentWasWrittenBy / writesTeacherComment. Enlaza un comentario con el professor que lo escribió.
- A qué práctica corresponde: isIn / hasComment. Enlaza el comentario con la práctica en la que se escribío.

REMINDER

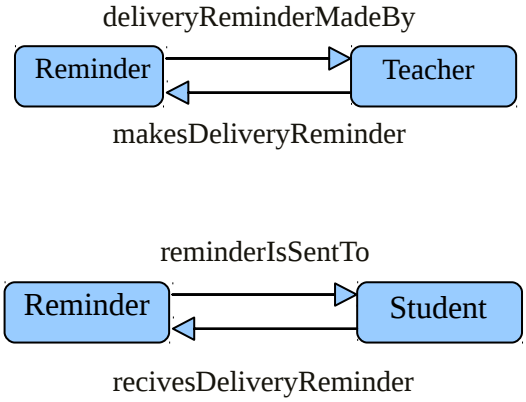


Figura 3.15 Conjunto de las propiedades de la clase Reminder.

Por último, la figura 3.15 muestra las dos propiedades que posee la clase Reminder:

- Quién lo hace: deliveryReminderMadeBy / makesDeliveryReminder. Enlaza a un recordatorio con el profesor que lo realiza.
- A quién va dirigido: reminderIsSentTo / recivesDeliveryReminder. Enlaza a un recordatorio con un estudiante.

A la hora de realizar la ontología para este proyecto, se decidió que el número total de cada una de las clases era el siguiente:

1. Practice: 4
2. Delivery: 7
3. Evaluation: 8
4. Teacher: 1
5. Student: 6
6. Reminder: 2
7. StudenComment: 3
8. TeacherComment: 1

Esto es debido a que anteriormente a realizar los Servicios Webs para los datos de el sistema de entregas real, se realizaron unos Servicios Web con los datos de estos objetos, sirvieron como base para realizar posteriormente los reales.

3.2 CREACIÓN DE LAS TABLAS MYSQL Y SU CARGA.

La base de datos fue creada para introducir toda la información que existe del sistema de entregas real de prácticas que será requerida para realizar los Servicios Web propios de la aplicación.

La versión de MySQL utilizada para el proyecto es la 5.1

Para almacenar la información se utilizó sdb, junto con su archivo sdb.ttl

El archivo sdb.ttl utilizado en el proyecto es el siguiente:

```
@prefix rdfs:     <http://www.w3.org/2000/01/rdf-schema#> .
@prefix rdf:      <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix ja:       <http://jena.hpl.hp.com/2005/11/Assembler#> .
@prefix sdb:      <http://jena.hpl.hp.com/2007/sdb#> .

# Store only - no connection details

<#mySstore> rdf:type sdb:Store ;
   sdb:layout        "layout2" ;
   sdb:connection     <#conn> ;
.
<#conn> rdf:type sdb:SDBConnection ;
   sdb:sdbType       "mysql" ;
   sdb:sdbName        "copyrealproyect" ;
   sdb:sdbHost        "localhost" ;
   sdb:sdbUser        "root" ;
   sdb:sdbPassword    "passroot" ;
   sdb:driver          "com.mysql.jdbc.Driver" ;
 .
```

La tabla creada consta de 4 tablas que son las siguientes:

```
mysql> use copyrealproyect;
Database changed
mysql> show tables;
+--------------------------+
| Tables_in_copyrealproyect |
+--------------------------+
| nodes                    |
| prefixes                 |
| quads                    |
| triples                  |
+--------------------------+
4 rows in set (0.11 sec)
```

Figura 3.16 Tablas creadas en la base de datos para el almacenamiento de los datos del Sistema real de entregas.

En la figura 3.16 se observan las 4 tablas que han sido creadas para el almacenamiento de los datos del sistema real de entregas. A continuación, se explicarán cada una de ellas por separado:

- La tabla “nodes”: corresponde a los grafos de RDF.

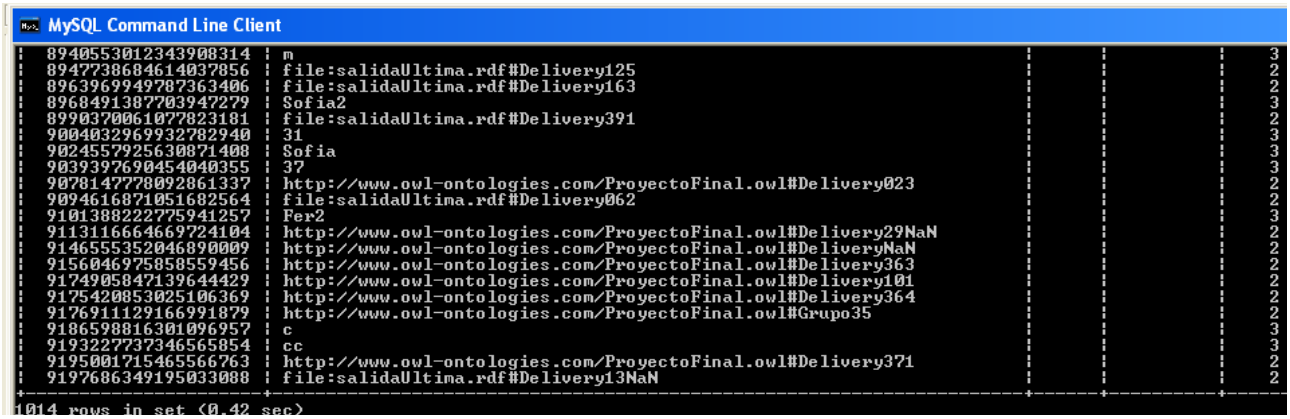


Figura 3.17 Tabla *nodes*

La tabla *nodes*, mostrada en la figura 3.17, almacena la representación de los términos de RDF, proporcionando dos mapeos diferentes, desde el nodo (segunda columna) al id del nodo (primera columna con calidad de 8 bytes) y viceversa. Ambos sentidos de mapeos son utilizados para diferentes funciones. Mientras que el primer caso se utilizada durante la carga y cuando se convierten los términos en consultas, el segundo caso se utiliza para convertir los resultados obtenidos con ids de nodos, en nodos de representación de Jena. A la tabla de *nodes* se le conoce como diccionario, ya que con el id somos capaces de conocer el nodo, que es lo que aporta significado visual para nosotros.

- La tabla “prefixes” contiene la información de todos los prefijos utilizados en el archivo rdf introducido en la base de datos. Este archivo contiene toda la información del sistema real de entrega de prácticas.

```
mysql> select * from prefixes;
+----------+-------------------------------------------------+
| prefix   | uri                                             |
+----------+-------------------------------------------------+
| :        | http://www.owl-ontologies.com/ProyectoFinal.owl# |
| owl:     | http://www.w3.org/2002/07/owl#                  |
| protege: | http://protege.stanford.edu/plugins/owl/protege# |
| rdf:     | http://www.w3.org/1999/02/22-rdf-syntax-ns#     |
| rdfs:    | http://www.w3.org/2000/01/rdf-schema#           |
| swrl:    | http://www.w3.org/2003/11/swrl#                 |
| swrlb:   | http://www.w3.org/2003/11/swrlb#                |
| xsd:     | http://www.w3.org/2001/XMLSchema#               |
| xsp:     | http://www.owl-ontologies.com/2005/08/07/xsp.owl# |
+----------+-------------------------------------------------+
9 rows in set (0.33 sec)
```

Figura 3.18 Tabla *prefixes*

Los prefijos mostrador en la figura 3.18 no forman parte del procesado de las consultas pero sí dan soporte para la presentación de los tripletes SDB en nuestro caso. Estos tripletes fueron explicados en el apartado 2.1.3.4.2 SDB del estado del arte.

- La tabla “quads” se crea para los grafos de RDF. Para nuestro proyecto no es necesaria así que permanece vacía.

```
mysql> select * from quads;
Empty set (0.05 sec)
```

Figura 3.19 Tabla *quads*

- La tabla “triples” es para las tripletas:

```
|  9195001715465566763 |  -8206516612553319 63 | -5479402559969115187 |
|  9195001715465566763 |  -8206516612553319 63 |   -64279151752153471 |
|  9195001715465566763 |  1225227694830770300 |  9039397690454040355 |
|  9195001715465566763 |  6006218113948065870 | -8161068526649467704 |
|  9195001715465566763 |  6006218113948065870 |  2684584859539031741 |
|  9197686349195033088 | -6430697865200335348 | -5941428619312581107 |
|  9197686349195033088 | -4986611568994189092 |  2193958602356199047 |
|  9197686349195033088 |  -8206516612553319 63 | -5479402559969115187 |
|  9197686349195033088 |  -8206516612553319 63 | -1610264463114189344 |
|  9197686349195033088 |  1225227694830770300 |  649730661157942 6769 |
|  9197686349195033088 |  6006218113948065870 | -8161068526649467704 |
+----------------------+----------------------+----------------------+
5394 rows in set (0.20 sec)
```

Figura 3.20 Tabla *nodes*

La figura 3.20 muestra un ejemplo de la tabla *nodes*, que está formada por 3 índices, dónde cada uno de los índices tiene la información sobre un triplete.

Una vez creadas las tablas donde se almacenarían los datos posteriormente, se llevó a cabo toda su carga. Para realizar dicho proceso, se utilizó la tecnología SDB. En primer lugar, como primer paso, fue necesario poseer el documento RDF con la información del sistema de entregas real. Sin embargo, dicho documento no existía aunque sí un documento XML proporcionado por el tutor del proyecto. Por lo tanto, previa carga en la base de datos, fue necesaria una transformación de un documento XML a un RDF con la tecnología SAXON, explicada en el apartado X.X de la memoria.

Una parte seleccionada aleatoriamente del documento XML para mostrar la estructura de éste es la siguiente:

```
<?xml version="1.0"?>
<log>
	<logentry revision="1849">
		<author>b</author>
		<date>2009-09-09T18:59:55.163283Z</date>
		<paths>
			<path action="M">/Group_36/Practica02/dictionary.c</path>
			<path action="M">/Group_36/Practica02/dictionary.h</path>
		</paths>
		<msg>Versión para septiembre</msg>
	</logentry>

	<logentry revision="1848">
		<author>b</author>
		<date>2009-09-09T18:59:07.232383Z</date>
		<paths>
			<path action="M">/Group_36/Practica01/sumList.c</path>
		</paths>
		<msg>Versión para septiembre</msg>
	</logentry>

	....
</log>
```

El documento está dividido en diferentes revisiones, objetos “revision”, que corresponden a diferentes entradas con información a cerca de las entregas, comentarios, etc de un grupo de prácticas en concreto y sobre una práctica determinada. En términos generales lo que se va a hacer es recorrer todo el documento XML y juntar en el RDF de salida toda la información conjunta para el mismo grupo y práctica, ya que su información puede estar en

entradas “revision” distintas. Los objetos para un grupo y práctica va a estar definido como “Delivery”+número de grupo + número de práctica.

Un ejemplo del resultado esperado es el siguiente:

```
<Delivery rdf:ID="Delivery275">
        <practiceGroup>27</practiceGroup>
        <practiceDelivered>Practice5</practiceDelivered>
        <dateOfDelivery>2009-09-09</dateOfDelivery>
        <dateOfDelivery>2009-08-24</dateOfDelivery>
        <dateOfDelivery>2008-12-18</dateOfDelivery>
        <dateOfDelivery>2008-12-15</dateOfDelivery>
        <dateOfDelivery>2008-12-03</dateOfDelivery>
        <deliveryMadeBy>a</deliveryMadeBy>
        <deliveryMadeBy>abel</deliveryMadeBy>
</Delivery>
```

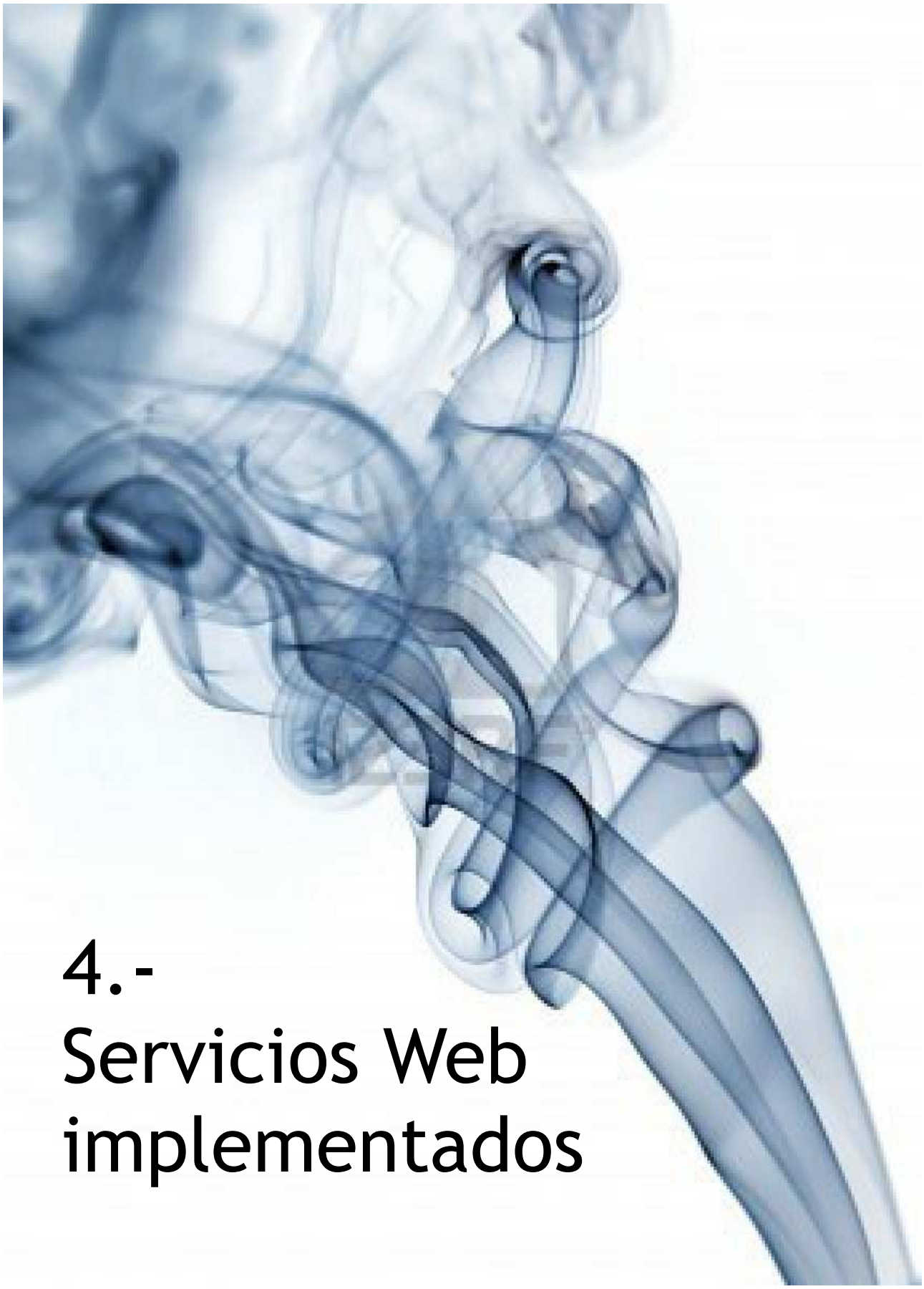
Por lo tanto, el proceso para conseguir la salida esperada con el documento XML de entreda es el siguiente:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE rdf:RDF [
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <!ENTITY protege "http://protege.stanford.edu/plugins/owl/protege#">
    <!ENTITY xsp "http://www.owl-ontologies.com/2005/08/07/xsp.owl#">
    <!ENTITY owl "http://www.w3.org/2002/07/owl#">
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">
    <!ENTITY swrl "http://www.w3.org/2003/11/swrl#">
    <!ENTITY swrlb "http://www.w3.org/2003/11/swrlb#">
    <!ENTITY base "http://www.owl-ontologies.com/ProyectoFinal.owl#">
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#">  ]>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
        xmlns:xs="http://www.w3.org/2001/XMLSchema" exclude-result-prefixes="xs"          version="2.0"
        xmlns:rdf="&rdf;">
        <xsl:strip-space elements="*"/>
        <xsl:output indent="yes"/>
        <xsl:template match="@*|node()">
                <xsl:copy>
                        <xsl:apply-templates select="@*,node()"/>
                </xsl:copy>
        </xsl:template>
        <xsl:template match="log">
           <rdf:RDF xmlns:rdf='&rdf;' xmlns:protege="&protege;" xmlns:xsp="&xsp;"
            xmlns:owl="&owl;" xmlns:xsd="&xsd;" xmlns:swrl="&swrl;" xmlns:swrlb="&swrlb;"
            xmlns:pf="&pf;"         xmlns:rdfs="&rdfs;">
         <xsl:variable name="log" as="element(logentry)*">
                xsl:apply-templates select="logentry"/>
        </xsl:variable>
        <xsl:copy>
            <xsl:for-each-group select="$log" group-by="paths/path[@action = 'M']/group">
                <xsl:variable name="group" as="xs:string" select="current-grouping-key()"/>
                <xsl:for-each-group select="current-group()" group-by="paths/path[@action='M']/practice">
                        <xsl:variable name="numberGroup" as="xs:string" select="$group"/>
                        <xsl:variable name="numberPractice" as="xs:string" select="current-grouping-key()"/>
                        <Delivery>
                                <xsl:attribute name="rdf:ID">
                                        <xsl:value-of select="concat('Delivery',$numberGroup,$numberPractice)"/>
                                </xsl:attribute>
                                <practiceGroup>
```

```
                                        <xsl:value-of select="$group"/>
                                    </practiceGroup>
                                    <practiceDelivered>
                                        <xsl:value-of select="concat('Practice',$numberPractice)"/>
                                    </practiceDelivered>
                                    <xsl:for-each-group select="current-group()/date" group-by="fecha">
                                        <dateOfDelivery>
                                            <xsl:value-of select="current-grouping-key()"/>
                                        </dateOfDelivery>
                                    </xsl:for-each-group>
                                    <xsl:for-each-group select="current-group()/author" group-by="autor">
                                        <deliveryMadeBy>
                                            <xsl:value-of select="current-grouping-key()"/>
                                        </deliveryMadeBy>
                                    </xsl:for-each-group>
                                </Delivery>
                        </xsl:for-each-group>
                </xsl:for-each-group>
            </xsl:copy>
        </rdf:RDF>
    </xsl:template>
        <xsl:template match="path[@action = 'M']">
                <xsl:copy>
                        <xsl:copy-of select="@*"/>
                        <xsl:variable name="tokens" as="xs:string*" select="tokenize(., '/')"/>
                        <group>
                                <xsl:value-of select="substring-after($tokens[2], '_')"/>
                        </group>
                        <practice>
                                <xsl:value-of select="number(replace($tokens[3], '[a-zA-Z]+([0-9]+)', '$1'))"/>
                        </practice>
                        <issue>
                                <xsl:value-of select="$tokens[4]"/>
                        </issue>
                </xsl:copy>
        </xsl:template>

        <xsl:template match="date">
                <xsl:copy>
                        <xsl:copy-of select="@*"/>
                        <xsl:variable name="tokens" as="xs:string*" select="tokenize(., 'T')"/>
                        <fecha>
                                <xsl:value-of select="$tokens[1]"/>
                        </fecha>
                        <hora>
                                <xsl:value-of select="$tokens[2]"/>
                        </hora>
                </xsl:copy>
        </xsl:template>

        <xsl:template match="author">
                <xsl:copy>
                        <xsl:copy-of select="author"/>
                        <xsl:variable name="tokens" as="xs:string*" select="tokenize(., 'T')"/>
                        <autor>
                                <xsl:value-of select="$tokens[1]"/>
                        </autor>
                </xsl:copy>
        </xsl:template>
        </xsl:stylesheet>
```



4.- Servicios Web implementados

Introducción.

El funcionamiento de cada uno de los Servicios Web de los que se compone mi PFC se diseñó de tal modo que en ellos juega un papel muy importante la fecha en la que nos encontramos cuando hacemos su uso, ya que los Servicios Web se planificaron para aplicaciones de un sistema de entregas real en tiempo real. Es decir, van a tener un funcionamiento u otro en relación a la fecha en la que nos encontremos.

Basándome en lo anterior, voy a definir los 3 casos temporales en los que nos podemos encontrar. Primeramente comentar que, partiendo de la definición de la clase Practica definida en el apartado anterior sobre la Ontología, cada objeto Practice tiene su fecha de inicio y final de entrega, es decir, no podrá ser entregada por parte del alumno en cualquier momento sino que tendrá que ser en una fecha comprendida en el intervalo de entrega. Éste, por tanto, será independiente para cada una de las prácticas y un mismo Servicio Web podrá tener comportamiento distinto en función del periodo en el que nos encontremos.

1.- Periodo de entrega no comenzado: El día actual es anterior al día de inicio del periodo de entrega.

2.- Dentro del periodo de entrega. El día actual es posterior al inicio de entrega pero anterior al límite de entrega.

3.- Periodo de entrega finalizado: El día de fin de entrega de la práctica es anterior a la fecha actual.

1.3.1.- Servicio Web “CalcularNotasAlumno”.

A la hora de elegir qué Servicios Web implementar, escogí éste ya que se puede llevar a cabo el control de cada uno de los alumnos por separado y ver la evolución de éstos. Por lo tanto, un parámetro obligatorio será el nombre o identificativo del alumno sobre el que se quiere hacer la petición.

Funcionalidad:

Este Servicio Web tiene como objetivo principal, calcular la nota global de un alumno teniéndose en cuenta únicamente las entregas realizadas con una fecha anterior a la fecha en la que se produce el cálculo de la nota, es decir, todas las entregas realizadas “a día de hoy” en el caso de que tengamos el sistema de entregas configurado con la fecha y hora actual. Se realizará el cálculo para el periodo 2.

- Además, calcula otro tipo de datos relacionados con la puntuación del alumno. Serán explicados a continuación.

- Importante: en la base de datos están almacenadas todas las entregas y para el cálculo se tendrán en cuenta las que han sido entregadas según la fecha ya que se sigue rigurosamente el orden cronológico de las entregas.

Métodos del Servicio Web:

Nota: Todos los métodos realizan una determinada función para el alumno pasado como parámetro del método.

- **calcularNA(String alumno):** proporciona la nota global del laboratorio de la asignatura donde dicha nota es el sumatorio de las notas individuales correspondientes a las prácticas entregadas en la fecha actual.

- **calcularNASegunEntregas(String alumnoNA):** proporciona la nota obtenida por el alumno sólo teniendo en cuenta las prácticas entregadas, no la nota global del laboratorio.

    - Ejemplo: Si el alumno sólo ha entregado 2 prácticas, la nota será en función de esas dos, como si el total de las prácticas fuese sólo esas.

- **numeroEntregasA(String alumnoN):** devuelve el numero de entregas realizadas por el alumno en la fecha actual de búsqueda.

- **dameSumatorioPesosPracticasHechasA(String alumnoN):** cada una de las prácticas tiene un peso distinto en el cálculo de la nota final del laboratorio y este método devuelve el sumatorio de los pesos de las prácticas realizadas por el alumnoN. En principio este método fue creado para ser utilizado por otros métodos del ServicioWeb.

- **practicasHechasPorALumnos(String alumnoN):** este método devuelve un Vector con las prácticas realizadas por el alumnoN.

Esquema de la implementación del Servicio Web:

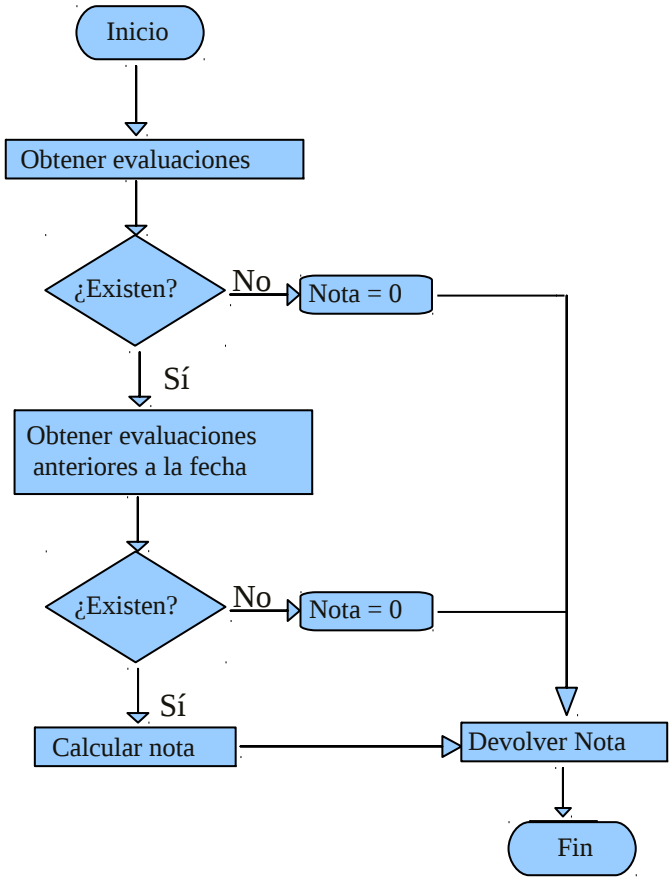


Figura 4.1 Diagrama de flujo del cálculo de la nota global perteneciente al Servicio Web “CalcularNotas”

Tal y como se muestra en la figura 4.1, se realiza la búsqueda de las evaluaciones que han sido mandadas a un alumno específico. Si no se encuentran, la nota del alumno será 0 puesto que no se le habrá corregido ninguna práctica. En caso afirmativo, entre todas ellas se seleccionarán las que sean anteriores a la fecha actual despreciando las posteriores puesto que están almacenadas en la base de datos pero para el cálculo son similares a si no se hubiesen entregado. Por lo tanto, se calculará la nota únicamente con las últimas seleccionadas.

Para otro cálculo alternativo, por ejemplo la “nota según solo las entregas” el proceso es similar pero se calculará y devolverá el dato correspondiente al método seleccionado.

El proceso para obtener las evaluaciones de un alumno anterior a la fecha actual es el siguiente:

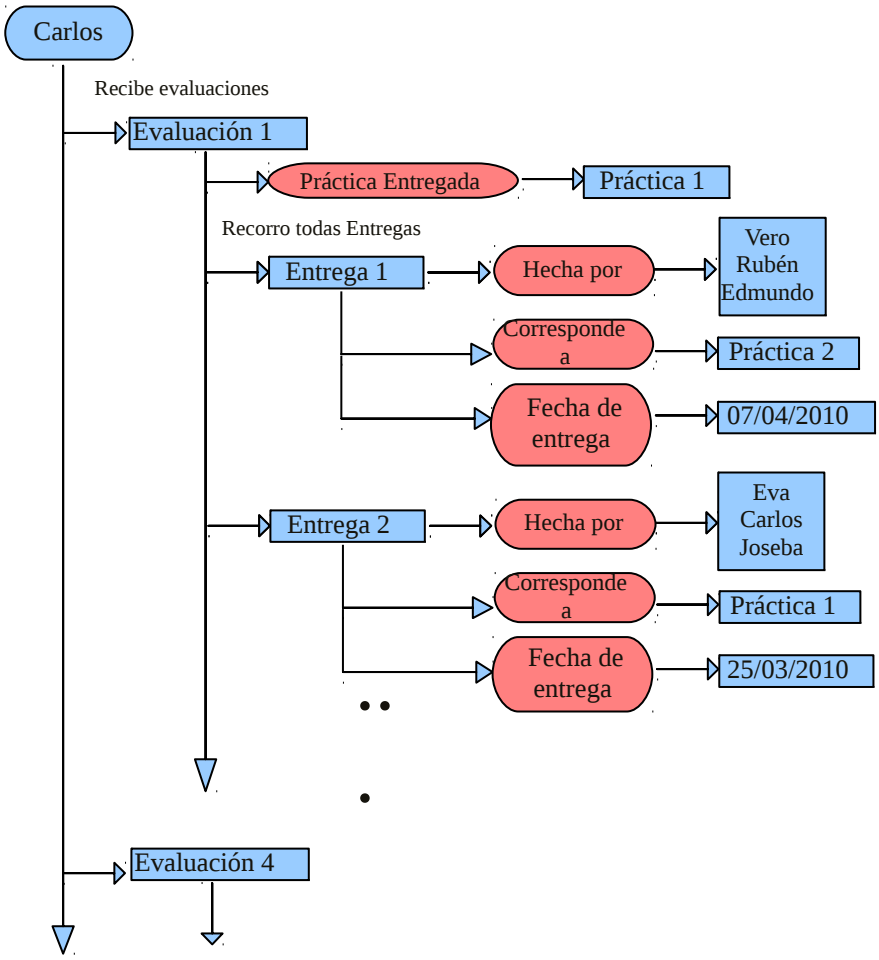


Figura 4.2 Implementación del proceso “Obtener Evaluaciones anteriores a una fecha dada”, perteneciente al diagrama de flujo de la figura X.X mostrada anteriormente.

Como se ve representado en la figura 4.2, en primer lugar, por un lado, se almacenan las evaluaciones que recibe el alumno para obtener las prácticas ha realizado. Por otro lado, buscamos en la base de datos todas las “Entregas” y almacenamos quién ha realizado la entrega, a qué práctica corresponde y la fecha de la entrega. Posteriormente, una vez que tenemos esa información, vamos a comparar los datos para poder realizar los cálculos, quedándonos con todas entregas de la base de datos que corresponden al alumno específico y a la práctica que ha sido corregido en la evaluación , y que sean anteriores a la fecha actual. Si la entrega cumple esas condiciones, tendremos en cuenta la Evaluación correspondiente para el cálculo de la nota. Esto es necesario ya que cada objeto de la ontología de Protégé almacena distinta información y es necesario requerir a distintos objetos para el cálculo de un dato.

1.3.2 Servicio Web “PrácticaSuplementoria”.

Este Servicio Web es una consecuencia personal de haber estudiado una ingeniería. No siempre es posible realizar todas las prácticas correctamente por falta de tiempo o por la dificultad de éstas, así que es posible que no todas ellas hayan sido realizadas correctamente para que el laboratorio esté aprobado.

Funcionalidad

Este Servicio Web tiene el objetivo de mandar una recomendación para realizar una práctica suplementoria. Como todos los Servicios Web realizados en el proyecto final de carrera, se tiene en cuenta el tiempo y las fechas de entregas para calcular los datos necesarios.

Se mandará la recomendación para los casos en los que al alumno le falte únicamente una práctica por ser entregada y en ese momento la nota global de la asignatura sea inferior a 5 y aquellos en los que todas las prácticas hayan sido entregadas y la nota global de la asignatura sea también inferior a 5.

Como caso contrario, no se mandará cuando al alumno le falte por entregar más de una práctica, ya que con la realización de la práctica suplementoria no se alcanzaría una nota global superior o igual a 5 en la asignatura. También cuando el alumno haya entregado todas prácticas a excepción de una pero la nota global de la asignatura sea superior a 5. Por último, cuando el alumno haya entregado todas las prácticas y la nota global sea superior a 5.

Métodos del Servicio Web:

Nota: Todos los métodos realizan una determinada función para el alumno pasado como parámetro del método.

- **mandarRecomendacionPractica(String alumnoARecomendar):** determina si a un alumno le tiene que ser mandada una recomendación para realizar una práctica adicional.
- **dameNumeroPracticasSinHacer(String alumnoARecomendar):** determina el número de prácticas que no han sido realizadas por el alumno.
- **dameVectorPracticasSinHacer(String alumnoARecomendar):** devuelve todas las prácticas que no han sido realizadas por el alumno.
- **dameNumeroPracticasHechas(String alumnoARecomendar):** determina el número de prácticas que han sido realizadas el alumno.
- **dameVectorPracticasHechas(String alumnoARecomendar):** devuelve todas las prácticas que han sido realizadas por el alumno.
- **dameNotaDelAlumno(String alumnoARecomendar):** devuelve la nota actual que posee el alumno. Se realizó porque era necesario para realizar los cálculos en los métodos anteriores pero pertenece al Servicio Web “CalcularNotasAlumno”.

- **dameNotaQueFaltaEnLaEvaluacionParaAprobar(String alumnoARecomendar):** calcula la nota que es necesaria para llegar a un mínimo de 5 en la evaluación global del laboratorio.

Esquema de la implementación del Servicio Web:

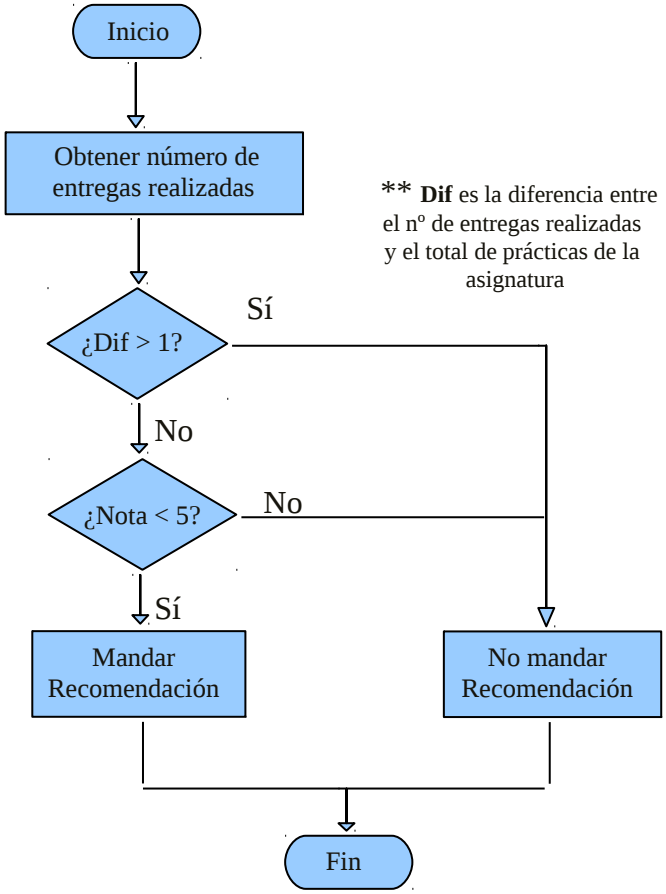


Figura 4.3 Diagrama de flujo del método mandarRecomendaciónPráctica perteneciente al Servicio Web "PrácticaSuplementoria".

Tal como se muestra en la figura 4.3, en primer lugar al Servicio Web se le es pasada un alumno en concreto para comprobar si es necesario o no mandarle una recomendación. A continuación, se obtiene el número de entregas que ha realizado y la nota correspondiente del alumno a día de hoy. Si se da el caso en el que el alumno ha entregado todas las prácticas o todas a excepción de uno y su nota es inferior a 5, se le mandará una recomendación para realizar una práctica suplementoria con el fin de que pueda obtener una calificación de aprobado en el laboratorio de la asignatura.

1.3.3 Servicio Web “RecordatorioGrupoSegunPractica”.

Funcionalidad.-

- El objetivo de este Servicio Web es mandar un recordatorio a aquellos grupos que no hayan realizado la práctica a 10 días del final del plazo de entrega siempre que haya comenzado el periodo de entrega de dicha práctica.

Métodos del Servicio Web.-

- **comprobarRecordatorioGrupo(String practice):** determina si el recordatorio de una práctica tiene que ser enviado y a qué grupos.

- **dameVectorGruposMandarRecordatorio(String practiceCompr):** crea un grupo formado por los grupos que cumplen los requisitos para serles enviado un recordatorio.

- **dameVectorGruposNoMandarRecordatorio(String practiceCompr):** crea un grupo formado por los grupos que no cumplen los requisitos para serles enviado un recordatorio.

- **dameDiasDesdeInicioEntrega(String practiceCompr):** calcula los días que han transcurrido desde el inicio del periodo de entrega de la práctica.

- **dameDiasParaInicioEntrega(String practiceCompr):** calcula los días que existen para que comience el inicio del periodo de entrega de la práctica.

- **dameDiasParaFinPeriodoEntrega(String practiceCompr):** calcula los días que quedan para que finalice el inicio del periodo de entrega de la práctica.

- **dameDiasDesdeFinPeriodoEntrega(String practiceCompr):** determina los días que han transcurrido desde que finalizó el periodo de entrega de la práctica.

Esquema de la implementación del Servicio Web.-

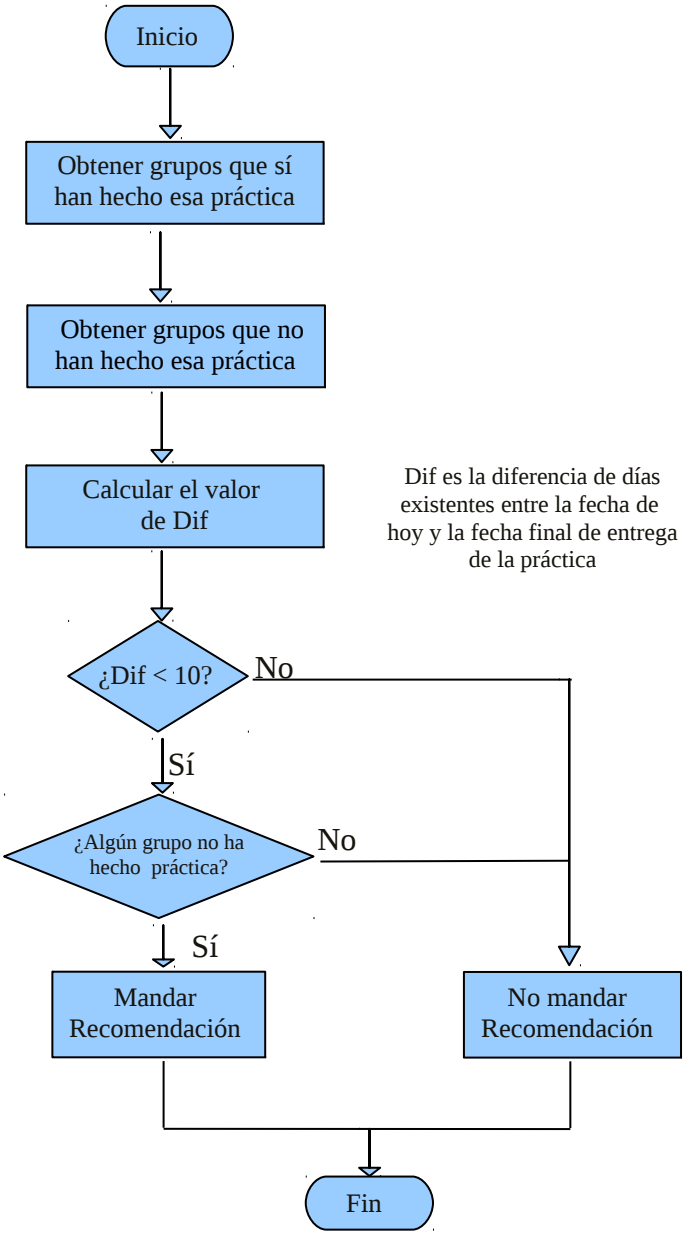


Figura 4.4 Diagrama de flujo del método comprobarRecordatorioGrupo perteneciente al Servicio Web “RecordatorioGrupoSegunPractica”.

En la figura 4.4 se explica visualmente la implementación del método “comprobarRecordatorioGRupo” del Servicio Web “RecordatorioGrupoSegúnPráctica”. En primer lugar se selecciona una práctica a ser analizada y se forman dos grupos, los grupos del laboratorio que no la han realizado y los que sí. A continuación, se comprueba si la fecha actual cumple los requisitos establecidos, en ese caso, que sean menos de 10 días para el final del plazo de entrega de la práctica. Con estos datos, si existe

algún grupo del laboratorio que no ha entregado la práctica y faltan menos de 10 días para la finalización del periodo de entrega, se mandará una recomendación a ese grupo, no a todos.

El proceso para crear los grupos de alumnos que han realizado y no una práctica es el siguiente:

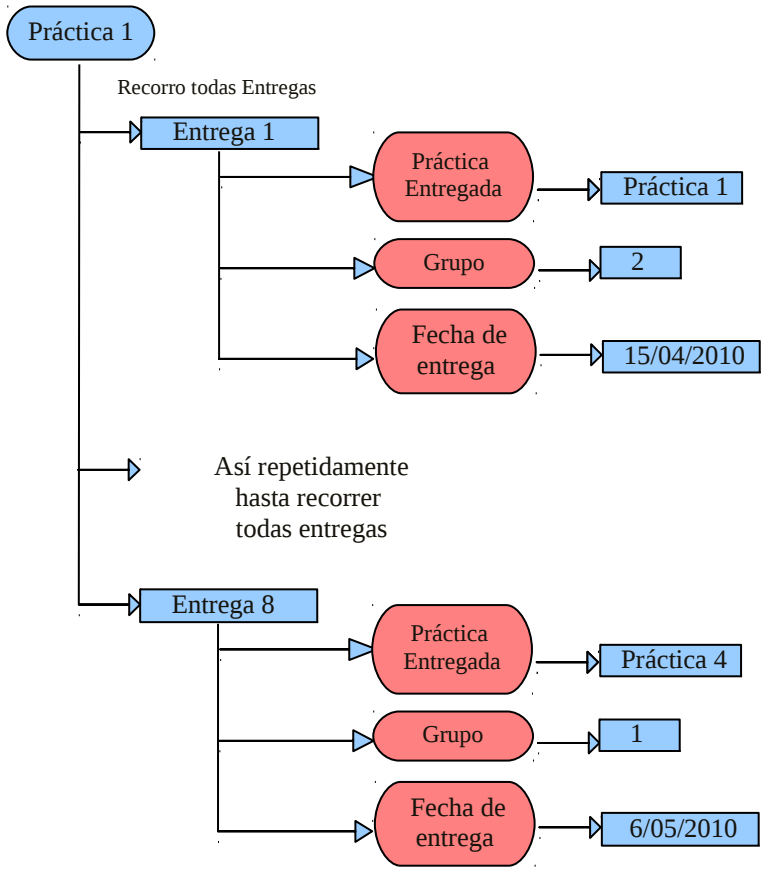


Figura 4.5 Proceso de creación de los grupos de alumnos que han realizado una práctica y los que no.

Una vez que se ha seleccionado una práctica, se recorren todas las entregas almacenadas en la base de datos y se seleccionan únicamente aquellas que pertenezcan a la misma práctica que se ha seleccionado en primer lugar. A continuación se comprueba que la fecha de entrega sea anterior a la fecha de hoy, es decir, que ya haya sido entregada. Si se cumplen estos requisitos, se obtiene el grupo al que pertenece la entrega y se añade al grupo de alumnos que sí han realizado la práctica. En caso de que la fecha sea

posterior, se añadirá ese grupo al grupo de alumnos que no han realizado la práctica.

1.3.4 Servicio Web “RealizarGrupos”.

Este Servicio Web surge para facilitar al profesor o encargado del sistema de entregas un mejor análisis de cómo están las entregas de las diferentes prácticas.

Funcionalidad.-

El objetivo de este Servicio Web es realizar una serie de agrupaciones de alumno en relación a diferentes datos. Los grupos son:

- Alumnos que pertenecen al mismo grupo del laboratorio.
- Alumnos que tienen un recordatorio para la realización de una práctica suplementoria.
- Alumnos que no tienen ningún recordatorio para la realización de una práctica suplementaria.
- Alumnos que tienen el laboratorio suspenso.
- Alumnos que tienen el laboratorio aprobado.
- Alumnos con una misma práctica suspensa.
- Alumnos con una misma práctica aprobada.
- Alumnos con un mismo número de entregas realizadas.

Métodos del Servicio Web.-

- **dameNombresAlumnosDelGrupo(int numGrupo):** crea tantos grupos como distintos grupos de laboratorio existan y devuelve el nombre de los alumnos que pertenecen al grupo pasado como parámetro del método.

- **DameVectorDeAlumnosConRecordatorio():** crea un grupo con todos los alumnos del laboratorio a los que les ha sido mandado un

recomendatorio para realizar una práctica suplementaria.

- **DameVectorDeAlumnosSinRecordatorio():** rea un grupo con todos los alumnos del laboratorio a los que les no ha sido mandado un recomendatorio para realizar una práctica suplementaria.

- **DameAlumnosSuspensos():** crea un grupo con todos los alumnos del laboratorio cuya nota global es inferior a 5.0.

- **dameAlumnosAprobados():** crea un grupo con todos los alumnos del laboratorio cuya nota global es superior o igual a 5.0.

- **dameAlumnosPracticaXSuspendida(String practica):** crea tantos grupos como distintas prácticas existan y devuelve el nombre de los alumnos cuya nota en la práctica pasada como parámetro del método sea inferior a 5.0.

- **dameAlumnosPracticaXAprobada(String practica):** crea tantos grupos como distintas prácticas existan y devuelve el nombre de los alumnos cuya nota en la práctica pasada como parámetro del método sea igual o superior a 5.0.

- **dameAlumnosConXEntregas(int xEntregas):** crea tantos grupos como distintas prácticas existan y devuelve el nombre de los alumnos con el número de entregas realizadas igual al número pasado como parámetro del método.

Esquema de la implementación del Servicio Web.-

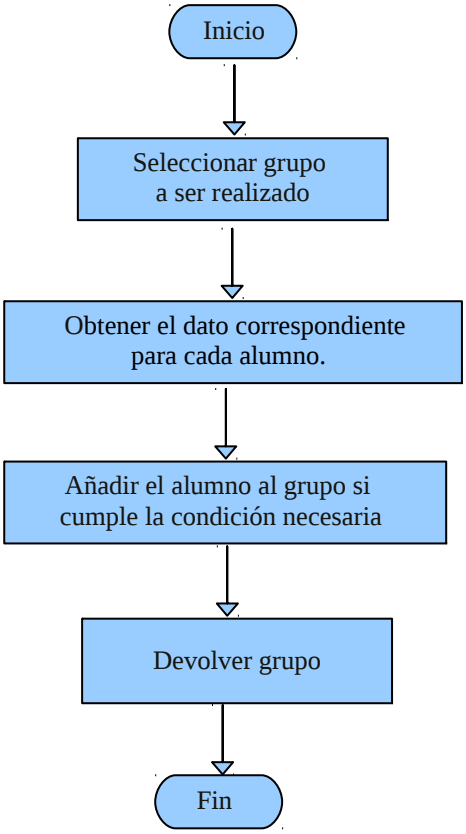


Figura 4.6 Implementación del Servicio Web “RealizarGrupos”.

Como se ve en la figura 4.6, una vez que es seleccionado el grupo a realizar, se extrae su característica selectiva y se calcula para cada uno de los alumnos. Posteriormente, se comprueba si se cumple determinada relación y se añade el alumno si la cumple. Por último, se devuelve el grupo deseado ya formado.

1.3.5 Servicio Web “SupervisiónPráctica”.

Funcionalidad.-

El objetivo de este Servicio Web es mandar un recordatorio si se cumple que si a 'x' días del final del plazo de entrega de la práctica número 'k', el 'y'% no ha entregado la práctica, se manda un aviso exclusivamente a los grupos que no hayan realizado dicha práctica. Los valores introducidos son variables. Para el Servicio Web realizado, son 10 días antes del final del plazo de entrega y el tanto por cierto a comparar es del 50%.

Métodos del Servicio Web.-

- **comprobarRecordatorio(String practice):** comprueba si es necesario mandar el aviso de la práctica introducida como parámetro del método a cualquier grupo del laboratorio de la asignatura.

- **dameTantoPorCientoAlumnosPracticaHechaHastaHoy(String practicaSerComprobada):** devuelve el tanto por ciento de los alumnos que ha realizado la práctica hasta el día de hoy. (Este método pertenece al Servicio Web “TantoPorCiento”).

- **gruposAMandarRecordatorioEntrega(String practicaSerComprobada):** devuelve el nombre de todos los grupos que cumplen la condición de ser enviado un aviso por no haber sido entregada la práctica pasada como parámetro del método.

Esquema de la implementación del Servicio Web.-

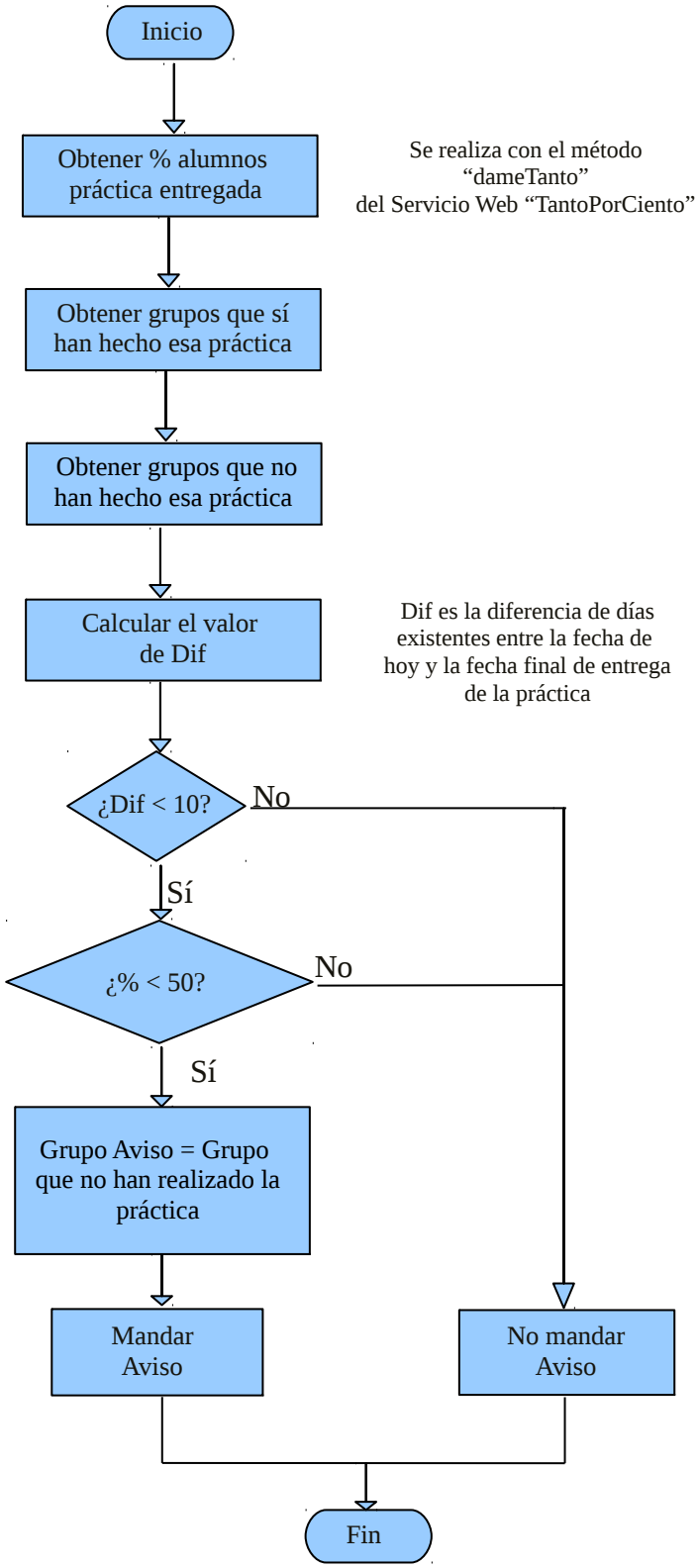


Figura 4.7 Diagrama de flujo del método comprobarRecordatorio perteneciente al Servicio Web "SupervisiónPráctica".

Tal como se muestra en la figura 4.7, este método tiene un diagrama similar al método 'comprobarRecordatorioGrupo' del Servicio Web 'RecordatorioGrupoSegunPractica'. En primer lugar se calcula el tanto por ciento de los alumnos que ha realizado la práctica en la fecha actual. Posteriormente, se crean los grupos de alumnos que han realizado dicha práctica y los que no, ya que se utilizarán al final del método. Una vez que se tienen estos datos, se procede a realizar las comprobaciones necesarias para ver si se cumple o no la condición de mandar una viso de la práctica a algún grupo, comentadas anteriormente. Si se cumple la condición, se mandará un aviso a aquellos grupos que se encuentren dentro del grupo de alumnos que no han realizado la práctica.

1.3.6 Servicio Web “TantosPorCiento”.

A la hora de realizar los Servicios Web, decidí crear éste ya que me parecía muy importante tener conciencia en todo momento de la cantidad de alumnos o grupos que iban entregando una práctica en concreto. Sirve, por ejemplo, para que el profesor pueda averiguar si existe algún problema con la práctica. Este Servicio Web puede ser utilizado a su vez, en otros Servicios Web, como por ejemplo, en que se ha comentado anteriormente a cerca del recordatorio de prácticas.

Funcionalidad.-

El objetivo de este Servicio Web es calcular el tanto por ciento de alumnos o de grupos que han realizado o no una determinada práctica en la fecha actual del cálculo. Sin embargo, dependerá del periodo en el que nos encontremos. Si el periodo de entrega no ha comenzado no se realizará el cálculo del tanto por ciento. Sí lo hará para los otros dos casos, es decir, para el caso en el que nos encontremos dentro del periodo de entrega y para el que ya haya finalizado. Para el primer caso de los dos, únicamente tendremos en cuenta las entregas realizadas desde la fecha de inicio de entrega hasta el día

actual, mientras que para el segundo caso se calculará el tanto por ciento teniendo en cuenta todas entregas que hayan sido realizadas en dicho intervalo, ambos días incluidos.

Métodos del Servicio Web.-

Este Servicio Web consta de 4 métodos diferentes, cada uno de ellos para calcular un tanto por ciento diferente correspondientes al:

- dameTantoPorCientoAlumnosSinLaPracticaHecha
- dameTantoPorCientoAlumnosConLaPracticaHecha
- dameTantoPorCientoGruposSinLaPracticaHecha
- dameTantoPorCientoGruposConLaPracticaHecha

Cada uno de ellos va a devolver un objeto de tipo Integer con el tanto por ciento correspondiente a, como los propios nombres de los métodos indica, diferentes grupos. Éstos son respectivamente, alumnos con la práctica hecha, alumnos sin la práctica entregada, grupos que han entregado y grupos que no lo han hecho.

Los parámetros de entrada de los métodos son la práctica sobre la que queremos hacer la consulta, el número de días respecto a la fecha actual sobre la que queremos consultar y la fecha actual.

El parámetro “días” nos permite obtener los resultados para días anteriores, es decir, si introduzco un valor de 2, me realizará la consulta como si estuviésemos dos días antes. Con ello yo puedo ir comprobando los sucesos ocurridos hasta el día en el que nos encontremos. Si el día actual se encuentra dentro del proceso de entrega pero la diferencia de días entre la fecha actual y el número de días introducidos, hace que la consulta sea anterior a la fecha de inicio, es decir, provoca que nos salgamos del intervalo de entrega, se tomará como referencia el primer día de entrega.

Esquema de la implementación del Servicio Web.-

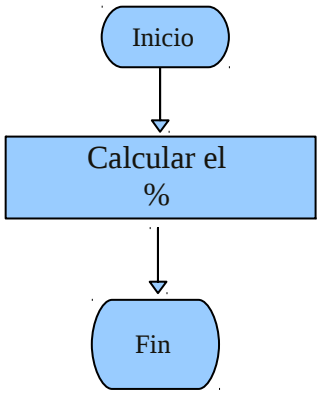


Figura 4.8 Diagrama de flujo del Servicio Web “TantosPorCiento”.

El proceso para calcular el tanto por ciento es el siguiente:

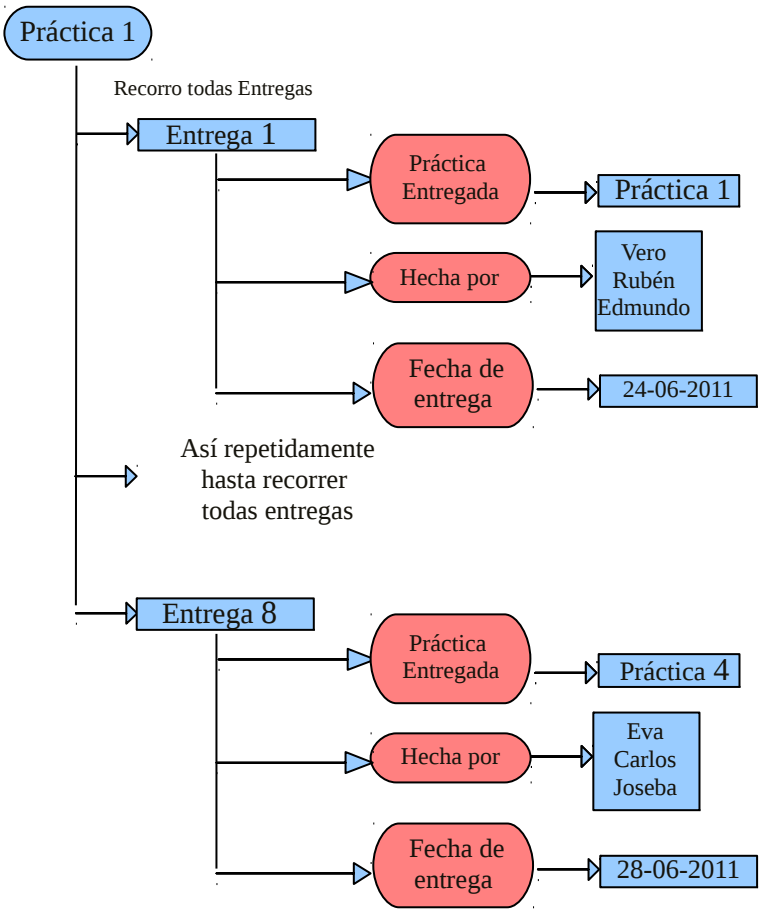


Figura 4.9 Implementación de cálculo del tanto por ciento, perteneciente al diagrama de flujo de la figura X.X mostrada anteriormente.

Para calcular el tanto por ciento, como se muestra en la figura, es necesario recorrer todas las entregas existentes en la base de datos. Se tendrán únicamente en cuenta para el cálculo aquellas cuya práctica entregada sea la práctica sobre la que queremos calcular el tanto por ciento.

Sin embargo, no tendremos en cuenta todas correspondientes a esa práctica ya que pueden existir en la base de datos pero haber sido entregadas fuera del periodo de entrega. Por ello, habrá que obtener las fechas de inicio y final de entrega y la fecha sobre la que queremos hacer la consulta (combinando la fecha actual y el número de días pasado como uno de los parámetros del Servicio Web), para saber así en qué caso nos encontramos y realizar una acción y otra como se comentó en el apartado de introducción de este capítulo.

A partir de esas entregas, se podrá crear el grupo que ha realizado o no dicha práctica, según el método al que estamos llamando.

Por último, se calculará el tanto por ciento a partir del número de alumnos o grupos que hayamos sacado anteriormente y el número total que existen en el laboratorio de la práctica.

1.3.7 Servicio Web “Cantidades”.

Este servicio web surgió como consecuencia del Servicio Web anterior ya que así como me resultaba útil saber el tanto por ciento, también lo era el número en sí, es decir, la cantidad exacta de alumnos o grupos con la práctica hecha o sin entregar.

Funcionalidad.-

El objetivo de este Servicio Web es calcular el número exacto de alumnos o de grupos cuya práctica ha sido entregada o no.

En este Servicio Web también es muy importante la línea temporal ya

que únicamente se realizará el cálculo de este número siempre y cuando el periodo de entrega haya comenzado o acabado ya, omitiendo su cálculo cuando nos encontremos en una fecha anterior a dico perioso.

Métodos del Servicio Web.-

El presente Servicio Web se compone de 4 métodos principales más un método auxiliar “recibirDatos” que nos recupera la información necesaria para poder realizar la correcta obtención de las cantidades que queremos devolver:

dameCantidadAlumnosSinLaPracticaHecha

dameCantidadAlumnosConLaPracticaHecha

dameCantidadGruposSinLaPracticaHecha

dameCantidadGruposConLaPracticaHecha

Cada uno de ellos va a devolver un objeto de tipo Integer con la cantidad exacta correspondiente a, como los propios nombres de los métodos indica, los diferentes grupos. Éstos son respectivamente, alumnos sin la práctica hecha, alumnos con la práctica entregada, grupos que no han entregado y grupos que sí lo han hecho.

Los parámetros de entrada de los métodos son la práctica sobre la que queremos hacer la consulta, el número de días respecto a la fecha actual sobre la que queremos consultar y la fecha actual.

El parámetro fecha es el mismo que el explicado en el Servicio Web TantosPorCiento.

Esquema de la implementación del Servicio Web.-

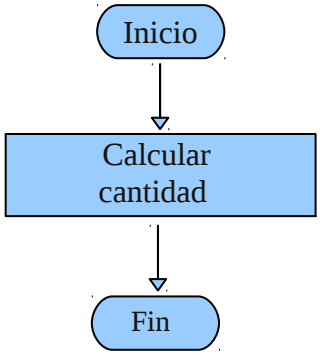


Figura 4.10 Diagrama de flujo del Servicio Web “Cantidades”.

El proceso para calcular la cantidad exacta que buscamos es similar al proceso del cálculo del tanto por ciento pero devolvemos el número de componentes del grupo que hemos creado, es decir, suprimimos el último paso del cálculo de %.

1.3.8 Servicios Web “CompruebaAlumnos” y “CompruebaGrupos”.

Estos Servicios Web surgieron después de pensar en un Servicio Web que mandase un email a los alumnos que no tuviesen una práctica entregada como recordatorio o bien un email informativo para indicar que la práctica sí que había sido recibida con éxito, para lo cuál necesitaría, en primer lugar, sus nombres para poder así recuperar su correo electrónico y mandar dichos emails.

Por tanto, estos Servicios Web van a devolver el nombre o el grupo de aquellos alumnos tanto con una práctica entregada como los que no la han entregado, siendo dos métodos diferentes para cada uno de los Servicios. También se respetará la condición de encontrarnos o bien en el periodo de entrega o bien una vez finalizado éste y la condición de que si al introducir el número de días nos salimos del periodo de entrega, tomaremos como

referencia el día de inicio.

Estos Servicios Web tienen que estar calibrados con los dos anteriores ya que tienen que cuadrar por ejemplo el número de alumnos o el tanto por ciento, con la cantidad de nombres que devolvemos en los métodos que lo definen.

Métodos de los Servicios Web.-

Los métodos que definen estos Servicios Web son 5 para cada uno de ellos, dos auxiliares para recibir la información que necesitamos “recibirDatos” y “obtenerEntregasEntre2Fechas” y 3 principales que realizan las funciones propias de tal servicio web. Éstos son “dameAlumnosSinLaPracticaHecha”, “dameAlumnosConLaPracticaHecha” y “hayAlumnos” para el Servicio Web “CompruebaAlumnos” y “dameGruposSinLaPracticaHecha”, ”dameGruposConLaPracticaHecha” y “hayGrupos” para el segundo. Los dos últimos de cada Servicio Web devuelvem un valor de tipo booleano que será “true” siempre que haya al menos un alumno/grupo que no ha entregado la práctica correspondiente.

Esquema de la implementación del Servicio Web.-

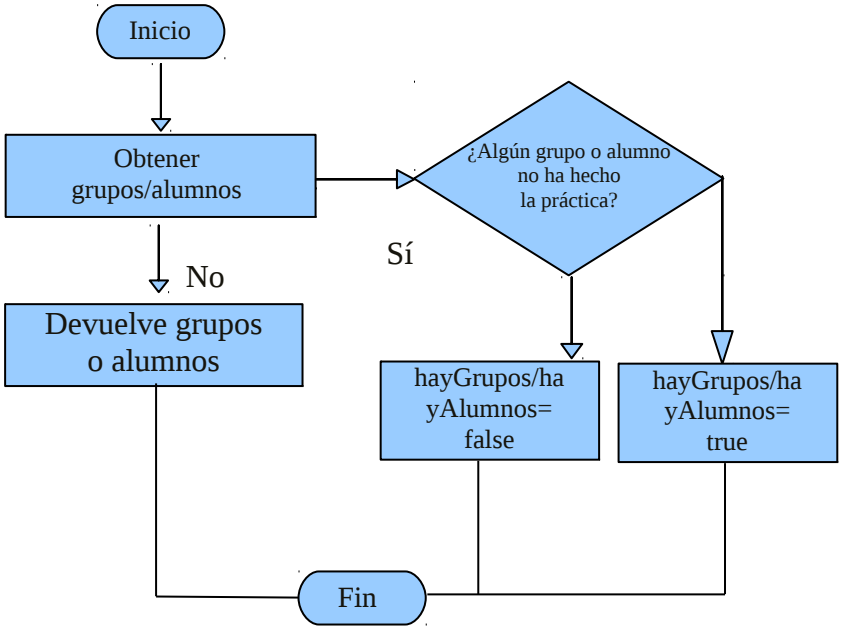


Figura 4.11 Diagrama de flujo del Servicio Web “CompruebaAlumnos/CompruebGrupos”.

Como se puede apreciar en la figura 4.11, obtenemos los grupos o alumnos que no han realizado una práctica en concreto y si éste es mayor de uno, el booleano “hayGrupos” será “true” y “false” en el caso contrario.

1.3.9 Servicios Web “MoverFechas”.

Este Servicio Web tiene el objetivo de retrasar la fecha límite de entrega de una práctica siempre y cuando se cumpla una cierta condición, modificable por parte del profesor introduciendo unos parámetros de entrada en el Servicio Web. Creo que este Servicio Web es útil para modificar el curso de las prácticas de una asignatura ya que pueden ocurrir factores sorpresas que modifiquen su rumbo y sea conveniente en algún caso retrasar el periodo de entrega, como por ejemplo, para dar facilidades a los alumnos para superar el laboratorio de la asignatura porque el número de entregas con la fecha de final se encuentra muy próxima y el número de entregas realizadas es muy bajo.

Métodos de los Servicios Web.-

El presente Servicio Web consta de 6 métodos en su totalidad, siendo el más relevante el método “moverFechaLimitePractica”, que será el que ejecute la modificación de la fecha, para lo cual obtendrá de la base de datos el dato a modificar y lo cambiará por el que le corresponda. Los parámetros de este método son la práctica de entrada, el número de días anteriores a la fecha actual, que corresponderá a la fecha en la que se hace la consulta, la fecha actual, el tanto por ciento de grupos mínimo que tiene que tener la práctica para no ser modificada la fecha y el número de días a retrasar en caso de que sea necesario. Es decir, se modificará la fecha de entrega el número de días introducidos siempre y cuando el tanto por ciento de grupos entregados en la fecha de consulta sea inferior al tanto por ciento que le pasamos como parámetro al Servicio Web.

Los métodos restantes son métodos auxiliares que se utilizarán para realizar el método anterior correctamente. Éstos son:

recibirDatos
obtenerEntregasEntre2Fechas
dameGruposSinLaPracticaHecha
hayGrupos
dameTantoPorCientoGruposSinLaPracticaHecha

Todos ellos van a ser utilizados para obtener los datos necesarios para el método principal. No se van a explicar en esta sección ya que han sido explicados todos ellos en Servicios Web anteriores.

Esquema de la implementación del Servicio Web.-

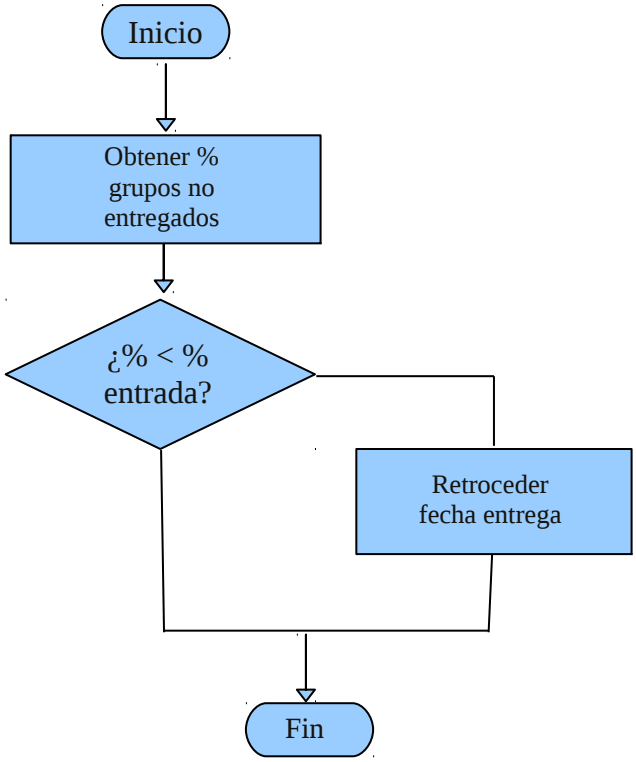


Figura 4.12 Diagrama de flujo del Servicio Web “MoverFechass”.

1.3.10 Servicio Web “MandarEmail”.

Funcionalidad.-

- Los objetivo de este Servicio Web son dos, el primero mandar un email a todos aquellos alumnos que no hayan realizado la entrega de la práctica en la fecha que se comprueba y el segundo, mandar un email como recordatorio para aquellos alumnos que sí que la han entregado. Tal fecha de comprobación tendrá que cumplir la condición de encontrarse dentro del periodo de entrega. Si el periodo de entrega no ha comenzado o ya ha finalizado, no se enviará ningún email ya que en el primero de los casos no tiene sentido mandar ningún recordatorio puesto que no es posible entregarla aún y nadie la puede haber entregado, y en el segundo, ya se da por suspendida o entregada.

- Estos eran los objetivos iniciales del Servicio Web pero a la hora de ser implementado, como no se poseían los email de cada uno de los alumnos, en lugar de enviar un email a cada uno de ellos, se envían todos los emails correspondientes a una única dirección para comprobar que el Servicio Web funciona correctamente.

- El email contiene datos dinámicos sobre cada uno de los alumnos, como son el nombre, la práctica que se comprueba, días que faltan para el final de entrega y fecha de entrega para el segundo de los casos.

- Para realizarlo, se ha utilizado el servidor de Gmail.

Métodos del Servicio Web.-

El Servicio Web MandarEmail consta de 5 métodos , de los cuales se podrían separar como 3 auxiliares (recibirDatos, obtenerEntregasEntre2Fechas, hayQueMandarAlgunRecordatorio) para conseguir toda la información que necesitarán los 2 métodos importantes (mandarEmailAlumnosConPracticaEntregada, mandarEmailAlumnosSinPracticaEntregada) que ejecutarán las acciones principales del método, es decir, mandar los emails tanto a los alumnos que no han entregado las prácticas como los que sí lo han hecho.

Esquema de la implementación del Servicio Web.-

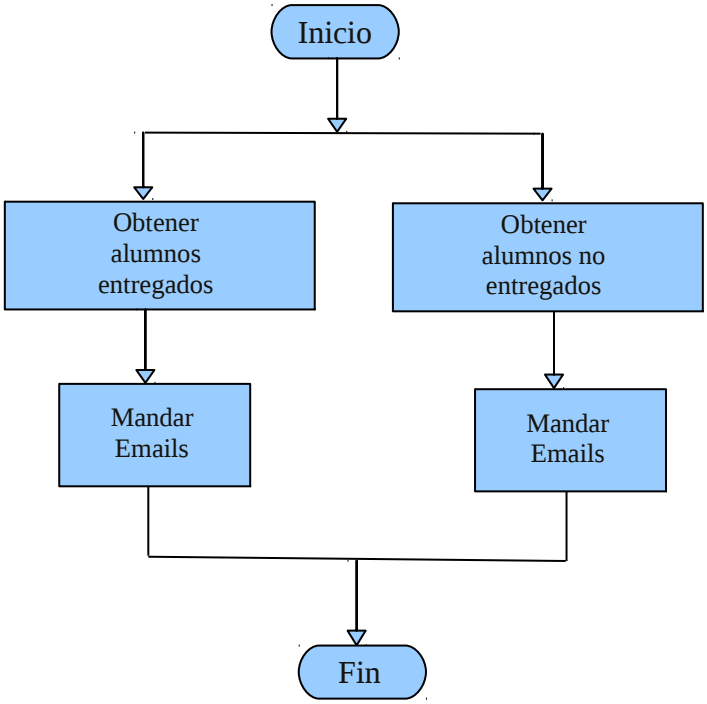
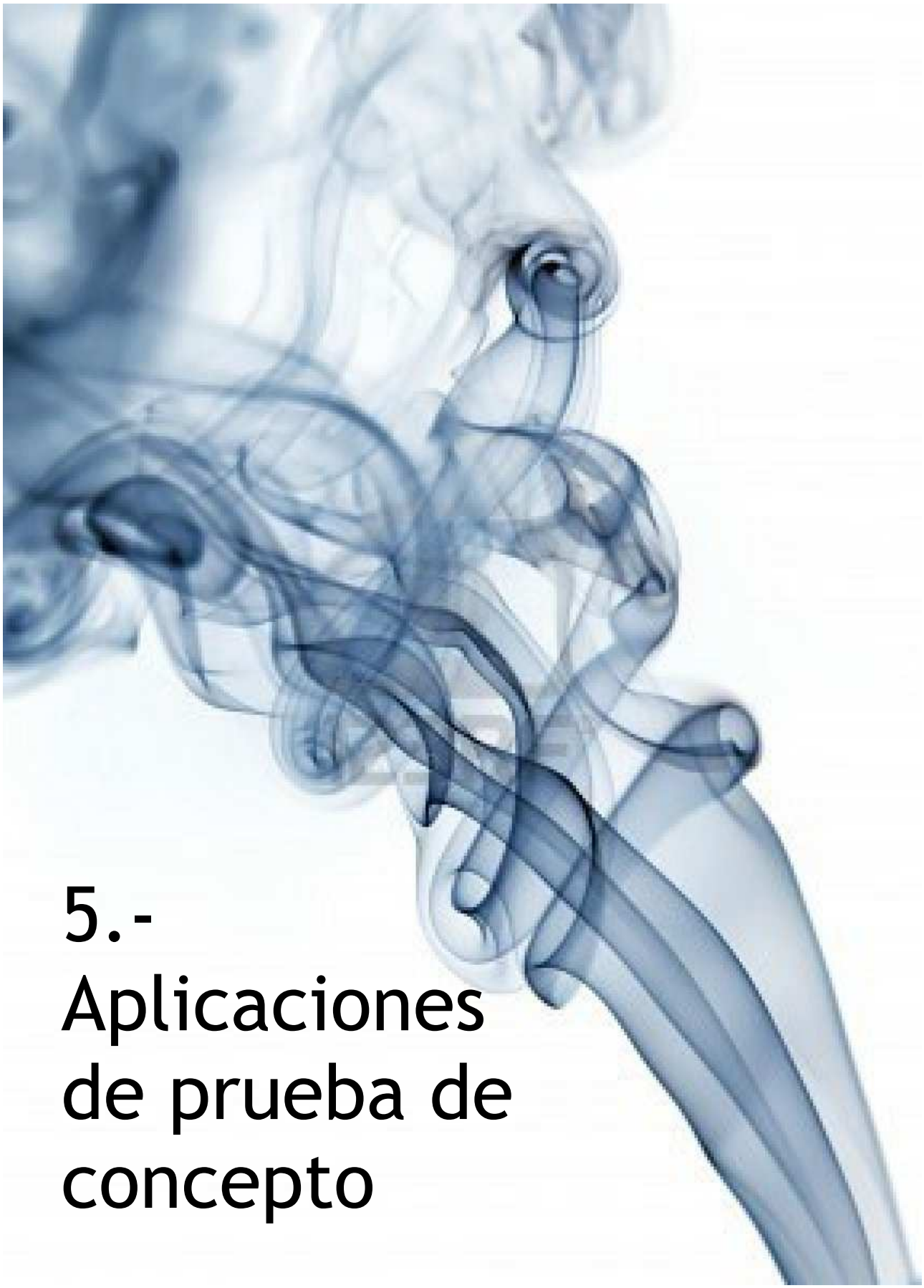


Figura 4.13 Diagrama de flujo del Servicio Web “MandarEmail”.

Como se puede apreciar en la figura 4.13, el diagrama de flujo del Servicio Web MandarEmail es muy sencillo. Únicamente se trata de recuperar los dos grupos de alumnos, es decir, los que sí han entregado la práctica que se está comprobando y los que no, y mandar un email a cada uno de ellos. Estos emails pueden ser los recordatorios de que la práctica no ha sido entregada y los



5.- Aplicaciones de prueba de concepto

5.1 INTERFAZ GRÁFICA

A la hora de realizar el proyecto decidí realizar una interfaz gráfica mediante la biblioteca gráfica de Java JSwing para poder consumir los diferentes Servicios Web de una manera rápida y sencilla.

El aspecto de la interfaz es el siguiente:

Figura 5.1 Representación de la interfaz en su situación inicial.

En rasgos generales y sin comentar detalles técnicos en este momento, como se puede ver en la figura 5.1, la interfaz consta de una lista de Servicios Web en la parte de la izquierda donde se seleccionará aquel Servicio Web que quiera ser llevado a cabo, de un pop-up con distintos componentes gráficos donde serán introducidos los parámetros de entrada necesarios para cada uno de ellos, una serie de botones que permitirán acciones como ejecutar los Servicios Web, cancelar o modificar los parámetros, entre otras, y un área donde serán mostrados los resultados.

5.1.1 Implementación de la Interfaz Gráfica

La interfaz posee un JPanel padre con una distribución de tipo GridBagLayout que permite alinear los componentes horizontal y verticalmente sin necesidad de que éstos sean del mismo tamaño. En este panel es donde se han insertado todos los componentes de la siguiente manera:

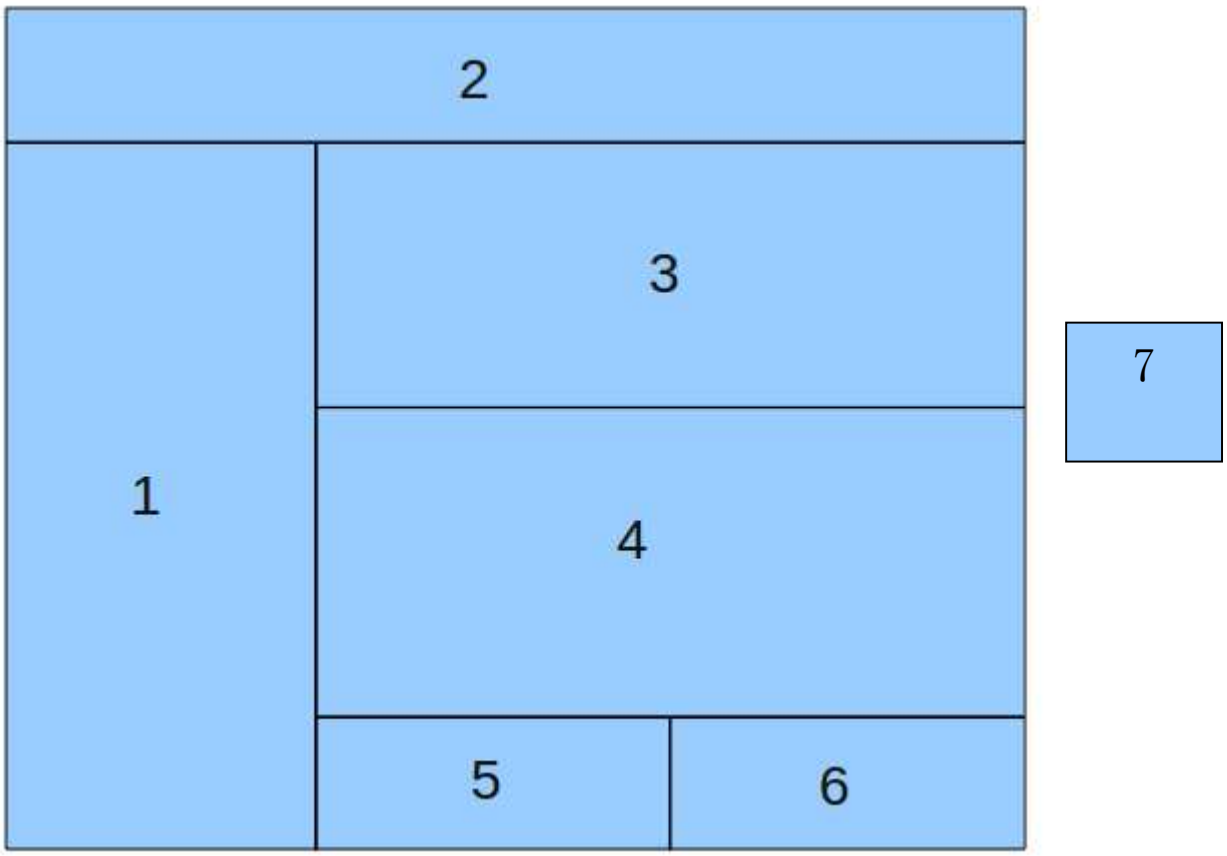


Figura 5.2 Distribución de la interfaz

A continuación van a ser descritos cada uno de los componentes de la figura 5.2.

1.- En primer lugar tenemos un panel de tipo JPanel llamado "panelServiciosWeb". Este panel es de tipo GridLayout, donde el número de filas es igual al número total de Servicios Web de la interfaz y el número de columnas es 1. Cada uno de los objetos de la lista es de tipo JCheckBox para poder seleccionar uno de ellos, dejando como no seleccionados el resto. Estos

componentes están relacionados con la clase OyenteCheckBox para manejar su funcionamiento que se explicará en el apartado 5.1.2. A continuación una imagen del panelServiciosWeb para ver más claramente cómo se han colocado todos los JCheckBox:

Figura 5.3 Visualización del "panelServiciosWeb".

2.- Una etiqueta de tipo JLabel llamada “tituloPagina” para mostrar el título de la página. Esta etiqueta permanecerá estática en todo momento.

Figura 5.4 Visualización de la etiqueta “tituloPagina”.

3.- Un JPanel “panelParametrosIntroducidos” con disposición de tipo GridBagLayout donde se mostrarán los valores de los parámetros introducidos para los Servicios Web, de tal manera que en todo momento el usuario está seguro de la búsqueda que va a realizar. Así mismo habrá dos botones para o bien seguir adelante y ejecutar el Servicio Web o bien cancelar los parámetros introducidos. Su distribución es la siguiente:

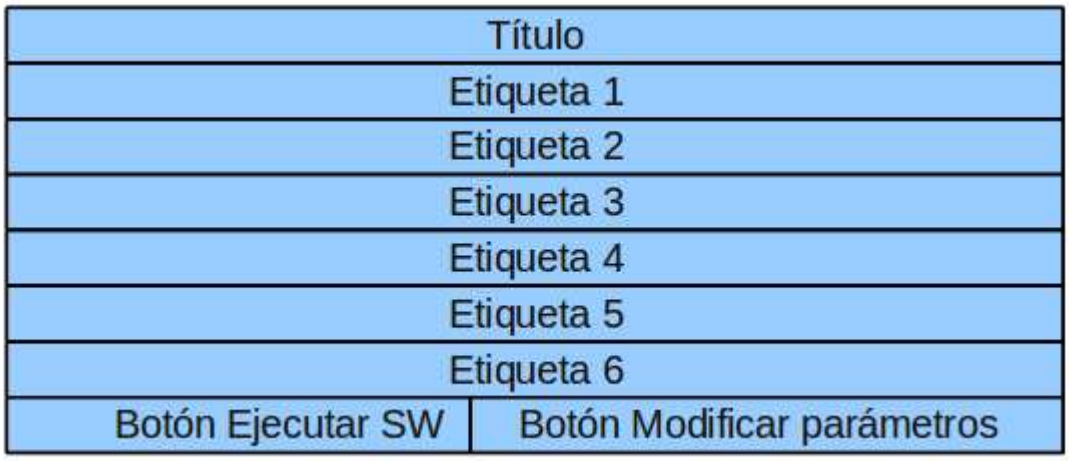


Figura 5.5 Distribución del panel “panelParametrosIntroducidos”.

Como se aprecia en la figura 5.5, el panel consta de una etiqueta estática para poner un título al panel, más 6 etiquetas cuyo contenido es dinámico y será rellenado con los valores introducidos desde el pop-up. En la parte inferior del panel se encuentran los dos botones relacionados con la clase “OyenteEventoBoton” que definirá sus acciones. Por un lado, el situado a la izquierda, cuyo valor es “Botón Ejecutar SW”, llevará a cabo, como su propio nombre indica, la llamada al Servicio Web seleccionado con los

parámetros introducidos. Por otro lado, el botón "Modificar parámetros" mostrará de nuevo el pop-up con los parámetros para poder ser modificados de nuevo. Su aspecto es el siguiente:

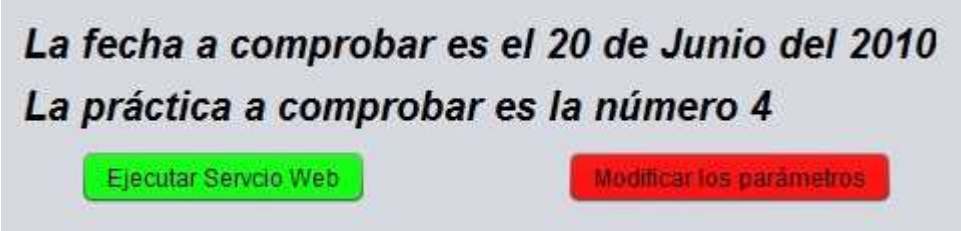


Figura 5.6 Visualización del panel "panelParametrosIntroducidos".

4.- Un JScrollPane con un área de texto de tipo JTextArea, donde se reflejará el resultado de los distintos Servicios.

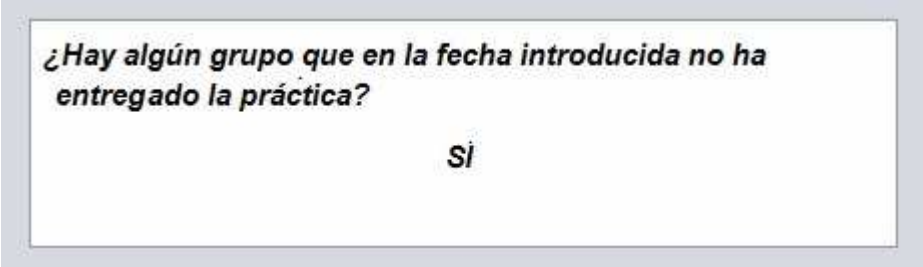


Figura 5.7 Visualización del scroll panel "areaScroll".

5.- En la quinta posición se encuentra un botón de tipo JButton llamado "botonEscogerOtroSW" con valor "Elegir otro SW" asociado a la clase "OyenteEventoDias" para manejar su funcionamiento. En caso de ser pulsado, la interfaz volverá a su situación inicial con todas las etiquetas vacías y todos los JCheckBox de los Servicios Web disponibles para poder ser seleccionados.

6.- A continuación, en la posición 6, se encuentra otro botón JButton llamado "botonUtilizarMismoSW" con valor "Utilizar mismo SW" asociado igualmente a la clase "OyenteEventoDias". En caso de ser pulsado, se abrirá de nuevo el pop-up para poder o bien modificar los parámetros introducidos o dejarlos como estaban y seguir adelante con la ejecución del Servicio Web seleccionado.

Figura 5.8 Visualización de los botones “botonEscogerOtroSW” y “botonUtilizarMismoSW”

7.- En la posición 7 se encuentra un JDialog para crear un pop-up para que el usuario introduzca los parámetros de los distintos Servicios Web. Todo su contenido se encuentra organizado con las propiedades de la clase GridBagLayout y su distribución es la siguiente:

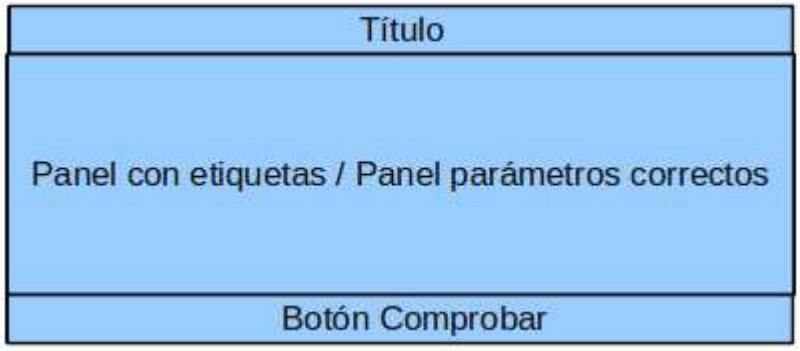


Figura 5.9 Distribución del JDialog “dialog”.

En relación a la figura 5.9, en primer lugar se encuentra una etiqueta cuyo contenido es estático para poner un título informativo al pop-up. A continuación hay un JPanel “panelEtiquetas“, donde se van a insertar los distintos componentes para llevar a cabo la elección de los parámetros de entrada de los Servicios Web. La cantidad de componentes será distinta para cada uno de ellos. Existe también un JPanel que será mostrado cuando la comprobación de los parámetros se haya realizado exitosamente. Por último, se encuentra un JButton asociado a la clase OyenteEventoBoton que se encargará de llamar a la acción de comprobar los parámetros utilizados.

El JDialog tiene este aspecto:

Figura 5.10 Visualización del JDialog "panelEtiquetas".

Como se visualiza en la figura 5.10, el JPanel para introducir los valores a los Servicios Web (parte central excluyendo el título y el botón) posee diferentes componentes propios de JSwing como son JComboBox, JLabel, JButton, ImageIcon. En la parte de la izquierda hay una serie de etiquetas informativas a cerca del parámetro de entrada y a la derecha se encuentran los componentes para su elección. Tanto para el caso de la fecha como para el de la práctica a elegir y el número de días que se va a retrasar, se trata de un desplegable (JComboBox), y para el caso del número de días frente al que se quiere hacer la consulta y el número de días a retrasar la fecha de entrega, se han utilizado varios JButton con objetos de tipo ImageIcon en su contenido. Los JComboBox están asociados a una clase ItemListener y los JButton están asociados a la clase OyenteEventoDias. Una vez que se ha hecho clic sobre el botón comprobar, se muestra un mensaje de acción realizada con éxito y se cierra automáticamente el JDialog ya que se ha realizado un Timer. El mensaje que se muestra es el siguiente:

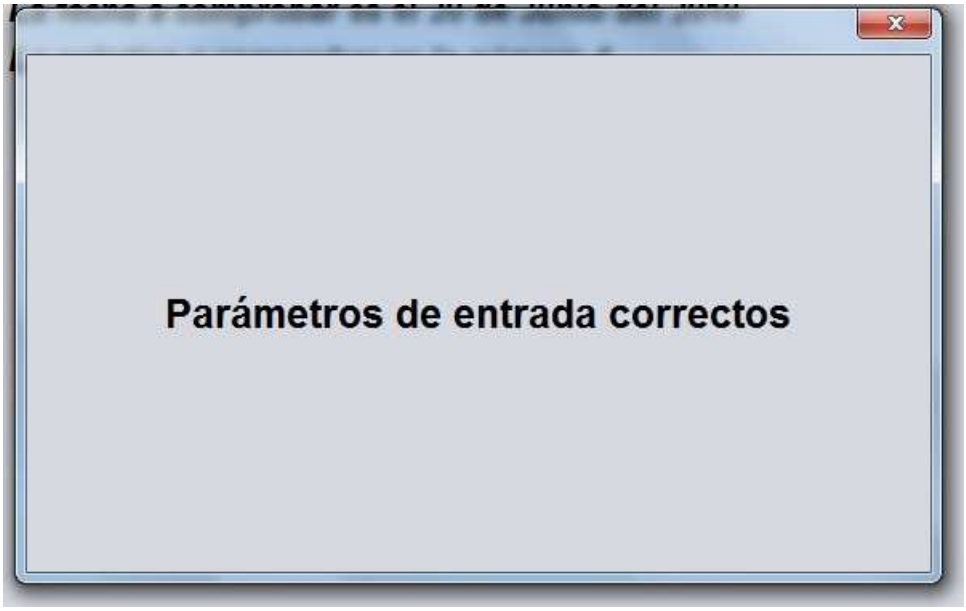


Figura 5.11 Visualización del JDialog “dialog”.

Adicionalmente se creó un menú de tipo JMenuBar para añadir información al usuario en caso de que tuviese alguna duda frente a la interfaz. Se encuentra en la parte superior a la izquierda. Por un lado, consta de un apartado de información de interés sobre la interfaz qué facilita información de cómo ha de ser ejecutada. Su aspecto es el siguiente:

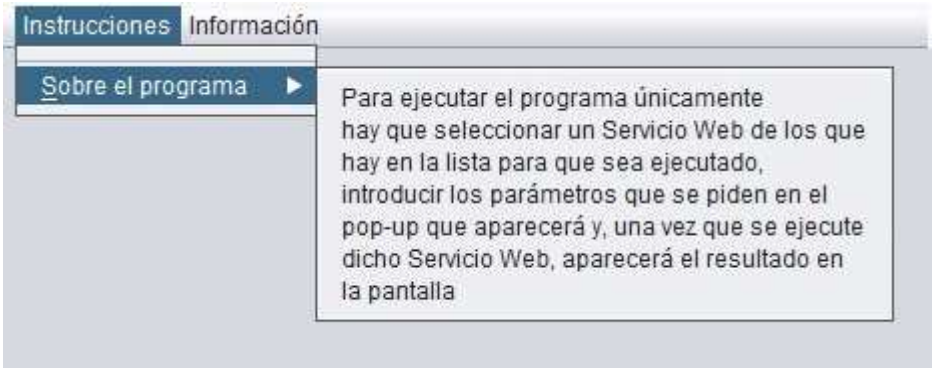


Figura 5.12 Menú información Interfaz

Y por otro lado, tenemos información a cerca de cada uno de los Servicios Web y aplicaciones:

Figura 5.13 Visualización del menú de Servicios Web y Aplicaciones

Al hacer clic sobre alguno de los submenús, tendremos como resultado un pop-up con la información del Servicio Web o aplicación escogida. Su aspecto es el siguiente:

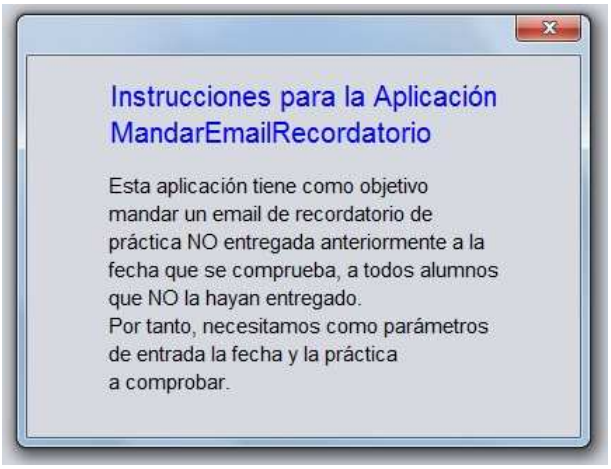


Figura 5.14 Visualización pop-up información de los Servicios Web y Aplicaciones

5.1.2 Clases para la gestión de los Eventos de JSwing.

Para llevar a cabo la gestión de los eventos de los diferentes componentes, se crearon 2 clases distintas:

Clase **OyenteEventoBoton**: Esta clase se creó para controlar la acción resultante de hacer click sobre los diferentes botones. Implementa la interfaz ActionListener, por lo que estará definido su método actionPerformed. Lo primero que se tiene que saber por tanto es qué botón ha sido pulsado:

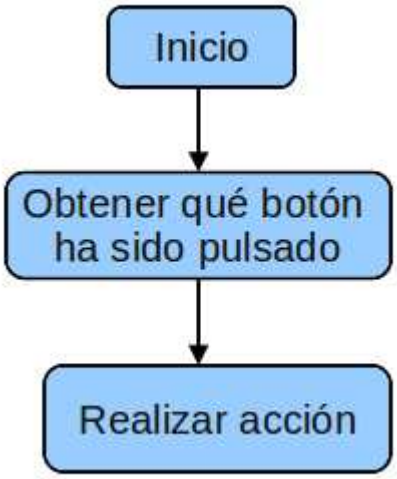


Figura 5.15 Diagrama de flujo de la clase OyenteEventoBoton

Una vez que controlamos que botón ha sido pulsado, se ejecuta la acción correspondiente, que será diferente para los distintos botones:

(Elegir los parámetros)

1. Botón “Comprobar”: comprueba que no hay error en los parámetros introducidos, muestra a continuación un mensaje de “Parámetros introducidos correctamente” y cierra el JDialog mostrando en el JPanel “panelParametrosIntroducidos” los valores de los seleccionados en el pop-up.

2. Botón “Modificar los parámetros“: una vez que han sido seleccionados cuáles son los parámetros de entrada del Servicio y se ha cerrado el pop-up, aparece este botón que lo que hará será abrir de nuevo el pop-up para modificar los valores.

3. Botones con imágenes “flecha”: estos botones se utilizan para aumentar o disminuir en 1 o 10 la cantidad de días frente a los que se quiere hacer la consulta, tanto por ciento a introducir o número de días a retrasar la fecha.

(Ejecutar el Servicio Web).

4. Botón “Ejecutar Servicio Web”: Va a ejecutar el Servicio Web seleccionado en el JCheckBox del panel de servicios web con los parámetros de entrada elegidos en el pop-up.

(Servicio Web ejecutado correctamente).

5. Botón “Elegir otro SW”: Si no ha habido ningún problema a la hora de ejecutar el Servicio Web y se ha mostrado correctamente su resultado, aparece este botón que lo que hace es poner la interfaz gráfica en su posición inicial, resetea todos los valores.

6. Botón “Utilizar mismo SW”: abre de nuevo el pop-up para elegir otros parámetros de búsqueda.

(Error a la hora de ejecutar el Servicio Web).

7. Botón “Cancelar”: Si se ha producido un error a la hora de realizar el Servicio, aparece este JButton que lo que hace es poner la interfaz gráfica en su posición inicial.

Clase **OyenteCheckBox:** esta clase fue creada para controlar el flujo de los JCheckBox que se encuentran en el panel de Servicios Web. Implementa la interfaz ItemListener por lo que está definido su método itemStateChanged para controlar lo que ocurre cuando pasa de estar seleccionado a no seleccionado y viceversa.

El funcionamiento es similar para todos servicios Web: si pasa a estar seleccionado, aparece el pop-up para introducir los parámetros y si pasa a no seleccionado, éste se esconde y todo vuelve a la posición inicial. Es decir, una vez pulsado uno, va a recoger la etiqueta que tiene asociada para saber qué Servicio Web va a ser llevado a cabo ya que para cada uno de ellos se

necesitan unos parámetros de entrada diferentes, por lo que únicamente se resaltarán aquellas etiquetas y campos de texto necesarios junto con el botón de comprobar.

El flujo de un objeto de esta clase sigue el siguiente:

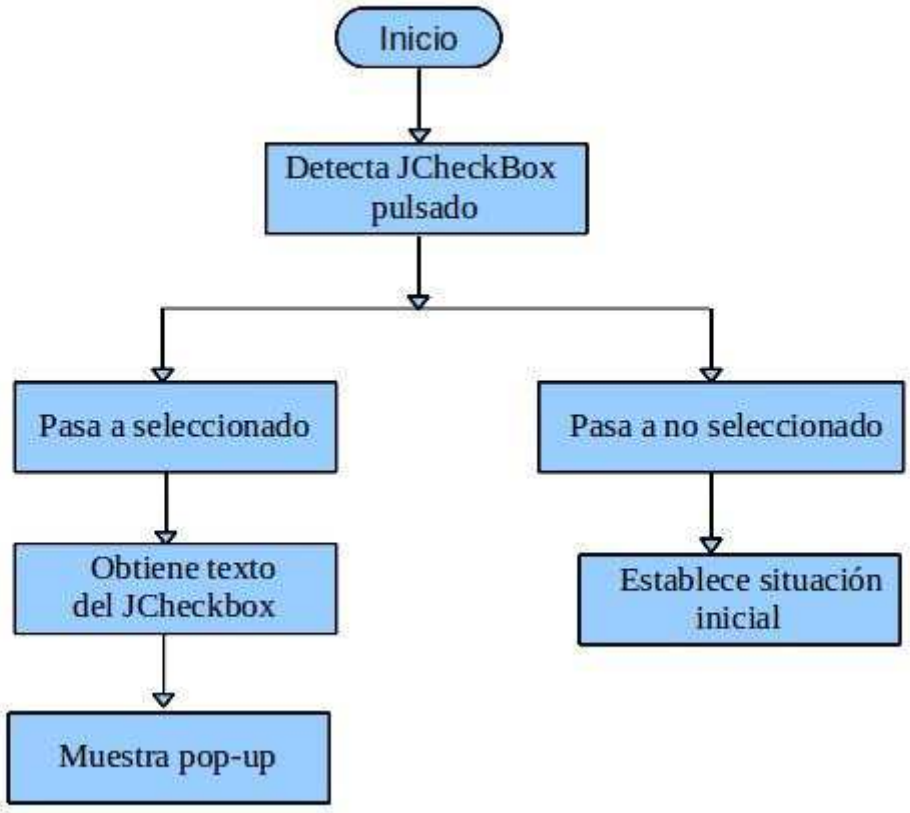


Figura 5.16 Diagrama de flujo de la clase OyenteCheckBox

5.1.3 Flujos que definen la interfaz.

El flujo que sigue la clase java principal para crear la interfaz gráfica es el siguiente:

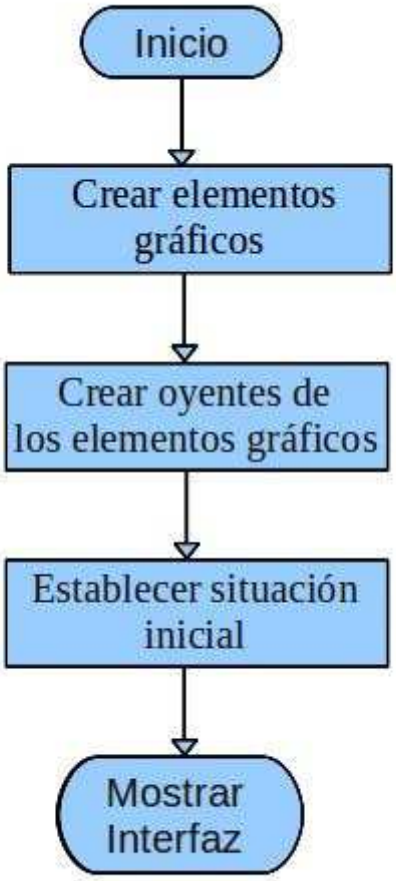


Figura 5.17 Flujo para crear la interfaz gráfica.

Como se ve en la figura 5.13, el primer método que se ejecuta es el de crear elementos gráficos, que no sólo se crean sino que se añaden a sus paneles correspondientes según la organización de la interfaz. A continuación se ejecuta el método crear oyentes de los elementos gráficos, donde se añade a cada elementos gráfico el oyente específico. Los tipos de eventos son los definidos en el apartado anterior. Por último se ejecuta el método que establece la situación inicial para su presentación.

El flujo que va a realizar un usuario con su interacción es el siguiente:

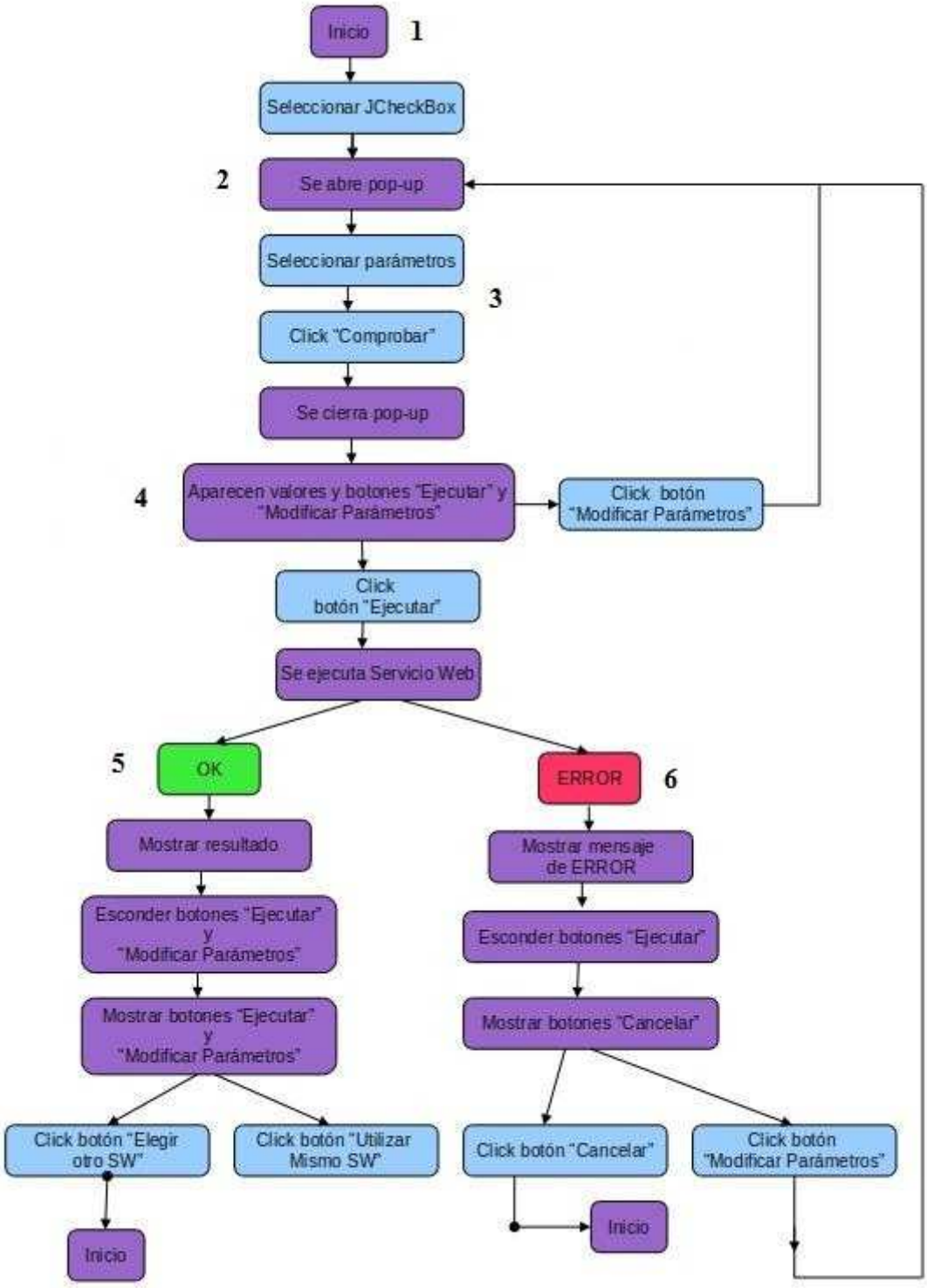


Figura 5.18 Flujo de la interfaz al completo

A continuación se va a explicar el flujo completo de la interfaz gráfica explicando la figura 5.15 y ayudándonos de pantallazos reales de la aplicación.

El primer lugar, una vez lanzada la aplicación Java, aparece la interfaz gráfica en su estado por defecto, que es con el panel de Servicios Web a la izquierda y el área de texto a la derecha (Punto 1).

Figura 5.19 Interfaz en su posición inicial

Posteriormente, el usuario tiene que elegir uno de los Servicios Web, obteniendo como resultado un pop-up donde se encuentran los parámetros a introducir (Punto 2).

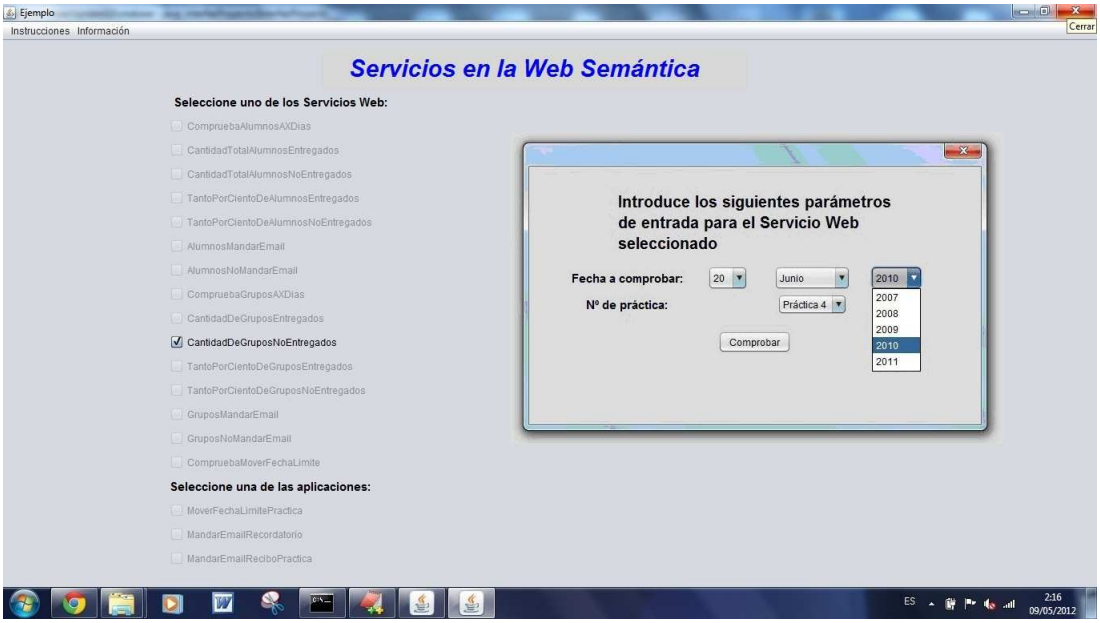


Figura 5.20 Interfaz en modo selección de servicio Web.

Una vez seleccionados los parámetros, el usuario tiene que hacer clic en el botón “Comprobar” para seguir adelante con el Servicio Web (Paso 3).

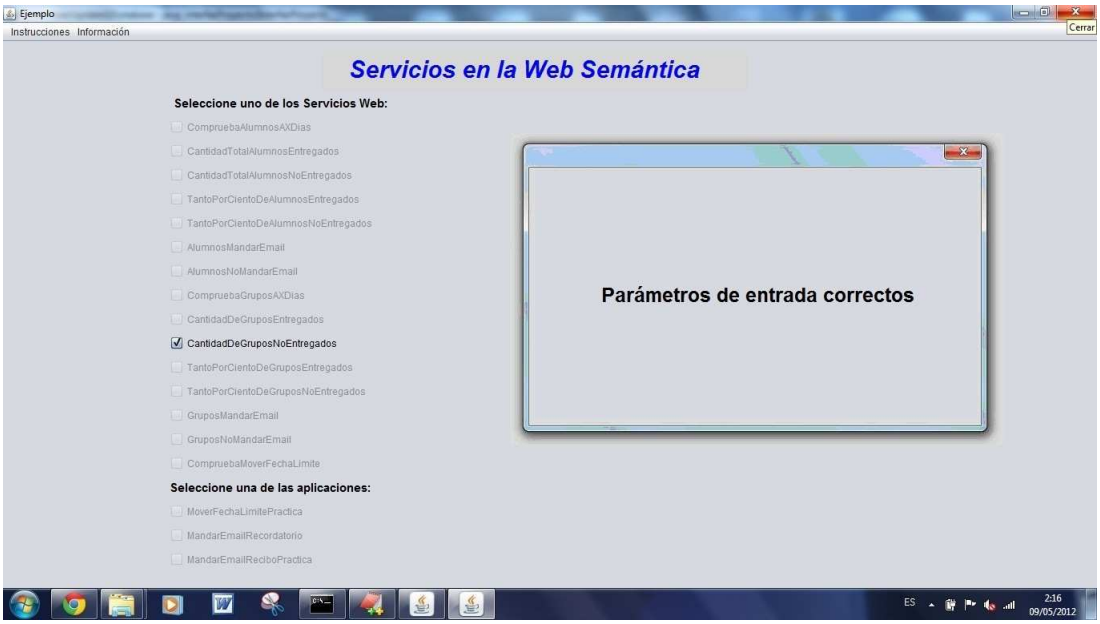


Figura 5.21 Interfaz en modo parámetros introducidos correctamente

A continuación, el pop-up desaparece automáticamente y en el panel principal aparecen los valores que hemos introducido y los botones “Ejecutar Servicio Web” y “Modificar Parámetros” (Paso 4).

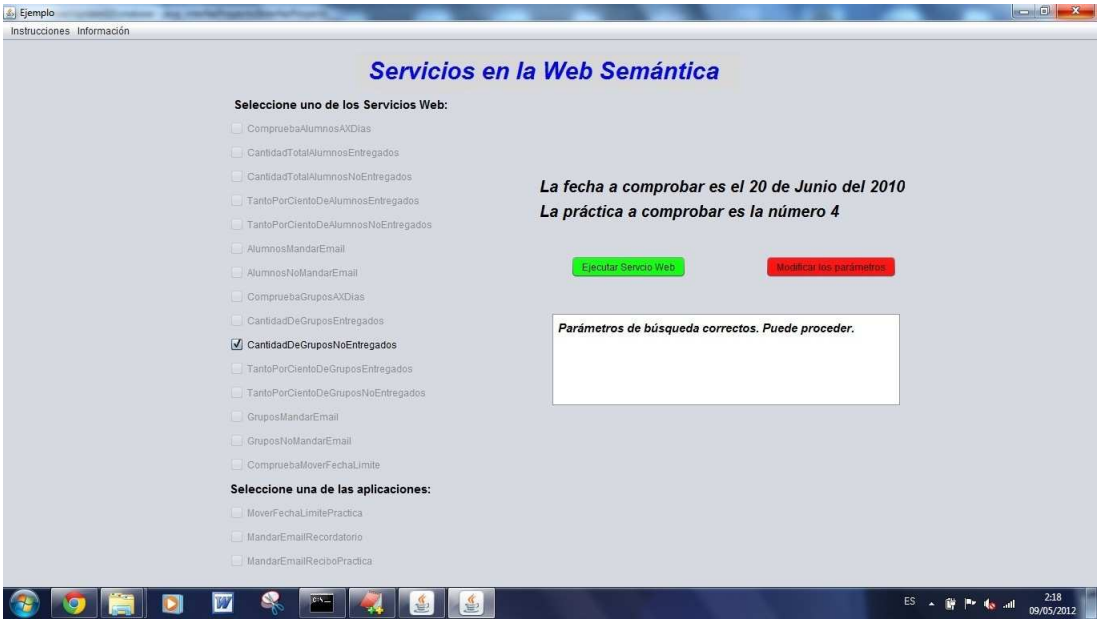


Figura 5.22 Interfaz lista para ejecutar Servicio Web

En este momento, el usuario tiene que elegir si ejecutar el Servicio Web o modifica los parámetros que ha introducido como entrada. Si elige modificar los parámetros de entrada, volverá a aparecer la imagen mostrada en el paso 2. Si elige ejecutar el Servicio Web, hay dos opciones, que se ejecute correctamente (Paso 5):

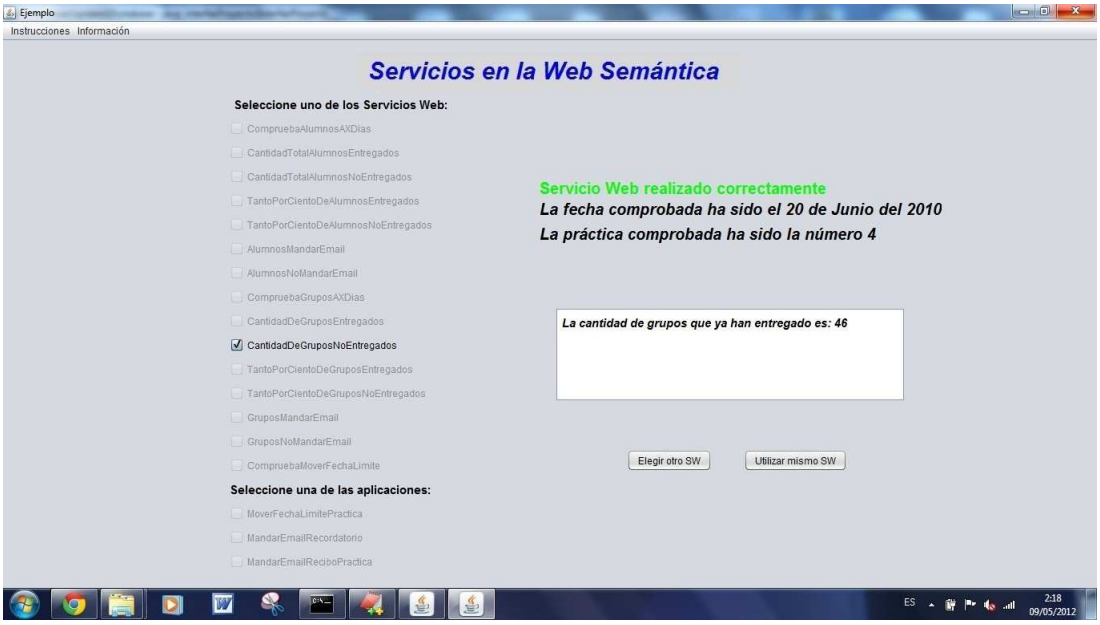


Figura 5.23 Interfaz con servicio Web realizado correctamente

Llegados a este punto, se puede hacer clic sobre “Elegir otro SW” para volver a la situación inicial” o pulsar sobre “Utilizar mismo SW” para que se abra otra vez el pop-up y realizar otra vez el mismo Servicio Web con parámetros diferentes.

O que se produzca un error (Paso 6):

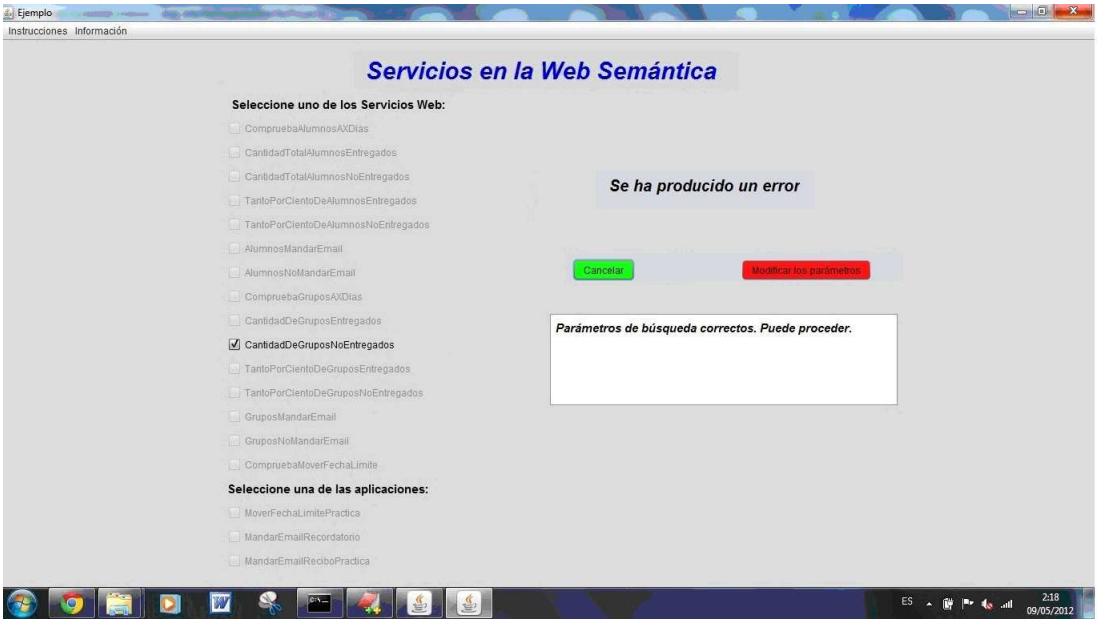


Figura 5.24 Interfaz con servicio Web realizado erróneamente

En este caso, se puede elegir pulsar el botón “Cancelar” para volver a la situación inicial o pulsar sobre el botón “Modificar los parámetros” para que se abra de nuevo el pop-up e intentarlo de nuevo.

5.2 INSTALACIÓN DE LA APLICACIÓN.

Para poder ejecutar la interfaz gráfica comentada en el apartado anterior, han sido necesarios los siguientes componentes:

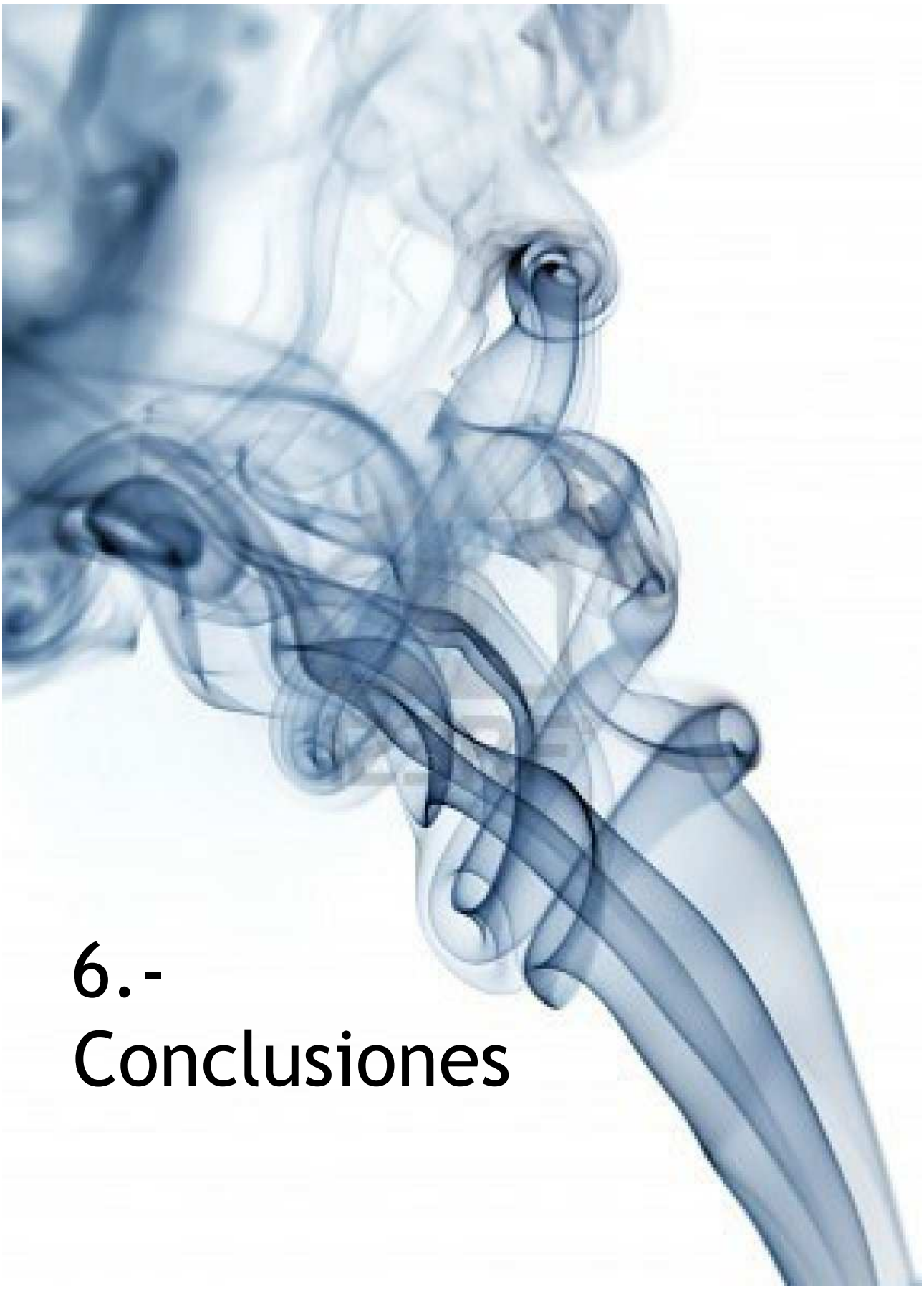
En primer lugar, es necesaria la carga de la base de datos de MySQL con el archivo RDF que hemos creado a partir del XML del sistema real de prácticas. Para ello, fue necesaria la instalación de SDB y la creación y llenado de las tablas comentadas en el apartado 3.2 del estado del arte.

En segundo lugar, se necesitan desplegar los Servicios Web, para lo cual es necesario instalar Jakarta-Tomcat y dentro de su carpeta webapp, incluir

la carpeta de AXIS. Se han introducido los archivos .jar de cada uno de los Servicios Web dentro de la carpeta “lib” de AXIS.

Toda la configuración se encuentra en un paquete a disposición del usuario por lo que únicamente tendría que descargarse el paquete en su ordenador y configurar las Variables de Entorno con las rutas apropiadas para que todo funcione correctamente.

Posteriormente tenemos que ejecutar 2 ejecutables, uno para iniciar el Tomcat y otro para lanzar la aplicación gráfica de la interfaz. Con estos dos ejecutables lanzados, ya se puede hacer uso de la interfaz sin problema alguno.



6.- Conclusiones

Una vez concluido el proyecto, considero muy importante comprobar si se han cumplido los objetivos que se propusieron en su planteamiento y análisis.

Los objetivos principalmente eran 3, crear una ontología para modelar los diferentes aspectos de un sistema de entrega de prácticas, realizar una serie de Servicios Web que fuesen aplicables a un Sistema de Entrega de prácticas real y, en tercer lugar, aplicar esos Servicios Web para crear una herramienta “real” con una interfaz gráfica que trabajase con un sistema de esas características utilizando para ello los Servicios Web creados anteriormente.

Para alcanzar el primer objetivo se utilizó la herramienta Protégé creándose una ontología capaz de definir todas aquellas clases y propiedades que se creyeron oportunas a la hora de realizar un sistema de entrega de prácticas.

El segundo objetivo se ha cumplido ya que se han realizado un total de 7 Servicios Web diferentes con propósitos distintos utilizando en cada uno de ellos procesos y tecnologías diferentes.

El tercer objetivo se alcanzó ya que se ha realizado una herramienta para poder pseudo-utilizar los Servicios Web con los datos de un sistema de entregas. Destaco la palabra pseudo-utilizar porque un proyecto de estas características puede ser muy extenso, mucho superior al alcance de un proyecto final de carrera. Son herramientas reales con muchísimas funcionales propias de equipos enteros de desarrollo. De todos modos, se ha intentado abarcar lo máximo posible y con la mejor calidad posible para realizar cualquier proceso.

Respecto a lo anterior, se podría añadir como trabajos futuros la inclusión de otros sistemas de entregas en el ya creado, realización de aplicaciones distribuidas utilizando más Servicios Web para otros servicios de e-learning,etc.

Otros Servicios Web que se podrían haber realizado en un proyecto final de carrera de estas características serían aquellos que adaptases en mucha

mayor medida las necesidades de cada uno de los estudiantes por separado. Éstos por ejemplo sería recomendación de algún trabajo en particular viendo las partes en las que más falla o en las que más apoyo necesita, es decir, se trataría de realizar recomendaciones adaptadas para cada uno de ellos.

Después de realizar el proyecto, mi conocimiento en relación al e-learning, y todo lo que éste conlleva, se ha visto aumentado considerablemente, así como el interés por todo lo relacionado con éste puesto que creo que las ventajas que se han conseguido con su uso son muy numerosas y con mucho potencial. Se ha eliminado la barrera espacio temporal existente en la educación siendo, bajo mi punto de vista, la característica mayormente destacable puesto que se está brindando la oportunidad de tener acceso a la educación a muchas personas que no la tendrían de no ser por el e-learning.

7.- Presupuesto

7.1 ESCENARIO DE DESARROLLO

Para calcular el presupuesto del presente proyecto se ha analizado el escenario de la fase de desarrollo que aproximadamente se ha extendido a un año de trabajo. Se pueden diferenciar las siguientes fases:

1.- Obtención de requisitos: en primer lugar, tuvo que ser discutido con el tutor qué es lo que se quería conseguir, qué se esperaba de este proyecto final de carrera. Para lograrlo fueron necesarias tanto reuniones presenciales en la universidad como conversaciones a través de emails. En ambas opciones fueron discutidos los diferentes servicios Web que utilizarían las técnicas de Web semántica para poder así manipular los datos relacionados con los sistemas de entregas.

2.- Análisis y diseño: una vez claro los objetivos y orientados hacia la meta a conseguir, fue primordial realizar una fase de análisis y diseño para ver cuáles eran las mejores tecnologías a utilizar o los procedimientos a seguir para poder alcanzar los objetivos y no quedarnos limitados tecnológicamente durante en proceso de implementación.

3.- Implementación: esta fase comprende el proceso completo de construcción de la aplicación, es decir, desde la instalación de los componentes necesarios como MySQL, Java, SDB, Jena o Tomcat hasta la carga de la base de datos, la codificación de los servicios Web y la interfaz gráfica y su puesta en correcto funcionamiento.

4.- Pruebas: la fase de pruebas se puede dividir en pruebas unitarias de cada uno de los Servicios Web por separado, la prueba del comportamiento de la interfaz con cada uno de sus componentes y la prueba de todo el conjunto final, es decir, llamada a los servicios Web desde la interfaz gráfica.

5.- Documentación: para finalizar el proyecto, está la fase de documentación, dejando así reflejado por escrito todo el trabajo realizado en este proyecto final de carrera.

7.2 COSTES

Los costes del proyecto se pueden clasificar en 3 tipos diferentes: personales, de material e indirectos. A continuación se explican más detalladamente cada uno de ellos.

7.2.1 Costes personales

Para poder calcular el coste de personal, se van a reflejar cada una de las tareas en las que se ha divido el trabajo y el número de horas que se han invertido en cada una de ellas para poder obtener así el número total de horas que han sido necesarias.

	Duración en meses	Coste en euros
Fase 1: Obtención de requisitos	1	2.694,39
Fase 2: Análisis y diseño	1	2.694,39
Fase 3: Implementación	6	16.166,34
Fase 4: Pruebas	1	2.694,39
Fase 5: Documentación	2	5.388,78
TOTAL	11	29.638,29

Tabla 7.1 Costes personales

Los datos considerados han sido 21 días laborables por cada mes y jornada de 8 horas diarias. Todos los roles necesarios para el desarrollo del proyecto (diseño, programación en distintos lenguajes, etc.) fueron asumidos por el mismo autor. El salario considerado ha sido el establecido en las

plantillas de la universidad para un Ingeniero, esto es, 2694,39€ por hombre y mes.

7.2.2 Costes de material

Como costes materiales para el proyecto se incluyen tanto los costes de software y hardware como material de oficina como pueden ser folios, tinta para la impresión de la memoria, bolígrafos, etc. En este caso, al tratarse todas las tecnologías involucradas de software libre, los gastos asociados son nulos.

Concepto	Coste	Cantidad	Coste Total
Portatil	750 €	1	750 €
Material oficina varios	25 €	1	25 €
		TOTAL	775 €

Tabla 7.2 Costes de material

7.2.3 Costes indirectos

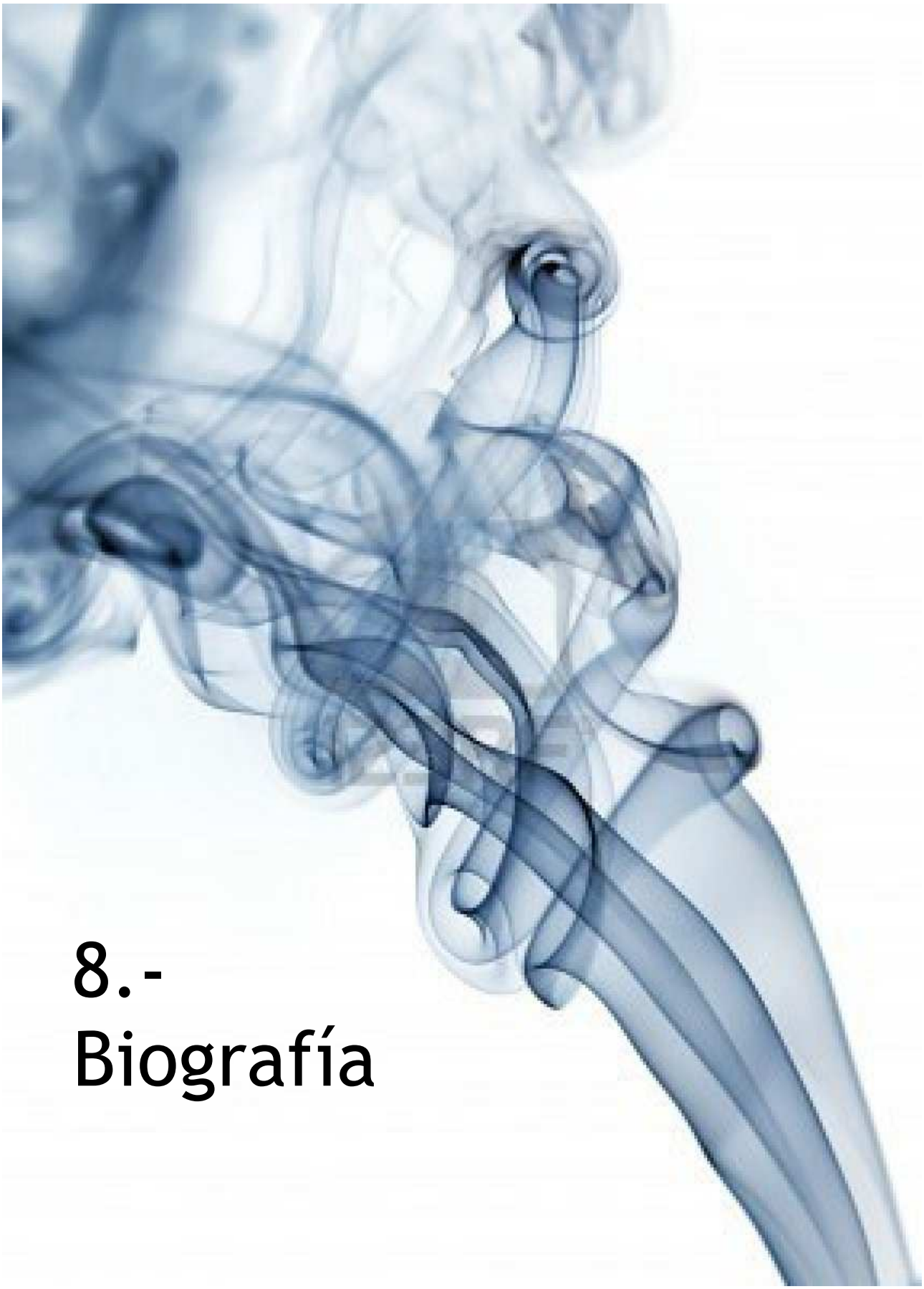
Se calculan como el 20% del coste total y engloban todos aquellos gastos como electricidad, conexión a Internet, etc. En este proyecto ascienden a 5927,658 euros.

7 .2.4 Resumen de costes

Concepto	Coste
Costes personales	29.638,29 E
Costes de material	775,00 €
Costes indirectos	5927,658 €
TOTAL	36.340,948 €

Tabla 7.3 Presupuesto total del proyecto.

El presupuesto total del proyecto asciende a 36.340,948 €.



8.- Biografía

[1] Web 2.0, Web 3.0 y Web 4.0. Website 2009.
http://www.icesi.edu.co/blogs/egatic/2009/05/21/web-20-web-30-y-web-40/

[2] “Web1.0 vs Web 2.0”. Documento pdf.
http://ddd.uab.cat/pub/dim/16993748n10a4.pdf

[3] Wikipedia. Website 2012. http://es.wikipedia.org/wiki/Web_2.0

[4] “Web 3.0 Web Semántica”. Website 2011.
http://web30websemantica.comuf.com/web30.htm

[5] “RDF, SPARQL y Jena“. Documento pdf Departamento de Ciencias de la Computación Universidad de Talca. 2010.
http://dcc.utalca.cl/~rangles/charlas/2010.10.20-Jena.pdf

[6] Validation Service. Website 2007. http://www.w3.org/RDF/Validator/

[7] “SPARQL Query Language for RDF. Official Websit”. Website 2008.
http://www.w3.org/TR/rdf-sparql-query/

[8] “OWL Web Ontology Language Overview Official Website”. Website 2004.
http://www.w3.org/TR/owl-features/

[9] “Apache Jena Official Website”. Website 2011-12.
http://jena.apache.org/

[10] “ARQ Official Website”. Website 2011-12.
http://jena.apache.org/documentation/query/index.html

[11] “SDB Official Website”. Website 2011-12.
http://jena.apache.org/documentation/sdb/index.html

[12] “TDB Official Website”. Website 2011-12.
http://jena.apache.org/documentation/tdb/index.html

[13] “Pablo Castells. Escuela Politécnica Superior. Universidad Autónoma de Madrid”. Documento pdf.
http://arantxa.ii.uam.es/~castells/publications/castells-uclm03.pdf

[14] “Presentación de e-learning”. Documento pdf.
http://www.proyectomono.cl/elearning/pdf/presentacion_e_learning.pdf

[15] Bases pedagógicas del e-learning. Revista de Universidad y Sociedad del Conocimiento. 2006. http://www.uoc.edu/rusc/3/1/dt/esp/cabero.pdf

[16] A. Tregobov, “The Web-Based Assignment Submission Systems”, Presented at NAWeb, 1998

[17] D. Jones, S. Behrens, “Online Assignment Management: An Evolutionary Tale”, Proceedings of the 36 th Hawaii International Conference on System

Sciences, 2003
[18] LRN Website. Website. http://www.dotlrn.org/index.html
[19] Wikipedia. WebCT. Website. http://en.wikipedia.org/wiki/WebCT
[20] Dokeos Official Website. Website. http://www.dokeos.com/
[21] Moodle Official Website. Website http://moodle.org/
[22] “Guía básica de subversion”. Documento pdf. http://quegrande.org/apuntes/EI/4/Com/practicas/09-10/guia_basica_de_subversion.pdf
[23] “GIT Oficial Website”. Website. http://git.or.cz/index.html