UNIVERSIDAD CARLOS III DE MADRID

[ESCUELA POLITÉCNICA SUPERIOR]

Ingeniería Informática

Buscador Semántico sobre una Red Social de preguntas de examen.

Realizado por: **Diego Jiménez López**

Dirigido por: **Ricardo Colomo Palacios**

octubre de 2009

Dedicado a la persona cuyo nombre

he tomado prestado.

“Non ho mai incontrato un uomo così ignorante
dal quale non abbia potuto imparare qualcosa.”

(“Nunca he encontrado un hombre tan ignorante
del cual no pueda aprender alguna cosa”)

Galileo Galilei (1564-1642)

RESUMEN

El desarrollo de la Enseñanza Asistida por Ordenador ha sido, desde sus comienzos, fruto del esfuerzo de diversas entidades, que compartían meta, pero no camino. Este desarrollo paralelo, ha provocado que el uso de las bases de conocimiento manejadas por cada aplicación sean casi absolutamente dependientes de la aplicación en sí. Hoy día, la tecnología tiende hacia la estandarización y el esfuerzo colaborativo entre distintas aplicaciones, así como entre usuarios y desarrolladores, de modo que existe la necesidad de adaptar los conceptos del E-Learning a las tecnologías actuales.

La aparición de la Web Semántica, y la Web Social, ofrece un marco de trabajo ideal para la organización de la información. Los conceptos de *Red Social*, y *Ontología,* aplicados a una base de conocimiento de preguntas de exámenes, permiten que la información sea fácilmente recopilada y recuperada por los usuarios.

El *Buscador Semántico sobre una Red Social de preguntas de examen*, crea un entorno colaborativo para distintos usuarios, donde pueden compartir elementos de evaluación a varios niveles. La información está organizada mediante tecnologías de Web Semántica, y estándares de e-learning, para ser aprovechada por aplicaciones cliente de servicios Web independientes. La capacidad semántica de la aplicación va mejorando a través de la experiencia de los usuarios, de modo que, con el paso del tiempo, los resultados de búsqueda se aproximarán a aquellos que un usuario medio espera encontrar.

ABSTRACT

The Computer-Assisted Learning development has been, since its early stages, the result of the effort of several entities, which shared the goal, but not the way. This parallel development had caused that the use of the knowledge bases, handled by each application, was almost absolutely dependent on the application itself. Nowadays, Technology is tending to standardization and cooperative effort among various applications, as well as among developers and users. This shows the need to adapt the E-Learning concepts to the current technologies.

The appearance of the Semantic Web, and the Social Web, provides an ideal framework for the organization of information. The concepts *Social Network* and *Ontology*, applied to a knowledge base of questions for exams, allow information to be easily gathered and recovered by users.

The *Semantic Search Engine, over a Social Network of questions for exams*, builds a cooperative environment for different users, where they can share assessment items on several levels. The information is organized through Semantic Web Technologies, and E-Learning standards, in order to be used by independent Web services client applications. The application semantic ability, is improved thanks to the users' experience, so, in time, the search results will come close to those that an average user expects.

ÍNDICE GENERAL

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1. INTRODUCCIÓN Y OBJETIVOS.

El objetivo de esta sección es permitir una aproximación inicial al dominio que se estudiará en el proyecto *Buscador Semántico sobre una Red Social de preguntas de examen*.

Introducción.

En este apartado se definirán los dominios que se estudiarán, de forma previa a su análisis en el *Estado del arte*. Del mismo modo, se indicará el contenido de cada una de las secciones que formarán el documento.

Descripción del ámbito de estudio.

Tal y como se verá en el apartado *Objetivos,* de esta misma introducción, el fin del proyecto es elaborar algún tipo de sistema que permita la compartición de exámenes (a nivel de pregunta), entre distintos usuarios.

El fin del entorno donde se encuadrará el proyecto es la educación, a través de una aplicación informática, de modo que un dominio fundamental que será necesario estudiar será el *e-learning*, que estudia no sólo la enseñanza a través de nuevas tecnologías, sino también la evaluación de los conocimientos adquiridos. El proyecto no estará encaminado a la enseñanza en sí misma, sino que se centrará tan sólo en los elementos de evaluación. En un primer momento, tan sólo se manejarán preguntas relacionadas con Ingeniería del Software para acotar el área de conocimiento.

La evolución del mundo de la Web en los últimos años, parece ir encaminado hacia unas aplicaciones con contenidos creados por y para los usuarios. Se pretende llevar este modelo hacia el mundo educativo, de modo que será conveniente analizar en profundidad este fenómeno, conocido como *Web Social* o *Web 2.0*.

Otra rama de evolución de la Web consiste en la llamada *Web Semántica*, que consiste en el uso de ciertas tecnologías para crear una Web con contenidos que sean comprensibles y computables por las máquinas. Si se pretende conseguir que la aplicación resultante del proyecto sirva contenidos de manera óptima, será conveniente investigar la información sobre Web Semántica existente, y buscar en qué modo podría aplicarse.

Conviene remarcar que el proyecto se referirá tan sólo al almacenamiento y compartición de las preguntas de examen. La elaboración de herramientas cliente del servidor, que permitan el uso de las preguntas obtenidas, será competencia de otros proyectos ajenos a este.

Estructura de la memoria.

La memoria del proyecto Servidor on-line de Exámenes queda estructurada en cinco partes fundamentales.

Esta misma sección consiste en una introducción que permita centrarse en el ámbito concreto que va a estudiarse, permitiendo una mejor comprensión de la memoria completa.

Tras la introducción, tendrá lugar el estudio del *Estado del Arte*, que consiste en la investigación previa de los dominios que, a priori, parecen relacionados con el proyecto que pretende desarrollarse. Las conclusiones de dicho Estado del Arte permitirán ver las carencias presentes en dichos dominios, con vistas a resolverlas durante el desarrollo del proyecto.

Una vez analizada la situación actual de los dominios elegidos, se definirán formalmente los problemas encontrados que se pretenden resolver con el desarrollo del proyecto. Esto tendrá lugar en la sección *Descripción del problema*.

En la sección *Descripción de la Solución*, se propondrá una forma en la que el proyecto puede resolver la problemática descrita en la sección anterior. Por un lado, se definirá una solución funcional que permita ver de qué funcionalidades dispondrá la aplicación resultante, y por otra una solución tecnológica que mostrará las tecnologías relevantes tanto para el desarrollo y gestión del proyecto, como para su funcionamiento.

En las *Conclusiones y líneas futuras* se indicará, una vez completado del desarrollo del proyecto, si la problemática que se encontró queda resuelta con la aplicación resultante. Por otro lado, indicará qué problemas quedan pendientes de resolver en su caso, además del rumbo que podrían tomar desarrollos posteriores basados en este proyecto.

En los *Apéndices* se adjuntarán documentos relacionados con el desarrollo que se consideren interesantes para una mejor comprensión del proyecto al completo.

La sección de *Bibliografía* contiene todos los documentos a los que se hayan hecho referencia en la memoria del proyecto formalmente, para permitir su consulta en caso de querer documentarse sobre los temas tratados.

Objetivos.

En este apartado, se establecerán las metas del proyecto. En primer lugar de forma textual, para después mostrarlas de forma esquemática.

Como ya se adelantó en la introducción, el objetivo fundamental del proyecto es poner a la disposición del público un repositorio en el que los usuarios puedan compartir preguntas de exámenes sobre Ingeniería del Software.

Llama la atención que, pese a haber transcurrido mucho tiempo desde que empezaron a diseñarse aplicaciones para hacer exámenes -y repositorios on-line para ellos- aún no se haya generalizado el uso de ningún estándar de almacenamiento o compartición, que permita la interconectividad entre aplicaciones docentes, y el aprovechamiento óptimo de las preguntas para todos los usuarios.

Un objetivo secundario será proponer el aprovechamiento de las tecnologías existentes con el fin de establecer una interfaz de comunicación con otras

aplicaciones, que permita la automatización en la recuperación de las preguntas de examen servidas. Se pretende minimizar la necesidad de la intervención humana a la hora de clasificar, seleccionar, y evaluar la calidad de las preguntas que se hayan compartido.

En el desarrollo del Servidor on-line de Exámenes, el fin será la construcción de un software orientado al tratamiento masivo de datos. Siempre se tendrá en mente la creación de una aplicación *"de producción"*, que no presente carencias en el momento que aumente el número de usuarios, o de datos manejados.

En resumen, los objetivos del proyecto Servidor on-line de Exámenes son:

1. Crear un **entorno de compartición de preguntas** de Ingeniería del Software a través de internet.
2. **Aprovechar tecnologías** de reciente implantación con fines educativos.
3. Crear una **interfaz** pública para que **otras aplicaciones** puedan servirse de los contenidos publicados, mediante el uso de estándares.

2. ESTADO DEL ARTE.

La investigación de las soluciones existentes hoy día, se centrará en tres dominios: E-Learning, Web Semántica y Web Social.

La estructura que se seguirá para la presentación de los resultados, será la siguiente: en esta misma introducción, se presentará una breve descripción de cada dominio, justificando su importancia para el proyecto. A continuación, se procederá con el análisis de la situación actual de cada dominio (en general, y aplicado a la ingeniería del software). Posteriormente, se presentarán las conclusiones obtenidas sobre la relación existente entre los dominios y sus carencias.

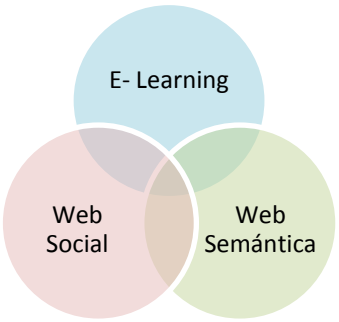

Ilustración 1 Esquema de los dominios en los que se enmarca el proyecto

Se denomina “E-Learning” a los sistemas relacionados con el aprendizaje a través de una vía telemática.

El objetivo del proyecto es poner a la disposición del público elementos de evaluación sobre ingeniería del software. Estos elementos permitirían la valoración de los conocimientos adquiridos sobre dicha área de conocimiento a distancia, luego se podría considerar como una herramienta de e-learning.

La “Web Semántica” abarca todas las tecnologías capaces de dar un significado a la información existente en la Web, para hacer posible una clasificación óptima de dicha información que permita aumentar la interoperabilidad entre sistemas automáticos, y reducir la intervención humana en su procesamiento.

Se pretende que la información compartida sea accesible y recuperable por el mayor número de usuarios posible. No hay que desestimar el uso de tecnologías y estándares de Web Semántica; lo que posibilitaría evitar las redundancias y descontextualizaciones en los elementos de evaluación compartidos.

La “Web Social”, o “Web 2.0”, es un concepto que cada día cobra más importancia. Consiste en la generación y mantenimiento de contenidos en un entorno Web, a cargo de los propios usuarios, de dicha información.

El proyecto puede enfocarse como la elaboración de una red social de elementos de evaluación para ingeniería del software, en la que cada usuario tenga la posibilidad de publicar información de su creación. Un estudio de las características de las redes sociales de mayor difusión, podría sentar las bases del protocolo de compartición de información.

A continuación, se muestra el análisis de cada dominio, y la forma en que podrían enfocarse hacia la Ingeniería del Software.

Web Social.

La Web Social, también llamada Web 2.0, es según Christian Van Der Hest S. (1): "*la transición que se ha dado de aplicaciones tradicionales hacia aplicaciones que funcionan a través del web enfocadas al usuario final. Se trata de aplicaciones que generen colaboración y de servicios que reemplacen las aplicaciones de escritorio*". En otras palabras, se trata del conjunto de aplicaciones Web en las que la experiencia del usuario final cobra protagonismo. También añade que "*El Web 2.0 no es precisamente una tecnología, sino es la actitud con la que debemos trabajar para desarrollar en Internet*". Esto indica que puede ser complicado definir qué es exactamente la Web Social.

Tim O'Reilly, autor del término "Web 2.0", dice en su artículo original al respecto (2) que la Web 2.0 no tiene una clara frontera, sino un núcleo gravitacional. Puede verse la Web 2.0 como un conjunto de principios y prácticas alrededor del cual giran las aplicaciones, a mayor o menor distancia.

No es sencillo definir a la Web Social, al menos en aspectos prácticos. A veces, es complicado definir cuándo un sitio queda encuadrado dentro de la "Web 1.0" y cuándo puede considerarse miembro de pleno derecho de la Web Social. Basándose en el artículo de Graham (3), se puede llegar a la conclusión de que hay tres tendencias fundamentales en los sitios que considerados de Web 2.0: **"Ajax, democracia, y no tratar mal a los usuarios"**.

Con *Ajax* se refiere a que se ha hecho posible que las páginas Web se parezcan cada vez más a aplicaciones de escritorio. Considera que se está produciendo una generación nueva de software, de las que no había desde que aparecieron los primeros microcomputadores.

Graham hace gran hincapié en la democratización de la información que supone la Web 2.0. Dice que ha quedado demostrado que el trabajo de los aficionados puede superar el de los profesionales, cuando disponen del sistema adecuado para canalizar sus esfuerzos. El autor del artículo hace ver cómo un filtro de calidad compuesto por usuarios masivos puede ser mucho más eficiente que el filtro que pueda imponer un editor profesional. Por supuesto, no hay que olvidar que todos los contenidos publicados en la Web Social tienen mayor o menor calidad, pero aquellos que realmente valen la pena son los que prevalecen con el tiempo.

Siguiendo con (3), llama la atención la característica de "no tratar mal a los usuarios" a la que el autor dedica una sección completa del artículo. Compara los sitios que se crearon durante la burbuja tecnológica con los actuales sitios de Web 2.0. Las de aquel entonces tenían, según el autor, diseños insultantes, y transmitían el mensaje de "Esta es nuestra Web, no la tuya". Graham cree que la raíz del problema es que las compañías de aquel entonces tenían la sensación de estar dando algo gratis, lo que les hacía comportarse de forma arrogante. Podían llegar a pensar que cuanto más daño se hiciera al usuario (en forma de anuncios, registros, spam, etc), mejor. La Web 2.0 está cambiado todo eso. Las Webs ya no son de las compañías, sino de los propios usuarios, quienes generan los contenidos. El autor da una serie de consejos para todas las "startups" relacionadas con Web Social que aparezcan:

- No ser nunca despótico.
- No hacer que los usuarios se registren a menos que sea necesario.
- No pedir la dirección a menos que haga falta por alguna razón.
- No hacer preguntas innecesarias.
- No enviar e-mails a menos que los usuarios los pidan explícitamente.
- No abrir los links en frames o en otras ventanas.
- Si hay versión gratuita y de pago, no ser muy restrictivo.
- Pecar de generosidad en cuanto a lo que se permite a los usuarios.

Graham (3) resume su opinión en que Web 2.0 significa usar la Web en la forma en que debe ser usada. Las *tendencias* que se observan ahora, no son más que la naturaleza inherente a la Web, emergiendo de los viejos modelos que se impusieron durante la Burbuja Tecnológica.

Volviendo a (2), se puede ver cómo el padre del término Web 2.0 insiste en la importancia de los datos, o mejor dicho, del control de los mismos. En una plataforma en la que los usuarios generan datos que otros usuarios consumen, ¿Quién es el dueño? O'Reilly prevé que a medida que las empresas comiencen a darse cuenta de que el control sobre datos puede ser su principal fuente de ventaja competitiva, se podrán encontrar mayores intentos de control. También llega a la conclusión de que el control de una fuente de datos única y difícil de replicar, que se enriquezca con el uso de los usuarios, puede ser el secreto del éxito corporativo en los nuevos tiempos.

O'Reilly(2) da dos directrices para las empresas que quieran embarcarse en proyectos de Web Social:

- Las operaciones deben convertirse en una competencia central de la compañía (*core competence*). Esto significa que, si bien el mantenimiento ya era algo importante, ahora debe de ser lo más frecuente posible, llegando incluso a automatizarlo.
- Los usuarios deben ser tratados como codesarrolladores. Ya sea liberando el código, teniendo en cuenta la importancia de sus aportaciones de contenidos, o simplemente permitiendo un desarrollo abierto, en el que cualquiera pueda aprovechar los servicios en sus propios sitios (caso de Google Maps, cuyos mapas son usados como base en otros sistemas).

Los análisis previos ya hacían bastante obvia la segunda afirmación, pero la primera es sorprendente. El autor presenta el ejemplo de Google, al igual que hace en la mayoría de sus postulados. Si Google no rastreara continuamente la Web para actualizar sus índices, ni buscara sin descanso formas de evitar los trucos ilegales de posicionamiento, no habría llegado a sus niveles de calidad actuales. Como bien dice el autor, posiblemente su topología de red, sistema de administración, y algoritmos de balanceo, sean secretos mejor guardados que los propios algoritmos de búsqueda.

Una vez descrito lo que es el concepto de Web Social en sí, y las implicaciones empresariales que supone, se describirán las tecnologías que, junto a la evolución del uso de la Web, han desencadenado el paso a la Web 2.0.

Tecnologías de Web Social

A partir del artículo de O'Reilly (2), se puede deducir que las tecnologías que suelen estar presentes en los proyectos de Web Social son:

- Ajax
- Sindicación (RSS)
- Orientación a servicios

Conviene aclarar que la presencia de todas estas tecnologías en cierto sistema no implica necesariamente que pertenezca a la Web Social. Es más, no es necesario que todas estén presentes, ya que como dice O'Reilly (2), "**la Web 2.0 no es una tecnología, sino una actitud**". A continuación se profundizará en cada una de estas tecnologías, o conjunto de ellas.

Ajax

Según publica Jesse James Garret en su artículo *"Ajax: A new approach to Web Applications"* (4): "*Ajax no es una tecnología. Es realmente muchas tecnologías, cada una floreciendo por su propio mérito, uniéndose de poderosas nuevas formas*".

Ajax es el acrónimo de *Asynchronous JavaScript + XML*. En términos generales, consiste en enviar una petición directamente al servidor a través de JavaScript, pudiendo tratar la respuesta para cargar tan sólo partes de una página, sin necesidad de cargarla entera. El problema de la actualización parcial de contenidos ha sido un lastre para los programadores de aplicaciones Web. Ajax ha resuelto este escollo que separaba a las aplicaciones Web de los programadores de aplicaciones de escritorio mediante peticiones asíncronas.

Volviendo al artículo de Garret (4), se comprueba de forma esquemática lo que ofrece de nuevo Ajax:

- Presentación basada en estándares usando XHTML y CSS
- Exhibición e interacción dinámicas usando el Document Object Model.
- Intercambio y manipulación de datos usando XML y XSLT
- Recuperación de datos asincrónica usando XMLHttpRequest
- y JavaScript uniendo todo.

En lugar de cargar un pagina Web, al inicio de la sesión, el navegador carga el motor Ajax (escrito en JavaScript y habitualmente metido en un frame oculto). Este motor es el responsable de renderizar la interfaz que el usuario ve, y comunicarse con el servidor en nombre del usuario.

El motor AJAX permite que la interacción del usuario con la aplicación suceda asincrónicamente (independientemente de la comunicación con el servidor). Así el usuario nunca estará mirando una ventana en blanco del navegador, esperando que el servidor haga algo.

Cada acción del usuario que normalmente generaría una petición HTTP, se transforma en una llamada JavaScript al motor Ajax. Cualquier respuesta a una acción de usuario que no necesite un viaje de vuelta al servidor (como una simple validación de datos, editar datos en memoria, e incluso alguna navegación) es manejada por el

motor por su cuenta. Si el motor necesita algo del servidor para responder, hace esa petición asíncronamente, normalmente mediante XML de forma transparente al usuario.

Como se adelantó unos párrafos más atrás, Ajax consigue que las aplicaciones Web se parezcan cada vez más a las aplicaciones clásicas de escritorio, y puedan funcionar como tales (3). Si la Web Social pretende crear un entorno amigable para los usuarios, es decir, que sientan que "están en casa", Ajax puede ser el primer paso para conseguirlo.

Sindicación (RSS)

La sindicación (o más correctamente "redifusión") de información en Web consiste en la transmisión de información desde una fuente de la misma hasta otros entornos, que se convierten a su vez en emisores al ofrecerla a más usuarios.

Los dos formatos que se usan más frecuentemente para la sindicación Web son RSS y Atom. En este documento se describirá RSS por ser el único que ha sido especialmente referenciado en las fuentes estudiadas sobre Web Social hasta el momento.

Según Mark Pilgrim(5): *"RSS es un formato para sindicar noticias y el contenido de sitios que las publiquen. Estos sitios pueden ser muy diferentes, desde grandes agencias de noticias, hasta blogs. Puede sindicarse cualquier cosa que pueda ser dividida en elementos textuales discretos. Una vez que la información de cada elemento está en formato RSS, un lector de RSS puede comprobar los cambios en el feed RSS del modo adecuado, pudiéndose publicar la información en multitud de sitios heterogéneos de forma simultánea".*

Existen varias versiones de RSS, aunque las más frecuentes, hoy día, son la 1.0 y la 2.0. La 1.0 está basada en RDF, mientras que la 2.0 es un formato por sí mismo, tan sólo basado en XML. Bajo estas líneas, se adjunta una tabla comparativa de las versiones de RSS extraída de (5):

Versión	Propietario	Ventajas	Estatus	Recomendación
0.90	Netscape		Obsoleto por 1.0	No usar
0.91	UserLand	Muy simple.	Oficialmente obsoleta por 2.0, pero todavía bastante popular.	Usar para sindicación básica. Fácil de migrar a 2.0 si se necesita más flexibilidad.
0.92, 0.93, 0.94	UserLand	Permite metadatos más ricos que 0.91	Obsoleta por 2.0	Usar 2.0 en su lugar
1.0	RSS-DEV Working Group	Extensibilidad basada en RDF vía módulos, no controlada por una única distribuidora.	Núcleo estable, desarrollo activo de módulos.	Usar para aplicaciones basadas en RDF o si necesitas módulos avanzados específicos de RDF.
2.0	UserLand	Extensibilidad vía módulos, ruta fácil de migración desde 0.9x	Núcleo estable, desarrollo activo de módulos.	Usar para sindicación rica en metadatos de propósito general.

Tabla 1 Comparativa de versiones RSS y recomendaciones

De esta tabla se deduce que, hoy en día, conviene centrarse en generar sindicación para las versiones de RSS 1.0 y 2.0, por ser las que se usan de forma más

generalizada. Aunque la 0.91 siga usándose, el estar "oficialmente obsoleta" en el momento de la redacción del artículo fuente, le da pocas garantías de futuro.

La sindicación ha permitido que cualquier usuario sea avisado de actualizaciones en los sitios Web que elija, sin necesidad de visitarlos uno a uno. Según (2), es el avance más significativo en la evolución desde la arquitectura de la Web 1.0 a la 2.0.

Orientación a servicios

La *"Guía breve de los Servicios Web"*(6), de la W3C, define los Servicios Web como *"conjunto de aplicaciones o de tecnologías con capacidad para interoperar en la Web. Estas aplicaciones o tecnologías intercambian datos entre sí con el objetivo de ofrecer unos servicios. Los proveedores ofrecen sus servicios como procedimientos remotos y los usuarios solicitan un servicio llamando a estos procedimientos a través de la Web"*. En la misma fuente, se indica que *"[…] proporcionan mecanismos de comunicación estándares entre diferentes aplicaciones, que interactúan entre sí para presentar información dinámica al usuario. Para proporcionar interoperabilidad y extensibilidad entre estas aplicaciones, y que al mismo tiempo sea posible su combinación para realizar operaciones complejas, es necesaria una arquitectura de referencia estándar".*

En el paradigma clásico, o la Web 1.0, las aplicaciones eran entes independientes completos, donde las interacciones con otras eran fruto de un diseño que lo buscase expresamente. Las arquitecturas de cada una eran herméticas hacia el exterior, y el usuario final sólo podía aprovechar lo que ofreciera cada una, y de la forma en que lo ofreciera.

La Web 2.0 busca la reutilización de elementos ya existentes (servicios). Si una compañía ya proporciona unos datos, o un tratamiento efectivo de los mismos ¿Por qué volver a desarrollarlo? La reutilización siempre ha sido uno de los pilares del desarrollo, pero solía reducirse al ámbito de una misma compañía o equipo. La orientación a servicios es mucho más que desarrollar un elemento software para que otros lo añadan: es dejar abiertas partes de unos sistemas, para que otros desarrolladores las añadan al suyo, y consigan una innovación beneficiosa mediante el ensamblado de servicios Web.

O'Reilly habla del ensamblado de servicios en estos términos (2):

"Cuando los componentes de tipo materia prima son abundantes, se puede crear valor simplemente ensamblándolas de forma novedosa o eficaz. Al igual que la revolución del PC proporcionó muchas oportunidades para la innovación en el ensamblado del hardware de tipo commodity (con compañías como Dell, que hizo de dicho ensamblando una ciencia, derrotando a compañías cuyo modelo de negocio requería de innovación en el desarrollo del producto), creemos que la Web 2.0 ofrecerá oportunidades a las empresas para superar a la competencia, siendo mejores en el aprovechamiento y la integración de los servicios proporcionados por otros".

Más adelante añade: "*Las aplicaciones Web 2.0 se construyen a partir de una red de servicios de datos que cooperan. Por tanto: ofrezca interfaces de Web Services*

y sindicación de contenidos, y reutilice los servicios de datos de otros. Apoye los modelos de programación ligeros que permitan sistemas débilmente acoplados".

En otras palabras, O'Reilly aconseja que las arquitecturas de las aplicaciones sean muy desacopladas, y que se permita que cualquier sistema pueda aprovechar servicios creados previamente de una forma sencilla.

La cuestión clave viene definida en cuanto a qué beneficio puede brindar, a los creadores de un servicio, el que sea aprovechado por otros (y más si es de forma gratuita). En este punto, es necesario volver a citar a O'Reilly(2) cuando dijo que *"el control de una fuente de datos única y difícil de replicar, que se enriquezca con el uso de los usuarios, puede ser el secreto del éxito corporativo en los nuevos tiempos"*. Si se ofrece un servicio que usan otras personas; aunque no dé un beneficio directo, da el control del mismo; este control puede suponer una ventaja estratégica importante.

Los dos tipos de Servicios Web más extendidos hoy día son:

- Servicios Ligeros (REST)
- Servicios basados en SOAP (*"Web Services"*)

Los servicios REST son cronológicamente los primeros. Se basan en un mero intercambio de mensajes HTTP con un servidor, sin seguir un estándar específico. Sus características principales es que se usan en comunicaciones cliente-servidor y que no mantienen estado; es decir, cada petición es independiente de otras.

Los servicios basados en SOAP son a los que se refiere la gente comúnmente cuando habla de Web Services o Servicios Web. Las características funcionales, en cuanto a comunicaciones cliente-servidor, y falta de estado, son similares a los servicios REST, con la aportación de un estándar para los mensajes y su implementación.

Se podría pensar que los servicios REST son adecuados para comunicaciones que no tengan valor empresarial, y los Web Services para aquellos que precisen de mayor seguridad o control del flujo de información.

Ejemplos de aplicaciones de Web Social

Una vez definida la Web Social, o Web 2.0, y adquiridas ciertas nociones sobre las tecnologías más habituales que emplean, se estudiarán algunos ejemplos de aplicaciones que podrían considerarse que pertenecen a este género.

Se analizarán algunos de los buques insignia de la Web Social: Wikipedia y Youtube, además de un sitio Web con un propósito más o menos cercano al del Servidor online de Exámenes, con mucha experiencia y cuyo nombre habla por sí solo: "El rincón del vago".

De cada ejemplo estudiado, se hablará sobre su aspecto general, el nivel de colaboración del usuario y las tecnologías que usa. Se añadirá el ranking que les da Alexa y su *pagerank* de Google como indicativo de su popularidad.

Wikipedia

La Wikipedia se define a sí misma como *"[…] una enciclopedia libre y políglota basada en la colaboración de sus contribuyentes por medio de la tecnología wiki*".

Ilustración 2 Imagen de la Wikipedia en castellano

Es un proyecto que se puso en marcha durante el año 2001, y cuyo objetivo es construir y mantener una enciclopedia libre gracias a la colaboración de los propios usuarios. Su idiosincrasia le permite estar actualizada casi inmediatamente con todos los acontecimientos que van teniendo lugar, a diferencia de las enciclopedias clásicas, que dependían de un grupo de editores. Por otro lado, estas mismas características hacen que la información que publica no sea siempre fiable.

El aspecto que presenta la aplicación es limpio y amigable. Es sencillo encontrar lo que se busque, y la estrecha interrelación entre todos los artículos permite ir pasando del tema inicial que se buscó a otros relacionados, siguiendo los enlaces. La financiación completa corre a cargo de donaciones, de modo que no existe ningún elemento publicitario más allá del anuncio de las propias donaciones.

Título	*Wikipedia*
URL	*http://wikipedia.org*
Versión WAP	*Sí*
Ranking Alexa	*8*
Pagerank Google	*9*
Idioma	*Aplicaciones distintas para cada idioma*
Código abierto	*Sí. El motor es descargable y se puede usar.*
Acceso libre	*Sí*
Publicación libre	*Sí*
Servicios web	*Búsqueda en portales con misma tecnología*
Ajax	*No*
Sindicación	*RSS, Atom*
Más información	*Estándar propio para publicación de artículos (wikimedia)*

Tabla 2 Análisis de la Wikipedia

Aunque la Wikipedia no use Ajax, ni haga un uso extendido de servicios web, no se puede dudar que pertenece al mundo de la Web Social. Ofrece sindicación, pero está restringida a los historiales de los artículos. Es un claro ejemplo de una aplicación que pertenece a la Web 2.0 no por las tecnologías que usa, sino por su propósito y forma de empleo.

Youtube

Es el portal de videos por excelencia. Un sitio donde cualquiera puede publicar sus videos para que otros usuarios los vean (siempre que no infrinjan las reglas básicas de Youtube).

Ilustración 3 Portada de Youtube en castellano

Inicialmente fue fundado por tres ex empleados de Paypal en 2005. La compañía fue creciendo hasta que fue comprada por Google. Las claves de su éxito residen en que se evita la necesidad de descargarse un video, o enviarlo mediante correo electrónico. Al difundirse por streaming, y ser convertidos a un formato flash, la duración y calidad del original es prácticamente irrelevante.

Youtube no sólo está encaminado al ocio, sino que muchas empresas ya lo han adoptado como escaparate; incluso se usa para hacer campaña política. Un video publicado en Youtube puede ser visto por millones de personas en poco tiempo.

El interfaz gráfico es la sencillez elevada a la máxima potencia. Si no se encuentra lo que se busca a la primera, o en los videos relacionados, seguramente es que no haya sido publicado. La integración con Google tras la compra del portal original ha ayudado a mejorar el sistema de búsqueda de vídeos.

Como suele ser habitual en las aplicaciones de Google, no hay publicidad visible, o al menos molesta, en ninguna de las páginas de Youtube. Tan sólo aparece la típica publicidad interna de otros productos de Google.

Título	*Youtube*
URL	*http://www.youtube.com*
Versión WAP	*Sí*
Ranking Alexa	*3*
Pagerank Google	*9*
Idioma	*Aplicación única multi-idioma.*
Código abierto	*No*
Acceso libre	*Sí*
Publicación libre	*Usuarios registrados*
Servicios Web	*Reproductor incrustado de videos en otras páginas*
Ajax	*Sí*
Sindicación	*RSS*
Más información	*Integración con servicios y usuarios Google*

Tabla 3 Análisis de Youtube

Es uno de los ejemplos más claros y populares de aplicación de la Web Social. No sólo usa las tecnologías que suelen llevar asociadas, sino que su propio fin es el de compartir información entre los usuarios.

Aunque no sea de código abierto, su sencillez hace muy fácil indagar en su funcionamiento. Un ejemplo de esto es que existen aplicaciones que son capaces de llegar hasta el archivo de vídeo y permitir al usuario que lo descargue, aunque Youtube no lo permita, o mejor dicho, “no lo ofrezca”.

Rincón del vago

Es una de las aplicaciones con más solera en el ámbito de la Web en castellano. Pone a disposición de los usuarios una extensa base documental orientada a trabajos, apuntes, exámenes, y otros materiales docentes.

Ilustración 4 Portada del Rincón del Vago.

Fue creada en 1998 por dos jóvenes de Salamanca, con la intención de que sirviera para que cualquier alumno tuviera a su disposición los trabajos y apuntes que otros alumnos pudieran prestarle. Más tarde pasó a formar parte de la red de portales de Orange.

Desde su creación, la polémica ha acompañado a este sitio web. Sus detractores siempre han esgrimido que facilita, incluso promueve, la copia de trabajos. Por otro lado, sus defensores afirman que tan sólo es un sitio donde documentarse.

El interfaz gráfico deja bastante que desear. Puede resultar confuso en ocasiones, y el abuso en el empleo de publicidad reduce sensiblemente el aprovechamiento óptimo del espacio. En algunos puntos cuesta distinguir qué parte es publicidad, y cuál es contenido.

La clasificación de la información se basa en el Sistema Educativo Español, de modo que dificulta su aprovechamiento para estudiantes de otras naciones. Esta propia clasificación tiende a ser un poco confusa en algunas secciones, de modo que es casi obligatorio el uso del buscador. Aún así, no se garantiza que el elemento obtenido mediante búsqueda sea relevante.

Título	*Rincón del vago*
URL	*http://www.rincondelvago.com*
Versión WAP	*No*
Ranking Alexa	*736*
Pagerank Google	*5*
Idioma	*Interfaz en castellano, contenido multingüe.*
Código abierto	*No*
Acceso libre	*Sí*
Publicación libre	*Usuarios registrados*
Servicios Web	*No*
Ajax	*No*
Sindicación	*No*
Más información	*Incluye secciones de ocio*

Tabla 4 Análisis del Rincón del Vago

Clasificar “El Rincón del Vago” como una aplicación de Web Social sería bastante riguroso. Los contenidos son creados y usados por los usuarios, pero éste es el único elemento a su favor.

No usa ninguna de las tecnologías que suelen asociarse con las aplicaciones de Web 2.0. Teniendo en cuenta que lleva más de 10 años en línea, esto podría deberse a que en su fundación aún prevalecían los paradigmas de la llamada Web 1.0. Se podría decir que este es un ejemplo de un sitio de Web Social que se denomina así ateniéndose únicamente a sus principios.

.

Web Semántica.

A menudo, el concepto de "Web semántica" resulta confuso y difuso. Conviene definir, antes de comenzar, qué es exactamente. Según la *"Guía Breve de Web Semántica"* del W3C(7), *"La Web Semántica es una Web extendida, dotada de mayor significado en la que cualquier usuario en Internet podrá encontrar respuestas a sus preguntas de forma más rápida y sencilla gracias a una información mejor definida. Al dotar a la Web de más significado y, por lo tanto, de más semántica, se pueden obtener soluciones a problemas habituales en la búsqueda de información gracias a la utilización de una infraestructura común, mediante la cual, es posible compartir, procesar y transferir información de forma sencilla. Esta Web extendida y basada en el significado, se apoya en lenguajes universales que resuelven los problemas ocasionados por una Web carente de semántica en la que, en ocasiones, el acceso a la información se convierte en una tarea difícil y frustrante".*

Es decir, consiste en dotar en la Web de significado a la información publicada, para facilitar la tarea de los sistemas de recuperación y acceso a la información. Esta incorporación de semántica a los datos, se lleva a cabo mediante la inclusión de "meta-datos", es decir, "datos sobre los datos", que indican la información que, si bien un humano puede deducir del contexto, y la "lectura entre líneas", una máquina necesita que sean especificada. La comprensión de esos datos por parte de las máquinas, se basa en las definiciones y relaciones entre los datos y metadatos, y las propias búsquedas. *"No se trata de una inteligencia artificial mágica que permita a las máquinas entender las palabras de los usuarios, es sólo la habilidad de una máquina para resolver problemas bien definidos, a través de operaciones bien definidas que se llevarán a cabo sobre datos existentes bien definidos"* (7).

Tecnologías de Web Semántica.

Tras esta breve introducción sobre qué es exactamente la Web Semántica, se puede pasar a describir cómo funciona con mayor detalle. Se podría decir que por un lado están las tecnologías de definición de metadatos, y por otro, las tecnologías capaces de interpretar esos metadatos para definir y resolver problemas.

Según (7): *"La Web Semántica utiliza esencialmente RDF, SPARQL, y OWL […]"*. Este documento se centrará en estos tres lenguajes, por ser los que tienen mayor difusión. Del mismo modo, se encuentra un acercamiento conceptual a las tres tecnologías en (7). Cito textualmente:

- ***RDF*** *proporciona información descriptiva simple sobre los recursos que se encuentran en la Web y que se utiliza, por ejemplo, en catálogos de libros, directorios, colecciones personales de música, fotos, eventos, etc.*
- ***SPARQL*** *es lenguaje de consulta sobre RDF, que permite hacer búsquedas sobre los recursos de la Web Semántica utilizando distintas fuentes datos.*
- ***OWL*** *es un mecanismo para desarrollar temas o vocabularios específicos en los que asociar esos recursos. Lo que hace OWL es proporcionar un lenguaje para definir ontologías estructuradas que pueden ser utilizadas a través de diferentes sistemas. Las ontologías,*

> *que se encargan de definir los términos utilizados para describir y representar un área de conocimiento, son utilizadas por los usuarios, las bases de datos y las aplicaciones que necesitan compartir información específica, es decir, en un campo determinado como puede ser el de las finanzas, medicina, deporte, etc. Las ontologías incluyen definiciones de conceptos básicos en un campo determinado y la relación entre ellos.*

En resumen: RDF permite expresar conocimiento. SPARQL permite hacer consultas sobre ese conocimiento, y OWL permite definir las relaciones entre distintos términos del conocimiento mostrado. Es necesario remarcar que estas tres tecnologías son complementarias al pertenecer a ámbitos distintos (representación, consulta y estructuración).

A continuación, se entrará más en detalle en cada una de las tecnologías.

Resource Description Framework (RDF).

Según (8), *el "Resource Description Framework" es un marco de trabajo para representar información en Web.* Como se adelantó en el apartado anterior, se encarga lo que vendría a ser "mostrar la información".

El desarrollo de RDF ha sido motivado para varios usos, entre otros, proveer información sobre recursos Web, y los sistemas que los usan; permitir el desarrollo de aplicaciones que precisen modelos de información abiertos, hacer procesable la información para máquinas a nivel global, permitir la combinación de información de diferentes aplicaciones para obtener una nueva información, y hacer posible que los agentes software automaticen tareas de acceso a información. RDF viene a ser un lenguaje común, basando en XML, mediante el cual todos estos elementos pueden entenderse (8).

RDF está basado en un modelo gráfico de datos (8). La estructura de cada expresión está basada en un conjunto de tres elementos: un sujeto, un predicado, y un objeto. A su vez, un conjunto de estas ternas, forman un Gráfico RDF. La notación de estos elementos suele ser mediante dos nodos unidos por una flecha, que siempre apunta al objeto.

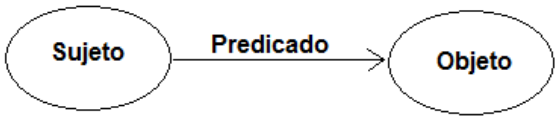

Ilustración 5 Representación gráfica de expresiones en RDF

La afirmación de una terna implica la afirmación de la relación expresada por el predicado, entre el sujeto y el objeto. Un gráfico RDF expresa la conjunción lógica de todas las expresiones que contiene.

Un ejemplo de expresión sería *"El análisis es una fase de desarrollo software"*. En este caso, el sujeto sería *"El análisis"*, el predicado *"es una"*, y el objeto *"fase de desarrollo software"*. Para representar todas las fases de desarrollo software, se necesitaría una terna por cada una de ellas. La conjunción de todas, expresaría la información que se quiere representar.

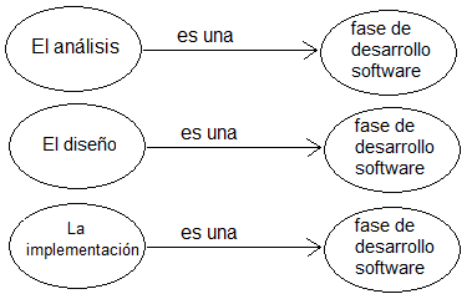

Ilustración 6 Ejemplo de gráfico RDF

Cada objeto puede ser a su vez el sujeto de otros predicados, o un "literal". Siguiendo con nuestro ejemplo, se podría tener la expresión *"Las fases de desarrollo software producen documentos"*; en la que *"Fase de desarrollo software"* pasaría a ser el sujeto. Si se construyera esa expresión, y la combinase con las anteriores, se vería que *"El análisis"*, es una *"fase de desarrollo software"*, y por el hecho de serlo, *"produce documentos"*. Es posible ampliar los nodos del gráfico RDF, incluyendo más expresiones, hasta conformar un vocabulario completo.

Existirían otras formas diferentes para la representación de la información que se han usado; por ejemplo, modificando las expresiones de forma que el objeto pase a ser el sujeto. Esto muestra que no existe una forma determinista de representar mediante RDF un vocabulario. Una mala elección de sujetos, predicados, u objetos, puede repercutir en que una representación no sea válida para todos los contextos. A la hora de elegir una representación RDF de un dominio, se han de valorar las diferentes opciones al alcance, y usar la que resulte más provechosa.

Para un ser humano, la comprensión del sujeto *"El análisis"*, o simplemente del predicado *"es una"*, no reviste mayor dificultad; sin embargo, una máquina necesita que se le explique qué es cada elemento. RDF resuelve esta cuestión a través de las referencias URI. Una referencia URI, o literal, usada como nodo, identifica qué representa el nodo. Una URI usada en un predicado representa la relación existente entre los elementos representados por los nodos que conecta. Un predicado URI podría ser a su vez un nodo en el gráfico(8). Cada URI identifica un recurso expresado por RDF.

En RDF se usan tipos (*datatypes)* para definir los posibles valores que puede tomar un nodo. Un tipo de datos consta de un espacio léxico, un espacio de valores, y un mapeo léxico-valor. Esto posibilita el referirse a un mismo elemento de diferentes formas. Según(8), la definición de tipos en RDF es compatible con los tipos en los XML Schemas. Cada elemento del espacio de valores puede estar relacionado con de 0 a N elementos del espacio léxico, mientras que cada elemento del espacio léxico está relacionado unívocamente con un elemento del espacio de valores.

La forma más clara de verlo es a través de un ejemplo: si se pretende representar los números naturales 1, 2 y 3 en binario y en decimal, el tipo de datos podría ser el que se muestra a continuación. Se ha forzado a dos bits para diferenciar la representación binaria de decimal en el 1.

Espacio de valores	{UNO, DOS, TRES}
Espacio léxico	{“1”, “2”, “3”, “01”,“10”,“11”}
Mapeo	{<”1”, UNO>, <”2”, DOS>, <”3”, TRES>, <”01”, UNO>, <”10”, DOS>, <”11”, TRES>}

Tabla 5 Ejemplo de datatype en RDF

Con este tipo de datos, se afirma que cada vez que se use DOS, se podrá estar hablando tanto del número decimal “2”, como del número binario “10”.

Los tipos de datos se definen de forma separada en RDF, y son referenciados en los grafos mediante URIs.

Para identificar como parte del léxico elementos como números o palabras, se usan literales. Cualquier elemento representado por un literal, puede ser a su vez representado por una URI, aunque (8) recomienda el uso de literales por ser a menudo más conveniente e intuitivo. Un literal puede ser el objeto de una expresión, pero nunca el sujeto o predicado.

Las ideas sobre el significado e inferencia en RDF se sostienen sobre el concepto formal de vinculación (*entailment*), tal y como se discute en (9). Una expresión RDF *vincula* a otra expresión RDF si todas las posibles combinaciones de cosas que hacen a la primera verdadera, lo hacen también a la segunda. Sobre esta base, si la verdad de la primera es presumida, o demostrada, esto puede transmitirse a la segunda.

Esta sección no pretende ser un manual de RDF, de modo que no se analizarán las vicisitudes de la sintaxis de este lenguaje. Para obtener información sobre la sintaxis, puede consultarse en (10) la recomendación de W3C sobre la misma.

RDF Vocabulary Description Language: RDF Schema.

Como se vio en la sección anterior, RDF es un lenguaje de propósito general para representar información en la Web. La especificación de RDF para definir vocabularios es RDF Schema (RDFS).

RDFS usa un sistema basado en clases y propiedades, similar al de la programación orientada a objetos(11), con la diferencia de que, en lugar de definir una clase en función de las propiedades que puedan tener sus instancias, RDFS define las propiedades según las clases de recursos a las que pueden aplicarse. Estos son los mecanismos de rol y ámbito(11). La cuestión es que las propiedades se aglutinan para formar clases, con unos ámbitos de aplicación. Básicamente, RDFS es una extensión semántica de RDF, que mediante recursos definidos con RDF permite la especificación de un vocabulario. Todas las clases son tipos de datos RDF (*datatypes*), mientras que las propiedades son predicados.

Los recursos pueden dividirse en grupos llamados clases(11), siendo llamados sus miembros *instancias*. Cada clase es, a su vez, un recurso que define sus propiedades mediante RDF, y suele identificarse con una URI. Para especificar que un recurso es una instancia de una clase, se usa la propiedad **rdf:type**.

Se denomina *extensión de la clase* al conjunto de instancias de una clase. Por ejemplo, en el ámbito de la Ingeniería del Software, podría existir la clase *"Participantes en el Análisis"*, cuya extensión podría estar formada (sin entrar en sutilezas) por *"Jefe de proyecto"*, *"Analista"* y *"Cliente"*. Una instancia puede pertenecer a más de una extensión de clase. Continuando con el ejemplo anterior, los elementos citados podrían pertenecer, a excepción de "*Cliente*", a la extensión de la clase "*Participantes en el Diseño*". Incluso puede darse el caso de que dos clases tengan exactamente la misma extensión de clases. El grupo de instancias de una clase se denota mediante la propiedad **rdfs:Class**(11).

Una clase puede contener subclases, que contendrán la propiedad **rdfs:subClassOf**. Del mismo modo, se usa el término "superclase" para referirse a propiedad inversa. La pertenencia a una clase es hereditaria de superclase a subclase. A modo de ejemplo, si "*Tareas del análisis"* es una subclase de *"Tareas de un proyecto"*, todas las instancias de la primera serán a su vez instancias de la segunda.

Todas las clases descritas por RDF son subclases de la clase **rdfs:Resource**, que a su vez es una instancia de **rsfs:Class**. Otras clases genéricas son **rdfs:Literal** (superclase de valores literales, como cadenas de texto o números), **rdfs:Datatype** (clase a la que pertenecen los tipos), **rdf:XMLLiteral** (clase de los literales con formato XML) y **rdf:Property** (clase de las propiedades). Se observa que todo el esquema de RDF para definir vocabularios, está definido a su vez en mediante RDFS.

A semejanza de las clases, las propiedades también pueden tener subpropiedades, existiendo una herencia en cuanto a los elementos que cumplen una propiedad (*"Si una propiedad P es una subpropiedad de la propiedad P', entonces todos los pares de recursos que están relacionados con P están también relacionados con P'."*(11)). A diferencia de las clases, no se ha definido una superpropiedad suprema de la que todas las propiedades sean subpropiedades.

En el árbol de clases y propiedades, que podría formarse mediante todas las jerarquías imaginables, aparecen los ámbitos (*ranges)* y dominios (*domains*) para seleccionar exactamente los elementos que formen parte de un vocabulario.

Para establecer que los valores de una propiedad son instancia de una clase, se usa la propiedad **rdfs:range**. La pertenencia a un ámbito establece que todos los elementos que sean el objeto en una expresión RDF con cierto predicado, pertenecen a la clase especificada.

Por ejemplo, la expresión:

"es revisado por" rdfs:range *"Responsables de documento"*

Afirma, según este hipotético (e inexacto) vocabulario de Ingeniería del Software, que todos los elementos "a la derecha" (los objetos) de las expresiones RDF que contengan el predicado *"es revisado por"*, pertenecen a la clase *"Responsables de documento".*

Dicho de forma gráfica, esta terna:

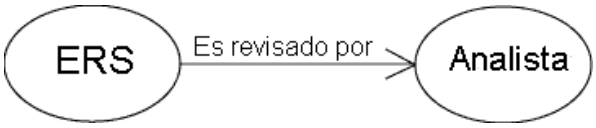

Ilustración 7 Ejemplo para rango RDFS

Indica que *"Analista"* es una instancia de la clase *"Responsables de documento"* gracias a la definición del ámbito anterior.

Los dominios son semejantes a los ámbitos, con la diferencia de que, en vez de indicar la clase a la que pertenece el objeto, están relacionados con el sujeto. Se denota el dominio mediante la propiedad **rdfs:domain**.

Siguiendo con el ejemplo anterior, se podría expresar lo siguiente:

"es revisado por" rdfs:domain *"Documentos"*

Lo que indica que todos los elementos que "son revisados por alguien", pertenecen a la clase "*Documentos*". Si se hubiera establecido este dominio, la Ilustración 7 diría que la ERS (especificación de requisitos software) es una instancia de la clase "*Documentos*".

Es importante destacar que RDFS únicamente define las relaciones entre diferentes elementos de un vocabulario, y limitaciones en cuanto a tipos o propiedades, pero no establece cómo deberían ser usadas estas relaciones(11). Dicho de otra forma, tan sólo da información gramatical, sin entrar en la semántica.

SPARQL Language for RDF.

SPARQL está diseñado para expresar consultas a través de diversas fuentes de datos almacenados como RDF o traducidos a RDF. SPARQL permite consultar patrones requeridos y opcionales a través de conjunciones y disyunciones. Los datos pueden mostrarse como conjuntos de resultados o grafos RDF.

La mayoría de las consultas en SPARQL contienen un conjunto de patrones triples (patrón de grafo básico). Los patrones triples son como las ternas de RDF, exceptuando que cada sujeto, predicado, y objeto, puede ser una variable. Un patrón básico de grafo coincide con un subgrafo de los datos RDF cuando los términos RDF de dicho subgrafo pueden ser sustituidos por las variables, y el resultado es un grafo RDF equivalente al subgrafo. Esto significa que fijando el sujeto, predicado, objeto, o cualquier combinación de estos elementos, la consulta devolverá el subconjunto, o subconjuntos, de datos del grafo que coinciden con ese patrón(12).

Por ejemplo, si se dispusiera de unos datos RDF sobre ingeniería del software, en el que se hubieran definido todos los documentos que produce cada fase de desarrollo, y se pretendiera saber los documentos que produce la fase de "*análisis"*, la consulta podría ser la siguiente:

SELECT ?v WHERE { "análisis" "produce" ?v}

Se observa la estructura de una consulta típica. Tras la palabra "select", se indican los elementos a mostrar en los resultados (en este caso, la variable v). En la cláusula *where*, se especifican las restricciones en este orden: sujeto, predicado, objeto. Suponiendo que el grafo contenga ternas del tipo fase – produce – documento, fijando la fase a "análisis", y el predicado a "produce", tan sólo se deja variable el objeto (el documento producido), de modo que los resultados obtenidos corresponderán exactamente con los documentos producidos durante la fase de análisis. Del mismo modo, se podría consultar qué fase de desarrollo produce el documento que se llame "ERS" con esta consulta:

SELECT ?v WHERE { ?v "produce" "ERS"}

O consultar todos los documentos producidos por todas las fases:

SELECT ?x ?y WHERE { ?x "produce" ?y}

De esta forma, se puede probar a fijar todos los elementos necesarios para obtener los resultados deseados. Se permite no fijar ningún elemento, aunque técnicamente no tendría sentido consultar "todo lo que está relacionado de alguna forma con todo".

Con total seguridad, limitar las búsquedas a coincidencias exactas en los literales será insuficiente. Por este motivo, SPARQL permite el uso de expresiones regulares a través de los filtros. Para ver un ejemplo, se supondrá que nuestro lenguaje de Ingeniería del Software incluye una definición de los roles existentes en un proyecto, siendo cada expresión de la forma (*nombre del rol) (gestiona) (ámbito)*. Si se quiere saber qué ámbitos gestionan cada uno de los roles que comienzan por la palabra "jefe", la consulta sería la siguiente:

SELECT ?rol WHERE {?rol “gestiona” ?ambito

FILTER regex (?rol, “^jefe”) }

La sintaxis de las expresiones regulares es la misma que se usa para XQuery y XPath, y está basada en las XML Schema Regular Expressions(12). Por supuesto, se puede especificar en qué lenguaje ha de consultarse un literal. Para definir que se habla de “phase” en el lenguaje *ois* (una hipotética Ontología de Ingeniería del Software) codificándolo de la siguiente manera: “phase”@ois.

Del mismo modo que se dispone de filtros para cadenas de texto, también existen para campos numéricos. Para filtrar una variable ?v, se usa en el WHERE el filtro FILTER (?v> 10000).

Para que la salida, en vez de mostrar un conjunto de resultados, devuelva un grafo RDF, tan sólo hay que sustituir la palabra SELECT por CONSTRUCT. También es posible que la salida sea un grafo RDF descriptivo de la solución; en este caso habría que usar DESCRIBE. Si tan sólo se pretende saber si existen resultados a una consulta, se usa ASK.

Otra de las ventajas de SPARQL es que permite que las consultas incluyan patrones o datos opcionales(12). Esto permite obtener datos en el resultado aunque la expresión RDF no tenga disponible este dato; de otro modo, los datos se descartarían para el resultado.

Si se usan patrones opcionales, los datos se mapearán a un grafo RDF siempre que estén disponibles, si no lo estuvieran, no llegará a hacerse el binding, pero no se descartarán como parte de los resultados.

Continuando con el ejemplo de los roles en un proyecto de ingeniería del software, se podría querer obtener información sobre los roles que participan en la implementación, y a quién supervisa cada uno. Si no se usaran patrones opcionales, los becarios no aparecerían en los resultados, al no supervisar a nadie. La forma de hacer la consulta sería la siguiente:

SELECT ?rol ?supervisado

WHERE { ?rol “participa en” “implementacion”

OPTIONAL { ?rol “supervisa” ?supervisado} }

Por supuesto, es posible incluir tantas cláusulas opcionales como se quiera, y filtros de igual modo que en unas restricciones normales.

A parte de restricciones en la consulta, se permite especificar la forma de mostrar los resultados. Están definidas las siguientes:

- **ORDER BY:** permite que los resultados sigan un orden basado en el valor de una o varias variables. Se puede especificar si la ordenación será ascendente o descendente (ASC(?nombre_variable) y DESC(?nombre_variable)) aunque por defecto lo hace de forma ascendente.

- **DISTINCT:** elimina los resultados repetidos (SELECT DISTINCT ?nombre_variable… WHERE…)
- **REDUCED:** elimina algunos de los resultados repetidos (SELECT REDUCED ?nombre_variable… WHERE…)
- **LIMIT:** reduce el número de resultados mostrados a los indicados por el límite (SELECT … WHERE {…} LIMIT *i*)
- **OFFSET:** los resultados se mostrarán a partir del i-ésimo, que no será incluido (SELECT … WHERE {…} OFFSET *i*).

Todas las restricciones de resultados pueden combinarse entre sí. Por ejemplo, para saber cuál es el tercer rol de Ingeniería del Software que participa en el análisis, ordenado alfabéticamente por el nombre, la consulta sería la siguiente:

```
SELECT ?rol

WHERE {?rol “participa en” “análisis”}

ORDER BY ?rol OFFSET 2 LIMIT 1
```

Existen muchas más consideraciones y particularidades sobre SPARQL, pero quedan fuera del ámbito de este estudio sobre la Web Semántica. Para más información, pueden consultarse las referencias.

Web Ontology Language (OWL).

Según (13): *"El Lenguaje de Ontologías Web (OWL) está diseñado para ser usado en aplicaciones que necesitan procesar el contenido de la información en lugar de únicamente representar información para los humanos"*. Si bien RDF era el lenguaje que permitía la descripción de recursos, OWL permite la definición de vocabularios complejos sobre ámbitos concretos (ontologías).

Antes de definir el lenguaje de ontologías, se debe puntualizar qué es en sí una ontología. En (14) se define de la siguiente forma: *"Una ontología define los términos a utilizar para describir y representar un área de conocimiento. Las ontologías son utilizadas por las personas, las bases de datos, y las aplicaciones que necesitan compartir un dominio de información (un dominio es simplemente un área de temática específica o un área de conocimiento, tales como medicina, fabricación de herramientas, bienes inmuebles, reparación automovilística, gestión financiera, etc.). Las ontologías incluyen definiciones de conceptos básicos del dominio, y las relaciones entre ellos, que son útiles para los ordenadores [...]. Codifican el conocimiento de un dominio y también el conocimiento que extiende los dominios. En este sentido, hacen el conocimiento reutilizable"*.

De este modo, OWL se encarga de enfocar los recursos definidos mediante RDF hacia la especificación completa de un vocabulario sobre un dominio específico.

Anteriormente, existían otros lenguajes de ontologías, pero eran desarrollados con objetivos muy concretos, alejados de la estandarización (14). El uso de RDF orientado a la especificación de ontologías permitió que las ontologías dispusieran de las siguientes capacidades (14):

- *Capacidad de ser distribuidas a través de varios sistemas*
- *Escalable a las necesidades de la Web*
- *Compatible con los estándares Web de accesibilidad e internacionalización*
- *Abierto y extensible*

Cuando se analizó la importancia de RDF en el apartado dedicado a dicho lenguaje, pudo parecer que bastaba para la definición de vocabularios; sin embargo, *"OWL extiende RDFS para permitir la expresión de relaciones complejas entre diferentes clases RDFS, y mayor precisión en las restricciones de clases y propiedades específicas Esto incluye por ejemplo: los recursos para limitar las propiedades de clases con respecto a número y tipo. Los recursos para inferir qué elementos que tienen varias propiedades son miembros de una clase en particular. Los recursos para determinar si todos los miembros de una clase tendrán una propiedad en particular, o si puede ser que sólo algunos la tengan. Los recursos para distinguir entre relaciones uno-a-uno, varios-a-uno o uno-a-varios, permitiendo que las "claves externas" de las bases de datos puedan representarse en una ontología. Los recursos para expresar relaciones entre clases definidas en diferentes documentos en la Web. Los recursos para construir nuevas clases a partir de uniones, intersecciones y complementos de otras. Por último, los recursos para restringir rangos y dominios para especificar combinaciones de clases y propiedades"*(14). En resumen, RDF es un lenguaje que se limita a exponer datos, usando una capacidad de expresión semántica muy reducida.

Además, como dice (13): *"La Web semántica se basará en la capacidad de XML para definir esquemas de etiquetas a medida y en la aproximación flexible de RDF para representar datos. El primer nivel requerido por encima de RDF para la Web semántica es un lenguaje de ontologías que pueda describir formalmente el significado de la terminología usada en los documentos Web. Si se espera que las máquinas hagan tareas útiles de razonamiento sobre estos documentos, el lenguaje debe ir más allá de las semánticas básicas del RDF Schema".* Esto quiere decir que, pensando en las herramientas de Web Semántica como una jerarquía de capas, RDFS se limitaría a exponer recursos, que son usados por OWL para dotarles de una semántica completa.

Una vez descrito lo que es una ontología, su utilidad, y la necesidad de crear un lenguaje estándar para definirla, se podrá pasar al análisis de OWL.

Según (13):*"OWL proporciona tres lenguajes, cada uno con nivel de expresividad mayor que el anterior, diseñados para ser usados por comunidades específicas de desarrolladores y usuarios"*. Estos lenguajes son OWL Lite, OWL DL y OWL Full. Sus principales diferencias radican en las restricciones que aplican a su vez sobre la especificación de OWL.

OWL Lite está pensado para ontologías con restricciones y clasificaciones jerárquicas simples. Al no ofrecer mucha complejidad, las herramientas que pueden procesar estos lenguajes son a su vez simples.

OWL DL (*Description Logics*) busca expresar mayor complejidad descriptiva sin ganar en complejidad computacional. Todas las conclusiones con computables y resolubles en tiempo finito.

OWL Full permite expresar relaciones más abstractas y mayor libertad sintáctica. (13) indica que: "*Es poco probable que cualquier software de razonamiento sea capaz de obtener un razonamiento completo para cada característica de OWL Full*".

Los lenguajes que pertenecen a cada rama de OWL están incluidos en la mayor, de este modo, un lenguaje de OWL Lite es a su vez OWL DL, y por lo tanto, también OWL Full. Las relaciones entre los distintos "dialectos" se expresan en el siguiente diagrama.

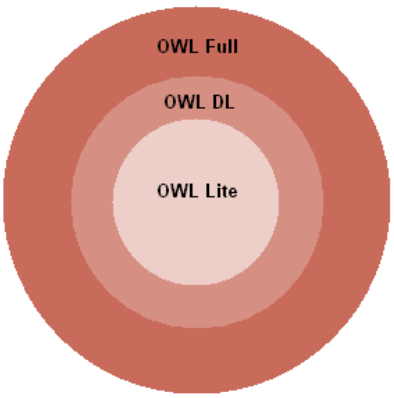

Ilustración 8 Relaciones de los lenguajes OWL

De las descripciones de los lenguajes OWL, se puede deducir que debe existir un compromiso, entre la riqueza de expresión que se busque en una ontología, y la capacidad de ser procesada por una máquina. Se podría hacer un símil con los distintos idiomas del lenguaje natural. Cuando más compleja es la gramática de una lengua, y más dada a los giros, permite una riqueza de expresión mayor, pero es más difícil de aprender para un hablante no nativo.

A diferencia del resto de tecnologías investigadas previamente, no se hará hincapié en la sintaxis y diferentes propiedades de OWL al no considerarse necesario para la comprensión de su utilidad y funcionamiento. Tan sólo es necesario tener en cuenta que extiende propiedades de RDFS, permitiendo la expresión de cardinalidades, además de relaciones lógicas entre propiedades (inversas, simétricas…). Por otro lado, permite el empleo de restricciones numerales (para todo, existe uno…), y operadores lógicos entre clases, tales como uniones o intersecciones. Para más información, se remite a (13).

Ontología para la Ingeniería del Software (SEOntology).

A lo largo del análisis de las diferentes tecnologías implicadas en la Web Semántica, se ha podido apreciar que todos los ejemplos iban encaminados al dominio de la Ingeniería del Software. Esto se debe a que será el dominio que se tomará como base para el desarrollo del Servidor on-line de Exámenes.

Una ontología de Ingeniería del Software típicamente provee conceptos de dicho dominio (qué son, cómo están relacionados, y cómo pueden relacionarse con otros) para representar y comunicar el conocimiento de Ingeniería del Software, e información de proyectos, a través de Internet (15).

Una buena práctica sería el desarrollo de una nueva ontología para la Ingeniería del Software, pero se trataría de un desarrollo complejo que formaría un proyecto completo por sí mismo. Se ha decidido basarse en una ontología ya desarrollada. En este apartado se analizarán los aspectos básicos de la *Software Engineering Ontology* (SEOntology), la primera ontología de Ingeniería del Software cuya implementación está disponible on-line (15).

SEOntology nació con el objetivo de proporcionar un lenguaje común para la compartición de conocimiento entre diferentes núcleos de un equipo de desarrollo involucrado en un proyecto de Ingeniería del Software (15)

La necesidad de este *lenguaje común* radica en la tendencia actual de la diversificación de entornos, durante el ciclo de desarrollo de los proyectos. No es raro que una empresa se dedique al análisis y diseño, y se encargue la implementación, o las pruebas, a una empresa externa, que no tiene por qué estar geográficamente próxima a la empresa inicial.

Por otro lado, SEOntology pretende que los conceptos teóricos de la Ingeniería del Software sean comprendidos y seguidos, ya que no es raro que cada empresa (incluso cada ingeniero del software) los adapte a sus necesidades o costumbres, provocándose inconsistencias en la documentación (15).

De forma esquemática, las razones para el desarrollo de SEOntology, según (16) son:

- Definir los conceptos de Ingeniería del Software de una forma más precisa, permitiendo un mejor procesamiento automático tal y como han evolucionado.
- Hacer explícitos los supuestos en el dominio de la Ingeniería del Software.
- Separar el conocimiento del dominio y subdominio del conocimiento de la instancia.
- Crear una referencia *multi-site* de desarrollo de software para las aplicaciones.
- Compartir una comprensión consistente del significado de la información de un proyecto de ingeniería.

El proyecto SEOntology no se ha limitado a la creación de la ontología en sí, sino que ha desarrollado una serie de aplicaciones que permiten la gestión de proyectos en entornos colaborativos basándose en la propia ontología (15).

El conjunto completo de conceptos de Ingeniería del Software se capturan como un dominio de conocimiento. Obviamente, no todos los proyectos usarán todos los conceptos reflejados en la ontología, sino que cada uno de ellos estará más cerca de uno de los subdominios (por ejemplo, la metodología de orientación a objetos). Los conceptos específicos del conocimiento se expresan como subdominios.

Cada instancia de conocimiento en SEOntology representa las particularidades y necesidades específicas de un proyecto. En este caso, los creadores han decidido separar completamente el dominio de conocimiento de las instancias de conocimiento (17). Esta separación permite a su vez delimitar por un lado el conocimiento teórico (conceptos, relaciones, subdominios, etc), y por otro los datos de proyectos, permitiendo diferentes instanciaciones en función del proyecto con el que se relacionen. Por ejemplo, pertenecería al dominio el concepto “Use_case”, y una de sus instancias sería cada uno de los casos de uso presentes el un proyecto.

Se han publicado dos niveles de abstracción: el genérico y el específico. El genérico representa los conceptos que son comunes al conjunto de conceptos de Ingeniería del Software, mientras que el específico representa el conjunto de conceptos que son específicamente usados para algunas categorías de proyectos particulares (18).

A través de SEOntology, un texto sobre Ingeniería del Software puede ser parseado para descubrir de forma dinámica y automática los conceptos relevantes y la relación entre ellos. Si además se especifican las instancias presentes en ese texto, se puede representar el mismo de forma esquemática, a través de los conceptos y relaciones de la ontología, para asegurar su comprensión y validez semántica (18).

Fijando la vista en el caso particular del Servidor on-line de Exámenes, se deduce que, gracias a SEOntology, podrá optimizarse el acceso a los recursos publicados, ya que la indexación podría seguir una estructura semejante a la ontología. Al mismo tiempo, podrá validarse automáticamente la relevancia de los contenidos publicables respecto al dominio de la Ingeniería del Software.

E-Learning.

Las nuevas tecnologías en general, y el uso de Internet en particular, ha ido penetrando de manera progresiva en cada una de las actividades cotidianas de las personas en los últimos años. Así, actividades tan cotidianas como hacer la compra, ir a los bancos, o comprar el periódico, se están sustituyendo por la compra on-Line, el uso de la banca electrónica o la visualización de contenidos en línea. Como no podía ser de otra manera, la educación también se ha visto "afectada" por la llegada de estas nuevas tecnologías.

Tal como describe D. Francisco José García Peñalvo, profesor de la Universidad de Salamanca, en su artículo "Estado actual de los sistemas de e-Learning" (19): el sector educativo ha encontrado en estas nuevas tecnologías un excelente medio para romper con las limitaciones geográficas y temporales que los esquemas tradicionales de enseñanza-aprendizaje conllevan, revolucionando, y cambiando a la vez, el concepto de educación a distancia. Su adopción y uso han sido amplios, lo que ha permitido un desarrollo rápido y consistente en el que la Web ha ido tomando distintas formas dentro de los procesos educativos.

La Web se convierte en la infraestructura básica para desarrollar los procesos de enseñanza-aprendizaje no presenciales, combinando servicios síncronos y asíncronos, lo que ha dado lugar a un modelo conocido como e-formación o e-learning, cada vez más valorado, no como sustituto de la formación presencial tradicional, sino más como un complemento que se ha de adaptar según las necesidades y nivel de madurez del público receptor de esta formación(20) que puede ir desde ser una actividad complementaria muy concreta y residual en los estudios de primaria y secundaria, a ser un modelo únicamente no presencial en la formación a distancia o formación continua empresarial. No obstante, las aproximaciones mixtas, que combinan actividades formativas presenciales y no presenciales, toman cada vez más fuerza y se posicionan como una importante alternativa ante los grandes retos que se afrontan con la integración del sistema universitario al nuevo Espacio Europeo de Educación Superior y el creciente peso de la formación a lo largo de toda la vida.

Para intentar definir el concepto anteriormente nombrado de e-learning y cuya descripción es el objetivo de este punto, de nuevo se hace referencia al artículo del profesor Peñalvo (19). Según el mismo, el concepto de e-learning se define de muchas formas diferentes fundamentalmente debido a que los actores que de él hacen uso son muy diversos, cada uno con su idiosincrasia y su ámbito de aplicación.

Desde la perspectiva de su concepción y desarrollo como herramienta formativa, los sistemas de e-learning tienen una dualidad pedagógica y tecnológica. Pedagógica en cuanto a que estos sistemas no deben ser meros contenedores de información digital, sino que ésta debe ser transmitida de acuerdo a unos modelos y patrones pedagógicamente definidos para afrontar los retos de estos nuevos contextos. Tecnológica en cuanto que todo el proceso de enseñanza-aprendizaje se sustenta en aplicaciones software, principalmente desarrolladas en ambientes Web, lo que le vale a estos sistemas el sobrenombre de plataformas de formación.

Desde la perspectiva de su uso se podría distinguir la visión que tienen sus usuarios finales, que con independencia de su madurez y formación, verán al sistema e-learning como una fuente de servicios para alcanzar su cometido formativo. No obstante, también es factible diferenciar una visión de organización, en la que se definen el alcance y los objetivos buscados con la formación basada en estos sistemas, distinguiéndose una visión académica y una visión empresarial.

Si se toma como referencia la raíz de la palabra, e-learning se traduce como "aprendizaje electrónico", y como tal, en su concepto más amplio puede comprender cualquier actividad educativa que utilice medios electrónicos para realizar todo o parte del proceso formativo.

Existen definiciones que abren el espectro del e-learning a prácticamente a cualquier proceso relacionado con educación y tecnologías, como por ejemplo la definición de la American Society of Training and Development que lo define como "*término que cubre un amplio grupo de aplicaciones y procesos, tales como aprendizaje basado en web, aprendizaje basado en ordenadores, aulas virtuales y colaboración digital. Incluye entrega de contenidos vía Internet, intranet/extranet, audio y vídeo grabaciones, transmisiones satelitales, TV interactiva, CD-ROM y más*".

Otros autores acotan más el alcance del e-learning reduciéndolo exclusivamente al ámbito de Internet, como Rosenberg(21) que lo define como: "*el uso de tecnologías Internet para la entrega de un amplio rango de soluciones que mejoran el conocimiento y el rendimiento. Está basado en tres criterios fundamentales:*

1. *El e-learning trabaja en red, lo que lo hace capaz de ser instantáneamente actualizado, almacenado, recuperado, distribuido y permite compartir instrucción o información.*
2. *Es entregado al usuario final a través del uso de ordenadores utilizando tecnología estándar de Internet.*
3. *Se enfoca en la visión más amplia del aprendizaje que van más allá de los paradigmas tradicionales de capacitación*".

Desde la perspectiva que ofrece la experiencia en el desarrollo y explotación de plataformas e-learning, es posible aventurarse a dar una definición propia de e-learning, como la capacitación no presencial que, a través de plataformas tecnológicas, posibilita y flexibiliza el acceso y el tiempo en el proceso de enseñanza-aprendizaje, adecuándolos a las habilidades, necesidades y disponibilidades de cada discente, además de garantizar ambientes de aprendizaje colaborativos mediante el uso de herramientas de comunicación síncrona y asíncrona, potenciando en suma el proceso de gestión basado en competencias.

En todas estas definiciones, así como en otras que se pueden encontrar en la bibliografía especializada, se acaba haciendo mención explícita o implícita a lo que se viene llamando en triángulo del e-learning(22), formado por la tecnología (plataformas, campus virtuales…), los contenidos (calidad y estructuración de los mismos se toman como elementos capitales para el éxito de una iniciativa de e-formación) y los servicios (siendo el elemento más variopinto que engloba la acción de los profesores, elementos de gestión, elementos de comunicación, elementos de evaluación…). Variando el peso de estos tres componentes se obtienen diferentes modelos de e-formación, de igual

forma que variando las variables y recursos con los que cuenta un profesor se obtienen diferentes políticas de docencia presencial.

En la práctica, para llevar a cabo un programa de formación basado en e-learning, se hace uso de plataformas o sistemas de software que permiten la comunicación e interacción entre profesores, alumnos y contenidos. Se tienen principalmente dos tipos de plataformas: las que se utilizan para impartir y dar seguimiento administrativo a los cursos en línea o LMS (Learning Management Systems) y, por otro lado, las que se utilizan para la gestión de los contenidos digitales o LCMS (Learning Content Management Systems).

Entre las herramientas más utilizadas para los ambientes o sistemas e-learning están los Sistemas de Administración de Aprendizaje o LMS, también ampliamente conocidos como plataformas de aprendizaje. Un LMS es un software basado en un servidor web que provee módulos para los procesos administrativos y de seguimiento que se requieren para un sistema de enseñanza, simplificando el control de estas tareas. Los módulos administrativos permiten, por ejemplo, configurar cursos, matricular alumnos, registrar profesores, asignar cursos a un alumno, llevar informes de progreso y calificaciones. También facilitan el aprendizaje distribuido y colaborativo a partir de actividades y contenidos preelaborados, de forma síncrona o asíncrona, utilizando los servicios de comunicación de Internet como el correo, los foros, las videoconferencias o el chat.

El alumno interactúa con la plataforma a través de una interfaz web que le permite seguir las lecciones del curso, realizar las actividades programadas, comunicarse con el profesor y con otros alumnos, así como dar seguimiento a su propio progreso con datos estadísticos y calificaciones. La complejidad y las capacidades de las plataformas varían de un sistema a otro, pero en general todas cuentan con funciones básicas como las que se han mencionado.

Los Sistemas de Administración de Contenidos de Aprendizaje o LCMS tienen su origen en los CMS (Content Management System) cuyo objetivo es simplificar la creación y la administración de los contenidos en línea, y han sido utilizados principalmente en publicaciones periódicas (artículos, informes, fotografías…). En la mayoría de los casos lo que hacen los CMS es separar los contenidos de su presentación y también facilitar un mecanismo de trabajo para la gestión de una publicación web. Los LCMS siguen el concepto básico de los CMS, que es la administración de contenidos, pero enfocados al ámbito educativo, administrando y concentrando únicamente recursos educativos y no todo tipo de información.

En esencia, se define entonces un LCMS como un sistema basado en web que es utilizado para crear, aprobar, publicar, administrar y almacenar recursos educativos y cursos en línea(23). Los principales usuarios son los diseñadores instruccionales que utilizan los contenidos para estructurar los cursos, los profesores que utilizan los contenidos para complementar su material de clase e incluso los alumnos en algún momento pueden acceder a la herramienta para desarrollar sus tareas o completar sus conocimientos.

Los contenidos usualmente se almacenan como objetos descritos e identificables de forma única. En un LCMS se tienen contenedores o repositorios para almacenar los recursos, que pueden ser utilizados de manera independiente o directamente asociados a la creación de cursos dentro del mismo sistema. Es decir que el repositorio puede estar disponible para que los profesores organicen los cursos o también pueden estar abiertos para que cualquier usuario recupere recursos no vinculados a ningún curso en particular, pero que les pueden ser de utilidad para reforzar los aprendido sobre algún tema.

El proceso de trabajo dentro de un LCMS requiere de control en cada fase del contenido, esto conlleva un proceso editorial para controlar la calidad de los contenidos creados, así como para permitir y organizar su publicación.

Siguiendo con el completo artículo del profesor Peñalvo (19), se puede concluir que los sistemas que promueven los procesos de enseñanza-aprendizaje a través de sistemas de e-learning tienen una gran importancia para consolidar la denominada Sociedad del Conocimiento. Estos medios abren la puerta para la formación básica o avanzada a una importante cantidad de personas, que pueden ver mejorada su cualificación personal o su situación profesional. Estos sistemas tienen un campo enorme de aplicación ya que la formación puede orientarse de forma complementaria a nivel de educación primaria y secundaria, de forma complementaria o exclusiva a nivel universitario, de postgrado o de formación continua, y de formación especial a medida en las empresas. No obstante, el campo del e-learning está en sus fases iniciales y le falta un largo camino por recorrer hasta alcanzar su madurez y consolidación.

Herramientas E-Learning

Actualmente se está dando un pleno auge de los sistemas de gestión de formación en línea, lo que supone que día a día vayan apareciendo nuevas herramientas tanto de libre distribución como de desarrollo privado.

Tras un arduo análisis de mercado de estas herramientas, se ha optado por profundizar en dos de las herramientas de libre distribución más extendidas hasta el momento como son Moodle y Claroline.

La elección de estas dos plataformas de e-learning para el presente estudio, se basa en la gratuidad de las mismas, la gran implantación que se está realizando en la actualidad, así como los amplios servicios de que disponen y que permitirán exponer la imagen típica de un LMS.

Para la realización de este análisis de las herramientas, se ha procedido a la implantación de las mismas en la máquina de desarrollo del proyecto “desarrollo.babieca.org”.

Moodle

Moodle, cuyo nombre viene del acrónimo inglés “*Modular Object-Oriented Dynamic Learning Environment*” (Entorno Modular de Aprendizaje Dinámico Orientado a Objetos), es un LMS desarrollado inicialmente en agosto de 2002 en la Universidad

Tecnológica de Curtin. Su creador, Martin Dougiamas; diseñó un sistema bajo una corriente de pensamiento que se ha denominado "pedagogía construccionista social" (24) que ha llevado a este sistema a convertirse uno de los entornos de e-learning más utilizados actualmente.

Resumiendo los aspectos técnicos muy brevemente, es necesario dejar constancia de que es un sistema desarrollado íntegramente en PHP abstrayéndose completamente del SGBD y utilizando una única BBDD para la persistencia de todos los datos.

Además de la filosofía de desarrollo del sistema, uno de los aspectos más importante de Moodle es la enorme funcionalidad y flexibilidad de la que está dotado el sistema. Dicha funcionalidad permite al administrador, tal como se verá posteriormente, configurar absolutamente todos los aspectos relevantes de un LMS así como crear una completa agenda de cada curso añadiendo recursos y actividades.

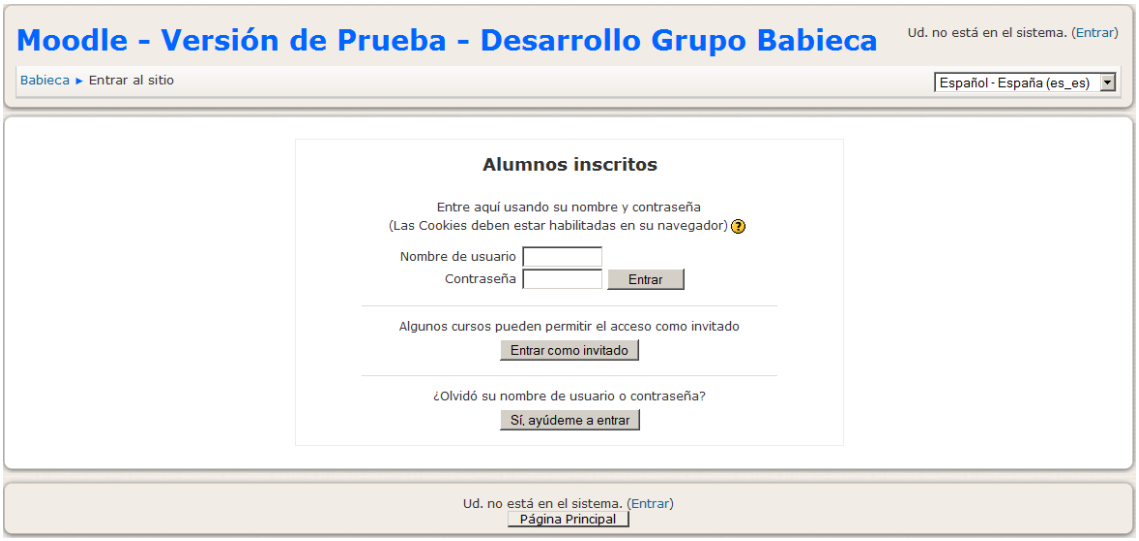

Ilustración 9 Interfaz de la Herramienta Moodle

La administración cuenta con una infinidad de opciones de configuración, sin embargo la exposición detallada de cada una de estas opciones recae fuera del ámbito de este proyecto. A continuación se enumerarán las opciones generales:

- Gestión de usuarios
- Gestión de cursos
- Administración de calificaciones
- Administración de ubicación e idioma.
- Administración módulos.
- Administración de seguridad.
- Administración de apariencia y portada.
- Administración del servidor.
- Administrar listas de enlaces.
- Administración de red.

- Generación de informes.

Dentro de la gestión y configuración de cada curso se encuentran las siguientes opciones:

- Configuración del curso.
- Asignación de roles.
- Gestión de Calificaciones.
- Gestión de grupos.
- Gestión de copias de seguridad.
- Administración de informes

Cada curso se encuentra estructurado a través de una agenda en la cual diariamente pueden ser añadidos recursos como etiquetas, páginas de texto y HTML, enlaces a documentos, directorios y paquetes con contenido IMS; así como actividades del tipo de Chat, consultas, cuestionarios, encuestas, foros, glosarios, lecciones y contenidos SCORM.

Para el análisis de esta herramienta se ha implantado en una máquina de desarrollo del proyecto accesible desde la URL http://moodle.desarrollo.babieca.org.

Tras la evaluación de cada una de las funcionalidades del sistema, se concluye que Moodle es uno de los LMS más completos, con mayor posibilidad de parametrización y todo bajo una filosofía *free software* lo que hace que sea uno de las herramientas e-learning más extendidas. Como puntos fuertes destacan: la adaptación a los estándares QTI y SCORM, que le permiten la importación de contenidos de una manera realmente sencilla a la vez que permitirán que los contenidos sean utilizados por otros. La disposición de los contenidos en forma de agenda dentro de cada curso, que permite al alumno tener una visión cronológica del propio curso de un solo vistazo. En lo referente a los recursos, cabe destacar la inclusión de BBDD, que permite la creación de manera totalmente dinámica de tablas de contenidos personales. Además de otros recursos típicos de un LMS se han incluido una gran variedad de actividades que el alumno puede ir realizando cada día. Entre estas actividades cabe destacar la inclusión de un Chat para una comunicación más ágil a modo de tutorías, así como foros y glosarios.

Para centrarse en el desarrollo posterior del proyecto, cabe destacar el completo sistema de evaluación mediante cuestionarios del que dispone el sistema. En él se contemplan gran variedad de tipos de preguntas como preguntas calculadas, descripciones, ensayos, emparejamientos, respuestas incrustadas, opción múltiple o respuestas cortas. Este sistema será analizado de una manera más completa posteriormente.

Claroline

El proyecto Claroline, cuyo logotipo es el rostro de Calíope, la musa griega de la poesía épica y la elocuencia; fue iniciado en el año 2000, en el Instituto Pedagógico Universitario de Multimedia de la Universidad Católica de Lovain (Bélgica). A él se han

ido uniendo una gran comunidad de profesores y desarrolladores y a partir del año 2004 el Centro de Investigación y Desarrollo, del Instituto Superior de Ingeniería Belga.

Técnicamente cabe destacar que la plataforma se encuentra escrita íntegramente en PHP y utiliza como sistema gestor de base de datos MySql, lo cual contribuye en gran medida a su amplia distribución.

Uno de los aspectos más interesantes de esta plataforma, es la implementación dentro de las especificaciones de SCORM e IMS. Tal como se verán en secciones posteriores, estas especificaciones marcan las bases de la accesibilidad, interoperabilidad y reusabilidad de los contenidos utilizados; además de ser uno de los principales objetivos de análisis del proyecto.

Ilustración 10 Interfaz de la Herramienta Claroline

Funcionalmente presenta las características propias de un sistema de gestión de contenidos (LMS). Las principales funcionalidades son:

- Gestión de cursos
- Gestión de agenda con tareas y plazos.
- Gestión de anuncios incluyendo el envío vía e-mail a usuarios registrados.
- Gestión de documentos y enlaces.
- Administración de cuestionarios y sistema de evaluación, incluyendo estadísticas de los mismos.
- Administración de secuencias de aprendizaje.
- Confección de ejercicios para que sean realizados por los estudiantes.
- Administrar foros de discusión tanto públicos como privados.
- Administrar listas de enlaces.
- Administración de grupos de estudiantes.
- Gestión los envíos de los estudiantes: documentos, tareas, trabajos, etc.
- Crear y guardar chats.
- Creación de un conocimiento compartido a través de “Wiki’s”.

Para el análisis de esta herramienta se ha implantado en una máquina de desarrollo del proyecto accesible desde la URL http://claroline.desarrollo.babieca.org.

Tras la evaluación de cada una de las funcionalidades, cabe destacar el sistema de gestión de cuestionarios que incluye, el cual permite la gestión de varios parámetros del cuestionario como son las fechas de realización, el tiempo, el número de intentos, etc. Además cuenta con diversos tipos de preguntas: elección simple, elección múltiple, rellenar huecos y relacionar. Por último en lo referente al sistema de evaluación, volver a hacer referencia en este punto a su adaptación al estándar QTI de IMS lo que le permite exportar los cuestionarios y ser interpretados por otro sistema.

Para finalizar el análisis de la herramienta Claroline hay que destacar también la gestión de ejercicios, que permite la configuración de varios parámetros como son: fechas de entrega, tipo de envío, ejercicio en grupo o individual, etc. Además permite al profesor la evaluación de cada uno de las entregas añadiendo la calificación y comentarios de los mismos.

Otras herramientas de e-learning

Dado que un mayor análisis de las herramientas de e-learning encontradas, recae fuera del alcance del estudio, se presenta la siguiente tabla a modo de resumen, donde puede encontrarse más información relativa a dichas herramientas.

Nombre	Referencia	Libre
.LRN	http://dotlrn.org/	x
ATutor	http://www.atutor.it	x
Authorware	http://www.adobe.com/	
Blackboard	http://www.blackboard.com/us/index.bbb	
Claroline	http://www.claroline.net/	x
Delfos LMS	http://franco-atigroup.com/delfos/	
Desire2Learn	http://www.desire2learn.com/	
Docebo	http://www.docebo.org	x
Dokeos	http://sourceforge.net/projects/dokeos	x
EKP	http://www.netdimensions.com/	
ILIAS	http://www.ilias.de	x
LON-CAPA	http://www.lon-capa.org/	x
Moodle	http://moodle.org/	x
SITEA	http://www.sitea.net/	
Skillfactory	http://www.skillfactory.com.mx/	
Vertice	http://www.verticelearning.com	

Tabla 6 Listado LMS's

Evaluación On-Line

No hay que olvidar que dentro del ámbito del e-learning, el presente proyecto se centra sobre el importantísimo proceso de evaluación on-line. Tal como expone Juan Carlos Lozano, en su artículo "Técnicas y Herramientas de Evaluación On-Line" (25): *"Evaluar es una de las etapas más importantes dentro del proceso de enseñanza-aprendizaje y no se debe confundir evaluación con calificación, ya que sólo es un aspecto más del proceso evaluativo que está relacionado con la valoración o notas finales y tiene una función exclusivamente acreditativa".*

Es decir, no sólo se evalúa al alumno para certificar el nivel de conocimientos adquirido a lo largo de todo el proceso, sino que también se evalúa para ofrecerle una retroalimentación sobre su aprendizaje, para que los docentes conozcan la efectividad de su actuación, para certificar los resultados, para evaluar la calidad de la metodología empleada, etc.

La formación online supera a otros tipos de formación tradicionales porque en ella se pueden evaluar, incluso en muchos casos de forma automática, los siguientes aspectos:

- La asistencia: se puede conocer el número de accesos, el tiempo empleado por los diferentes participantes de la acción formativa, etc. y esto puede servir para justificar las horas lectivas del curso.
- Las aportaciones: se puede conocer también el grado participación los participantes de la acción formativa (alumnos, docentes y coordinador), el número de mensajes enviados, intervenciones en los foros, etc.
- Los conocimientos: a través de técnicas e instrumentos de evaluación como las autoevaluaciones, ejercicios, exámenes, etc. se puede medir el grado de aprendizaje alcanzado por el alumno.
- El proceso formativo en su totalidad: se puede medir el grado de eficacia y eficiencia del curso, su atractivo, su usabilidad, etc.

La formación online cuenta con unas posibilidades casi ilimitadas para realizar la evaluación. Tanto en las plataformas como en los contenidos online se pueden incluir herramientas de evaluación interactivas y dinámicas que ofrecen por un lado, un feedback inmediato al alumno sobre los resultados alcanzados, y por otro lado, permiten a los gestores de la formación disponer de datos cuantitativos generados automáticamente por el sistema, que facilitan enormemente la tarea de evaluar.

A pesar de estas facilidades tampoco habrá que descartar en muchos casos la intervención y juicio de un docente o tutor que evalúe aspectos cualitativos, a través de la actuación del alumno en los distintos contextos y por supuesto mediante pruebas de toda la vida, que implique una mayor elaboración por parte del alumno, como por ejemplo, un proyecto fin de curso.

De este modo, aparecen técnicas objetivas (valoración cuantitativa), técnicas subjetivas (valoración cualitativa) y técnicas mixtas.

Siguiendo con la clasificación del profesor Lozano, se encuadran en técnicas objetivas los cuestionarios y ejercicios interactivos del tipo de: preguntas de elección múltiple, doble alternativa, asociación de parejas, rellenado de huecos, ordenación de elementos, identificación de términos y clasificación de elementos. Otras técnicas objetivas son juegos interactivos, como son los crucigramas, sopa de letras y puzles.

En el otro lado, serían técnicas subjetivas: la exposición oral y redacción escrita, resolución de problemas, dinámica de grupos, representación de roles, casos prácticos, así como el proyecto fin de curso.

Para obtener una descripción de estas técnicas puede consultarse el artículo de Juan Carlos Lozano en su totalidad (25).

Actualmente están empezando a proliferar gran cantidad de herramientas de pago incluso algunas de código libre y/o libre distribución para la evaluación on-line. A continuación se expondrán de manera muy breve una pequeña representación de las herramientas encontradas que han sido elegidas como representación de los múltiples sistemas analizados:

Autor Vértice

La Herramienta de Autor(26) es un software desarrollado por vértice e-learning cuya última versión es la 2.1. Autor es compatible con la versión 1.2 de SCORM por lo que se podrán generar y exportar contenidos bajo esta especificación. El sistema dispone de editores de contenidos, esquemas flash y evaluaciones integrados. Además cuenta con una sencilla edición de páginas así como el uso de plantillas para la generación de contenidos.

Perception Scantron

Perception(27) es la herramienta desarrollada por Scantron para la creación de test y encuestas a través de Internet y que completa su gama de software de evaluación. Perception cuenta con gran variedad de preguntas, configuraciones de test, opciones de seguridad e incluso opciones de reporte. Todas estas características se encuentran en la página oficial del sistema(27). Además de Perception cabe la pena destacar el software de generación de exámenes ExamView de Scantron que permite generar exámenes automáticamente de 2000 bancos de preguntas.

CrearTest.com

CrearTest.com(28) es una iniciativa online para la creación y realización de test, a través de Internet. Si bien no añade ninguna novedad tecnológica ni funcional significativa a las herramientas analizadas, se ha querido incluir en este pequeño análisis por ser unas de las primeras herramientas de evaluación en aplicar el nuevo concepto de Web Social; concepto tan importante en este proyecto.

CrearTest es una sencilla herramienta a través de la cual, cualquier persona puede crear y compartir sus propios exámenes para que otros usuarios los pueda realizar. La funcionalidad que ofrece es muy limitada ya que únicamente se contempla la creación y realización de exámenes. Además no hay ningún control sobre los contenidos publicados lo que hace que la información publicada sea escasamente útil,

y su recuperación altamente compleja. Por último resaltar la poco creativa interfaz de usuario lo que hace poco atractivo su uso.

iTest

iTest(29) es la continuidad del Centro de Estudios Felipe II adscrito a la Universidad Complutense de Madrid al proyecto Exanet(30) desarrollado por alumnos de este mismo centro durante el curso académico 2003/04 tal como se describe en un artículo publicado por la profesora Nuria Joglar(31) en 2007.

iTest permite la evaluación y la auto-evaluación en academia a todos los niveles educativos. iTest genera exámenes tipo test al azar, los corrige automáticamente, permite revisarlos *a posteriori*, por parte del alumno y genera listados de calificaciones y estadísticas sobre los resultados para el profesor.

Por último, no puede olvidarse que todo buen sistema LMS debe tener un motor de evaluación. Es por esto por lo que también deben tenerse en cuenta aquellos sistemas LMS referenciados con anterioridad y que no se han expuesto en esta sección por no caer en la redundancia en las referencias.

Estandarización E-Learning

Uno de los puntos más importantes y sobre el que más hincapié se está haciendo en el desarrollo de las nuevas tecnologías en general, y del e-learning en particular, es la definición de estándares. Mediante la utilización de estos estándares, se garantiza la independencia de los contenidos generados por los LCMS y utilizados por los LMS, de forma que se cumplan estas cuatro especificaciones que debería cumplir cualquier CMS:

- Accesibilidad: los contenidos son independientes de la plataforma en la que estén.
- Interoperabilidad: el contenido puede ser usado en diferentes plataformas
- Reusabilidad: los contenidos pueden ser utilizados en diferentes sistemas.
- Durabilidad: el contenido podrá utilizarse sin importar los cambios en la tecnología con la cual se elaboró.

Con la aplicación de los estándares se posibilita la libre elección de los proveedores de contenidos y herramientas, así como la reutilización de los cursos en diferentes plataformas y sistemas.

Actualmente hay diversos estándares utilizables, como son el AICC (desarrollado por la industria de la aviación de EEUU), IEEE LTSC (Instituto de Ingenieros Electrónicos e Informáticos), IMS (del Global Learning Consortium), y el más utilizado y extendido: SCORM. A continuación se explicará brevemente este último, así como IMS ya que será el estándar sobre el que se basará el presente proyecto.

SCORM

El Modelo Referenciado de Objetos de Contenido Compartible (**SCORM** , por sus siglas en inglés, *Sharable Content Object Reference Model.*) representa el conjunto de especificaciones que permiten desarrollar, empaquetar y entregar materiales educativos. Estos paquetes pueden incluir páginas Web, gráficos, Javascript, Flash y cualquier otro contenido que pueda ser representado en un navegador Web.

Las especificaciones de SCORM(32), distribuidas por ADL, detallan cómo deben de publicarse los contenidos y usarse los metadatos; también, incluyen las especificaciones para representar la estructura de los cursos por medio de XML y el uso de API.

Se puede decir que SCORM consta de tres componentes:

- Empaquetamiento de contenidos: se refiere a la manera en que se guardan los contenidos de un curso, el modo en que están ligados entre sí y la forma en la que se entregará la información al usuario. Todos estos datos se concentran en un archivo llamado imanifest.xml

- Ejecución de comunicaciones: detalla el ambiente para ejecutar la información y consta de dos partes, los comandos de ejecución y los metadatos del estudiante.

- Metadatos del curso: son de dos tipos, los que incluyen la información del curso en sí, y los que contienen el material del estudiante.

Con la aplicación de SCORM se hace posible el crear contenidos que puedan importarse dentro de sistemas de gestión de aprendizaje diferentes, siempre que estos soporten el estándar.

Puesto que SCORM no será utilizado en el presente proyecto, no se ha creído oportuno indicar los detalles técnicos y funcionales del mismo, para una mayor información puede consultarse la página oficial de la especificación (32).

IMS

El IMS Global Learning Consortium(IMS GLC)(33) es una compañía global, sin fines de lucro cuyo objetivo principal es la creación de estándares en el ámbito del e-learning. Actualmente IMS GLC está formada por más de 140 organizaciones del ámbito tecnológico y de la educación(33).

Desde su creación en 1997, IMS ha desarrollado alrededor de 20 normas las cuales son ampliamente utilizadas en aplicaciones tecnológicas de aprendizaje de todo el mundo. Estas normas incluyen estandarización de meta-datos, empaquetado de contenidos, servicios empresariales, preguntas y test, información de alumnos, definición de vocabulario, herramientas de interoperabilidad así como otros aspectos dentro del e-learning. Si bien es realmente interesante cada una de estas normas, por recaer fuera del alcance de este estudio, se realizará una pequeña descripción de una de ellas, ya que es el estándar que se va a usar en el presente proyecto: IMS Question & Test Interoperability Specification (QTI)(34).

La especificación QTI describe un modelo de representación de datos para las preguntas, los test, los datos generados así como los correspondientes informes de resultados. Por lo tanto, esta especificación permite el intercambio de datos entre sistemas LCMS, LMS, así como cualquier otra herramienta que gestione este tipo de información.

El modelo de datos QTI está descrito de una manera abstracta, utilizando UML para facilitar la unión de una amplia gama de modelo de datos, herramientas y lenguaje de programación, además, para el intercambio de datos se utiliza XML. Además hay que tener en cuenta que QTI ha sido diseñado para apoyar la interoperabilidad y la innovación por lo que han sido definidos varios puntos de extensión del estándar que pueden ser utilizados para un diseño a medida de cada herramienta.

Básicamente QTI está diseñado para:

- Proporcionar un formato del contenido bien documentado a través del cual se puedan almacenar e intercambiar test y preguntas, independientemente de la herramienta utilizada para crearlos.
- Apoyar el despliegue de los datos a través de una amplia gama de LMS.
- Proporcionar un sistema con la capacidad de informar de los resultados obtenidos en los test de una forma coherente.

A continuación se presenta un pequeño diagrama con los roles y sistemas representados en QTI:

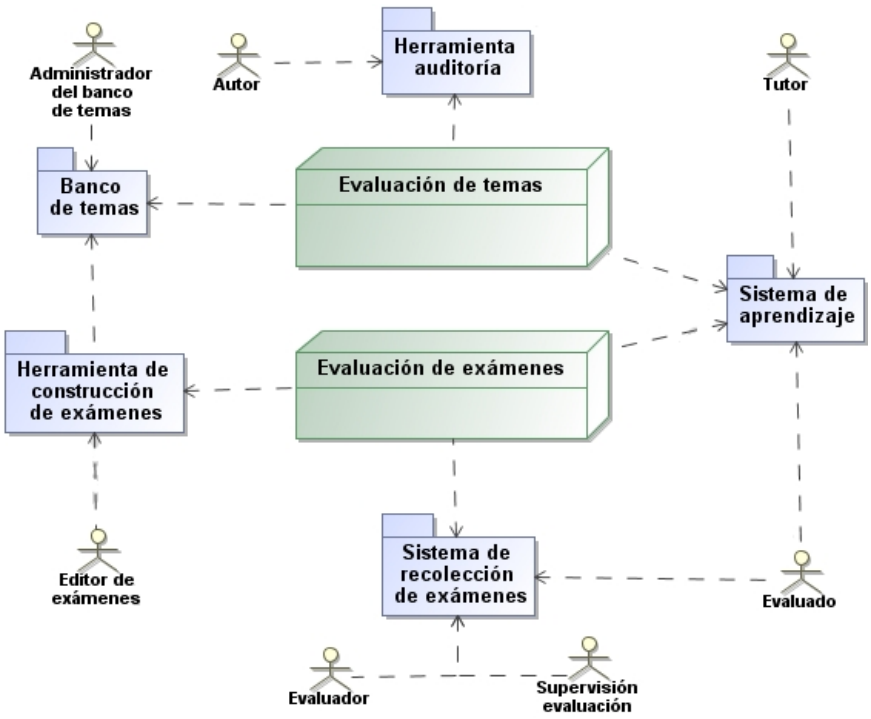

Ilustración 11 Diagrama de Roles y Sistemas que intervienen en QTI

Sistemas identificados por la especificación QTI:

- Herramienta de Auditoria: sistema utilizado por un autor para crear o modificar una evaluación.
- Banco de Temas: un sistema de recopilación y gestión de temas.
- Herramienta de construcción de exámenes: un sistema de creación de exámenes.
- Sistema de recopilación de exámenes: sistema que gestiona automáticamente de la entrega de los resultados de una evaluación al evaluador.
- Sistema de aprendizaje: lo denominado comúnmente LMS. Aquel sistema que gestiona todo lo relacionado con las tareas de gestión de aprendizaje.

Roles identificados por la especificación QTI:

- Autor: en situaciones simples los temas pueden ser creados por un solo autor, sin embargo en situaciones más complejas un tema puede tener que pasar a través de un proceso de creación con una control de calidad asociado. En esta situación se identifica la figura del autor que a través de la herramienta de auditoría valida esos temas creados.
- Administrador del banco de temas: encargado de la gestión de los elementos de las evaluaciones.
- Editor de Exámenes: se encarga de la construcción de exámenes, extrayendo los temas del banco de temas.
- Supervisor de Evaluación: aquella persona o sistema encargado de la supervisión de la correcta entrega de una evaluación.
- Evaluador: persona o sistema encargado de realizar la evaluación de una evaluación.
- Tutor: cualquier persona que participe en la gestión, dirección y tareas de apoyo en el proceso de aprendizaje del alumno.
- Evaluado: aquella persona que está siendo evaluada.

Conclusiones del estado del arte.

Tras haber analizado con cierta profundidad los dominios propuestos, se advierte que no se produce una colaboración estrecha entre los tres, al menos en el ámbito de este proyecto. Algunos dominios pueden plantear soluciones a los problemas existentes en otros.

La Web 2.0, y la Web Semántica, pueden dar solución a los problemas existentes actualmente en el e-Learning, como son la creación de contenidos y su recuperación. La experiencia colaborativa de un grupo grande de personas puede crear muchos más contenidos educacionales que un equipo reducido de las mismas. Si estos contenidos están además almacenados de forma que un sistema pueda *comprenderlos*, se aumenta espectacularmente la cantidad de gente que puede acceder a ellos, por no hablar de la posibilidad de usar estos contenidos como *servicios* para otros sistemas.

Por otro lado, la Web Social puede resolver problemas de la Web Semántica, como son el enriquecimiento semántico de contenidos. Para una persona, o equipo de personas, puede ser complicado llegar a los niveles adecuados de riqueza contextual, pero un grupo grande (a nivel de red social), puede llevar a cabo esta tarea de forma más completa.

Aparece, del mismo modo, el caso inverso. La Web Semántica puede optimizar la colaboración de los recursos en una red social. Si los contenidos que se *comparten* tuvieran a la vez un *significado* que facilitase el acceso a ellos, y que pudiera ser comprendido por cualquier sistema para su integración, se obtendría realmente ante un nuevo paradigma en la Web.

Estos tres dominios muestran diferentes caminos para llegar a un objetivo común: la total integración del Ser Humano con las nuevas tecnologías, para ámbitos de su desarrollo personal e intelectual (definido por algunos como "*El Homo Conexus"*), pero aún falta por definir la encrucijada donde estos caminos se encontrarán.

En la siguiente sección se describirán los problemas encontrados en estos dominios relacionados con el ámbito de este proyecto, que pretende ser un eslabón más en la cadena de tecnologías que posibilitarán que una persona pueda formarse, a la vez que colabora en la formación de otros, en una sociedad donde todo el mundo tiende a estar conectado.

3. DESCRIPCIÓN DEL PROBLEMA.

Los recursos sobre exámenes que están disponibles en Internet, no se encuentran clasificados por dominios absolutos, sino que obedecen a una **categorización arbitraria**, en la que es imposible localizar los elementos deseados si no se está familiarizado con la misma. Por ejemplo, un estudiante de una universidad que busque exámenes de una asignatura en la que esté matriculado, puede no encontrar exámenes de una asignatura idéntica en otra universidad, si no se da la casualidad de que los nombres de las asignaturas sean semejantes. En resumen: los elementos están clasificados según características que dependen de los autores, no de los recursos en sí.

Si bien el acceso vía categorías, es ya de por sí complicado; las búsquedas automáticas de recursos no ofrecen mucha más ayuda. Se basan en **algoritmos de búsqueda estándar**, que buscan concordancias de palabras o frases en los contenidos (en el mejor de los casos), en ocasiones, las concordancias de búsqueda se basan únicamente en el título o la descripción que el autor haya querido dar.

Por otro lado, no existe una uniformidad en cuanto a la **interfaz de recuperación de información**. Una base de conocimiento de elementos de evaluación, puede decidir basarse en la distribución de exámenes completos a nivel de fichero; normalmente documentos de MS Word, PDF, texto plano o HTML. En la mayoría de los casos, es imposible usar estas bases de conocimiento como un servicio externo, aprovechable desde las propias herramientas de evaluación. Es necesaria la intervención humana para la búsqueda de recursos, evaluación de los mismos, e integración de los datos en sistemas que automaticen la evaluación de aprendizaje.

No hay que olvidar que la mayor parte de estas redes se basan en la valoración que dé a los recursos el propio administrador del sitio. Es quien decide qué elementos se publican, y cuáles son filtrados. Las decisiones de las personas encargadas de esta purga son las que diferencian un buen recurso de uno inútil. Obviamente, cada administrador, o grupo de administradores, puede tener **criterios de evaluación de recursos distintos**, de modo que es el usuario final quien ha de ver por sí mismo qué elementos le sirven y cuáles no.

En resumen, existen las siguientes carencias graves en los sistemas actuales de puesta en común de elementos de evaluación:

1. Falta de estandarización en la clasificación y almacenamiento de elementos de evaluación.
2. Inexistencia de interfaz unificada de acceso a los elementos de evaluación a través de otros sistemas.
3. Deficiencias en los sistemas de búsquedas de los recursos compartidos.

Estos tres problemas pueden resumirse en la siguiente conclusión:

Necesidad de intervención humana en el proceso de divulgación, clasificación, evaluación, y acceso, a los elementos de evaluación educacional compartidos.

4. DESCRIPCIÓN DE LA SOLUCIÓN.

En esta sección se describirá la propuesta que se elaborará para resolver los problemas encontrados en la sección Descripción del problema. En primer lugar se enumerarán los requisitos funcionales, junto a una breve descripción de cada uno. Seguidamente, se especificarán las principales herramientas que se usarán para su desarrollo, y por último se escogerá una metodología de desarrollo para todo su ciclo de vida.

Funcionalidades.

El Servidor on-line de Exámenes (de ahora en adelante, SOLE), consistirá, grosso modo, en un entorno de acceso público (o a lo sumo controlado), en el que se podrán almacenar preguntas de exámenes, para el posterior acceso a las mismas mediante aplicaciones cliente.

El ámbito de la solución quedará enmarcado tan sólo en el desarrollo del servidor, y las interfaces con aplicaciones cliente externas, quedando dichas aplicaciones pendientes para otros proyectos. Para facilitar las comunicaciones, se emplearán estándares en la representación de los datos, y para optimizar su recuperación, tecnologías de Web Semántica. Cabe destacar que el sistema quedará por ahora reducido a preguntas tipo test sobre Ingeniería del Software. La elección de un dominio tan concreto responde a la necesidad de su especificación para el empleo de la Web Semántica. El idioma de las preguntas será inglés para hacer más sencilla su clasificación automática.

Debido a las carencias de la ontología usada (SEOntology), que no tiene definidas las instancias de las clases, será imprescindible la creación de un sencillo sistema de enriquecimiento del buscador, de modo que los usuarios puedan definir términos a los que se aplican los contextos presentes en la ontología.

Módulos funcionales

Esta descomposición en módulos no pretende formar parte del análisis. Su objetivo es definir de una forma intuitiva los diferentes grupos de funcionalidades de los que debería disponer la aplicación.

Es necesario remarcar que en la implementación del prototipo que acompañará a esta memoria, pudieran no haberse incluido todos, o al menos no de forma completa. Los análisis posteriores pueden llevar, así mismo, a la inclusión de nuevas funcionalidades, o la modificación de las aquí expuestas.

Los módulos no son completamente independientes entre sí, de modo que el desarrollo de uno probablemente permita enriquecer el funcionamiento de otros.

De forma esquemática, aunque no jerarquizada, el aplicativo SOLE dispondrá de los siguientes módulos funcionales principales:

Almacenamiento de preguntas[1]

Estará formado por el grupo de funcionalidades que permitan mantener y gestionar las preguntas publicadas en el servidor.

Se permitirá el alta de preguntas, la modificación de las mismas, y su baja en algunos casos. La categorización de preguntas se basará en la ontología correspondiente, pudiéndose acceder a ellas a través de un directorio. Aunque el proyecto SOLE esté basado en el uso de la ontología SEOntology, para Ingeniería del Software, el sistema estará preparado para una fácil inclusión de otras ontologías.

Del mismo modo, en principio se usarán preguntas en inglés, pero deberá contemplarse la posibilidad de ampliación a otros idiomas en un futuro.

Se podrá obtener información de dos tipos de cada pregunta: el propio contenido (en formato QTI), y meta información de la misma (fecha de publicación, valoración de los usuarios…). La información de ambos tipos que se almacenará será especificada durante el desarrollo del módulo.

Las preguntas podrán darse de alta de dos formas: una a una mediante interfaz gráfico de usuario, o masivamente a través de un fichero XML que cumpla el estándar QTI. Dicho fichero podrá transmitirse directamente desde el interfaz del servidor, o a través de un servicio web.

Motor semántico

Consistirá en aplicar la información que se obtenga de la ontología para dar significado a las preguntas. Deberá ser capaz de determinar con qué clase o clases de la ontología está relacionada, basándose en la información que proporcione QTI sobre dicha pregunta. Como ya se ha mencionado, el algoritmo de clasificación deberá ser independiente de la ontología usada, con vistas a futuras ampliaciones del proyecto a otras ontologías.

Una parte opcional de dicho motor, que no sería al fin y al cabo más que una adaptación del propio clasificador semántico, consistiría en ofrecer un servicio de clasificación público de texto plano, para dar valor añadido a la aplicación.

Buscador de preguntas

Será un servicio público que permitirá obtener un listado de preguntas almacenadas en el servidor, siguiendo el estándar QTI.

El buscador usará técnicas de Web semántica, apoyado en el Motor Semántico, para la creación de un índice de preguntas basado en la ontología correspondiente. Las claves de búsqueda serán a su vez procesadas por el clasificador semántico para poder obtener un listado de preguntas relacionadas con la búsqueda, ordenado por relevancia.

[1] A partir de ahora, cada vez que se haga referencia a “las preguntas”, esta denominación responderá tanto al enunciado de las preguntas, como a sus respuestas opcionales. En caso de referirse tan sólo al enunciado, se hará de forma explícita.

Con el objetivo de mantener la escalabilidad, será necesario indicar, a parte de las claves de búsqueda, el dominio sobre el que se quiere buscar (especificado por la ontología).

Las búsquedas podrán realizarse desde la aplicación del servidor, o a través de un servicio web.

Aprendizaje del motor semántico

Como se mencionó anteriormente, las carencias de SEOntology hacen necesario enriquecer al motor semántico.

El módulo de aprendizaje consistirá en una serie de funcionalidades que permitirán proponer instancias de las clases de una ontología, así como valorar las ya propuestas. Estas instancias serán usadas por el motor semántico para la clasificación.

En otras palabras, se indicará al clasificador que cierta palabra es una instancia de cierta clase, con lo que se le ayudará a encontrar coincidencias.

Gestión de usuarios

Por un lado, consistirá en la gestión típica de usuarios de cualquier aplicación, es decir: registro de usuario, login y logout.

El valor que añadirá la gestión de usuarios permitirá el empleo de funcionalidades orientadas a Web 2.0: valoración de preguntas, comentarios, “usuarios amigos”, niveles de privacidad en preguntas publicadas (privada, “sólo amigos”, pública…), así como el uso, de manera opcional, de un sistema de karma de usuarios. Este sistema permitiría aumentar la relevancia en las valoraciones que haga un usuario dependiendo de la cantidad y tipo de acciones que lleve a cabo en el servidor.

La gestión de usuarios implica la gestión de permisos sobre diferentes acciones.

Administración del servidor

No hay que olvidar a un tipo especial de usuarios: los administradores. Este tipo de usuarios podrán llevar a cabo acciones especiales, tales como la moderación en los elementos publicados (edición o censura), y de los propios usuarios.

Los administradores tendrán, a su vez, la posibilidad de configurar aspectos técnicos de la aplicación, tales como la configuración de la base de datos.

Mantenimiento del servidor

Será necesaria la ejecución de tareas en segundo plano para garantizar la ejecución óptima de todas las funcionalidades de la aplicación.

Con cierta periodicidad, y de forma transparente al usuario en la medida de lo posible, se ejecutarán tareas de reindexación, copias de seguridad, y reclasificación de preguntas para aprovechar los términos propuestos por los usuarios en el módulo de aprendizaje.

Descripción de la solución.

Sindicación

Se encargará de mantener un sistema de sindicación RSS para elementos específicos, tales como las últimas preguntas subidas por cierto usuario. Durante el desarrollo de este módulo se especificarán exactamente qué elementos serán sindicables, así como la versión concreta de RSS que se usará.

Herramientas.

En la presente sección se expondrá una solución técnica, sin entrar en detalles de diseño e implementación, que responda al problema definido. Para ello, se describirá brevemente la solución elegida así como las herramientas a utilizar en el desarrollo de la solución, clasificadas por el papel que desempeñan dentro del proyecto.

A muy alto nivel, se describe la solución elegida, como la implementación de una aplicación Web bajo los estándares Java EE(35).

Para el diseño de los componentes de negocio, se utilizará una herramienta de modelado UML como MagicDraw(36) que permitirá un diseño bajo la arquitectura MDA y generación de código automática con una herramienta como AndroMDA(37).

La comunicación entre sistemas externos se hará mediante una filosofía SOA a través de Servicios Web implementados en parte mediante frameworks de desarrollo como Spring integrado en la herramienta AndroMDA .

El Backend de la aplicación se desarrollará utilizando un mapeo de datos objeto-modelo relacional mediante la herramienta Hibernate(38), utilizando MySQL(39) como gestor de BBDD. De todas formas, se usará programación en base de datos para aquellos procesos del servidor con elevada transaccionalidad.

El Frontend de la aplicación se implementará mediante un Framework de desarrollo AJAX como ZK(40) integrado en el entorno de desarrollo Eclipse a través del plugin ZK Studio. También se utilizará la API de Struts(41) para la implementación del control del MVC.

Por último, resaltar el uso de APIS's como JAXB(42) para el tratamiento de XML que será necesario para las comunicaciones con otras aplicaciones, así como el de JENA 2(43) para tratar la información de las ontologías.

Herramientas de modelado y generación de código

Las herramientas de modelado permiten realizar los diseños de las aplicaciones siguiendo estándares como UML para posteriormente poder pasar el modelo a código a través de herramientas de generación automáticas.

MagicDraw UML Community Edition

MagicDraw UML(36) es un excelente programa para ser utilizado como modelado de UML. Provee soporte completo para metamodelos UML 2.0, incluyendo diagramas de clases, casos de uso, comunicación, secuencia, estado, actividad, implantación, paquetes, componentes, estructuras compuestas y de distribución. Adicionalmente, MagicDraw provee soporte explicito para perfiles UML y diagramas personalizables dando significado al modelo visual de arquitecturas.

MagicDraw genera automáticamente sus partes del modelo, todo de acuerdo con los patrones de diseño establecidos: GoF, Java, EJB, JUnit, Esquemas XML, WSDL, CORBA, IDL y cualquier otro patrón personalizable.

Además tiene infinidad de características más que pueden consultarse en la página oficial de la aplicación(36).

AndroMDA

AndroMDA(37) es un framework MDA Open Source, cuya función principal es la de generar código a partir de un modelo, generalmente UML en formato XMI producido por herramientas de modelado. Existen gran cantidad de plugins de AndroMDA (cartuchos y librerías) a través de los cuales se pude generar diversos componentes en forma de código para un gran número de lenguajes entre los que se encuentra Java.

A pesar de que AndroMDA puede ser utilizado en cualquier aplicación, se suele utilizar más en aplicaciones con tecnología Java EE(35). AndroMDA puede crear un nuevo proyecto Java EE desde cero, en donde el código es generado a partir de un modelo realizado en UML. Entre las grandes posibilidades que ofrece AndroMDA está la de generar el código para Hibernate(38), EJB, Spring(44), WebServices y Struts(41); integrando el código generado automáticamente al proceso de construcción. AndroMDA es muy eficiente generando código para la aplicación a partir de un modelo, generando la columna vertebral del proyecto, lo que permite que los desarrolladores se mantengan enfocados a la lógica de negocio.

Herramientas de desarrollo

A pesar de que la mayoría de herramientas descritas en esta sección podrían clasificarse como herramientas de desarrollo, únicamente se han incluido en este punto aquellas utilizadas para la edición de código, complicación, y depuración ya sea en un entorno visual o no.

Java Enterprise Edition 5 SDK

Java Platform, Enterprise Edition o Java EE(35) (anteriormente conocido como Java 2 Platform, Enterprise Edition o J2EE hasta la versión 1.4), es una plataforma de programación—parte de la Plataforma Java—para desarrollar y ejecutar software de aplicaciones en Lenguaje de programación Java con arquitectura de N niveles distribuida, basándose ampliamente en componentes de software modulares ejecutándose sobre un servidor de aplicaciones.

La plataforma Java EE está definida por una especificación similar a otras especificaciones del Java Community Process. Java EE es también considerada informalmente como un estándar debido a que los suministradores deben cumplir ciertos requisitos de conformidad para declarar que sus productos son conformes a Java EE.

Java EE incluye varias especificaciones de API, tales como JDBC, RMI, e-mail, JMS, Servicios Web, XML, etc. y define cómo coordinarlos. Java EE también configura algunas especificaciones únicas para Java EE para componentes. Estas incluyen Enterprise JavaBeans (EJB), servlets, portlets (siguiendo la especificación de Portlets Java), JavaServer Pages (JSP) y varias tecnologías de servicios web. Esto permite al desarrollador crear una Aplicación de Empresa portable entre plataformas y escalable, a la vez que integrable con tecnologías anteriores. Otros beneficios añadidos son, por

ejemplo, que el servidor de aplicaciones puede manejar transacciones, la seguridad, escalabilidad, concurrencia y gestión de los componentes desplegados, significando que los desarrolladores pueden concentrarse más en la lógica de negocio de los componentes en lugar de en tareas de mantenimiento de bajo nivel.

En su versión 5, la plataforma Java EE pasa a ser el estándar de la industria para la aplicación de la arquitectura orientada a servicios (SOA) así como de la próxima generación de aplicaciones Web. Se centra en hacer más fácil el desarrollo conservando la riqueza de la plataforma J2EE 1.4. Además ofrece nuevas y actualizadas funciones, como Enterprise JavaBeans (EJB) 3.0, tecnología JavaServer Faces (JSF), y la última API de servicios Web.

Eclipse Ganymede

Eclipse(45) es un entorno de desarrollo integrado de código abierto multiplataforma para desarrollar aplicaciones. Desarrollado originalmente por IBM, actualmente es desarrollado por la Fundación Eclipse, una organización independiente sin ánimo de lucro que fomenta una comunidad de código abierto y un conjunto de productos complementarios, capacidades y servicios.

La versión actual de Eclipse dispone de las siguientes características:

- Editor de texto
- Resaltado de sintaxis
- Pruebas unitarias con JUnit
- Control de versiones con CVS
- Integración con Ant
- Asistentes (wizards): para creación de proyectos, clases, tests, etc.
- Refactorización

Asimismo, existen multitud de plugins para diversas tareas como la integración con Hibernate(38) y Subversión(46) que permiten extender la funcionalidad del entorno.

ZK

ZK(40) es un framework AJAX, escrito completamente en Java, de código abierto, que permite el desarrollo de una rica interfaz de usuario para aplicaciones Web sin usar JavaScript y con una programación simple y escasa.

El núcleo de ZK es un mecanismo conducido por eventos basado en AJAX, sustentado sobre 70 componentes XUL y 80 componentes XHTML, y un lenguaje de marcación para diseñar interfaces de usuario. Los programadores diseñan las páginas de su aplicación en componentes XUL/XHTML con una gran cantidad de características, y los manipulan con eventos disparados por la actividad del usuario

final. Es similar al modelo de programación encontrado en las aplicaciones basadas en GUI de escritorio.

ZK utiliza el acercamiento llamado centrado-en-el-servidor para la sincronización de componentes, el pipelining entre clientes y servidores se hace automáticamente por el motor, y los códigos de Ajax son completamente transparentes para los desarrolladores de aplicaciones Web. Por lo tanto, los usuarios finales obtienen una interacción y respuesta similar a las de una aplicación de escritorio, mientras que la complejidad del desarrollo es similar a la que tendría la codificación de aplicaciones de escritorio.

Además de la programación basada en componentes, de manera similar a Swing, ZK soporta un lenguaje de marcación para la definición de una potente interfaz de usuario llamada ZUML.

- ZUML está diseñado para que desarrolladores no expertos diseñen interfaces de usuario de forma eficiente.
- ZUML permite a un desarrollador mezclar diferentes tipos de lenguaje de marcación, tales como el lenguaje XUL de Mozilla y XHTML, todos ellos en la misma página.
- ZUML permite a los desarrolladores embeber scripts en lenguaje Java (interpretado por BeanShell) y usar expresiones para manipular los componentes y acceder a los datos.

El desarrollo con ZK estará integrado en el IDE Eclipse, a través de un plugin de dicho entorno llamado ZK Studio que forma parte del propio desarrollo de ZK.

Frameworks y API's de desarrollo

Además de las herramientas de desarrollo descritos anteriormente, se han utilizado una serie de API's y frameworks de desarrollo en Java para facilitar algunas tareas.

Struts

Struts(41) es una herramienta de soporte para el desarrollo de aplicaciones Web bajo el patrón MVC bajo la plataforma Java EE(35). Struts se desarrollaba como parte del proyecto Jakarta de la Apache Software Foundation, pero actualmente es un proyecto independiente conocido como Apache Struts.

Struts se basa en el patrón del Modelo-Vista-Controlador (MVC). Cuando se programan aplicaciones Web con el patrón MVC con un único controlador, aparece un grave problema, ya que el controlador se convierte en lo que se conoce como "fat controller", es decir un controlador de peticiones. Struts surge como la solución a este problema ya que implementa un solo controlador (ActionServlet) que evalúa las peticiones del usuario mediante un archivo configurable (struts-config.xml).

Struts permite reducir el tiempo de desarrollo. Su carácter de "software libre" y su compatibilidad con todas las plataformas en que Java Entreprise esté disponible, lo convierte en una herramienta altamente disponible.

Spring

El Spring Framework(44) (también conocido simplemente como Spring) es un framework de código abierto de desarrollo de aplicaciones para la plataforma Java.

A pesar de que Spring Framework no obliga a usar un modelo de programación en particular, se ha popularizado en la comunidad de programadores en Java al considerársele una alternativa, y sustituto, del modelo de Enterprise JavaBean. Por su diseño el framework ofrece mucha libertad a los desarrolladores en Java y soluciones muy bien documentadas y fáciles de usar para las prácticas comunes en la industria.

Mientras que las características fundamentales de este framework pueden emplearse en cualquier aplicación hecha en Java, existen muchas extensiones y mejoras para construir aplicaciones basadas en Web por encima de la plataforma empresarial de Java (Java Enterprise Platform).

Hibernate

Hibernate(38) es una herramienta de mapeo objeto-relacional para la plataforma Java (y disponible también para .Net con el nombre de NHibernate) que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación, mediante archivos declarativos (XML) que permiten establecer estas relaciones.

Como todas las herramientas de su tipo, Hibernate busca solucionar el problema de la diferencia entre los dos modelos de datos coexistentes en una aplicación: el usado en la memoria de la computadora (orientación a objetos) y el usado en las bases de datos (modelo relacional). Para lograr esto permite al desarrollador detallar cómo es su modelo de datos, qué relaciones existen y qué forma tienen. Con esta información Hibernate le permite a la aplicación manipular los datos de la base operando sobre objetos, con todas las características de la POO. Hibernate convertirá los datos entre los tipos utilizados por Java y los definidos por SQL. Hibernate genera las sentencias SQL y libera al desarrollador del manejo manual de los datos que resultan de la ejecución de dichas sentencias, manteniendo la portabilidad entre todos los motores de bases de datos con un ligero incremento en el tiempo de ejecución.

Hibernate está diseñado para ser flexible en cuanto al esquema de tablas utilizado, para poder adaptarse a su uso sobre una base de datos ya existente. También tiene la funcionalidad de crear la base de datos a partir de la información disponible.

Hibernate ofrece también un lenguaje de consulta de datos llamado HQL (Hibernate Query Language), al mismo tiempo que una API para construir las consultas programáticamente (conocida como "criteria").

Hibernate para Java puede ser utilizado en aplicaciones Java independientes o en aplicaciones Java EE, mediante el componente Hibernate Annotations que implementa el estándar JPA, que es parte de esta plataforma.

JAXB

JAXB(42) es una API de Java para el tratamiento de información XML proporcionando un mapeo directo entre XML y los objetos Java. Dado un esquema, que especifica la estructura de los datos XML, el compilador JAXB genera un conjunto de clases de Java que contienen todo el código para analizar los documentos XML basados en el esquema. Una aplicación que utilice las clases generadas puede construir un árbol de objetos Java que representa un documento XML, manipular el contenido del árbol, y regenerar los documentos del árbol, todo ello en XML sin requerir que el desarrollador escriba código de análisis y de proceso complejo.

Las principales ventajas de la API JAXB son las siguientes:

- Usa tecnología Java y XML
- Garantiza datos válidos
- Es rápida y fácil de usar
- Puede restringir datos
- Es personalizable y extensible

JENA 2

Framework desarrollado por HP Labs para manipular metadatos desde una aplicación Java. Desde su versión 1, JENA(43) implementaba soporte para RDF, y ya en su versión 2 incluye una API para el manejo de ontologías soportando OWL.

Además se incluyen los siguientes componentes:

- Un Parser y API de RDF.
- API de Ontologías con soporte para OWL, DAML y RDF Schema.
- Subsistema de Razonamiento.
- Soporte para Persistencia.
- RDQL: Lenguaje de consultas de RDF.

JQTI

Básicamente JQTI(47) es un intérprete para el estándar IMS QTI v2.1 el cual está desarrollado sobre XML. QTI es bastante diferente a la mayoría de estándares sobre XML, ya que no sólo contiene datos, sino también contiene instrucciones para el procesamiento de estos datos. Estas instrucciones, que hacen referencia a las evaluaciones en sí, especifican básicamente cómo han de presentarse las preguntas, así como los contenidos evaluados. De hecho se podría definir la especificación QTI como un propio lenguaje de programación, cuya especificación viene dada por el propio documento XML.

El intérprete JQTI incluye una serie de características que lo hace realmente interesante para el procesamiento de la información bajo el estándar QTI: incluye

apoyo para leer, analizar e interpretar los documentos XML QTI, así como para la creación de nuevos contenidos en QTI sobre XML; además de permitir la validación y el informe de errores de los documentos en QTI.

Por último resalta la distribución de JQTI bajo la licencia BSD lo que permite usarlo libremente desde otros desarrollos como el de SOLE.

Log4j

Log4j(48) es una biblioteca open source desarrollada en Java por la Apache Software Foundation, que permite a los desarrolladores de software elegir la salida y el nivel de granularidad de los mensajes o "logs" en tiempo de ejecución y no en tiempo de compilación, como es comúnmente realizado, lo que se consigue mediante el uso de archivos de configuración externos.

Log4j tiene una serie de características que hace realmente interesante su uso para la generación de ficheros de control y monitorización del sistema. Incluye seis niveles de prioridad de mensaje, se pueden configurar varias salidas de texto simultáneamente, incluso configurar nuestra propia salida tipo servidor remoto, base de datos etc. Por otro lado, es totalmente configurable desde un fichero XML, lo que hace que su configuración sea altamente flexible y con una gran facilidad de uso gracias a la API que dispone.

ROME

ROME(49) es un conjunto de herramientas open source, orientadas a la gestión de feeds RSS y Atom, distribuida bajo licencia Apache 2.0.

Las diferentes bibliotecas de ROME permiten publicar, generar, parsear, y convertir entre diferentes formatos de sindicación.

Los formatos soportados por ROME son:

- RSS 0.90
- RSS 0.91 Netscape
- RSS 0.91 Userland
- RSS 0.92
- RSS 0.93
- RSS 0.94
- RSS 1.0
- RSS 2.0
- Atom 0.3
- Atom 1.0

El uso de ROME en el proyecto SOLE permitirá la generación de feeds RSS abstrayéndose del manejo de XML a bajo nivel.

Por un lado, el manejo de objetos Java facilitará la sindicación de diferentes elementos, dando escalabilidad al módulo de sindicación. Por otro lado, al simplificar la conversión entre distintos formatos, la migración de uno a otro sería afrontable sin demasiado esfuerzo.

Herramientas de gestión de datos

Uno de los principales puntos en todo sistema es la persistencia de datos. Como en la mayoría de aplicaciones, para el presente sistema se hará uso de un gestor relacional de base de datos encargado de almacenar los datos de la aplicación en tablas relacionales.

MySQL

MySQL(39) es un gestor de base de datos sencillo de usar y increíblemente rápido. También es uno de los motores de base de datos más usados en Internet, la principal razón de esto es que es gratis para aplicaciones no comerciales.

Las características principales de MySQL son las siguientes:

- Es un gestor de base de datos relacional. Una base de datos relacional es un conjunto de datos que están almacenados en tablas entre las cuales se establecen unas relaciones para manejar los datos de una forma eficiente y segura. Para usar y gestionar una base de datos relacional se usa el lenguaje estándar de programación SQL.

- Es Open Source. El código fuente de MySQL se puede descargar y está accesible a cualquiera, por otra parte, usa la licencia GPL para aplicaciones no comerciales.

- Es una base de datos muy rápida, segura y fácil de usar. Gracias a la colaboración de muchos usuarios, la base de datos se ha ido mejorando optimizándose en velocidad. Por eso es una de las bases de datos más usadas en Internet.

- Existe una gran cantidad de API's para todos los lenguajes (incluido Java) que facilita en gran medida la comunicación con el gestor.

- A partir de la versión 5, permite el uso de procedimientos almacenados en la base de datos, funciones, tablas temporales, etc.

Servidor de Aplicaciones

Tal y como se ha comentado insistentemente, el sistema a desarrollar se trata de una aplicación Web por lo que otro punto importante en la solución tecnológica es la elección del servidor de aplicaciones sobre el que se desplegará la aplicación. A pesar de que en un primer momento se pensó en la utilización de Tomcat por su simplicidad, facilidad de uso y extensión entre los desarrolladores; se abandonó la idea una vez que se decidió la utilización de EJB's para la implementación de los servicios de negocio del sistema, ya que Tomcat únicamente es un contenedor de Servlets y no de EJB's.

Glass Fish

GlassFish(50) es un servidor de aplicaciones que implementa las tecnologías definidas en la plataforma Java EE y permite ejecutar aplicaciones que siguen esta especificación, tomando como base de desarrollo el Application Server de Sun

Microsystems. Es gratuito, de código libre y se distribuye bajo la licencia CDDL y la GNU GPL.

Glash Fish implementa la plataforma Java EE 5, por lo que soporta las últimas versiones de tecnologías como: JSP, JSF, Servlets, EJBs, Java API para Servicios Web (JAX-WS), Arquitectura Java para Enlaces XML (JAXB), Metadatos de Servicios Web para la Plataforma Java 1.0, y muchas otras tecnologías. El servidor Glash Fish está rodeado por una gran comunidad de desarrolladores con el mismo nombre, que están en continuo desarrollo detectando y corrigiendo errores a la vez que dotan al servidor de nuevas funcionalidades.

Herramientas de colaboración

Unas de las herramientas más importantes y muchas veces olvidadas en todo desarrollo software, son las herramientas de gestión de proyectos y colaboración. Este tipo de herramientas facilitan el trabajo en entornos colaborativos, además de facilitar el acceso al código y documentación generada.

SVN / Tortoisse

Subversión(46) es un software de sistema de control de versiones diseñado específicamente para reemplazar al popular CVS, el cual posee varias deficiencias. Es software libre bajo una licencia de tipo Apache/BSD y se le conoce también como svn por ser ese el nombre de la herramienta de línea de comandos. Una característica importante de Subversion es que, a diferencia de CVS, los archivos versionados no tienen cada uno un número de revisión independiente. En cambio, todo el repositorio tiene un único número de versión que identifica un estado común de todos los archivos del repositorio en cierto punto del tiempo.

Las principales características de subversión son las siguientes:

- Se sigue la historia de los archivos y directorios a través de copias y renombrados.
- Las modificaciones (incluyendo cambios a varios archivos) son atómicas.
- La creación de ramas y etiquetas es una operación más eficiente; Tiene costo de complejidad constante (O(1)) y no lineal (O(n)) como en CVS.
- Se envían sólo las diferencias en ambas direcciones (en CVS siempre se envían al servidor archivos completos).
- Puede ser servido mediante Apache, sobre WebDAV/DeltaV. Esto permite que clientes WebDAV utilicen Subversion en forma transparente.
- Maneja eficientemente archivos binarios (a diferencia de CVS que los trata internamente como si fueran de texto).

- Permite selectivamente el bloqueo de archivos. Se usa en archivos binarios que, al no poder fusionarse fácilmente, conviene que no sean editados por más de una persona a la vez.

- Cuando se usa integrado a Apache permite utilizar todas las opciones que este servidor provee a la hora de autentificar archivos (SQL, LDAP, etc.).

Como clientes de subversión se utilizará Subclipse(51), plugin integrado con Eclipse, así como TortoiseSVN(52) para ser utilizado visualmente integrado con el sistema operativo.

dotProject

dotProject(53) es una herramienta orientada a la Gestión de Proyectos. Para eso se orienta a la administración de recursos para desarrollar un producto, cuya producción requiera de un conjunto de actividades o tareas que se desarrollen entre ellas en forma paralela o independiente.

La aplicación consta de un conjunto de entidades ordenadas jerárquicamente las cuales permiten brindar la funcionalidad del producto. A continuación se mencionan las entidades más importantes de dotProject:

- Compañías: Son las entidades que agrupan proyectos, actividades y usuarios.

- Departamentos: Son áreas dentro de las compañías, que permiten agrupar usuarios en dicho nivel.

- Usuarios/Contactos: dotProject tiene usuarios los cuales son capaces de loguearse a dotProject y trabajar dentro del esquema de permisos que posea el rol de dicho usuario. Los contactos son usuarios especiales que asignados a un determinado proyecto pueden recibir por ejemplo: correo, actualizaciones y noticias pero no necesariamente deben tener acceso al sistema dotProject. Los usuarios y contactos pertenecen a una compañía.

- Proyectos: Es la entidad que contiene el grupo de tareas necesarias para desarrollar un determinado producto.

- Actividades: son las tareas asignadas dentro de un proyecto. Son los componentes sobre los cuales se controla: la duración, dependencias, recursos asignados y progreso. Las actividades deben de pertenecer a un único proyecto.

- Diagramas de Gantt: Permite ver en forma grafica las actividades ordenadas jerárquicamente, mostrando las dependencias y solapamientos de las mismas.

- Tickets: para administrar todos los problemas relacionados a un proyecto.

- Archivos: Permite almacenar archivos dentro de un proyecto permitiendo su versionado.

- Foros: Permite la creación de foros de discusión dentro de cada proyecto para distribuir información y discutir temas relativos al proyecto del foro.

- Administración del Sistema: Contiene la actividades relacionadas a la administración de usuarios, roles y configuración del sistema.

Descripción de la solución.

Metodología de desarrollo.

Tras analizar diferentes metodologías tradicionales de desarrollo tales como Métrica v3(54), Merise(55), así como los Estándares de Desarrollo Software propuestos por la ESA(56); se ha concluido que las características del proyecto a desarrollar como solución al problema planteado, no se adaptaban a este tipo de metodologías. Es por ello por lo que se ha buscado un nuevo tipo de metodologías más flexibles como son las metodologías ágiles.

Metodologías Ágiles	Metodologías Tradicionales
Basadas en heurísticas de prácticas en producción de código	Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo
Especialmente preparados para cambios durante el proyecto	Cierta resistencia a los cambios
Proceso menos controlado, con pocos principios	Proceso mucho más controlado, con numerosas políticas/normas
El cliente es parte del equipo de desarrollo	Existe un contrato prefijado, el cliente interactúa con el equipo de desarrollo mediante reuniones
Grupos pequeños y trabajando en el mismo sitio	Grupos grandes y posiblemente distribuidos
Pocos roles	Más roles
Menos énfasis en la arquitectura del software	La arquitectura del software es esencial y se expresa mediante modelos
Tiempos Flexibles	Tiempos Estrictos
Simplicidad de aplicación	Complejas estructuras

Tabla 7 Comparativa de las Metodologías Ágiles frente a las Tradicionales

Metodologías ágiles

Las metodologías ágiles empezaron a desarrollarse en el año 2001 cuando diecisiete expertos en los modelos de desarrollo software, se reunieron para definir los métodos que estaban surgiendo frete a las metodologías tradicionales. De esta reunión surgió el siguiente *Manifiesto Ágil*(57):

"Estamos poniendo al descubierto mejores métodos para desarrollar software, haciéndolo y ayudando a otros a que lo hagan. Con este trabajo hemos llegado a valorar:

- A los individuos y su interacción, por encima de los procesos y las herramientas.
- El software que funciona, por encima de la documentación exhaustiva.
- La colaboración con el cliente, por encima de la negociación contractual.
- La respuesta al cambio, por encima del seguimiento de un plan.

Aunque hay valor en los elementos de la derecha, valoramos más los de la izquierda."

Tras los cuatro valores descritos, los firmantes redactaron los siguientes, como los principios que de ellos se derivan:

- Nuestra principal prioridad es satisfacer al cliente a través de la entrega temprana y continua de software de valor.

- Son bienvenidos los requisitos cambiantes, incluso si llegan tarde al desarrollo. Los procesos ágiles se doblegan al cambio como ventaja competitiva para el cliente.

- Entregar con frecuencia software que funcione, en periodos de un par de semanas hasta un par de meses, con preferencia en los periodos breves.

- Las personas del negocio y los desarrolladores deben trabajar juntos de forma cotidiana a través del proyecto.

- Construcción de proyectos en torno a individuos motivados, dándoles la oportunidad y el respaldo que necesitan y procurándoles confianza para que realicen la tarea.

- La forma más eficiente y efectiva de comunicar información de ida y vuelta dentro de un equipo de desarrollo es mediante la conversación cara a cara.

- El software que funciona es la principal medida del progreso.

- Los procesos ágiles promueven el desarrollo sostenido. Los patrocinadores, desarrolladores y usuarios deben mantener un ritmo constante de forma indefinida.

- La atención continua a la excelencia técnica enaltece la agilidad.

- La simplicidad como arte de maximizar la cantidad de trabajo que no se hace, es esencial.

- Las mejores arquitecturas, requisitos y diseños emergen de equipos que se auto-organizan.

- En intervalos regulares, el equipo reflexiona sobre la forma de ser más efectivo y ajusta su conducta en consecuencia.

A pesar de que este tipo de metodologías centra como uno de los puntos principales la comunicación con el cliente, y el trabajo en contacto con él, punto que obviamente no se cumplirá en el presente desarrollo por no elegir tal figura; se ha elegido este nuevo concepto de metodología de desarrollo por aportar la suficiente flexibilidad en el trabajo y diseño que el ámbito del proyecto necesita.

Tras analizar las principales propuestas de metodologías ágiles, finalmente se ha optado por seguir una de las pioneras en este ámbito: XP (Extreme Programming)(58). A continuación se expondrás las principales características de esta metodología así como la descripción del ciclo de vida que se propone y seguirá el sistema a desarrollar.

XP (Extreme Programming)

La programación extrema es una metodología ágil desarrollada por Kent Back para la construcción de software. La misma se basa en la simplicidad, comunicación y reutilización del código desarrollado. Su idea básicamente consiste en desarrollar sub-versiones del software, realizando entregas con frecuencia (cada dos semanas o menos). Cada sub-versión, se debe codificar de manera simple para que funcione correctamente. El diseño (o mini-diseño) se hace a medida que se avanzan sobre las sub-versiones. El código y el diseño se van modificando continuamente, añadiéndoles funcionalidades.

A pesar de que XP es especialmente flexible en este punto, se definen los siguientes roles: programador, cliente, encargado de pruebas, encargado de seguimiento, responsable global y gestor. Para el presente proyecto todos estos roles son asumidos por los dos desarrolladores del proyecto, a excepción del rol de cliente y encargado de seguimiento que será compartido con el propio tutor del mismo.

A diferencia de las metodologías tradicionales donde se plantea un desarrollo en vertical, con cuatro procesos claramente diferenciados (análisis, diseño, implementación y pruebas) los cuales se realizan de manera secuencial en grandes periodos de tiempo cada uno; o de las metodologías iterativas las cuales proponen un desarrollo en espiral, cerrando mini-proyectos a partir de pequeños análisis; XP propone una única etapa de análisis en la cual se defina el alcance del proyecto, para después ir resolviendo pequeños problemas planteados en el análisis, realizando paralelamente los procesos de diseño, implementación y pruebas lo que desemboca en pequeñas entregas que van resolviendo requisitos de usuario y son validadas por el propio usuario.

En el gráfico adjunto puede observarse claramente las características comentadas de cada una de las metodologías:

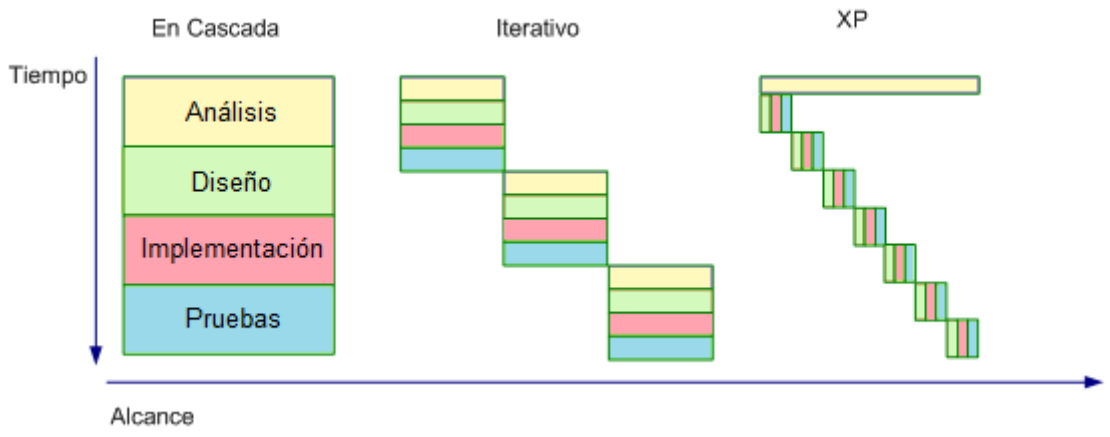

Ilustración 12 Comparativa entre las diferentes metodologías

El uso de XP se basa en trece prácticas básicas:

1. Equipo completo: está formado por el equipo de desarrollo, el cliente y el responsable del proyecto.

2. Planificación: se observan las historias de usuario y se planifica en qué orden se van a realizar. Puesto que la planificación puede o no respetarse, es necesario revisarla constantemente.

3. Pruebas del cliente: el cliente, con la ayuda de los desarrolladores, propone sus propias pruebas.

4. Versiones pequeñas: las sub-versiones deben ser lo suficientemente pequeñas como para poder implantarse y probarse en un periodo de no más de una semana.

5. Diseño simple: a medida que se implementa el código se deben de ir dando algunos retoques a la interfaz, analizando cada una de ellas y eliminando partes innecesarias de la misma.

6. Pareja de programadores: dos personas aportan más al código que una sola.

7. Desarrollo guiado por las pruebas automáticas: es necesario usar un programa que testee las historias de usuarios.

8. Mejorar el código: mientras de codifica debe mejorarse el código; extraer funcionalidades comunes, eliminar código innecesario, etc.

9. Integración continua: cuando se tenga una nueva funcionalidad, debe recompilarse y probarse.

10. El código es de todos: cualquier persona dentro del equipo de desarrollo deber conocer el código y modificarlo.

11. Normas de codificación: de debe definir un estilo común de codificación.

12. Metáforas: hay que buscar frases que definan cómo funcionan las distintas partes del programa, así todo el quipo entenderá de qué se está hablando.

13. 0 horas extras y días 0: se quiere señalar que no deben realizarse horas extras, y no dejar días sin hacer nada. Tiempo: aproximadamente 40 horas por semana.

Ciclo de Vida de XP

El ciclo de vida ideal de XP consiste de seis fases(59):

1. Exploración: En esta fase, los clientes plantean a grandes rasgos las historias de usuario que son de interés para la primera entrega del producto. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto. Se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo. La fase de exploración toma de pocas semanas a pocos meses, dependiendo del tamaño y familiaridad que tengan los programadores con la tecnología.

2. Planificación de la Entrega (*Release*): En esta fase el cliente establece la prioridad de cada historia de usuario, y correspondientemente, los programadores realizan una estimación del esfuerzo necesario de cada

una de ellas. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente. Una entrega debería obtenerse en no más de tres meses. Esta fase dura unos pocos días. Las estimaciones de esfuerzo asociado a la implementación de las historias la establecen los programadores utilizando como medida el punto. Un punto, equivale a una semana ideal de programación. Las historias generalmente valen de 1 a 3 puntos. Por otra parte, el equipo de desarrollo mantiene un registro de la "velocidad" de desarrollo, establecida en puntos por iteración, basándose principalmente en la suma de puntos correspondientes a las historias de usuario que fueron terminadas en la última iteración. La planificación se puede realizar basándose en el tiempo o el alcance. La velocidad del proyecto es utilizada para establecer cuántas historias se pueden implementar antes de una fecha determinada o cuánto tiempo tomará implementar un conjunto de historias. Al planificar por tiempo, se multiplica el número de iteraciones por la velocidad del proyecto, determinándose cuántos puntos se pueden completar. Al planificar según alcance del sistema, se divide la suma de puntos de las historias de usuario seleccionadas entre la velocidad del proyecto, obteniendo el número de iteraciones necesarias para su implementación.

3. Iteraciones: Esta fase incluye varias iteraciones sobre el sistema antes de ser entregado. El Plan de Entrega está compuesto por iteraciones de no más de tres semanas. En la primera iteración se puede intentar establecer una arquitectura del sistema que pueda ser utilizada durante el resto del proyecto. Esto se logra escogiendo las historias que fuercen la creación de esta arquitectura, sin embargo, esto no siempre es posible ya que es el cliente quien decide qué historias se implementarán en cada iteración (para maximizar el valor de negocio). Al final de la última iteración el sistema estará listo para entrar en producción. Los elementos que deben tomarse en cuenta durante la elaboración del Plan de la Iteración son: historias de usuario no abordadas, velocidad del proyecto, pruebas de aceptación no superadas en la iteración anterior y tareas no terminadas en la iteración anterior. Todo el trabajo de la iteración es expresado en tareas de programación, cada una de ellas es asignada a un programador como responsable, pero llevadas a cabo por parejas de programadores.

4. Producción: La fase de producción requiere de pruebas adicionales y revisiones de rendimiento antes de que el sistema sea trasladado al entorno del cliente. Al mismo tiempo, se deben tomar decisiones sobre la inclusión de nuevas características a la versión actual, debido a cambios durante esta fase. Es posible que se rebaje el tiempo que toma cada iteración, de tres a una semana. Las ideas que han sido propuestas y las sugerencias son documentadas para su posterior implementación (por ejemplo, durante la fase de mantenimiento).

5. Mantenimiento: Mientras la primera versión se encuentra en producción, el proyecto XP debe mantener el sistema en funcionamiento al mismo tiempo que desarrolla nuevas iteraciones. Para realizar esto se requiere de tareas de soporte para el cliente. De esta forma, la velocidad de desarrollo puede bajar después de la puesta del sistema en producción. La fase de mantenimiento puede requerir nuevo personal dentro del equipo y cambios en su estructura.

6. Muerte del Proyecto: Es cuando el cliente no tiene más historias para ser incluidas en el sistema. Esto requiere que se satisfagan las necesidades del cliente en otros aspectos como rendimiento y confiabilidad del sistema. Se genera la documentación final del sistema y no se realizan más cambios en la arquitectura. La muerte del proyecto también ocurre cuando el sistema no genera los beneficios esperados por el cliente o cuando no hay presupuesto para mantenerlo.

Tomando estas fases como base de desarrollo, el ciclo de vida de XP viene definido mediante el siguiente diagrama.

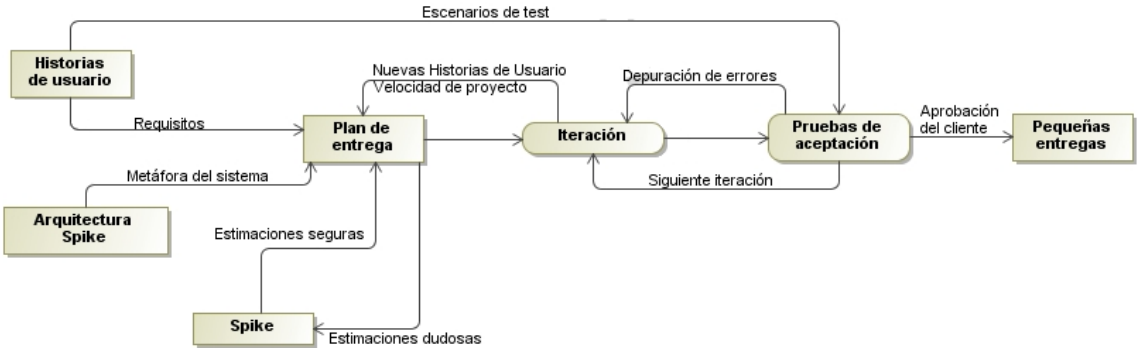

Ilustración 13 Ciclo de Vida de XP

Las condiciones especiales del proyecto a desarrollar, donde no hay un cliente que demande un software, sino que los requisitos de usuario nacen a partir de la experiencia de los desarrolladores y tutor del proyecto, así como de un estudio de mercado previo; hace que sea imposible aplicar el presente ciclo de vida directamente. Por este motivo se ha creado un ciclo de vida específico para el presente proyecto, partiendo como base el propuesto por XP.

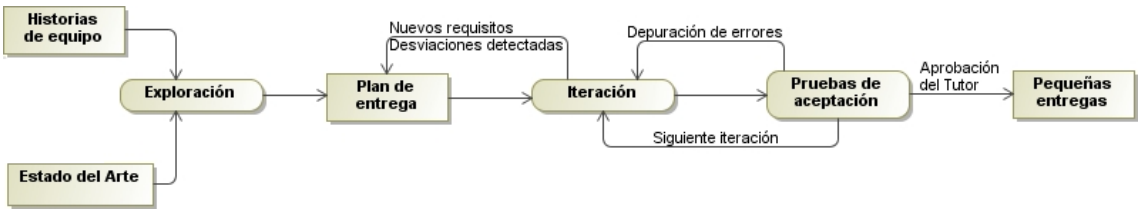

Ilustración 14 Ciclo de Vida del proyecto SOLE

Los principales cambios encontrados son la desaparición del usuario como fuente de requisitos así como los sistemas de pruebas automáticas, a la vez que se han incluido nuevas fuentes de requisitos como es el estudio de mercado y la experiencia previa del equipo en este tipo de herramientas. También se ha incluido la figura del tutor como parte fundamental en la aprobación de cada uno de los elementos desarrollados.

También es importante resaltar la inclusión explicita de la fase de exploración anteriormente descrita y no incluida en el ciclo de vida de XP.

5. CONCLUSIONES Y LÍNEAS FUTURAS.

Tras el desarrollo de la primera versión del Buscador Semántico sobre una Red Social de preguntas de examen, procede al análisis de los resultados, y especificación de la hoja de ruta para siguientes versiones.

Conclusiones.

La principal conclusión que se extrae del desarrollo del proyecto es que la aplicación resultante responde adecuadamente a los problemas planteados. Dichos problemas consistían en los siguientes:

1. Falta de estandarización en la clasificación y almacenamiento de elementos de evaluación.
2. Inexistencia de interfaz unificada de acceso a los elementos de evaluación a través de otros sistemas.
3. Deficiencias en los sistemas de búsquedas de los recursos compartidos.

Un análisis sobre el prototipo resultante del proyecto, ofrece los siguientes datos:

1. Los elementos de evaluación se almacenan siguiendo el estándar QTI 2.1, de dominio público, y que puede ser estudiado e implementado por cualquier entidad. La información se clasifica para su recuperación siguiendo la Ontología SEOntology.

2. Los elementos de evaluación se recuperan mediante un XML que sigue el esquema de QTI 2.1. Cualquier aplicación puede implementar los servicios Web necesarios para acceder a la información.

3. La búsqueda se realiza a nivel de pregunta, y no de examen, siguiendo una categorización de las claves de búsqueda, y comparándola con el índice categorizado que se actualiza cada vez que se da de alta una pregunta. Los resultados de búsqueda son mejorados por la experiencia de los usuarios, al ser ellos mismos quienes educan al motor semántico.

El único punto negativo que se observa, es que la potencia del motor semántico, tanto para clasificación como para búsqueda, depende de que los usuarios. En los primeros momentos, las búsquedas no serán muy exactas, necesitándose una gran promoción de la aplicación para que vaya mejorando con el tiempo.

Líneas futuras.

En el desarrollo del prototipo, han quedado varias ideas en el tintero, que no han sido implementadas finalmente. La mayoría de estas cuestiones atañen a la usabilidad de la aplicación, y tratan de mejorar la experiencia del usuario y administrador.

La hoja de ruta para siguientes versiones contiene los siguientes hitos:

1. Implementar la interfaz Web del buscador.
2. Implementar los servicios pendientes:
    a. Edición de preguntas.
    b. Eliminación de preguntas.
    c. Creación de preguntas a partir de QTI.
    d. Eliminación de comentarios.
3. Implementar los componentes que han quedado aplazados:
    a. Gestor de Redes
    b. Comunicador Usuarios
    c. Administración.
    d. Sindicación.
    e. Demonio.

Tras completar los elementos pendientes de la primera versión, se podría comenzar la explotación de la aplicación, con una primera versión beta. Se han encontrado las siguientes mejoras, que podrían ser desarrolladas a partir de esta primera versión beta:

1. Mejora sustancial de la interfaz gráfica Web.
2. Añadir soporte para nuevos tipos de preguntas, soportados por QTI 2.1:
    a. Preguntas con varias interacciones.
    b. Preguntas a desarrollar.
    c. Etc.
3. Añadir nuevas ontologías/dominios.
4. Permitir que el usuario no especifique el dominio de la pregunta, y que sea el motor semántico quien escoja la más adecuada.
5. Sistema de intercambio de mensajes privados entre usuarios.
6. Desarrollo de redes de usuario adicionales, con distintos niveles de acceso (redes privadas, redes públicas).
7. Acceso a preguntas mediante un sistema de directorio, estructurado según los índices del buscador.
8. Nuevos diccionarios de idiomas para la interfaz Web.
9. Internacionalización de las preguntas almacenadas, pudiéndose buscar preguntas en idiomas concretos.
10. Mejorar el algoritmo de clasificación, orientándolo a preguntas.
11. Clases para sugerir conceptos, y conceptos para valorar aleatorios.

6. APÉNDICES.

En esta sección se adjuntarán documentos que se consideren relevantes para un mejor conocimiento del aplicativo SOLE desarrollado. Se dividirán los anexos en iteraciones de desarrollo, siguiendo la metodología XP.

De cada iteración, se adjuntará un documento de diseño y uno de pruebas, así como cualquier otro documento que haya sido generado y que se considere relevante.

Análisis del Sistema de Información.

Este documento contendrá toda la documentación generada en la tarea de Análisis del proyecto Servidor on-line de Exámenes (en adelante, SOLE).

Introducción.

El Servidor on-line de Exámenes es un repositorio on-line en el que los visitantes pueden compartir preguntas de exámenes de Ingeniería del Software.

En la *Propuesta de Solución* de la memoria del proyecto se llevó a cabo un primer esbozo del sistema de información. En el este documento, se detallarán con mayor profundidad sus características.

La metodología que se seguirá para el desarrollo del proyecto será XP, de modo que la fase de análisis se completará para el sistema completo, y no debería modificarse durante el resto del ciclo de vida del proyecto.

Estructura del documento.

No existe ningún estándar de documentación para la metodología XP. Se deja a elección de cada grupo de desarrollo que establezca, tanto los documentos que generará, como la estructura de los mismos. A parte de la introducción, este documento de análisis tendrá tres partes fundamentales: una descripción general del aplicativo, una descripción individual de cada componente, y una planificación.

En la descripción general se especificará la arquitectura a alto nivel de la aplicación, se concretarán los distintos roles de usuarios, y se definirán las interfaces de comunicación con otros sistemas. En la descripción modular, se mostrará de cada módulo sus requisitos funcionales y de interfaz, así como los casos de uso que implementarán las funcionalidades propias. En la planificación, se llevará a cabo una priorización y estimación temporal del desarrollo del sistema.

Relación con otros proyectos.

De forma paralela a SOLE, y con una estrecha colaboración entre ambos proyectos, Borja Blanco Iglesias está desarrollando el aplicativo GEO (Gestor de exámenes on-line). Esta aplicación será cliente de servicios SOLE, de modo que las necesidades y características de la aplicación no se limitarán a un entorno teórico de cliente-servidor, pudiendo llevarse a cabo pruebas sobre entornos reales.

Del mismo modo, se tendrán en cuenta los hitos del proyecto GEO para la priorización del desarrollo de componentes.

Descripción general.

En esta sección se analizará el aplicativo al completo, sin detallar las características individuales de cada componente.

El sistema consiste básicamente en una Red Social en la que los distintos usuarios comparten preguntas de examen. La información que se maneja, puede ser accedida desde el propio sitio web de la aplicación, o a través de aplicaciones externas que usen los servicios. Para el acceso a las preguntas, se usará un buscador semántico, que las indexará basándose en una ontología, que en un primer momento será SEOntology.

Los usuarios podrán subir preguntas, comentar las existentes, y formar redes *de amigos* con otros usuarios, lo que les permitirá compartir sus preguntas a varios niveles de privacidad.

El buscador semántico aprenderá de la experiencia de los usuarios, que podrán enseñarle a mejorar el análisis de preguntas para su clasificación. Los usuarios podrán sugerir términos para los distintos conceptos que contenga la ontología; así como valorar los términos que hayan propuesto otros usuarios.

Usuarios, roles y escenarios.

El Servidor on-line de exámenes presenta cuatro visiones de la aplicación: usuario del sitio, usuario a través de aplicación cliente, administrador del servidor, y demonio de mantenimiento.

Los usuarios del sitio (a partir de ahora, simplemente *Usuarios*), y los administradores del servidor (*Administradores*), interactuarán con las funcionalidades del aplicativo a través de una interfaz web que formará parte del propio servidor.

Los usuarios de aplicaciones cliente (*Clientes*), ejecutarán las distintas funcionalidades a través de servicios, desde aplicativos externos. Por el momento no se contempla la posibilidad de administración del servidor a través de aplicaciones externas. En la sección *Interfaces con otros sistemas*, se detallan los servicios que podrán consumir los clientes.

El demonio de mantenimiento (*Demonio*) será un agente software interno al servidor, que disparará determinadas tareas de mantenimiento automático.

A priori, los diferentes roles estarán presentes en todos los módulos funcionales del aplicativo, a excepción del Demonio, que tan sólo será partícipe del Módulo de mantenimiento del servidor, y del Cliente, que tan sólo se comunicará con el *Manejador de Servicios*.

En resumen, los escenarios existentes en SOLE serán los siguientes:

- Escenario del Usuario.
- Escenario del Cliente.
- Escenario del Administrador.
- Escenario del Demonio.

Arquitectura de SOLE.

En esta sección se describirán los componentes de alto nivel existentes en la aplicación, así como las relaciones entre ellos, y los componentes externos que podrán conectarse al aplicativo.

Los módulos funcionales, obtenidos en la solución funcional, no equivaldrán necesariamente a componentes software del aplicativo. En la mayoría de los casos, un módulo funcional estará formado por varios componentes, es más, un componente podrá ser usado por varios módulos funcionales.

Se ha optado por una arquitectura Cliente-Servidor basada en componentes. Cada componente será técnicamente independiente del resto, aunque se harán llamadas unos a otros a través de EJBs. En el diagrama adjunto, se muestran con líneas continuas las llamadas internas. Las interfaces de comunicación con Web Services se definirán en este mismo documento, pero las interfaces de comunicación interna entre componentes no serán definidas hasta las tareas de diseño.

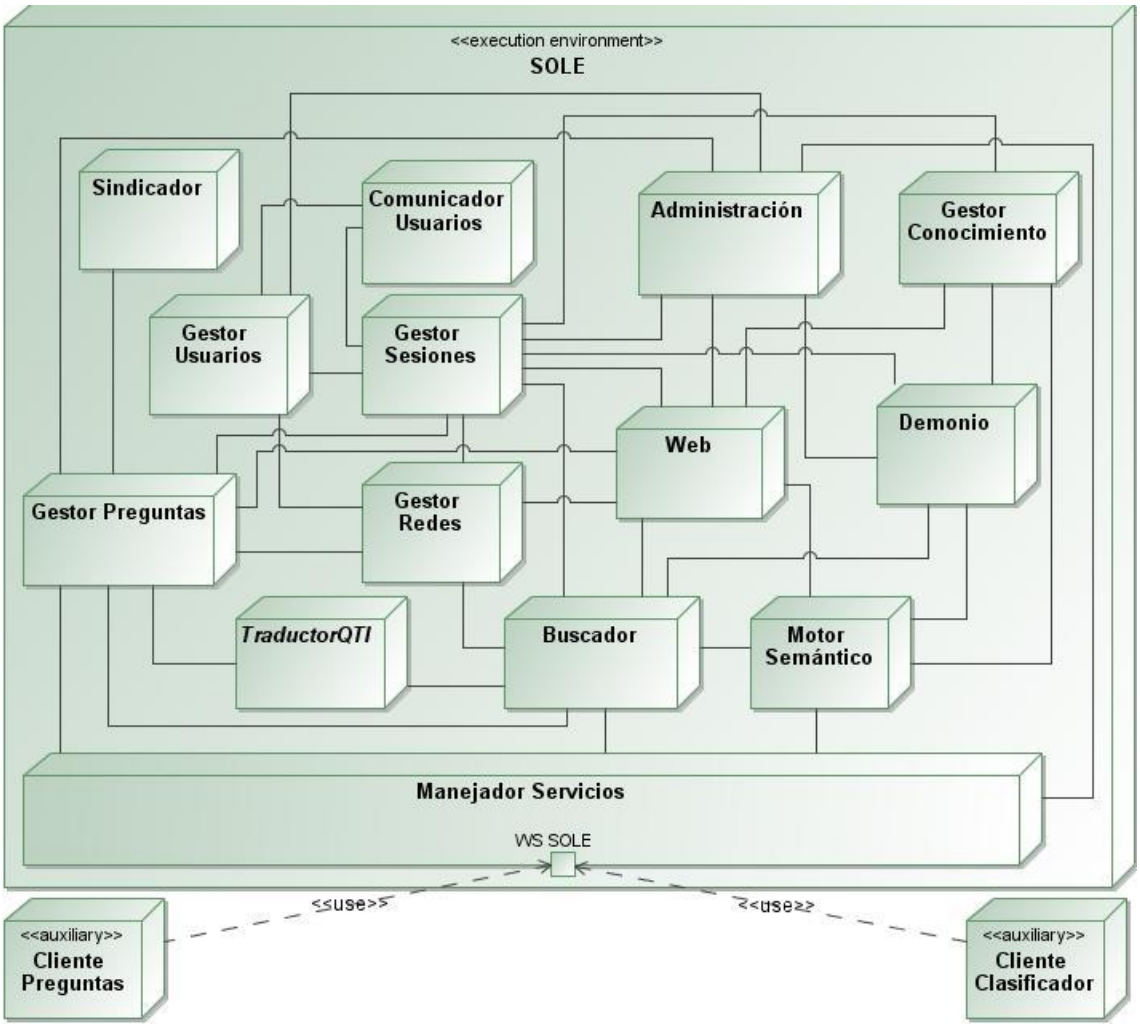

Ilustración 15 Arquitectura global de SOLE

En un primer momento, todos los componentes serán desplegados en una sola máquina, aunque su diseño irá encaminado a que puedan ser distribuidos. En el diagrama arquitectónico adjunto, puede observarse el entorno de ejecución principal

con los nodos propios de la aplicación. Se han añadido los sistemas externos que, de forma principal, se comunicarán con la aplicación. Se ha obviado el nodo correspondiente a la base de datos; cada componente gestionará su propia persistencia, de modo que se trataría de información redundante que no ayudaría a la comprensión del conjunto arquitectónico.

A continuación se describirá cada uno de los nodos presentes en la arquitectura:

- **SOLE** (entorno de ejecución): se refiere al entorno del servidor de aplicaciones donde se desplegarán los nodos.
- **Manejador servicios**: centralizará en un único punto las llamadas a los servicios Web.
- **Gestor Preguntas**: implementará las tareas básicas para la persistencia de preguntas: consulta, alta, modificaciones, etc.
- **Traductor QTI**: permitirá transformar una pregunta, o conjunto de ellas, en un XML que siga el estándar QTI, además de la operación inversa.
- **Gestor Usuarios**: ofrecerá los servicios relacionados con la gestión de usuarios: registro, consulta, modificación de datos, etc.
- **Gestor Sesiones**: aparte de la autenticación de usuarios (login, logout), mantendrá un registro de todas las sesiones iniciadas en el servidor.
- **Comunicador Usuarios**: permite que los usuarios reciban mensajes al suceder determinados eventos.
- **Gestor Redes**: manejará las operaciones relacionadas con redes de usuarios.
- **Buscador**: consistirá en dos partes fundamentales, no representadas en el diagrama: el indexador, y el buscador en sí. El indexador gestionará el índice de preguntas que consultará el buscador para devolver los resultados. No hay que confundir la búsqueda de preguntas, con la consulta del Gestor de Preguntas.
- **Motor semántico**: se usará a la hora de clasificar preguntas en el indexador, y al clasificar los términos de búsqueda en el buscador. Se ofrecerá del mismo modo un servicio de clasificación de texto plano, aprovechando la funcionalidad implementada.
- **Gestor Conocimiento**: estará formado por todas las operativas relacionadas con los términos sugeridos por los usuarios para enriquecer el Motor Semántico. Además de la gestión de los términos sugeridos, será el encargado de informar al Motor Semántico durante las clasificaciones.
- **Sindicador**: administrará los distintos feeds RSS que se generen durante el uso del aplicativo. Los ficheros generados podrán ser leídos por lectores RSS externos.
- **Administración**: encapsulará las operativas de configuración y mantenimiento del servidor, además de las tareas de administración de preguntas y usuarios.
- **Demonio**: agente encargado de disparar automáticamente las tareas de administración y mantenimiento del servidor.

- **Web**: interfaz web de usuario a través del que se invoquen las operaciones permitidas.
- **Cliente preguntas** (auxiliar): ejemplo de aplicación externa que consuma los servicios de la Gestión de Preguntas y el Buscador. Necesitará autenticación para las tareas cuyo nivel de permiso sea el de usuario.
- **Cliente Clasificador** (auxiliar): ejemplo de aplicación externa que consuma los servicios de clasificación de texto plano.

Interfaces con otros sistemas.

SOLE interactuará con clientes de los servicios web ofrecidos. En esta sección, se detallarán todas las posibles comunicaciones con dichos sistemas, incluyendo los parámetros de entrada y de salida.

La interfaz de Web Service será gestionada por el *Manejador de Comunicaciones*. Será el componente que reciba las peticiones, y las distribuya entre el resto de componentes, antes de devolver la respuesta. En este apartado, se describirá cada una de las funcionalidades ofrecidas por el servidor a través de Servicio Web.

Información Servidor	
Descripción	Devuelve información útil del servidor para los clientes de Web Services.
Entrada	-

Salida	• Versión QTI
	• Ontologías Servidor
	  ○ Nombre
	  ○ Descripción
	  ○ URI

Tabla 8 Descripción Servicio "Información Servidor"

Alta Pregunta	
Descripción	Da de alta una pregunta en el servidor.
Entrada	• Usuario • Contraseña (encriptada) • QTI con las pregunta a subir • URI ontología • Visibilidad pregunta (*pública, redes, privada*) • Redes visible (si visibilidad redes)
Salida	• QTI de la pregunta incluyendo ID • Errores encontrados

Tabla 9 Descripción Servicio "Alta Pregunta"

Modificación Pregunta	
Descripción	Modifica una pregunta ya dada de alta en el servidor.
Entrada	• Usuario • Contraseña (encriptada) • QTI de la pregunta incluyendo ID • URI ontología • Visibilidad pregunta (*pública, redes, privada*) • Redes visible (si visibilidad redes)
Salida	• QTI guardado en el servidor • Errores encontrados

Tabla 10 Descripción Servicio "Modificación Pregunta"

Baja Pregunta	
Descripción	"*Elimina*" del servidor una pregunta dada de alta.
Entrada	• Usuario • Contraseña (encriptada) • ID de la pregunta
Salida	• Errores encontrados

Tabla 11 Descripción Servicio "Baja Preguntas"

Consultar Pregunta	
Descripción	Obtiene toda la información sobre una pregunta.
Entrada	• ID pregunta

Salida	• QTI Pregunta
	• Autor
	• Fecha de alta
	• Fecha última modificación
	• Valoración media (0-10)
	• Número de votos
	• Veces marcada como inapropiada
	• Enlace a pregunta
	• Últimos comentarios
	  ○ Usuario
	  ○ Fecha
	  ○ Comentario

Tabla 12 Descripción Servicio "Consultar Pregunta"

Valoración Preguntas	
Descripción	El usuario vota y opcionalmente comenta una pregunta
Entrada	• Usuario • Contraseña (encriptada) • ID de la pregunta • Nota (-1-10) • Comentario

Salida	• (Metainformación actualizada)
	• ID pregunta
	• Autor
	• Fecha de alta
	• Fecha última modificación
	• Valoración media (0-10)
	• Número de votos
	• Veces marcada como inapropiada
	• Enlace a pregunta
	• Últimos comentarios
	  ○ Usuario
	  ○ Fecha
	  ○ Comentario

Tabla 13 Descripción Servicio "Valoración Preguntas"

Búsqueda Preguntas	
Descripción	Obtiene las preguntas que sigan ciertas condiciones.
Entrada	• Usuario • Contraseña (encriptada) • URI ontología • Keywords • Buscar todas (sin comparar keywords) • Criterio ordenación (relevancia, valoración usuarios, fecha alta/modificación) • Orden (ascendente, descendente) • Visibilidad (públicas, redes, privadas) • Red (una o varias si visibilidad = redes). • Número de preguntas

Salida	• QTI con resultados.

Tabla 14 Descripción Servicio "Búsqueda Preguntas"

Clasificar texto	
Descripción	Analiza un texto siguiendo una ontología que exista en el servidor, y lo clasifica.
Entrada	• URI ontología • Texto

Salida	• Resultados
	  ○ Clase
	  ○ Conceptos

Tabla 15 Descripción Servicio "Clasificar Texto"

Redes usuario	
Descripción	Obtiene las redes a las que pertenece un usuario
Entrada	• Usuario • Contraseña (encriptada)

Salida	• Redes
	  ○ Nombre de la red
	  ○ Tipo (propia, amigos)

Tabla 16 Descripción Servicio "Redes Usuario"

Descripción modular.

A continuación se estudiarán cada uno de los módulos que se han obtenido tras la elaboración de la arquitectura de SOLE.

Gestor Preguntas

Se trata de un componente típico de gestión, enfocado al dominio de las "preguntas". Es necesario remarcar que la denominación "pregunta" abarca no sólo el enunciado, sino también las posibles respuestas y la solución de la misma. Dicho de otra forma, las "preguntas" serán los elementos que en QTI se denominan "Assessment Items".

Los tipos de preguntas soportados será un subconjunto de aquellos definidos por el estándar QTI 2.1. El sistema deberá estar preparado para ir añadiendo nuevos tipos de preguntas soportados por QTI con el tiempo. En la primera versión, se gestionarán únicamente preguntas tipo test multirrespuesta.

Las operaciones soportadas por el Gestor de Preguntas serán:

- **Añadir pregunta**: guarda en el sistema una pregunta con los datos aportados. Será necesario ser usuario registrado para ejecutarlo.
- **Modificar pregunta**: modifica uno o varios datos asociados a una pregunta. Un usuario sólo podrá modificar sus propias preguntas.
- **Eliminar pregunta**: marca una pregunta como *eliminada*, sin borrarla físicamente del sistema. Un usuario sólo podrá eliminar sus propias preguntas.
- **Obtener pregunta**: obtiene toda la información relacionada con una pregunta.
- **Consultar preguntas**: obtiene una lista de preguntas que cumplan ciertas condiciones relacionadas con sus datos y/o metadatos.
- **Votar pregunta**: añade un voto a una pregunta. Los votos serán de 0 a 10, pudiéndose además marcar la pregunta como *inapropiada* (voto -1).
- **Comentar pregunta**: añade un comentario a una pregunta. Un usuario podrá comentar cualquier pregunta las veces que quiera.
- **Modificar voto pregunta**: modifica el voto dado a una pregunta. Un usuario sólo podrá modificar sus propias valoraciones.
- **Eliminar comentario**: marca como "eliminado" un comentario. Sólo podrán hacerlo los administradores y los autores de la pregunta.

Hay que resaltar la diferenciación que se hace entre los *datos* de una pregunta, y los *metadatos* de la misma. Esta separación obedece a la necesidad de distinguir la información de la pregunta que se limita a elementos como enunciado, respuestas posibles, soluciones, etc.; de la información adicional que se necesita para su gestión, y que será importante para los usuarios.

La información sobre preguntas que se mostrará en las consultas de la primera versión, a parte los datos propios de QTI, será la siguiente:

- **Autor**: usuario que dio de alta la pregunta en el sistema.
- **Fecha de alta**: fecha en la que la pregunta fue creada.

- **Fecha de modificación**: fecha de la última edición de la pregunta.
- **Enlace**: hipervínculo HTTP a la pregunta completa. Será útil cuando las consultas se hagan desde clientes del Servicio Web
- **Valoración media**: media de los votos obtenidos. Será un número de 0 a 10. Las valoraciones de pregunta *inapropiada* no entrarán en el cálculo.
- **Número de votos**: veces que ha sido valorada la pregunta.
- **Pregunta inapropiada**: veces que la pregunta ha sido votada como *inapropiada*.
- **Últimos comentarios**: lista de los diez últimos comentarios que haya recibido la pregunta. Cada comentario tendrá el siguiente contenido:
    - **Usuario**: usuario que hizo el comentario.
    - **Fecha/hora**: fecha y hora en la que se hizo.
    - **Comentario**: texto del comentario.

Una pregunta tendrá asociado un nivel de visibilidad, como veremos más adelante. Los niveles posibles serán: privada (sólo la ve el usuario), compartida en red (véase *Gestor de Redes*) y pública (visible para todos los usuarios).

Traductor QTI

Será el componente que traduzca los objetos de negocio a un XML que siga el estándar QTI 2.1, y a la inversa. Tendrá que manejar datos de forma masiva, y será muy importante la velocidad de traducción para que no forme un cuello de botella en las comunicaciones.

Dado que en un primer momento el sistema no soportará todos los tipos de preguntas especificados por QTI 2.1, deberá poder ser ampliado con facilidad. Será así mismo fundamental una correcta gestión de errores para que no comprometa la estabilidad del sistema la entrada de un fichero que no sea válido.

Las operaciones soportadas por el Traductor QTI serán:

- **Obtener QTI**: obtiene un fichero XML que siga el estándar QTI a partir de una o varias preguntas.
- **Interpretar QTI**: a partir de un fichero XML que siga el estándar QTI 2.1, obtiene todas las preguntas compatibles con el sistema.

Gestor Usuarios

Es el componente que se encarga de la gestión de usuarios, en lo referente a consultas, altas, y modificaciones de datos. Los usuarios administradores son también usuarios al fin y al cabo, y sus acciones serán gestionadas por el Gestor de Usuarios, con la salvedad del alta de nuevos administradores, cuyo ámbito pertenece al componente de Administración, tal y como se verá al estudiar dicho.

Las operaciones soportadas por el Gestor de Usuarios serán:

- **Registro Usuario**: da de alta un nuevo usuario en el sistema.
- **Modificar Usuario**: modifica los datos de un usuario.
- **Buscar Usuario**: permitirá encontrar a un usuario del servidor a través de su alias.

Gestor Sesiones

Es el componente que se encarga de la gestión de permisos. Será requerido por todos los componentes del sistema que tengan interfaz externa directa, por lo que sus operaciones han de ser muy eficientes para no entorpecer el resto de funcionalidades. Además, mantendrá un registro diario de las sesiones que se hayan iniciado en el servidor, volcando el contenido a un fichero de log cuando se ejecuten las tareas de mantenimiento.

Las operaciones soportadas por el Gestor de Sesiones serán:

- **Login usuario**: inicia sesión con el usuario dado.
- **Logout usuario**: cierra la sesión del usuario logado, y pasa a ser un usuario anónimo.
- **Validar permiso**: comprueba si el usuario logado, o un usuario anónimo en el caso de no estar logado, está autorizado a realizar una acción dada.
- **Guardar sesiones**: trasladará la información de sesiones almacenada hasta el momento a un fichero de texto plano, y se eliminará del sistema.
- **Cerrar sesiones inactivas**: realizará un logout automático de aquellas sesiones en las que no se haya realizado ninguna operación en un máximo de tiempo parametrizado.
- **Obtener acciones permitidas**: mostrará las acciones que pueden realizarse dependiendo del rol con el que se haya iniciado la sesión (o anónimamente, si no se ha iniciado).
- **Obtener usuario:** devolverá información sobre el usuario con el que se ha iniciado sesión.

Comunicador usuarios

Se encargará de enviar mensajes a los usuarios cuando sucedan ciertos eventos. En un primer momento, estos mensajes serán únicamente e-mails, aunque en el futuro podría estudiarse la implementación de un sistema de mensajes entre usuarios.

Las operaciones soportadas por el Gestor de Redes serán:

- **Enviar aviso**: se enviará una comunicación estándar a un usuario, cuyo contenido dependerá del tipo de comunicación.
- **Enviar mensaje**: se enviará un mensaje personalizado a un usuario.

Gestor Redes

Una red de usuarios será un conjunto de usuarios, que por la mera pertenencia a dicha red, tendrán acceso a los recursos que se hayan puesto disponibles. En un primer momento, las únicas redes de las que podrá formarse parte, serán “de amigos”, es decir, un usuario podrá incluir a otro dentro de su “red de amigos”, por lo que el usuario añadido tendrá acceso a los recursos de red que comparta el usuario que lo añadió. La peculiaridad de estas redes, respecto a las posibles redes futuras de usuarios, consistirá en que será necesaria cierta “reciprocidad”. Cuando un usuario

reciba una solicitud de otro para formar parte de su red de *amigos*, si acepta, cada uno pasará a formar parte de la red de *amigos* del otro.

A parte de las redes *de amigos*, existirá en la primera versión otro tipo de redes: las propias de usuario. Estas redes, aunque aparentemente contradigan al concepto de *red*, estarán formadas únicamente por un usuario, cada uno de la suya propia, sin posibilidad de aumentar. Su utilidad reside en organizar la visibilidad de los recursos compartidos por los usuarios de una forma lógica y estructurada. No podrá dejar de formarse parte de dicha red.

Las operaciones soportadas por el Gestor de Redes serán:

- **Darse de alta en red**: un usuario pasa a formar parte de una red de usuarios. En un primer momento, se tratarán de redes *de amigos*, será necesaria la autorización del *propietario* de la red (es decir, del otro usuario), y serán recíprocas.
- **Darse de baja en red**: el usuario deja de formar parte de una red de forma inmediata, y sin confirmación. En las redes de amigos, implicará que el otro usuario se dé de baja automáticamente en la red propia.

Motor semántico

Su objetivo fundamental es clasificar textos ateniéndose a una ontología que se le especifique. Se usará a la hora de clasificar preguntas que vayan a ser indexadas, aunque se aprovechará su presencia para que pueda usarse como clasificador de textos convencionales.

Las operaciones soportadas por el Motor Semántico serán:

- **Clasificar texto**: analizará el texto indicado, según la ontología dada, para decidir qué clases de la ontología, e instancias, son las más significativas.
- **Clasificar pregunta**: aplicará la clasificación de texto al contenido de una pregunta, para obtener las clases de la ontología dada, e instancias, que más se ajustan a ella.

Gestor Conocimiento

En muchas ocasiones, la información contenida en la definición de una ontología no será suficiente para un funcionamiento correcto del motor semántico. El gestor de conocimiento permitirá la retroalimentación del motor semántico, al obtener de los usuarios instancias de las clases de una ontología.

A parte de sugerir conceptos (instancias de clases), los usuarios podrán también valorar las instancias sugeridas por otros. De este modo, el motor semántico podrá distinguir aquellos conceptos fiables de los que no lo son.

Las operaciones soportadas por el Gestor de Conocimiento serán:

- **Sugerir concepto**: el usuario propondrá un concepto como instancia de una clase de una ontología.
- **Valorar concepto**: el usuario evaluará la fiabilidad de un concepto como instancia de una clase de una ontología.

- **Obtener conceptos**: se mostrarán los conceptos sugeridos y validados por usuarios correspondientes a una clase de una ontología.

Buscador

Manejará el *índice* de preguntas, y el *buscador semántico*, que usará dicho índice para mostrar conjuntos de preguntas que correspondan con la búsqueda efectuada.

El índice agrupará conjuntos de preguntas basándose en clases de una ontología. Aunque en un primer momento sólo se vaya a manejar una única ontología, el sistema debería estar preparado para gestionar varias a la vez. La organización del índice será en tres niveles:

1. Ontología (correspondiente a un dominio dato)
2. Clase (correspondiente a un concepto semántico dentro del dominio)
3. Instancia (correspondiente a una palabra o palabras en las que pueda expresarse la clase que las contiene).

Cada terna ontología-clase-instancia contendrá un conjunto de identificadores de preguntas que hayan sido indexados.

Debido a las características de aprendizaje del motor semántico, es posible que la clasificación que corresponda a una pregunta pueda variar con el tiempo, por lo que será necesario que las preguntas se puedan reindexar para perfeccionar el índice a medida que el motor aprenda.

El buscador hará consultas sobre el índice. Para hacer una búsqueda, será necesario especificar sobre qué ontología se quiere buscar, así como las claves de búsqueda, criterio de ordenación, y número máximo de resultados que se mostrarán.

Los criterios de ordenación serán: relevancia, valoración de los usuarios, fecha de alta/última modificación.

Opcionalmente, se podrá buscar limitando el ámbito de la consulta a aquellas cuyo autor forme parte de una red concreta, y especificando el nivel de privacidad. Hay que recordar que se han definido dos tipos de redes: de *amigos*, e *individuales*, por lo que si quieren buscarse las preguntas de un usuario concreto, habrán de buscarse aquellas de *la red* formada por ese usuario. Naturalmente, tan sólo se mostrarán preguntas que sean visibles para quien haga la búsqueda. El resultado será un fichero XML que siga la especificación QTI.

Para versiones futuras, se podría usar una búsqueda avanzada, con limitaciones relacionadas con la metainformación de la pregunta (por ejemplo, con una valoración de los usuarios superior a X).

Hay que destacar que el buscador recibirá consultas de dos fuentes: a través de la web de SOLE, y a través de Web Service. En el caso de una búsqueda por Web, los resultados se mostrarán formateados para una fácil lectura, aunque se permitirá al usuario descargar el fichero QTI de resultados. Cuando se seleccione una pregunta de los resultados, se mostrará toda su información y metainformación.

Las operaciones soportadas por el Buscador serán:

- **Indexar pregunta**: una pregunta ya clasificada se incluye en el índice.
- **Reindexar pregunta**: una pregunta ya indexada y nuevamente clasificada se reubica en el índice.
- **Buscar preguntas**: recibe los criterios de búsqueda, y devuelve un fichero QTI con los resultados que haya en el índice.

Sindicador

Generará feeds RSS 2.0, con información sobre preguntas añadidas. En la primera versión, los feeds disponibles serán:

- Últimas preguntas añadidas por ontología.
- Últimas preguntas añadidas por un usuario.

Los distintos feeds actualizarán automáticamente cada vez que se añada una pregunta al sistema, y podrán ser leídos por cualquier lector RSS 2.0.

La única operación que realizará el Sindicador será:

- **Actualizar feeds**: se añadirá una pregunta al feed de ontología, y el de usuario.

Administración

La administración del servidor permitirá configurar y parametrizar el comportamiento del mismo, además de realizar operaciones de supervisión de la información almacenada.

En un primer momento, todas las operaciones se realizarán desde una interfaz independiente con autenticación, aunque en el futuro sería aconsejable poder hacer ciertas tareas de administración desde la interfaz normal de usuario.

Las operaciones soportadas por el Administrador serán:

- **Configurar servidor**: indicará al sistema los datos técnicos necesarios para su funcionamiento (por ejemplo, configuración de la base de datos).
- **Configurar aplicación**: indicará al sistema los datos de parametrización que sean necesarios (por ejemplo, intervalos de ejecución del demonio).
- **Administrar preguntas**: permitirá gestionar preguntas (alta, baja, modificación), así como las valoraciones de las mismas, para por ejemplo evitar comentarios fuera de lugar.
- **Administrar usuarios**: permitirá gestionar usuarios (alta, baja, modificación).
- **Administrar administradores**: permitirá gestionar administradores (alta, baja, modificación).

Demonio

Será un proceso que se ejecute en segundo plano. Permitirá hacer tareas de mantenimiento del sistema, así como la reindexación de preguntas según los datos sugeridos por los usuarios.

El proceso deberá poder ejecutarse con el mínimo impacto posible para los usuarios, por lo que será necesario que sea lo más eficiente posible. Deberá diseñarse teniendo en cuenta que el tamaño de los datos puede aumentar considerablemente.

Las operaciones que automatizará el demonio serán, por este orden, y para cada ejecución:

1. **Asimilar datos M.A.:** obtendrá los datos relevantes, recibidos a través del módulo de aprendizaje, y los incorporará al motor semántico.
2. **Reindexar preguntas**: en base a los datos del motor semántico, volverá a clasificar preguntas, y actualizará sus datos de indexación en el buscador.
3. **Volcado sesiones**: la información de sesiones almacenada se guardará en un fichero de texto plano para su conservación.

Para agilizar el proceso de reindexación de preguntas, no se aplicará a todas las almacenadas en cada ejecución. Para la primera versión, será necesario hacer una estimación de tiempo de ejecución para calcular el número de preguntas **n** reindexables en una hora. En cada ejecución se aplicará la reindexación a las **n** preguntas que lleven más tiempo sin reindexarse.

Web

Los usuarios podrán ejecutar la mayoría todas las acciones descritas desde una interfaz Web dinámica, a excepción de las referidas al *demonio*. La interfaz Web deberá seguir el ejemplo de las redes sociales actuales en cuanto a usabilidad, acceso rápido y simple a los distintos contenidos, y empleo de tecnologías tales como Ajax para que sea similar a una aplicación de escritorio.

Manejador Comunicaciones

Será el componente que despliegue los Servicios Web. Redirigirá cada llamada al componente correspondiente. En la primera versión, los métodos disponibles a través de servicio web serán:

- Obtener información del servidor.
- Alta de pregunta.
- Modificación de pregunta.
- Eliminación de pregunta.
- Valoración de preguntas.
- Búsqueda de preguntas.
- Consultar pregunta.
- Clasificación de texto.
- Redes Usuario

Planificación.

El objetivo de esta sección es trazar una hoja de ruta para las próximas etapas de diseño, implementación, y pruebas; siguiendo con la metodología XP.

El criterio que se seguirá para asignar el orden de cada componente, obedecerá tanto a dependencias internas y externas, como a la estimación del esfuerzo necesario para completar cada desarrollo.

Dependencias internas.

Durante el diseño de la arquitectura de SOLE, se hicieron las estimaciones iniciales de dependencias entre componentes. Se usará esta estimación para trazar la hoja de ruta, aunque durante el diseño pudieran descubrirse nuevas dependencias, o pudiera desaparecer alguna.

En la siguiente matriz, pueden verse de forma vertical los componentes, y de forma horizontal, de qué componente o componentes dependen.

	Manejador servicios	**Gestor preguntas**	**Traductor QTI**	Gestor usuarios	Gestor sesiones	Comunicador usuarios	Gestor redes	Buscador	Motor semántico	Gestor conocimiento	Sindicador	Administración	Demonio	Web
Manejador servicios		X						X	X			X		
Gestor preguntas			**X**		X		X				X			
Traductor QTI		**X**												
Gestor usuarios														
Gestor sesiones				X										
Comunicador usuarios				X										
Gestor redes				X	X	X								
Buscador		X	X		X		X		X					
Motor semántico		X								X				
Gestor conocimiento					X									
Sindicador														
Administración		X		X	X									
Demonio								X	X	X		X		
Web	X	X		X	X			X	X	X	X	X		

Tabla 17 Matriz de dependencias entre componentes.

Se observa una doble dependencia en los componentes *Gestor preguntas* – *Traductor QTI*. En este caso, habrá que dividir los componentes para su desarrollo en módulos funcionales más pequeños.

A continuación, se muestra un resumen de dependencias, ordenada de forma orientativa por los componentes de los que depende (descendente), y por los que dependen de él (ascendente).

Componente	Componentes de los que depende	Componentes que dependen de él
Gestor usuarios	0	5
Sindicador	0	2
Gestor sesiones	1	6
Gestor conocimiento	1	3
Traductor QTI	1	2

Comunicador usuarios	1	1
Motor semántico	2	4
Gestor preguntas	3	6
Administración	3	3
Gestor redes	3	2
Manejador servicios	4	1
Demonio	4	0
Buscador	5	2
Web	9	0

Tabla 18 Resumen de dependencias.

Basándose en esta tabla, se estimará el orden en el que debieran desarrollarse los componentes para respetar las dependencias. Para el caso de doble dependencia, se dividirá el desarrollo de los componentes implicados.

Componente	División	Orden
Gestor usuarios		1
Sindicador		2
Gestor sesiones		3
Gestor conocimiento		4
Comunicador usuarios		5
Gestor preguntas	Todo menos QTI	6
Traductor QTI		7
Gestor preguntas	Relacionado QTI	8
Motor semántico		9
Administración		10
Gestor redes		11
Manejador servicios		12
Demonio		13
Buscador		14
Web		15

Tabla 19 Prioridad basada en dependencias internas

Dependencias externas.

Las dependencias procedentes de otros proyectos, que afectan al desarrollo de SOLE vienen dadas por el Proyecto GEO, de desarrollo paralelo. GEO será un cliente de los servicios de SOLE, por lo que necesitará la definición de los servicios (en forma de WSDL) de todos los métodos a los que podrá llamar. Para resolver esta dependencia, se desarrollará en primer lugar la interfaz externa del componente *Manejador de servicios*, para más adelante completar su desarrollo con las llamadas a los componentes.

Se creará un prototipo que despliegue los servicios y dé respuestas, aunque la funcionalidad interna aún no esté implementada. Con esta priorización, el orden de los desarrollos queda así:

Componente	División	Orden
Manejador servicios	Despliegue servicios	1
Gestor usuarios		2
Gestor sesiones		4
Gestor conocimiento		5
Comunicador usuarios		6
Sindicador		3
Gestor preguntas	Todo menos QTI	7
Traductor QTI		8

Gestor preguntas	Relacionado QTI	9
Motor semántico		10
Administración		11
Gestor redes		12
Manejador servicios	Conexión con resto componentes	13
Demonio		14
Buscador		15
Web		16

Tabla 20 Prioridad basada en dependencias externas

Estimación de esfuerzo.

En este apartado, se hará una estimación en días del tiempo necesario para el desarrollo de cada componente (o división de componentes). Esta estimación servirá, no sólo para la planificación del desarrollo del proyecto, sino que también puede dar lugar a que se decida modificar el orden de algún hito.

Componente	División	Días
Manejador servicios	Despliegue servicios	2
Gestor usuarios		7
Sindicador		1
Gestor sesiones		7
Gestor conocimiento		5
Comunicador usuarios		1
Gestor preguntas	Todo menos QTI	15
Traductor QTI		7
Gestor preguntas	Relacionado QTI	2
Motor semántico		20
Administración		12
Gestor redes		7
Manejador servicios	Conexión con resto componentes	1
Demonio		4
Buscador		12
Web		20
TOTAL		**123**

Tabla 21 Estimación de esfuerzo

Prioridad.

En este apartado, se aplicarán las prioridades de cada componente (o división funcional) respecto al proyecto. El desarrollo de algunos componentes, aunque no tengan dependencias en cierto punto, puede ser menos urgente que el desarrollo de otros. Para obtener el orden final de desarrollo, se establecerá una prioridad, desarrollándose los componentes en el orden resultante de aquel que tenga mayor prioridad, y que no tenga dependencias previas pendientes de desarrollo. La prioridad variará de 1 a 3, siendo 1 la mayor prioridad, y 3 la menor.

Componente	División	Prioridad	Orden
Manejador servicios	Despliegue servicios	1	1
Gestor usuarios		2	2
Gestor sesiones		2	3
Sindicador		3	4
Gestor preguntas	Todo menos QTI	1	5
Traductor QTI		1	6
Gestor preguntas	Relacionado QTI	1	7
Gestor conocimiento		2	8
Motor semántico		2	9

Comunicador usuarios		3	10
Gestor redes		2	11
Buscador		1	12
Administración		3	14
Demonio		3	15

Tabla 22 Orden de desarrollo basado en prioridad

Para seguir fielmente la Metodología XP, hay que tener en cuenta que cada desarrollo debería ser funcional por sí mismo, por lo que a partir de este punto, se dividirá el desarrollo del componente *Web* en varios, atendiendo cada parte a la interfaz gráfica de cierta funcionalidad. El desarrollo de la parte del componente Web que esté relacionado con otro desarrollo, se hará de forma conjunta; es decir: cuando se desarrolle el Gestor de Usuarios, se desarrollará también su interfaz Web. Del mismo modo, la conexión del manejador de servicios con su componente correspondiente se hará durante el desarrollo del propio componente.

Si analizamos la tabla, observamos que el *Sindicador*, siendo un componente que aporta poco valor al proyecto, se habría de desarrollar de los primeros por la dependencia existente en el *Gestor de Preguntas*, debido a que se sindicará cada pregunta que se dé de alta. Para resolver esta cuestión, se volverá a dividir el desarrollo del *Gestor de Preguntas*, separando la parte que depende de la sindicación, para permitir que el *Sindicador* se desarrolle en una fase más avanzada del proyecto. En la duración estimada en días del desarrollo, se incluye el tiempo dedicado al desarrollo de la parte Web correspondiente.

Componente	División	Orden	Prioridad	Días
Manejador servicios	Despliegue servicios	1	1	2
Gestor usuarios		2	2	10
*Gestor sesiones**		3	2	8
Gestor preguntas	Todo menos QTI, Sindicación	4	1	16
Traductor QTI		5	1	7
*Gestor preguntas**	Relacionado QTI	6	1	3
Gestor conocimiento		7	2	8
*Motor semántico**		8	2	21
Comunicador usuarios		9	3	1
Gestor redes		10	2	8
*Buscador**		11	1	16
Administración		12	3	16
Demonio		13	3	4
Sindicador		14	3	2
*Gestor preguntas**	Sindicación	15	3	1

Tabla 23 Planificación final

Cada hito de entrega se ha identificado con un asterisco (*) al lado del nombre del componente, o división funcional, correspondiente. En resumen, las entregas correspondientes a los desarrollos serán las siguientes:

- Primera entrega (20 días)
  - Manejador de servicios (despliegue de servicios)
  - Gestor usuarios
  - Gestor sesiones
- Segunda entrega (26 días)
  - Gestor preguntas (sin sindicación)

    - Traductor QTI
- Tercera entrega (29 días)
    - Gestor de conocimiento
    - Motor semántico
- Cuarta entrega (25 días)
    - Comunicador usuarios
    - Gestor redes
    - Buscador
- Quinta entrega (23 días)
    - Administración
    - Demonio
    - Sindicador
    - Gestor de preguntas (sindicación)

Estimación de costes.

El objetivo de esta sección es realizar un primer cálculo sobre los costes totales y beneficios que reportará el desarrollo del proyecto.

Los tipos de gastos se dividirán en: gastos de personal, gastos de material fungible, gastos de equipo informático, y otros gastos. En último lugar, se indicarán el beneficio y el riesgo estimado.

Cuando se concluya el desarrollo, se llevará a cabo un análisis de los costes finales, para comprobar la fidelidad de esta estimación.

Gastos de personal.

Gastos derivados de los recursos humanos asociados al proyecto. Se ha estimado que el desarrollo tendrá una duración de 25 semanas, con una dedicación de 40 horas semanales de un único recurso, con unos honorarios de 30€ la hora. Según los datos dados, los gastos de personal vienen reflejados en la siguiente tabla.

Actividad	Semanas Estimadas	Coste Actividad (€)
Análisis	7	8.400
Primera entrega	3	3.600
Segunda entrega	4	4.800
Tercera entrega	5	6.000
Cuarta entrega	4	4.800
Quinta entrega	4	4.800
Total costes	**25**	**32.400**

Tabla 24 Gastos de personal

Gastos de equipo informático.

Los equipos utilizados en el proyecto tienen estimado un periodo de amortización lineal de 5 años. A continuación, se expone una tabla con los precios de compra así como la amortización durante el periodo de duración del proyecto.

Equipos informáticos	Precio de compra (€)	Amortización semanal (€)	Amortización del proyecto
Servidor Desarrollo (1)	600	2,31	57,75
Ordenadores personales (1)	960	3,69	92,25
Impresoras (1)	120	0,46	11,50
Total costes			**161,50**

Tabla 25 Gastos de equipo informático

Gastos de material fungible.

Gastos provenientes de material no inventariable, derivado del desarrollo del proyecto.

Material	Precio unitario	Unidades	Coste total
Cartuchos de tinta	12	3	36,00
Paquetes de folios (500 uds.)	3	2	6,00
Material de oficina	-	-	50,00
Total costes			**92,00**

Tabla 26 Gastos de material fungible

Otros gastos.

Gastos provenientes de conceptos no clasificados anteriormente.

Gastos	Coste mensual (€)	Costes total (7 meses)
Hosting	15	105,00
Dominio	-	9,00
Suministros (agua, luz, teléfono, adsl, gas)	120,00	840,00
Viajes y dietas	-	50,00
Otros gastos	-	100,00
Total costes	**135,00**	**1.104,00**

Tabla 27 Otros gastos

Resumen del presupuesto.

Se presentan todos los gastos calculados anteriormente. Se suma un 20% de beneficio, y un índice medio de riesgo del 15%.

	Estimaciones
1. Gastos de Personal	32.400,00
2. Equipos Informáticos	161,50
4. Material Fungible	92,00
6. Otros	1.104,0
Total	**33.757,50**
Beneficio (20%)	6.571,50
Riesgo (15%)	5.063,63
Subtotal	45.392,63
I.V.A. (16%)	7.262,82
Total presupuesto	**52.655,45**

Tabla 28 Resumen del presupuesto

Primera iteración.

Este documento contendrá la documentación relativa a diseño y pruebas para los componentes *Gestión de Usuarios* y *Gestión de Sesiones*, así como el despliegue de servicios del *Manejador de Servicios*. También se incluye la interfaz Web asociada a estos componentes.

Manejador de servicios (despliegue).

Esta parte de la iteración procurará la definición e implementación de una interfaz de servicios para que los proyectos paralelos puedan tener una base con la que interactuar. En primer lugar, se creará la estructura del proyecto, y posteriormente, se harán las implementaciones mínimas de servicios para que las llamadas devuelvan prototipo de respuesta.

Diseño.

La aplicación estará contenida en un único fichero EAR, de nombre babiecasole-RC1.0.ear para esta iteración. El fichero contendrá, a parte de las librerías necesarias, dos ficheros WAR, siendo uno la interfaz web (sole-web.war), y otro la aplicación web de despliegue de Web Services (sole-webservices.war). También se incluirán dos ficheros EJB JAR, sole-common-1.0.jar y sole-core-1.0.jar, que contendrán los servicios EJB y la implementación del proyecto.

Las aplicaciones se desplegarán en el servidor de desarrollo http://desarrollo.babieca.org, y se les asignarán las siguientes rutas:

- http://sole.desarrollo.babieca.org/sole-ws: Web Services.
- http://sole.desarrollo.babieca.org/sole-web: interfaz Web de usuario.

El manejador de servicios quedará enmarcado en el paquete org.babieca.sole.servicesmanager. Para la funcionalidad necesaria en esta iteración (definir la interfaz de servicios web), se implementarán dos paquetes:

- org.babieca.sole.servicesmanager.services: clases con los métodos de los servicios.
- org.babieca.sole.servicesmanager.vo: clases con los objetos que se devuelven o pasan por parámetro.

Cabe destacar que, aunque se usen objetos en las llamadas a servicios Web, al usar la API de XFire, estos objetos serán transformados en elementos XML en las llamadas SOAP, cuya definición se incluirá en el propio WSDL, para evitar la dependencia con Java.

A continuación se muestra el modelo UML2 usado en ambos paquetes para la generación de código con AndroMDA.

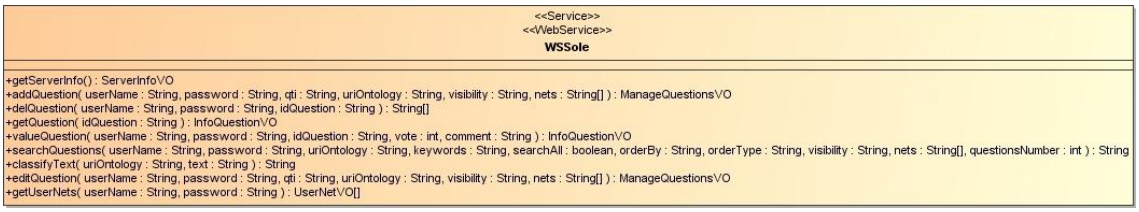

Ilustración 16 Diagrama de clases org.babieca.sole.servicesmanager.services

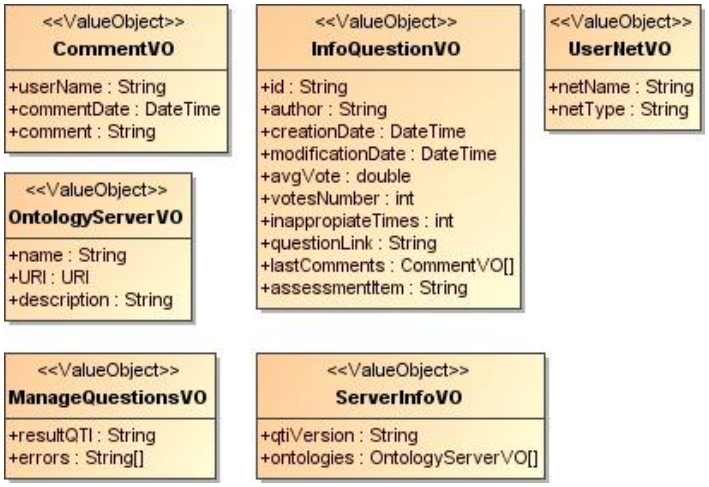

Ilustración 17 Diagrama de clases org.babieca.sole.servicesmanager.vo

Aunque se haya generado un WSDL completo, que incluye documentación y nombres concretos de parámetros, se seguirá obteniendo un WSDL con la petición estándar al servicio en:

http://sole.desarrollo.babieca.org/sole-ws/services/WSSole?wsdl.

El WSDL completo se incluye con la documentación del código fuente.

Pruebas.

Las pruebas del despliegue de servicios consisten en crear una aplicación cliente de servicios, y probar las llamadas a cada uno de ellos. Como en esta iteración tan sólo se implementará un prototipo de la funcionalidad de cada uno, no será necesario comprobar la corrección de los datos obtenidos, tan sólo que el contenido sea meramente válido (siempre o casi siempre, se obtendrá la misma respuesta). Del mismo modo, no hay escenarios de pruebas distintos, ya que no se realizará de momento autenticación ni validación de parámetros, más allá de que estén presentes.

En resumen, las pruebas que se harán consistirán en:

- Desplegar el EAR desarrollado sin errores de despliegue.
- Obtener el WSDL a través de la petición estándar.
- Aplicación cliente auxiliar, que se conecte a los servicios desplegados, y realice peticiones de:
  - Obtener información del servidor.
  - Añadir pregunta.

- Modificar pregunta.
- Borrar pregunta.
- Obtener pregunta.
- Valorar pregunta.
- Buscar preguntas.
- Clasificar texto.
- Obtener redes de usuario.

Gestor Usuarios.

Es el componente que se encarga de la gestión de usuarios, en lo referente a consultas, altas, y modificaciones de datos. Los usuarios administradores son también usuarios al fin y al cabo, y sus acciones serán gestionadas por el Gestor de Usuarios, con la salvedad del alta de nuevos administradores, cuyo ámbito pertenece al componente de Administración, tal y como se verá al estudiar dicho.

Diseño.

Las operaciones soportadas por el Gestor de Usuarios son:

- **Registro Usuario**: da de alta un nuevo usuario en el sistema.
- **Modificar Usuario**: modifica los datos de un usuario.
- **Buscar Usuario**: permitirá encontrar a un usuario del servidor a través de sus datos.

El módulo gestor de usuarios estará contenido en un paquete llamado usersmanager. A su vez, en su interior se encontrarán los subpaquetes *services* para las clases de servicios EJB, y *domain* para las entidades. En este caso no existirá paquete para los Value Objects, ya que se usarán tan sólo los generados por AndroMDA a partir de las entidades.

A continuación se muestran los diagramas de clases de los paquetes que se han de desarrollar:

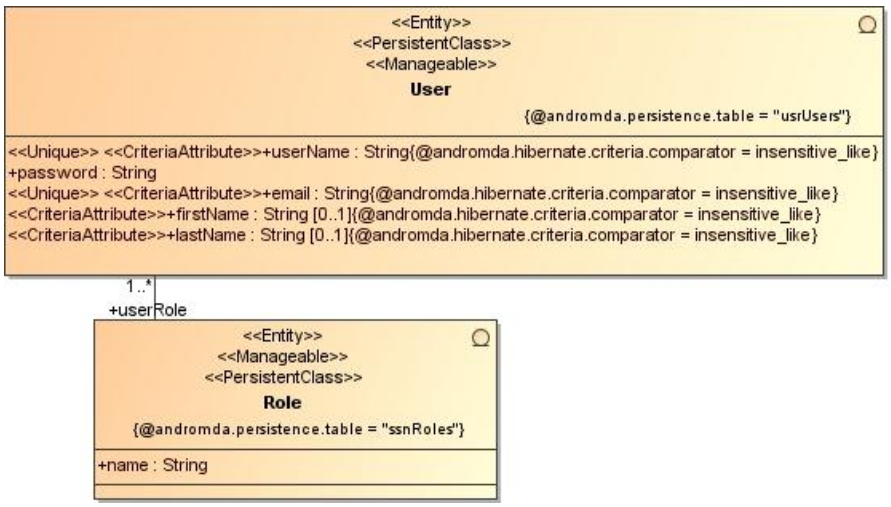

Ilustración 18 Diagrama de clases org.babieca.sole.usersmanager.domain

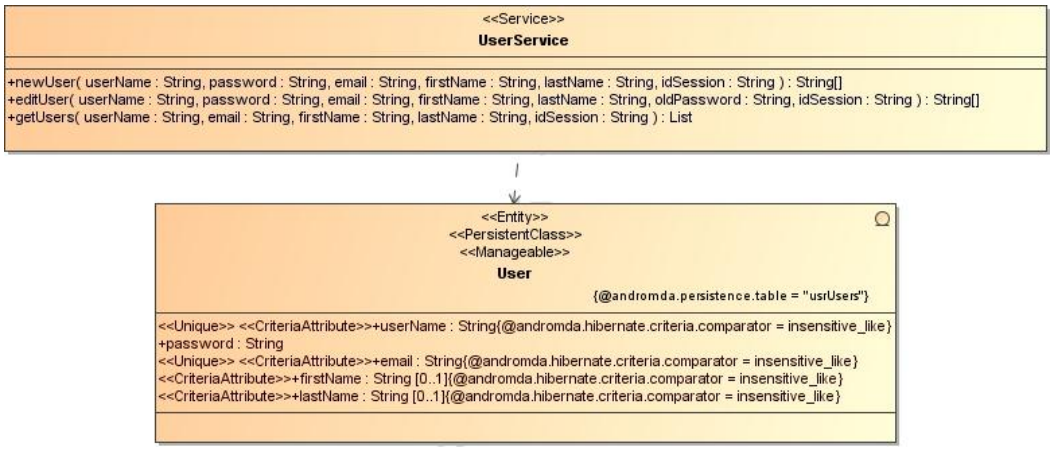

Ilustración 19 Diagrama de clases org.babieca.sole.usersmanager.services

La entidad *Role*, mostrada en el primer diagrama, pertenece al gestor de sesiones. Su aparición en este diagrama de clases se debe tan sólo a la necesidad de expresar la relación entre ambas entidades.

Se observa que, en el segundo diagrama, vuelve a aparecer la entidad User. Esto tan sólo obedece a la necesidad de expresar la dependencia entre la clase UserService y User, con el fin de que AndroMDA genere las clases DAO.

Pruebas.

Las pruebas de aceptación para la gestión de usuarios consistirán en dar de alta un usuario, obtener sus datos, y modificar su información, comprobando que los datos reflejados por la interfaz concuerdan con lo almacenado en la base de datos.

Gestor Sesiones.

Es el componente encargado tanto de la autenticación de usuarios, como de la gestión de permisos.

Diseño.

La gestión de la sesión se hará almacenando en la sesión Http un identificador codificado, a través del cual se podrá obtener el rol con el que se ha iniciado sesión, y por tanto, las acciones a las que se tienen acceso. De esta forma, para poder saltarse las medidas de seguridad, sería necesario conocer el identificador que la aplicación ha asignado a una sesión de un rol con más permisos.

En la base de datos, se almacenará un registro de todas las sesiones iniciadas, con información útil tanto para el funcionamiento de la aplicación, como para un posible rastreo de incidencias.

Los permisos se gestionarán parametrizando las acciones, y realizando siempre una comprobación preliminar, aún en el caso de tratarse de una operación que no requiera autenticación. Esto permite una mayor flexibilidad en el sistema, ya que se podrán modificar los permisos **en tiempo real**, incluso para operaciones que a priori son permitidas a usuarios anónimos.

Las acciones se relacionarán con cada rol que pueda llevarla a cabo. Habrá dos formas de comprobar si se tiene permiso para una cierta acción: a través del

identificador de sesión (en el caso de operaciones a través de Web), y por medio del usuario y contraseña (en el caso de operaciones a través de Web Service).

Las operaciones soportadas por el Gestor de Sesiones serán:

- **Login usuario**: inicia sesión con el usuario dado.
- **Logout usuario**: cierra la sesión del usuario logado, y pasa a ser un usuario anónimo.
- **Validar permiso**: comprueba si el usuario logado, o un usuario anónimo en el caso de no estar logado, está autorizado a realizar una acción dada.
- **Guardar sesiones**: trasladará la información de sesiones almacenada hasta el momento a un fichero de texto plano, y se eliminará del sistema.
- **Cerrar sesiones inactivas**: realizará un logout automático de aquellas sesiones en las que no se haya realizado ninguna operación en un máximo de tiempo parametrizado.
- **Obtener acciones permitidas**: mostrará las acciones que pueden realizarse dependiendo del rol con el que se haya iniciado la sesión (o anónimamente, si no se ha iniciado).
- **Obtener usuario:** devolverá información sobre el usuario con el que se ha iniciado sesión.

Las operaciones masivas sobre los datos de sesiones (cerrar sesiones inactivas, y guardar sesiones), que serán ejecutadas por el componente *Demonio*, serán implementadas a bajo nivel mediante procedimientos almacenados en la base de datos, para asegurar un óptimo rendimiento en el tratamiento de los datos.

El gestor de sesiones estará incluido en el paquete org.babieca.sole.sessionsmanager. A su vez contendrá los subpaquetes *domain, vo,* y *services*. Bajo estas líneas, se muestran los diagramas de clases de cada paquete.

Ilustración 20 Diagrama de clases org.babieca.sole.sessionsmanager.domain

En la entidad Session se observan dos operaciones: saveSessions y closeInactiveSessions. Esta son las operaciones que se ejecutarán mediante procedimientos almacenados. Al ubicarlas en una entidad, AndroMDA genera las clases de implementación DAO desde las que se hace la llamada a la ejecución de los procedimientos almacenados.

Ilustración 21 Diagrama de clases org.babieca.sole.sessionsmanager.vo

En este caso, se opta por implementar un Value Object para expresar los elementos que devuelve la operación de login. El idSession es el identificador encriptado con el que se identifican las sesiones. La forma de obtenerlo será codificando mediante MD5 una cadena formada por el nombre de usuario, el instante en milisegundos de inicio de sesión, y la IP desde la que se ha iniciado la sesión. Para que se diera un caso de identificador duplicado de sesión, sería necesario que el mismo usuario iniciara sesión desde la misma dirección IP y en el mismo momento exacto.

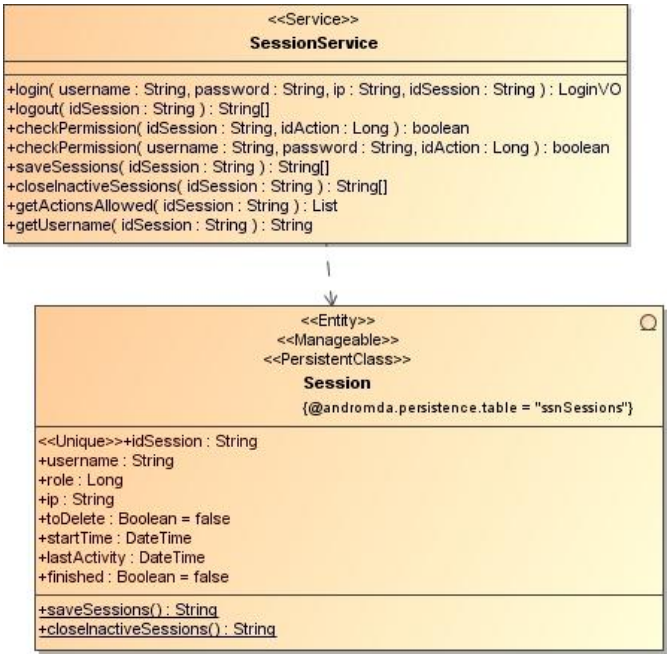

Ilustración 22 Diagrama de clases org.babieca.sole.sessionsmanager.services

En este diagrama, nuevamente se muestra la entidad Session para expresar la dependencia entre ambas clases, y que AndroMDA genere las clases DAO.

Pruebas.

Las pruebas de aceptación para el gestor de sesiones consistirán en:

1. Intentar ejecutar todas las acciones antes de iniciar sesión, comprobado que únicamente permite ejecutar las acciones permitida para el rol anónimo.

2. Iniciar sesión y comprobar el contenido de la tabla ssnSessions.
3. Ejecutar de nuevo la lista de acciones, comprobando que se permiten únicamente las acciones que se correspondan al rol del usuario con el que se ha iniciado sesión.
4. Comprobar que en la tabla ssnSessions se ha actualizado el momento de última actividad.
5. Cerrar sesión, y comprobar que nuevamente se permiten sólo las acciones asociadas al rol anónimo.
6. Asegurarse de que la sesión aparece como cerrada en la tabla ssnSessions.
7. Iniciar de nuevo sesión, y no efectuar ninguna operación durante un tiempo mayor del parametrizado como de máxima inactividad.
8. Ejecutar el procedimiento almacenado ssnCloseInactiveSessions. Comprobar que la sesión se ha cerrado automáticamente en la tabla ssnSessions, y que no se permite ejecutar ninguna operación asociada al rol de usuario registrado.
9. Ejecutar el procedimiento almacenado ssnSaveSessions, y comprobar que ambas sesiones aparecen marcadas para ser borradas.

Estas pruebas deberán repetirse cada vez que se desarrolle un nuevo componente, para asegurar una correcta asignación de permisos.

Interfaz Web.

En esta iteración se desarrolla la interfaz relacionada con la gestión de usuarios y sesiones. Se ha optado por desarrollar el esqueleto completo de la web, para en el futuro tan sólo tener que añadir elementos de nivel inferior.

Diseño.

Se han fijado las siguientes reglas de interfaz en esta iteración:

- El login y logout no afectará a la parte principal del sitio, permitiendo que si el usuario se encuentra con que necesita iniciar sesión para cierta acción, no pierda la ruta de navegación en la que se encontrase.
- El menú de acciones se generará automáticamente al iniciar y cerrar sesión, habilitando tan sólo aquellas operaciones para las que el usuario tenga permiso.
- Se separará, en la medida de lo posible, el código Java del código ZUML en las páginas.
- Se usarán restricciones basadas en expresiones regulares para los campos de entrada de datos, con el fin de evitar el *Cross-Site Scripting*.
- Todos los textos de la interfaz se mostrarán en inglés en primer momento, pero se usará desde el principio un sistema de internacionalización basado en el idioma del navegador. El diccionario que se mostrará por defecto cuando no se encuentre el idioma del navegador será el inglés. Se permitirá que para mensajes del sistema de ZK (“*procesando*”, etc) se muestren los mensajes en el idioma del navegador.

- Los errores relacionados con datos que haya introducido el usuario, siempre se capturarán y se mostrará un error de “valor incorrecto”, asociado con el campo, tabla, o botón donde se hayan introducido los datos, buscando una visibilidad aceptable.
- Se usará una hoja de estilos CSS cuando sea posible y coherente (no se usará en elementos que sólo vayan a usarse una vez).
- Se usará el método de protección “captcha” para el registro de usuarios.

Pantallas.

Se muestran capturas de pantalla de la interfaz, tomadas en el navegador Mozilla Firefox 3.0.11.

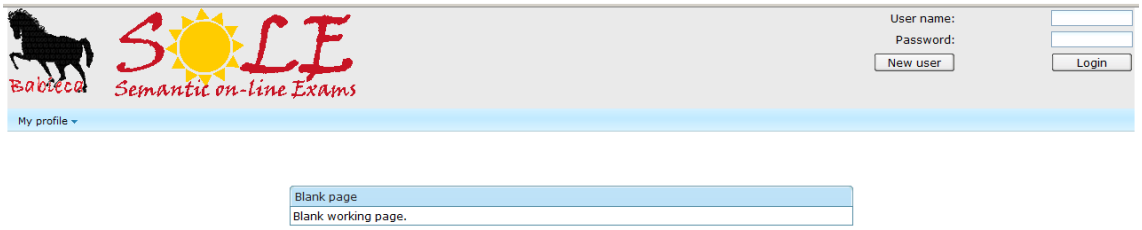

Ilustración 23 Pantalla principal SOLE.

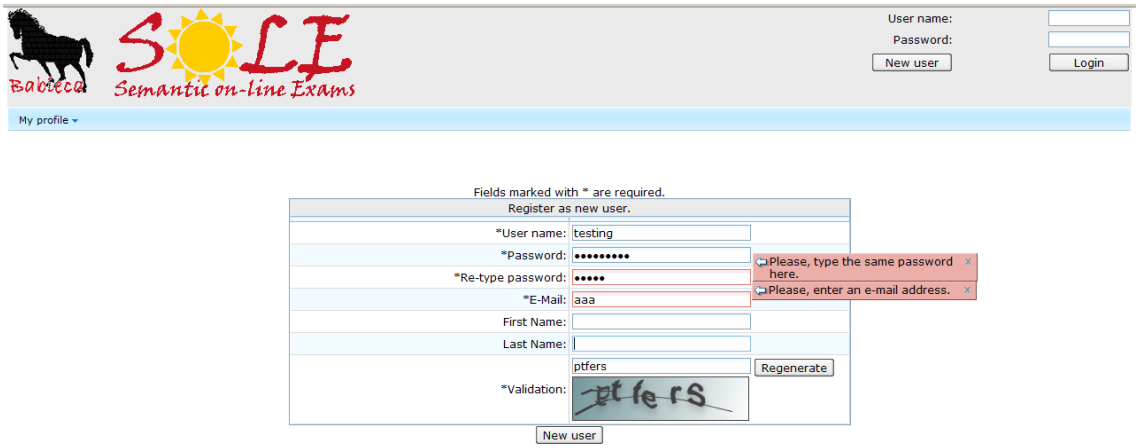

Ilustración 24 Registro de usuario.

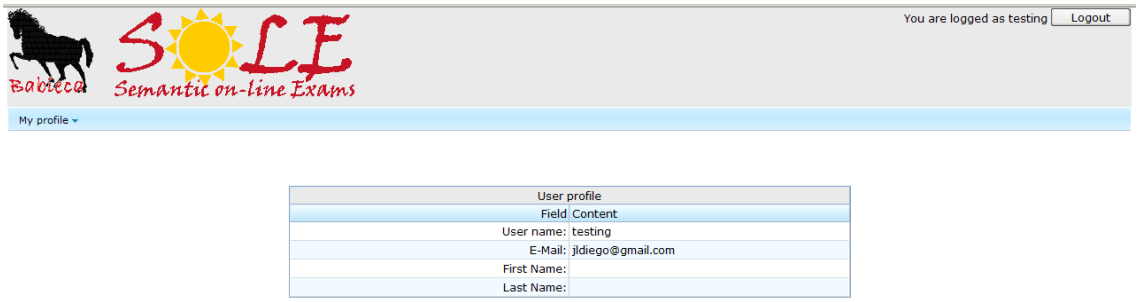

Ilustración 25 Ver perfil de usuario logado.

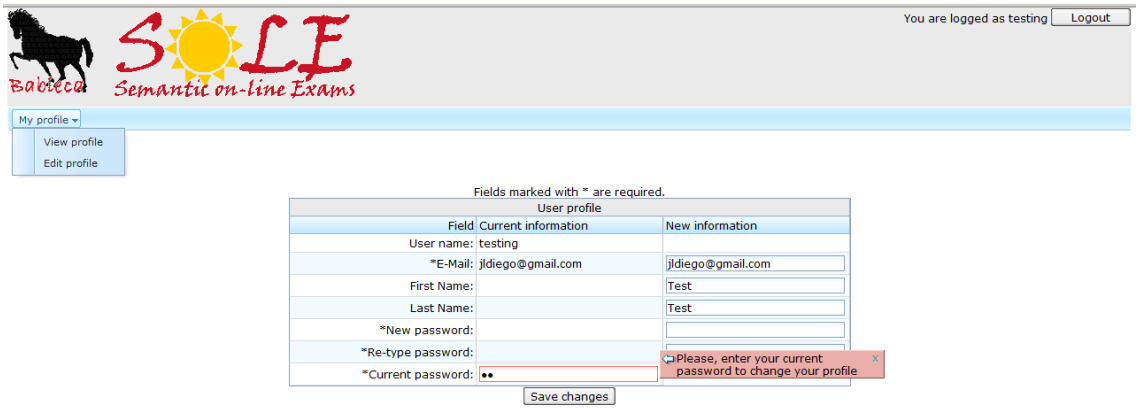

Ilustración 26 Editar perfil de usuario logado.

Segunda iteración.

Este documento contendrá la documentación relativa a diseño y pruebas para los componentes *Gestión de Preguntas* y *Traductor QTI.* También se incluye la interfaz Web asociada a estos componentes, y los servicios web relacionados pasan a usar la implementación real.

La aplicación resultante de esta iteración, estará contenida en el fichero babiecasole-RC2.0.ear, y siendo en las mismas condiciones que la iteración previa.

Revisión de la planificación.

Debido a la magnitud de este componente, ha sido necesario reestructurar tanto la propia iteración, como el alcance de la aplicación prototipo del proyecto.

En primer lugar, se ha decidido fusionar en un único componente el Gestor de Preguntas, y el Traductor QTI. Esto se debe a la estrecha dependencia existente entre ambos, y a que la mayoría de los servicios ofrecidos por el componente Traductor QTI, serían consumidos por el Gestor de Preguntas. Fusionar ambos componentes, provoca que, en la mayoría de las llamadas al Traductor QTI, dicha llamada sea interna, con la consiguiente mejora de rendimiento.

Se han postergado para futuras etapas de desarrollo, fuera del ámbito de la aplicación prototipo presentada con el proyecto, ciertos servicios del Gestor de Preguntas. La causa de dicho aplazamiento, reside en que su implementación no ayudaría a comprobar si la solución propuesta resuelve los problemas encontrados en el Estado del Arte. Los métodos necesarios aparecen en el diseño del componente, pero no se han implementado. Los servicios cuya implementación se aplazará son:

- Modificar pregunta.
- Eliminar pregunta.
- Eliminar comentario.
- Cargar pregunta mediante fichero QTI.

Respecto a siguientes iteraciones, se ha optado por prorrogar a futuras versiones el desarrollo de ciertos componentes, cuya implementación no supondría una base sólida en el análisis de resultados de la solución propuesta. Los componentes que no formarán parte del prototipo serán los pertenecientes a la última iteración de la planificación, es decir:

- Administración.
- Demonio.
- Sindicador.
- Gestor preguntas (sindicación).

Por último, se obviará la implementación de la interfaz Web de los componentes que dispongan de una interfaz a través de servicios Web, al poder ser usados desde el prototipo de aplicación cliente desarrollado por Borja Blanco. Los componentes que, en principio, dispondrán de interfaz Web serán:

- Gestor de conocimiento.

- Gestor Redes.

Gestor de Preguntas.

Es el componente encargado del alta, modificación, y consulta de preguntas, así como de su valoración por parte de los usuarios. Cabe destacar que el componente Gestor de Preguntas ha sido fusionado con el componente Traductor QTI.

Diseño.

La aplicación estará contenida en un único fichero EAR, de nombre babiecasole-1.1.ear para esta iteración. El fichero contendrá, a parte de las librerías necesarias, dos ficheros WAR, siendo uno la interfaz web (sole-web.war), y otro la aplicación web de despliegue de Web Services (sole-webservices.war). También se incluirán dos ficheros EJB JAR, sole-common-1.0.jar y sole-core-1.0.jar, que contendrán los servicios EJB y la implementación del proyecto.

Para el diseño de los servicios principales del Gestor de Preguntas, se ha optado por una arquitectura en capas del siguiente modo:

1. Interfaz (Web, o Servicio Web)
2. Servicio EJB
3. Método DAO
4. Procedimiento almacenado

Tan sólo se usan procedimientos almacenados en los casos que sea necesario ejecutar consultas sobre gran número de tablas de la base de datos, o actualizaciones masivas de datos. El encapsulamiento de ciertos métodos a través de DAO, permite que la ejecución de esos métodos se proteja por transacciones de la Base de Datos. Esta protección de la transaccionalidad, se debe a que, en la mayoría de los casos, se producirán inserciones y actualizaciones en multitud de tablas de la base de datos. Un error cuando ya se hubiera producido alguna actualización o inserción, provocaría que los datos almacenados quedaran en un estado inconsistente.

Para los servicios que no requieran actualizaciones/inserciones en varias tablas, o manejo de datos masivos, tan sólo se usarán los dos primeros niveles.

Los servicios implementados en esta versión del prototipo, para el Gesto de Preguntas son:

- **Añadir pregunta**: guarda en el sistema una pregunta con los datos aportados. Será necesario ser usuario registrado para ejecutarlo.
- **Obtener pregunta**: obtiene toda la información relacionada con una pregunta.
- **Consultar preguntas propias**: obtiene una lista de las preguntas cuyo autor sea el propio usuario.
- **Votar pregunta**: añade un voto a una pregunta. Los votos serán de 0 a 10, pudiéndose además marcar la pregunta como *inapropiada* (voto -1). Si un usuario vuelve a votar una pregunta, el efecto será modificar su voto anterior.

- **Comentar pregunta**: añade un comentario a una pregunta. Un usuario podrá comentar cualquier pregunta las veces que quiera.
- **Es autor**: acredita que cierto usuario es autor de cierta pregunta. Servicio auxiliar para ser usado por otros.
- **Obtener QTI pregunta**: obtiene el contenido de una pregunta (elemento AssessmentItem) en formato QTI 2.1. Este servicio pertenecería al antiguo componente Traductor QTI.
- **Obtener QTI preguntas**: obtiene el contenido de una lista de preguntas (elemento AssessmentTest) en formato QTI 2.1. Este servicio pertenecería al antiguo componente Traductor QTI.

Dados los diversos orígenes que puede tener una llamada a un servicio (a través de interfaz Web, o Web Service), el diseño interno de la capa de implementación de servicios se ha estructurado en varios métodos para un servicio que llaman a un núcleo común.

La necesidad de la división viene impuesta, principalmente, por el diferente tratamiento de las sesiones en cada una de las interfaces. Mientras que las sesiones en Web son persistentes, y se asigna una sesión a cada usuario, en las llamadas desde Web Service, no existe dicha sesión, y siempre se ha de indicar el nombre de usuario y contraseña.

El componente Gestor de Preguntas, se ha dividido en tres paquetes:

- org.babieca.sole.questionsmanager.domain
- org.babieca.sole.questionsmanager.services
- org.babieca.sole.questionsmanager.vo

Al igual que en componentes anteriores, el paquete *services* contendrá las clases relacionadas con la implementación de servicios; el paquete *vo* las clases adicionales de objetos que se devuelvan en métodos, o se reciban como parámetros, y el paquete *domain*, las clases relacionadas con las entidades y DAO.

Bajo estas líneas, se muestran los diagramas de clases de cada paquete.

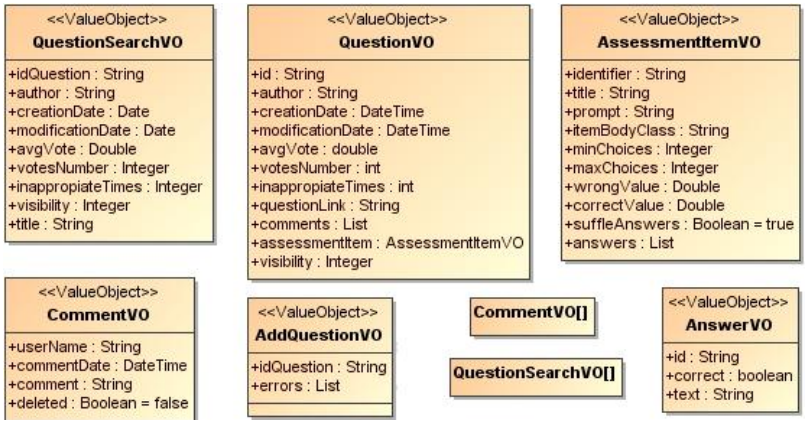

Ilustración 27 Diagrama de clases org.babieca.sole.questionsmanager.vo

Se puede observar cómo ha sido necesario diferenciar entre varios objetos distintos, del tipo Value Object, para ciertas operaciones. En los casos que se ha

optado por la creación de un nuevo objeto, se ha debido a que los objetos creados por defecto por AndroMDA para cada entidad no eran suficientes. En el caso de las preguntas (entidad Question, como se verá a continuación), un objeto QuestionValueObject contiene exclusivamente los datos de la entidad pregunta, pero en el caso de ser necesario obtener a la vez datos sobre el elemento de evaluación (AssessmentItem), el uso de QuestionValueObject obliga a una nueva consulta sobre los datos, repercutiendo en el rendimiento. La creación de objetos agregados permite una única obtención de datos, reduciendo el acoplamiento de niveles.

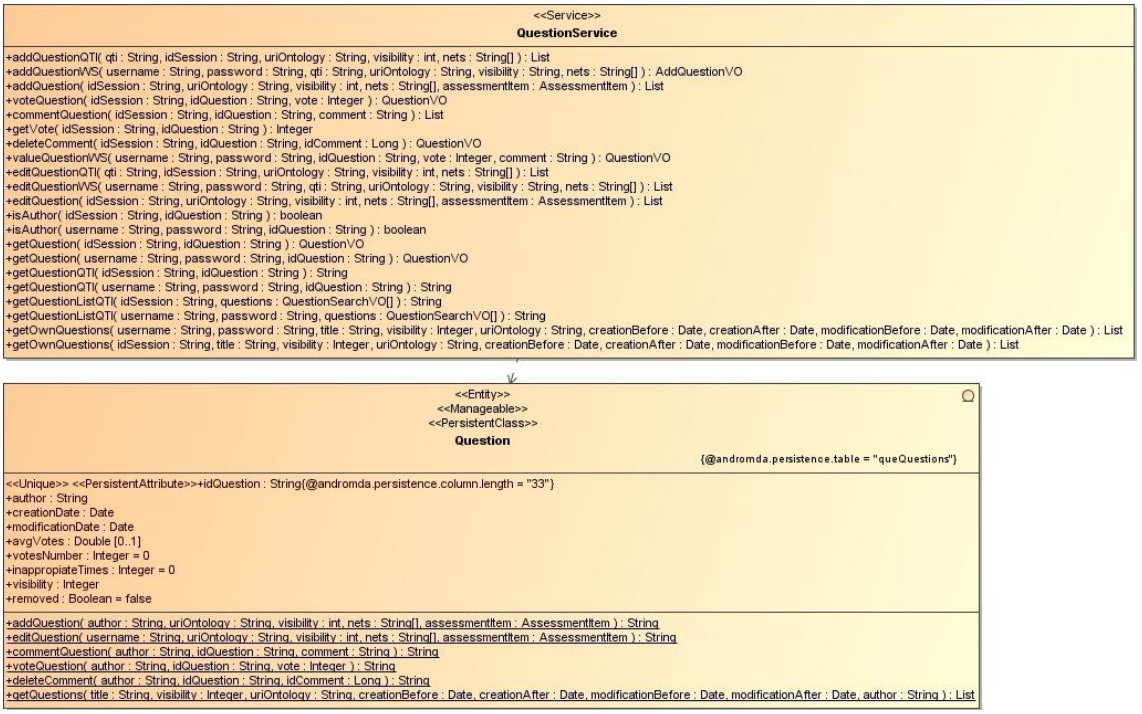

Ilustración 28 Diagrama de clases org.babieca.sole.questionsmanager.services

El paquete *services* contiene una única clase (QuestionService), aunque se muestra en el diagrama la dependencia con la entidad *Question*, que es necesaria para poder acceder al DAO de Question desde QuestionService. Cada método de QuestionService corresponde a un servicio del Gestor de Preguntas. Es necesario puntualizar que, cada uno de los servicios enumerados anteriormente corresponde a uno o varios métodos de esta clase, tal y como se describió con anterioridad. Algunos de los métodos que aparecen en el diagrama no han sido implementados, por las cuestiones expuestas en el apartado *Revisión de la planificación* de esta iteración.

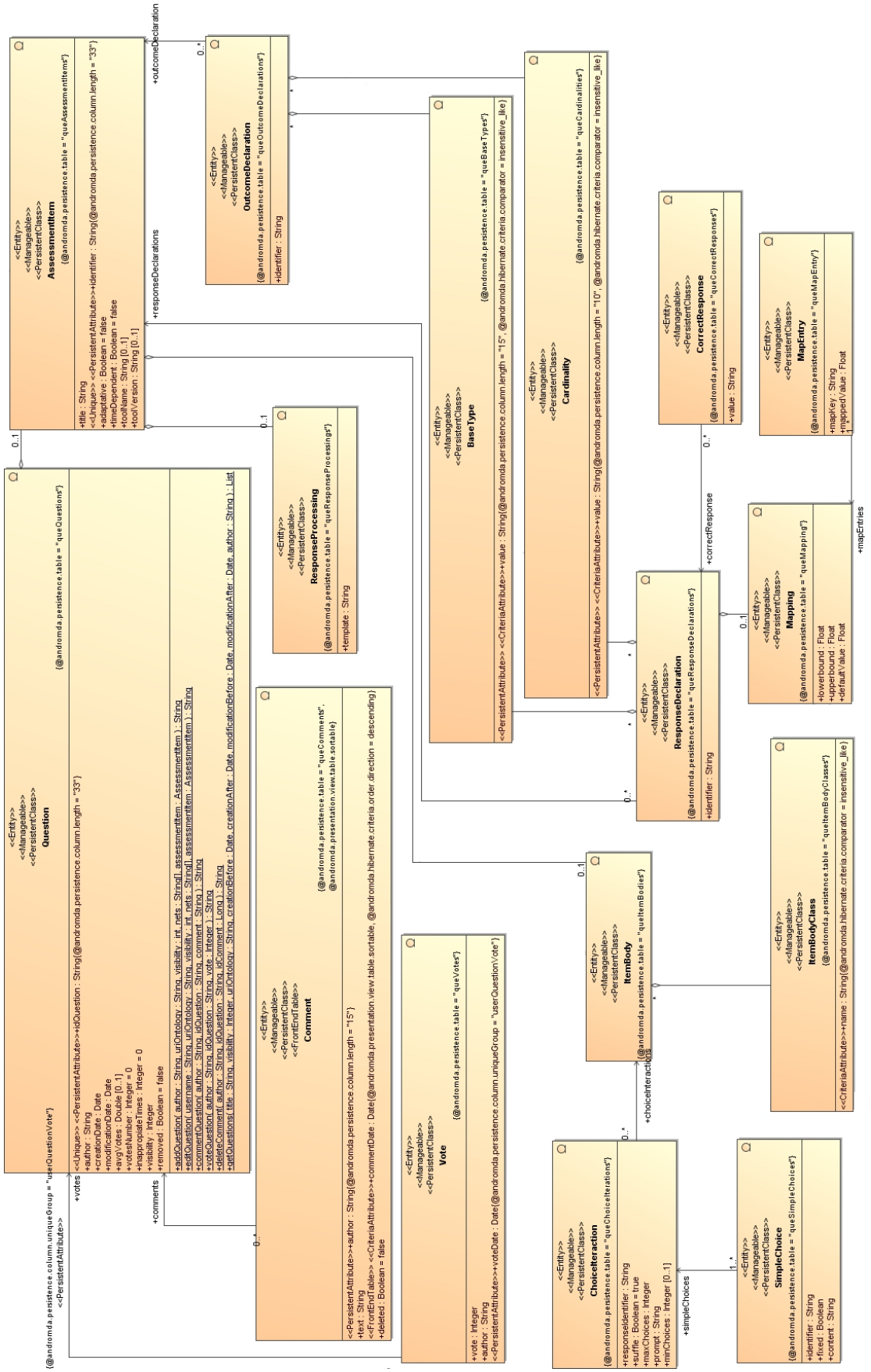

Ilustración 29 Diagrama de clases org.babieca.sole.questionsmanager.domain

El paquete *domain* es el más extenso del componente Gestor de Preguntas. Uno de los desafíos planteados en el desarrollo de esta iteración es el almacenamiento de los datos aportados por el estándar QTI 2.1 para su uso en la aplicación. Se desestima el almacenamiento de un XML que siguiera el esquema de QTI, por razones de rendimiento, y de limitación de contenido a 8000 caracteres. También queda desechada la opción de crear una única entidad para las preguntas que contenga únicamente la información de QTI que se utilizará en esta versión (preguntas tipo test multi-respuesta), debido a que la escalabilidad de la aplicación quedaría mermada en un grado demasiado elevado (sería necesario modificar toda la aplicación en el momento que se incorporara algún nuevo tipo de pregunta).

Se ha optado por crear una entidad *Question*, que contiene toda la meta-información de la pregunta que usa la aplicación, y que no está contenida en un AssessmentItem de QTI. Por otro lado, se define la entidad *AssessmentItem*, y las demás entidades relacionadas, que contienen la información del estándar de forma fiel. Cada *Question* tiene referenciado su correspondiente *AssessmentItem*, de modo que se separa la información propia de la aplicación, de aquella que viene especificada en QTI.

En otras palabras, se ha trasladado la parte del esquema QTI 2.1 que va a usarse a un esquema de entidades equivalente.

Cuando se pretenda añadir un nuevo tipo de pregunta, tan sólo habrá que crear, en el peor de los casos, nuevas entidades que contendrán la información que no se almacena en esta versión, y modificar los métodos correspondientes, no siendo necesario el cambio de la estructura de la base de datos.

Por otro lado, puede comprobarse que en la entidad *Question*, ya se almacenan datos estadísticos sobre los votos de la pregunta. Se ha optado por esta solución para evitar que cada vez que se quieran leer dichos datos, sea necesario consultar todos los datos sobre votos de esa pregunta. Cada vez que se vota una pregunta, los datos estadísticos se actualizan.

Pruebas.

El buen funcionamiento de este componente repercutirá en el correcto funcionamiento del resto de la aplicación. Será necesario un exhaustivo proceso de testing para verificar que se cumplen los requisitos funcionales.

Se diferencian dos tipos de pruebas: desde la interfaz Web, y desde el cliente de Web Services. Para realizar las pruebas de cliente Web Services, se usará la aplicación GEO, desarrollada por Borja Blanco. Tras cada prueba, se recomienda acceder a la información de la pregunta a través de la interfaz Web para comprobar los cambios.

Las pruebas de aceptación que se realizarán desde el cliente de Web Services serán:

- Obtener información de preguntas
  - Existentes y públicas (visible)
  - Existentes, privadas del propio usuario (visible)

    - Existentes, privadas de otro usuario (no visible)
    - No existentes (no visible)
- Dar de alta varias preguntas variando niveles de visibilidad, y usando:
    - QTI 2.1 válido, y soportado por SOLE.
    - QTI 2.1 válido, con elementos no soportados.
    - QTI con errores.
- Comentar preguntas
    - Existentes
    - No existentes
    - Existentes, pero no visibles
- Votar preguntas
    - Existentes y no votadas por el usuario
    - Existentes y ya votadas por el usuario
    - No existentes
    - Existentes, pero no visibles

Las pruebas de aceptación que se realizarán desde la interfaz Web serán:

- Obtener información sobre las preguntas del usuario.
    - Antes de dar de alta preguntas.
    - Habiendo dado de alta preguntas.
    - Filtros de preguntas.
- Obtener información de preguntas
    - Existentes y públicas (visible)
    - Existentes, privadas del propio usuario (visible)
    - Existentes, privadas de otro usuario (no visible)
    - No existentes
- Dar de alta varias preguntas variando los datos de entrada para comprobar el comportamiento de la aplicación.
- Comentar preguntas variando los datos de entrada para comprobar el comportamiento de la aplicación.
- Votar preguntas
    - No votadas por el usuario
    - Ya votadas por el usuario

Interfaz Web.

En esta iteración se desarrolla la interfaz relacionada con la gestión de preguntas. Se ha cambiado la hoja de estilos de la Web, siguiendo los consejos de un usuario experto.

Pantallas.

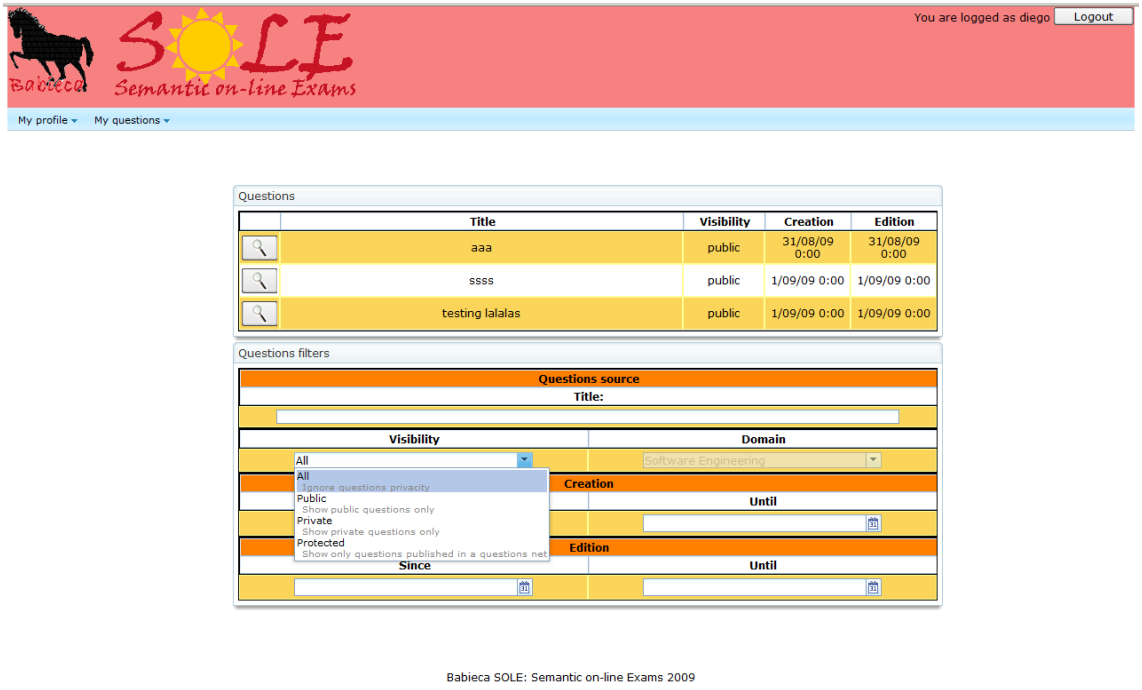

Ilustración 30 Pantalla de búsqueda de pregunta propias.

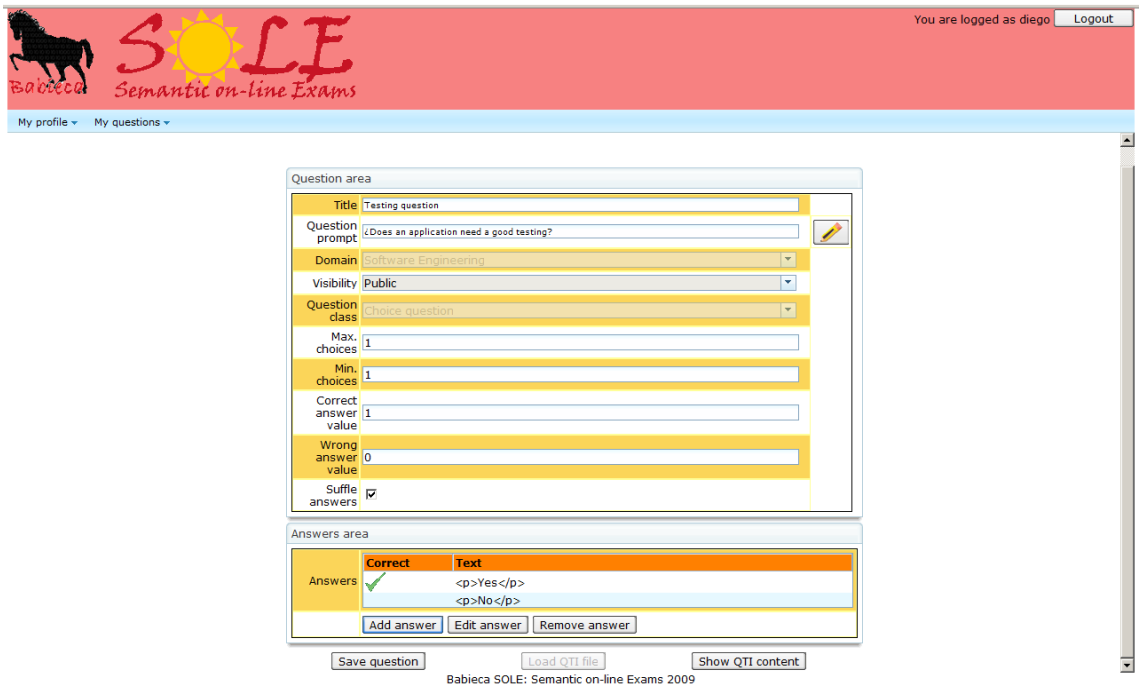

Ilustración 31 Pantalla de añadir pregunta.

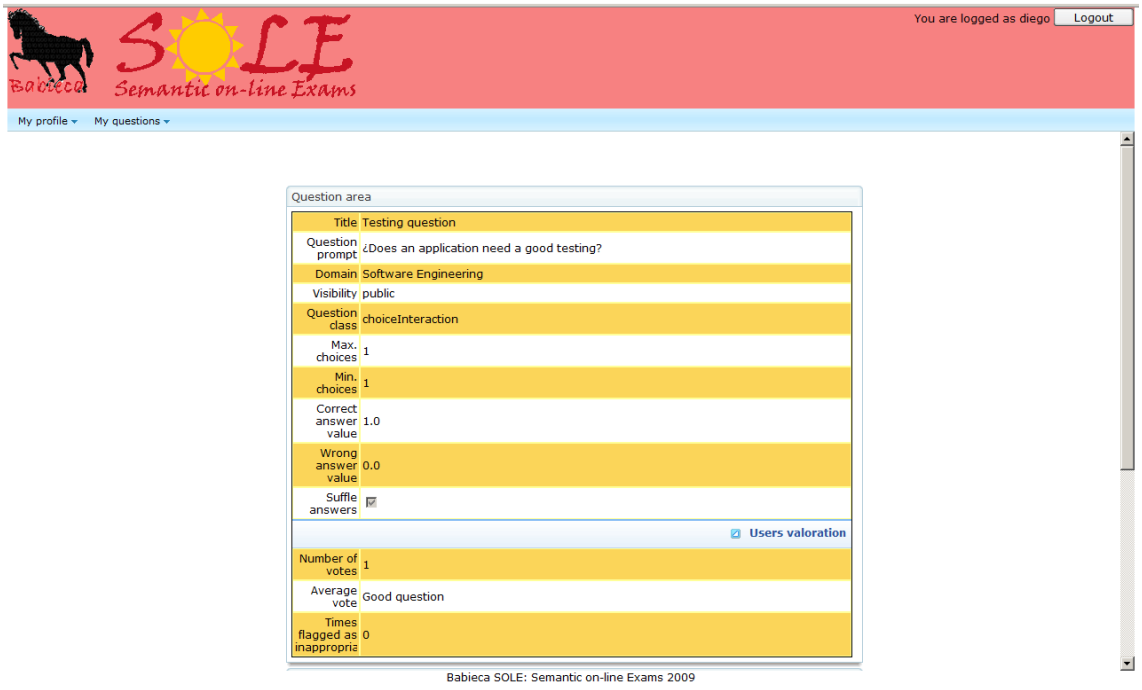

Ilustración 32 Pantalla de ver pregunta (Question area).

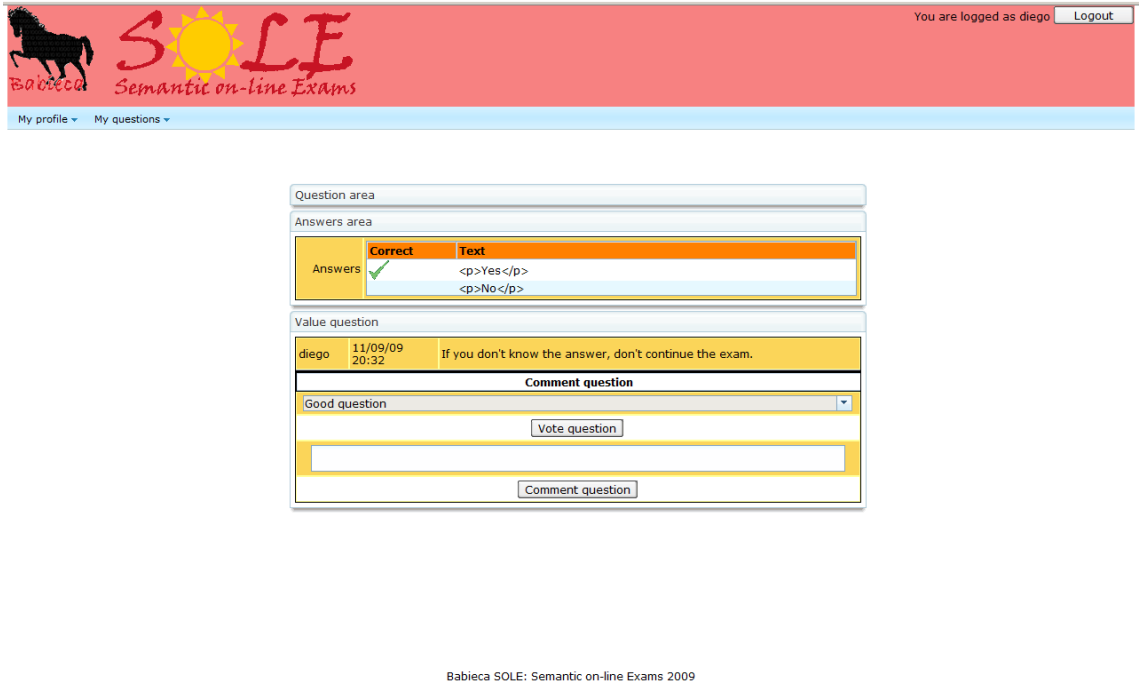

Ilustración 33 Pantalla de ver pregunta (Answers area y Value Question).

Tercera iteración.

Revisión de la planificación.

Dada la inamovilidad de la fecha de presentación de resultados, y la desviación de la planificación, será necesario revisar nuevamente la iteración pendiente (la cuarta).

Dicha iteración contendrá únicamente el *Buscador Semántico*, postergándose para futuros desarrollos tanto la gestión de redes de usuarios, como el comunicador de usuarios, ya que su presencia no influirá en las conclusiones finales del proyecto.

Al eliminar la gestión de redes de usuarios, no será necesario el desarrollo de ninguna interfaz de usuario en la cuarta iteración.

Motor Semántico.

Es el componente encargado de la realización de operaciones relacionadas con Web Semántica a la hora de interpretar preguntas, orientada a su indexación en el buscador.

La aplicación estará contenida en un único fichero EAR, de nombre babiecasole-1.2.ear para esta iteración. El fichero contendrá, a parte de las librerías necesarias, dos ficheros WAR, siendo uno la interfaz web (sole-web.war), y otro la aplicación web de despliegue de Web Services (sole-webservices.war). También se incluirán dos ficheros EJB JAR, sole-common-1.0.jar y sole-core-1.0.jar, que contendrán los servicios EJB y la implementación del proyecto.

Para el diseño de los servicios principales del Gestor de Preguntas, se ha optado por una arquitectura en capas del siguiente modo:

1. Interfaz (Web, o Servicio Web)
2. Servicio EJB
3. Método DAO
4. Procedimiento almacenado

Tan sólo se usan procedimientos almacenados en los casos que sea necesario ejecutar consultas sobre gran número de tablas de la base de datos, o actualizaciones masivas de datos. El encapsulamiento de ciertos métodos a través de DAO, permite que la ejecución de esos métodos se proteja por transacciones de la Base de Datos. Esta protección de la transaccionalidad, se debe a que, en la mayoría de los casos, se producirán inserciones y actualizaciones en multitud de tablas de la base de datos. Un error cuando ya se hubiera producido alguna actualización o inserción, provocaría que los datos almacenados quedaran en un estado inconsistente.

Para los servicios que no requieran actualizaciones/inserciones en varias tablas, o manejo de datos masivos, tan sólo se usarán los dos primeros niveles.

Diseño.

Los servicios implementados en esta versión del prototipo, para el Motor Semántico son:

- **Clasificar texto**: analizará el texto indicado, según la ontología dada, para decidir qué clases de la ontología, e instancias, son las más significativas.
- **Clasificar pregunta**: aplicará la clasificación de texto al contenido de una pregunta, para obtener las clases de la ontología dada, e instancias, que más se ajustan a ella.
- **Filtrar stop-words**: eliminará las palabras superfluas del idioma seleccionado en un texto dado.

Dados los diversos orígenes que puede tener una llamada a un servicio (a través de interfaz Web, o Web Service), el diseño interno de la capa de implementación de servicios se ha estructurado en varios métodos para un servicio que llaman a un núcleo común.

La necesidad de la división viene impuesta, principalmente, por el diferente tratamiento de las sesiones en cada una de las interfaces. Mientras que las sesiones en Web son persistentes, y se asigna una sesión a cada usuario, en las llamadas desde Web Service, no existe dicha sesión, y siempre se ha de indicar el nombre de usuario y contraseña.

El componente Motor Semántico, se ha dividido en tres paquetes:

- org.babieca.sole.semanticengine.domain
- org.babieca.sole.semanticengine.services
- org.babieca.sole.semanticengine.vo

Al igual que en componentes anteriores, el paquete *services* contendrá las clases relacionadas con la implementación de servicios; el paquete *vo* las clases adicionales de objetos que se devuelvan en métodos, o se reciban como parámetros, y el paquete *domain*, las clases relacionadas con las entidades y DAO.

Bajo estas líneas, se muestran los diagramas de clases de cada paquete.

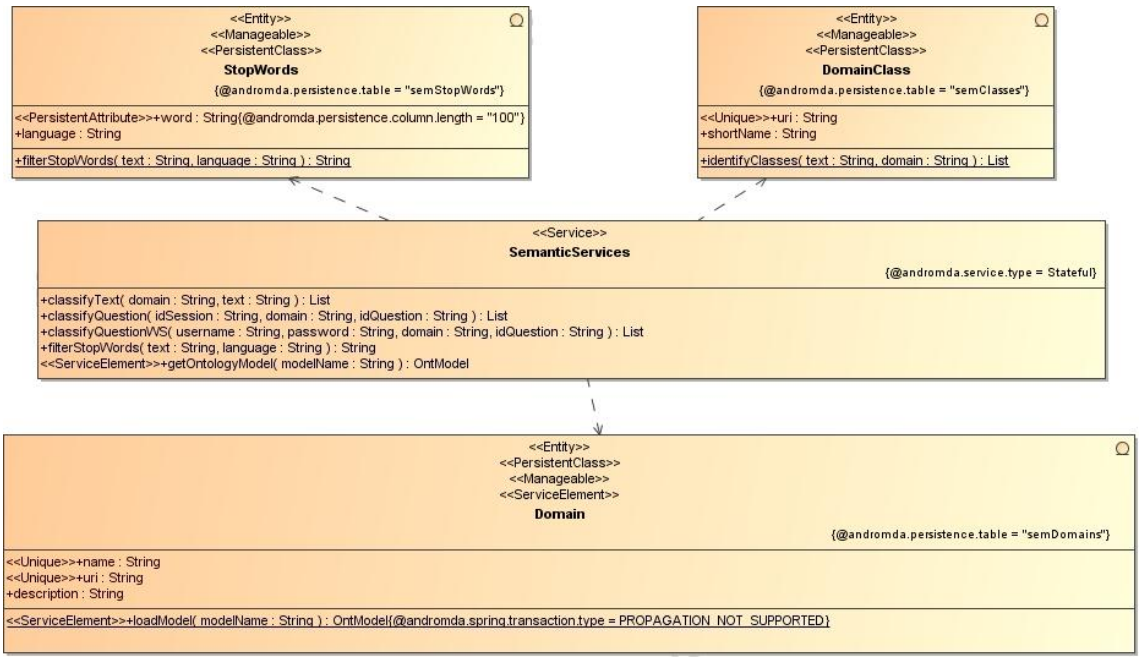

Ilustración 34 Diagrama de clases org.babieca.sole.semanticengine.services

El paquete *services* contiene una única clase SemanticServices, estereotipada como servicio. Se adjuntan las dependencias a las entidades Domain, StopWords y DomainClass para tener acceso a sus métodos DAO.

Ha sido necesario que el servicio getOntologyModel sea Stateful, con el fin de evitar en la medida de lo posible las cargas innecesarias de las ontologías en modelos del API Jena.

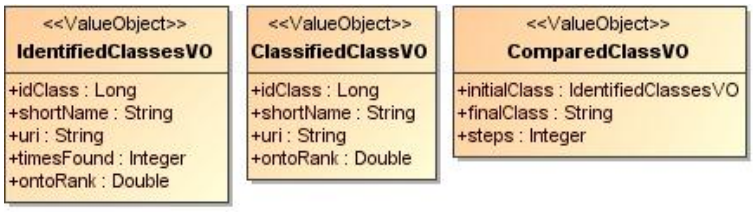

Ilustración 35 Diagrama de clases org.babieca.sole.semanticengine.vo

El paquete *vo* contiene las clases *ValueObject* auxiliares que han sido necesarias para el manejo de valores devuelto por los métodos de negocio en algunos casos, y para el algoritmo de identificación de clases, que se detallará más adelante.

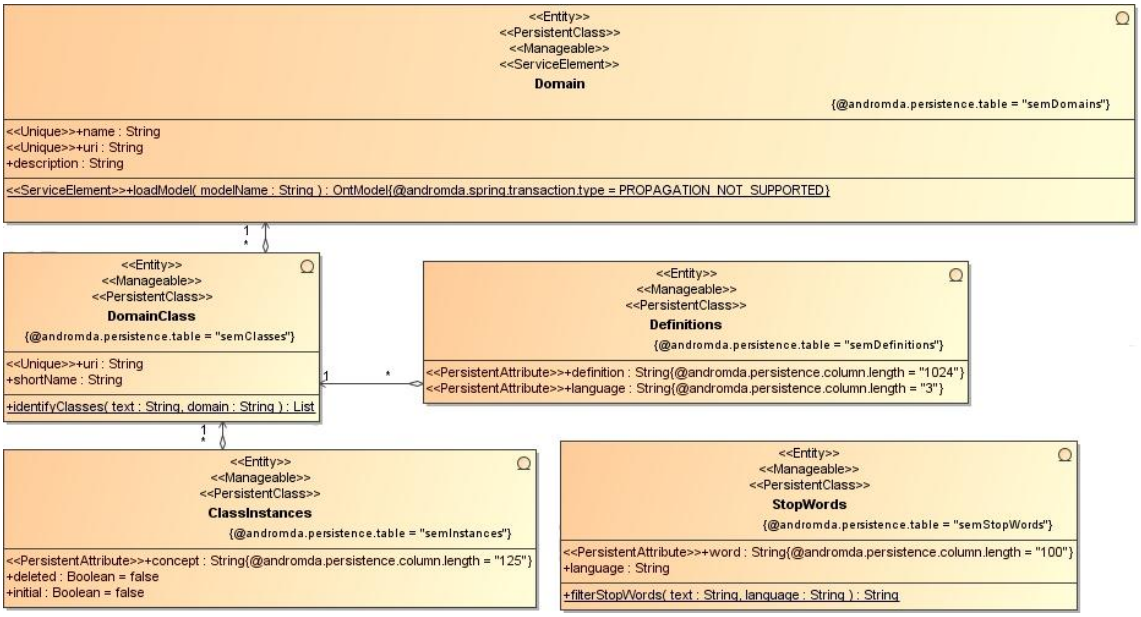

Ilustración 36 Diagrama de clases org.babieca.sole.semanticengine.domain

En el paquete *domain* se incluyen las entidades necesarias para el funcionamiento del motor semántico. El contenido inicial de la base de datos para *semDomain*, *DomainClass*, *Definitions* y *ClassInstances* se obtiene a través de la rutina de asimilación de ontología. Las StopWords se han obtenido de un fichero de texto estándar de Stop Words en inglés.

Estas clases se usan para el algoritmo de identificación de clases. No deben confundirse con las clases propias del modelo, generadas por Jena, que no se incluyen en el diagrama por no formar parte del diseño interno del motor semántico.

Algoritmo de clasificación.

Se trata de la rutina fundamental que permite un buen funcionamiento de clasificación. Recibe como datos de entrada el texto que debe clasificarse, junto con el dominio (ontología) por el que se clasificará, devolviendo una lista con las URIs de las clases de la ontología que se consideran más relevantes, junto a la puntuación de relevancia otorgada (a partir de ahora, *OntoRank*).

En primer lugar, se filtran las stop-words, signos de puntuación, etc, del texto de entrada. Este proceso tiene lugar en programación a nivel de base de datos para hacerlo lo más eficiente posible.

Una vez filtrado el texto, se procede a la identificación de las clases de la ontología. Este proceso se realiza buscando coincidencias de las instancias de cada clase en el texto filtrado. Al igual que el filtrado de stop words, se ejecuta mediante procedimiento almacenado en la base de datos, para optimizar el rendimiento del manejo de datos masivos. A cada clase detectada, se le asigna un *OntoRank*, que en la identificación de clases viene dado para cada una por el tanto por diez de apariciones de instancias, respecto al total de palabras del texto filtrado.

Según se haya configurado en el sistema, será necesario reducir el número de clases detectadas. Tras las pruebas de desarrollo, se ha optado por limitar a 3 el número de clases que se devolverán.

El proceso de reducción de clases supone un cuello de botella en el rendimiento del algoritmo de clasificación, debido a la dimensionalidad de los datos manejados.

En primer lugar, se obtiene la matriz semántica de clases, donde las columnas son clases detectadas, y las filas corresponden a la superclase de la ontología de la clase detectada, llegando hasta el nodo raíz, o hasta el límite de saltos que se haya configurado (2 en el prototipo desarrollado).

Seguidamente, se buscan coincidencias en las diferentes columnas entre las clases, buscando aquellos nodos que, estando en un nivel superior, permitan referirse a dos clases detectadas. El OntoRank de la clase resultante, corresponderá a la suma de los OntoRank previos, tras aplicarles una penalización por nivel ascendido en el árbol (25% en el prototipo desarrollado). La penalización es, al igual que el resto de parámetros del algoritmo, configurable en tiempo real.

Esta forma de reducción de clases está justificada por las normas de unidad temática en los textos de lenguaje natural. Es más probable que en un texto se mencionen temas relacionados, que temas completamente distintos, por lo que subiendo niveles en el árbol de clases de la ontología, se espera una reducción significativa del número de clases detectadas, sin perder rigurosidad en los datos.

Tras aplicar la reducción, si aún existieran más clases de las configuradas como máximas para el resultado, se devolverán las N clases con mayor OntoRank resultante de la reducción.

Al aplicar el algoritmo a una pregunta almacenada en el servidor, para la primera versión, se concatenarán el texto del título, enunciado y preguntas. Para

versiones futuras, se prevé la aplicación de distintos pesos a las clases detectadas en cada elemento, y una posterior recombinación de clases según los nuevos pesos.

Pruebas.

El buen funcionamiento de este componente repercutirá en el correcto funcionamiento del componente *Buscador Semántico*, que será desarrollado en la siguiente iteración. Cada prueba consistirá en la clasificación de textos, probando distintos valores en los parámetros de configuración.

Para las pruebas de clasificación, se ha desarrollado una interfaz de pruebas, accesible desde la URL http://sole.desarrollo.babieca.org/sole-web/test.zul.

Gestor del conocimiento.

Es el componente encargado de la mejora continua del motor semántico a través de la experiencia de los usuarios.

Cada usuario puede sugerir instancias de las clases de las ontologías manejadas (en esta versión, *SEOntology*). A su vez, puede valorar las instancias sugeridas por otros usuarios, decidiendo si son correctas, incorrectas, o si no está seguro.

Periódicamente, el *Demonio* asimilará las sugerencias, y formarán parte del motor semántico aquellas que tengan una valoración suficiente. Del mismo modo, dejarán de usarse para la detección de clases aquellas cuya valoración hubiera descendido por debajo del límite parametrizado.

Diseño.

Los servicios implementados en esta versión del prototipo, para el Gestor del Conocimiento son:

- **Sugerir concepto**: el usuario propondrá un concepto como instancia de una clase de una ontología.
- **Valorar concepto**: el usuario evaluará la fiabilidad de un concepto como instancia de una clase de una ontología.
- **Asimilar conceptos**: se establecen como instancias de las clases de dominios aquellas instancias sugeridas por los usuarios, y valoradas positivamente por los mismos.

En este componente, no se produce ninguna llamada con origen en Web Service, por lo que no será necesaria la descomposición de servicios según el origen.

El componente Gestor del Conocimiento, se ha dividido en tres paquetes:

- org.babieca.sole.knowledgemanager.domain
- org.babieca.sole.knowledgemanager.services
- org.babieca.sole.knowledgemanager.vo

Al igual que en componentes anteriores, el paquete *services* contendrá las clases relacionadas con la implementación de servicios; el paquete *vo* las clases adicionales de objetos que se devuelvan en métodos, o se reciban como parámetros, y el paquete *domain*, las clases relacionadas con las entidades y DAO.

A continuación, se muestran los diagramas de clases de cada paquete.

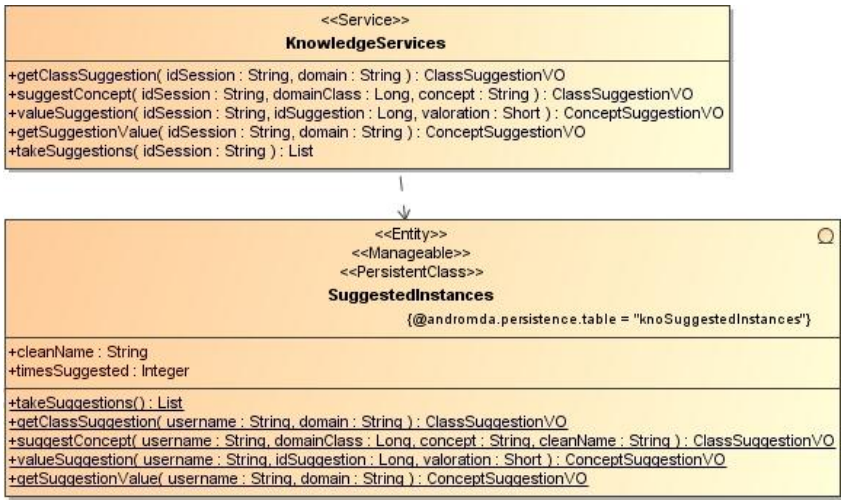

Ilustración 37 Diagrama de clases org.babieca.sole.knowledgemanager.services

El paquete *services* contiene una única clase KnowledgeServices, estereotipada como servicio. Se adjuntan las dependencias a la entidad SuggestedInstances para tener acceso a sus métodos DAO.

Para los métodos correspondientes a los servicios *Sugerir Concepto*, y *ValorarConcepto*, se usan dos métodos: uno que obtiene el elemento sobre el que se va a efectuar la acción, y otro que la realiza, y a su vez devuelve el siguiente objeto sobre el que efectuar la misma acción. El método de asimilación de sugerencias no necesita preparación para su ejecución.

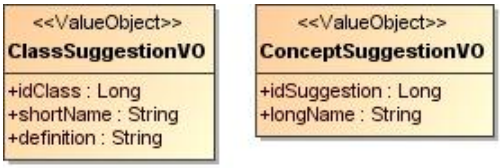

Ilustración 38 Diagrama de clases org.babieca.sole.knowledgemanager.vo

El paquete *vo* contiene las clases *ValueObject* auxiliares que han sido necesarias para el manejo de valores devuelto por los métodos de negocio.

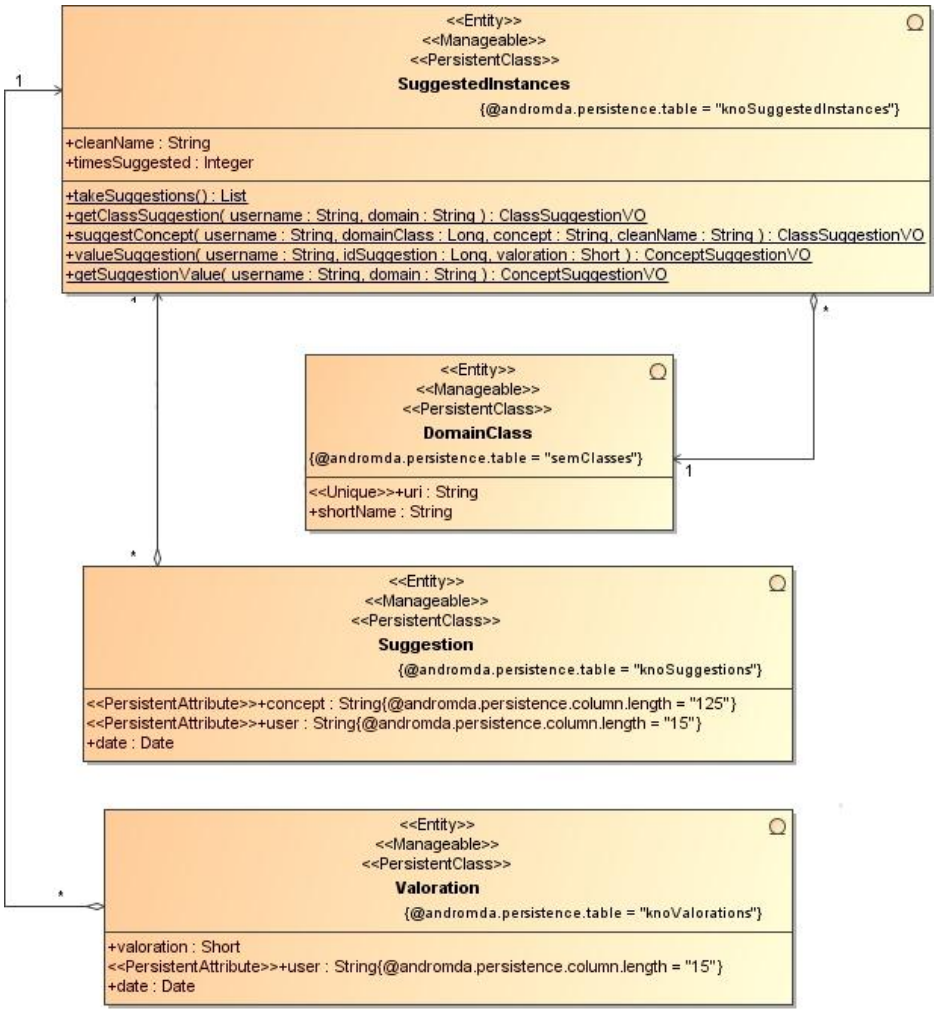

Ilustración 39 Diagrama de clases org.babieca.sole.knowledgemanager.domain

En el paquete *domain* se incluyen las entidades necesarias para el funcionamiento del gestor del conocimiento.

El almacenamiento de sugerencias y valoraciones se ha diseñado como un master-detail, donde, para cada sugerencia genérica, se guardan en distintas entidades sus valoraciones, y las sugerencias en sí de los usuarios. En el caso de las sugerencias de usuarios, esto se debe a que distintos usuarios pueden sugerir la misma instancia, de forma exacta, o con variaciones de stop words. En la entidad maestra, se almacena el concepto filtrado de stop words, y en la tabla de sugerencias, el concepto exacto sugerido por el usuario.

Las valoraciones podrán ser positivas, negativas, o neutras. Para esta versión, se ha establecido que un voto positivo vale 2, uno negativo 0 y uno neutro 1. Para que se considere relevante una sugerencia, su voto medio debe ser superior a 1,6 (dato configurable en tiempo real).

Pruebas.

Se definen tres tipos de pruebas para el Gestor del Conocimiento, relacionado cada uno con uno de los servicios ofrecidos. Se harán pruebas a todos los niveles, desde los procedimientos almacenados, a la interfaz de usuario.

Pruebas de sugerencia de conceptos:

- Sugerir para una misma clase conceptos con variación de stop words.
- Sugerir conceptos vacíos.
- Sugerir conceptos existentes.
- Sugerir conceptos para clases inexistentes.

Pruebas de valoración de conceptos:

- Valorar todos los conceptos disponibles y solicitar valorar más.
- Valorar conceptos ya valorados.
- Valorar conceptos inexistentes.

Pruebas de asimilación de conceptos:

- Comprobar que un concepto es asimilado y se usa en el motor semántico.
- Valorar negativamente el concepto anterior las veces que sean necesarias para que deje de cumplir la puntuación mínima, ejecutar la asimilación, y comprobar que ya no se usa en el motor semántico.
- Valorar positivamente el concepto anterior las veces que sean necesarias para que vuelva a cumplir la puntuación mínima, ejecutar la asimilación, y comprobar que nuevamente se usa en el motor semántico.

Interfaz Web.

En esta iteración se desarrolla la interfaz relacionada con la gestión del conocimiento, que consiste en las interfaces de sugerir concepto, y valorar sugerencia.

Pantallas.

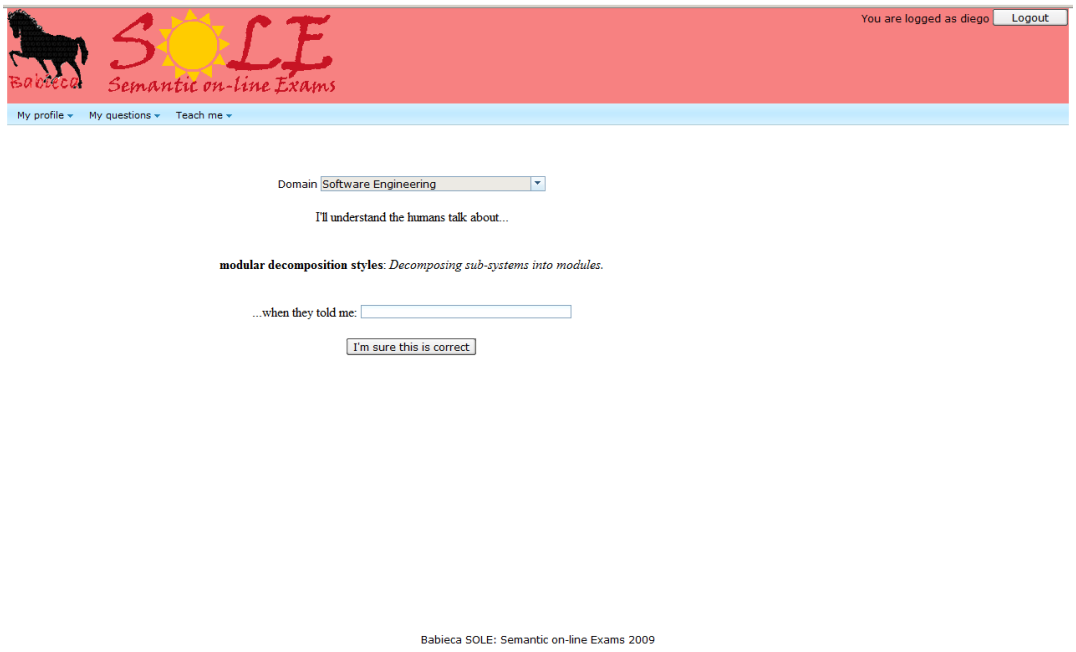

Ilustración 40 Pantalla de sugerir concepto.

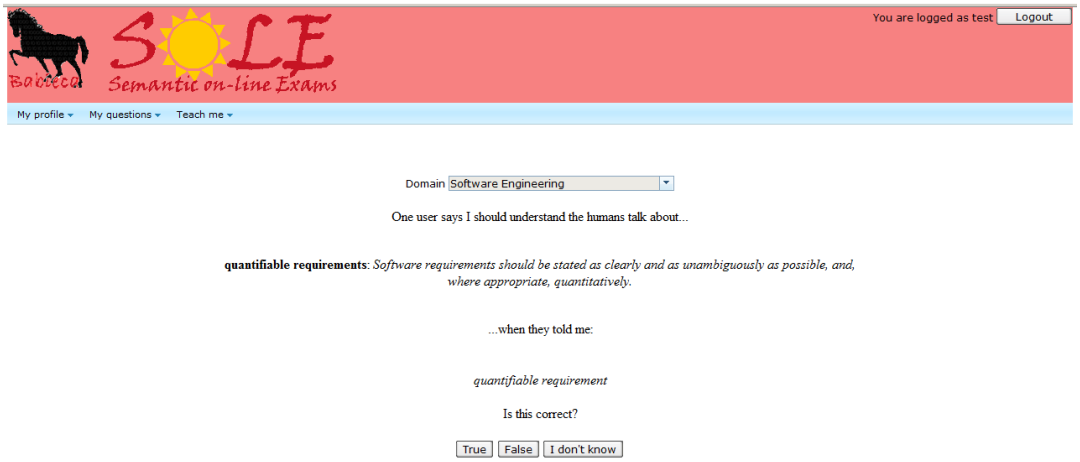

Ilustración 41 Pantalla de valorar sugerencia.

Cuarta iteración.

Buscador.

Se trata del componente encargado tanto del mantenimiento del Índice de Preguntas, como de las búsquedas sobre el mismo.

La aplicación estará contenida en un único fichero EAR, de nombre babiecasole-1.3.ear para esta iteración. El fichero contendrá, a parte de las librerías necesarias, dos ficheros WAR, siendo uno la interfaz web (sole-web.war), y otro la aplicación web de despliegue de Web Services (sole-webservices.war). También se incluirán dos ficheros EJB JAR, sole-common-1.0.jar y sole-core-1.0.jar, que contendrán los servicios EJB y la implementación del proyecto.

Para el diseño de los servicios principales del Gestor de Preguntas, se ha optado por una arquitectura en capas del siguiente modo:

5. Interfaz (Web, o Servicio Web)
6. Servicio EJB
7. Método DAO
8. Procedimiento almacenado

Tan sólo se usan procedimientos almacenados en los casos que sea necesario ejecutar consultas sobre gran número de tablas de la base de datos, o actualizaciones masivas de datos. El encapsulamiento de ciertos métodos a través de DAO, permite que la ejecución de esos métodos se proteja por transacciones de la Base de Datos. Esta protección de la transaccionalidad, se debe a que, en la mayoría de los casos, se producirán inserciones y actualizaciones en multitud de tablas de la base de datos. Un error cuando ya se hubiera producido alguna actualización o inserción, provocaría que los datos almacenados quedaran en un estado inconsistente.

Para los servicios que no requieran actualizaciones/inserciones en varias tablas, o manejo de datos masivos, tan sólo se usarán los dos primeros niveles.

Diseño.

Los servicios implementados en esta versión del prototipo, para el Buscador son:

- **Indexar pregunta**: una pregunta ya clasificada se incluye en el índice.
- **Reindexar pregunta**: una pregunta ya indexada y nuevamente clasificada se reubica en el índice.
- **Buscar preguntas**: recibe los criterios de búsqueda, y devuelve un fichero QTI con los resultados que haya en el índice.

Dados los diversos orígenes que puede tener una llamada a un servicio (a través de interfaz Web, o Web Service), el diseño interno de la capa de implementación de servicios se ha estructurado en varios métodos para un servicio que llaman a un núcleo común. Aunque el prototipo que se desarrollará para esta versión no contenga las búsquedas desde Web, se respetará esta estructura para permitir la creación de los servicios EJB correspondientes.

Se crearán dos Índices distintos: el *índice de Clases*, y el *Índice de Preguntas*, o *Índice Inverso*. La función del primero consistirá en dar la información sobre las preguntas indexadas para cierta clase de ontología (función clásica de búsqueda), mientras que el segundo estará orientado a obtener información de indexación de cada pregunta, para facilitar sus reindexaciones futuras. Ambos índices deben ser actualizados de forma atómica, para evitar inconsistencias.

La estructura del Índice de Clases seguirá la estructura en árbol de cada ontología en uso. Se optará por la creación de todos los nodos del índice durante el proceso de añadido de ontología, con el fin de optimizar los tiempos durante la indexación de una pregunta. La otra opción sería ir creando los nodos correspondientes a las clases durante las indexaciones, que en algunos casos requeriría tratamiento de datos masivos.

El proceso de búsqueda, comenzará con la clasificación semántica de las claves de búsqueda, de la que se obtendrán las claves del índice de clases, para luego ir recuperando las preguntas. El sistema de recuperación de preguntas del Índice de Clases será el siguiente, hasta completar el número de preguntas solicitadas, o en su defecto, el máximo posible:

1º. Preguntas indexadas para cada clase obtenida de las claves de búsqueda, ordenadas por el OntoRank de cada una.

2º. Preguntas indexadas para las subclases de las clases obtenidas de las claves de búsqueda, ordenadas por el OntoRank de cada una, que corresponderá al de la clase inicial, con una penalización por nivel.

3º. Preguntas indexadas para las superclases de las clases obtenidas de las claves de búsqueda, ordenadas por el OntoRank de cada una, que corresponderá al de la clase inicial, con una penalización por nivel.

Será necesario asegurarse de que las preguntas se recuperan y devuelven como resultado de forma única, ya que una pregunta podrá estar clasificada por varias clases, y es posible el caso de que una pregunta se indexe para dos clases donde una sea sublcase de otra.

La reindexación de preguntas se producirá de forma automática cuando esté desarrollado el *Demonio*. El periodo de ejecución deberá ser obtenido de forma empírica, tras comprobar cuánto se tarda en clasificar e indexar preguntas, con el fin de reducir al mínimo posible el tiempo que el Índice no vaya a ser accesible a los usuarios. El objetivo será obtener el número de preguntas clasificables e indexables en un máximo de cinco minutos, y en cada ejecución, se reindexarán las N preguntas que lleven más tiempo sin haber sido indexadas.

El componente Buscador, se ha dividido en dos paquetes:

- org.babieca.sole.searchengine.domain
- org.babieca.sole.searchengine.services

Al igual que en componentes anteriores, el paquete *services* contendrá las clases relacionadas con la implementación de servicios, y el paquete *domain*, las clases relacionadas con las entidades y DAO.

A continuación, se muestran los diagramas de clases de cada paquete.

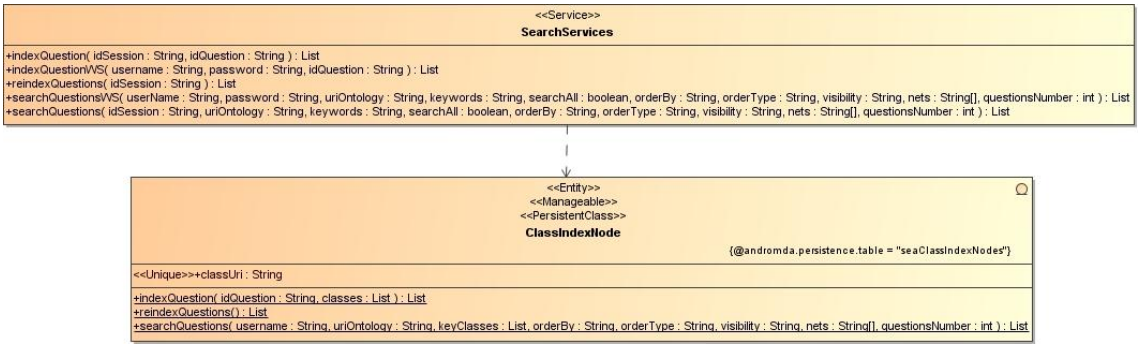

Ilustración 42 Diagrama de clases org.babieca.sole.searchengine.services

El paquete *services* contiene una única clase estereotipada como *Service*. Los métodos de esta clase corresponden con los servicios del Buscador. Todos los servicios, a excepción de *reindexar preguntas,* tienen doble origen.

En el diagrama aparece la entidad *ClassIndexNode* para mostrar la dependencia existente entre ambas, que permite la ejecución de métodos DAO.

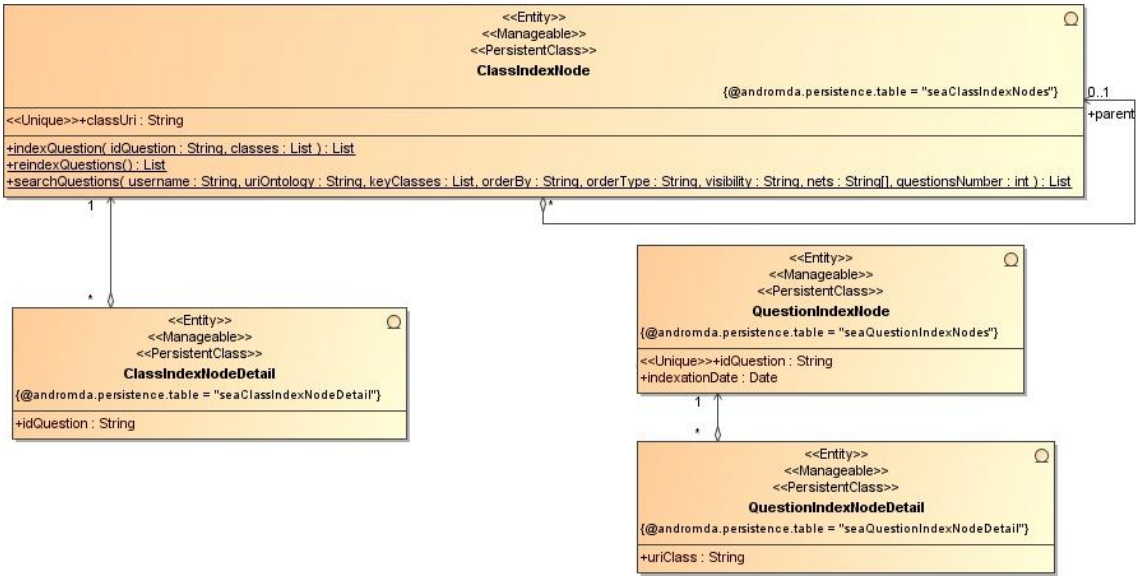

Ilustración 43 Diagrama de clases org.babieca.sole.searchengine.domain

El paquete *domain* contiene cuatro entidades, que se dividen en dos master-detail: uno para el Índice de Clases, y otro para el Índice de Preguntas.

En el caso del Índice de Clases, se observa una agregación sobre sí misma. Esto responde a la necesidad de expresar la estructura en árbol de dicho índice. Cada nodo, tendrá una referencia a su nodo padre.

Todos los métodos DAO se han incluido en la clase ClassIndexNode. Estos métodos realizarán operaciones sobre ambos índices, pero la necesidad de su

ejecución atómica, y dentro de una única transacción, provoca que la ejecución deba ser en un único método para cada servicio.

Pruebas.

Se definen tres tipos de pruebas del Buscador, relacionado cada uno con uno de los servicios ofrecidos. Se harán pruebas a todos los niveles, desde los procedimientos almacenados, a la interfaz de servicio web.

Pruebas indexación de preguntas:

- Indexar una pregunta clasificada para una única clase.
- Indexar una pregunta clasificada para N clases.
- Indexar una pregunta que no se haya podido clasificar.

Pruebas de búsqueda de preguntas (probando con cambios en todos los posibles parámetros de entrada):

- Búsqueda clasificada para una única clase.
- Búsqueda clasificada para N clases.
- Búsqueda que no pueda clasificarse.
- Búsqueda que devuelva menos preguntas de las solicitadas.
- Búsqueda que devuelva exactamente las preguntas solicitadas.
- Búsqueda que pudiera devolver más preguntas de las solicitadas.
- Búsqueda que pudiera incluir preguntas no visibles para el usuario que busque.

Pruebas de reindexación de preguntas:

- Ejecutar la reindexación sin cambios de clasificación para comprobar que la ejecución es determinista.
- Ejecutar la reindexación tras haber actualizado el Gestor del Conocimiento, incluyendo cambios en las clasificaciones.
- Ejecutar la reindexación mientras se producen búsquedas, para comprobar la disponibilidad del sisitema.

Análisis de costes.

En este apartado, una vez finalizado el desarrollo del proyecto, se calcularán los costes reales que ha conllevado el mismo, comparándolos con la estimación realizada en el comienzo de la fase de desarrollo.

Los resultados del análisis de costes deberían servir para mejorar las estimaciones en proyectos futuros.

Gastos de personal.

Gastos derivados de los recursos humanos asociados al proyecto. La duración del proyecto ha sido de 32 semanas, con una dedicación de 30 horas medias semanales de un único recurso, con unos honorarios de 30€ la hora. Según los datos dados, los gastos de personal vienen reflejados en la siguiente tabla.

Actividades	Semanas Estimadas	Coste Actividad Estimado (€)	Semanas de Desarrollo	Coste Actividad Real (€)
Análisis	7	8.400	12	10.800
Primera entrega	3	3.600	6	5.400
Segunda entrega	4	4.800	12	10.800
Tercera entrega	5	6.000	1	900
Cuarta entrega	4	4.800	1	900
Quinta entrega	4	4.800	-	-
Total costes	**31**	**32.400**	**38**	**28.800**

Tabla 29 Gastos de personal

Gastos de equipo informático.

Los equipos utilizados en el proyecto tienen estimado un periodo de amortización lineal de 5 años. A continuación, se expone una tabla con los precios de compra así como la amortización durante el periodo de duración del proyecto.

Equipos Informáticos	Precio de Compra (€)	Amortización Semanal (€)	Amortización Estimada	Amortización Real
Servidor Desarrollo (1)	600	2,31	57,75	73,92
Ordenadores personales (1)	960	3,69	92,25	118,08
Impresoras (1)	120	0,46	11,50	14,72
Total costes			**161,50**	**206,62**

Tabla 30 Gastos de equipo informático

Gastos de material fungible.

Gastos provenientes de material no inventariable, derivado del desarrollo del proyecto.

Material	Precio unitario	Unidades estimadas	Coste total estimado	Unidades consumidas	Coste total
Cartuchos de tinta	12	3	36,00	1	12,00
Paquetes de folios (500 uds.)	3	2	6,00	2	6,00
Material de oficina	-	-	50,00	-	3,00
Total costes			**92,00**		**21,00**

Tabla 31 Gastos de material fungible

Otros gastos.

Gastos provenientes de conceptos no clasificados anteriormente.

Gastos	Coste mensual (€)	Costes total	Coste mensual	Costes total
Hosting	15	105,00	15	120,00
Dominio	-	9,00	-	9,00
Suministros (agua, luz, teléfono, adsl, gas)	120,00	840,00	130,00	960,00
Viajes y dietas	-	50,00	-	8,00
Otros gastos	-	100,00	-	95,96
Total costes	**135,00**	**1.104,00**	**145,00**	**1.192,96**

Tabla 32 Otros gastos

Resumen del presupuesto.

Se presentan todos los gastos calculados anteriormente. Se suma un 20% de beneficio, y un índice medio de riesgo del 15%.

	Estimaciones	Gasto Total
1. Gastos de personal	32.400,00	28.800,00
2. Equipos	161,50	206,92
4. Fungible	92,00	21,00
6. Otros	1.104,0	1.192,96
Total	**33.757,50**	**30.220,88**
Beneficio	6.571,50	10.108,12 (33,45%)
Riesgo (15%)	5.063,63	5.063,63
Subtotal	45.392,63	45.392,63
I.V.A. (16%)	7.262,82	7.262,82
Total presupuesto	**52.655,45**	**52.655,45**

Tabla 33 Resumen del presupuesto

Se puede comprobar cómo el gasto en personal se ha reducido drásticamente. Esto obedece al hecho de que la dedicación horaria semanal se redujo sensiblemente, por lo que el aumento en la duración temporal del proyecto no aumentó los costes.

En el caso de haberse completado todas las iteraciones, con todas las funcionalidades previstas, el gasto en personal podría haberse equiparado al propuesto, con lo que el aumento en el resto de gastos habría llevado al empleo del capital riesgo, y probablemente, a la reducción del beneficio.

7. Bibliografía

1. **Van Der Hest S., Christian.** ¿Qué es la Web 2.0? [En línea] Maestros del Web, 27 de octubre de 2005. http://www.maestrosdelweb.com/editorial/web2/.

2. **O'Reilly, Tim.** What is Web 2.0. *What is Web 2.0.* [En línea] 30 de septiembre de 2005. http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html.

3. **Graham, Paul.** Web 2.0. [En línea] noviembre de 2005. http://www.paulgraham.com/web20.html.

4. **Garrett, Jesse James.** Ajax: A New Approach to Web Applications. *Adaptative Path.* [En línea] 18 de febrero de 2005. http://adaptivepath.com/ideas/essays/archives/000385.php.

5. **Pilgrim, Mark.** What Is RSS. *XML.com.* [En línea] 18 de diciembre de 2002. http://www.xml.com/lpt/a/2002/12/18/dive-into-xml.html.

6. **W3C.** Guía breve de los Servicios Web. [En línea] 9 de enero de 2008. http://www.w3c.es/Divulgacion/GuiasBreves/ServiciosWeb.

7. —. Guía Breve de la Web Semántica. [En línea] 7 de Febrero de 2008. http://www.w3c.es/Divulgacion/Guiasbreves/WebSemantica.

8. —. Resource Description Framework (RDF): Concepts and Abstract Syntax. *Resource Description Framework (RDF): Concepts and Abstract Syntax.* [En línea] 10 de febrero de 2004. http://www.w3.org/TR/rdf-concepts/.

9. —. RDF-Semantics. [En línea] W3C, 10 de febrero de 2004. http://www.w3.org/TR/2004/REC-rdf-mt-20040210/.

10. —. RDF/XML Syntax Specification (Revised). [En línea] 10 de febrero de 2004. http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/.

11. —. RDF Vocabulary Description Language 1.0: RDF Schema. [En línea] 10 de febrero de 2004. http://www.w3.org/TR/2004/REC-rdf-schema-20040210/.

12. —. SPARQL query languaje for RDF. *SPARQL query languaje for RDF.* [En línea] 15 de enero de 2008. http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/.

13. —. Visión general del Lenguaje de Ontologías Web (OWL). *Visión general del Lenguaje de Ontologías Web (OWL).* [En línea] 10 de febrero de 2004. http://www.w3.org/2007/09/OWL-Overview-es.html.

14. —. Preguntas frecuentes sobre el Lenguaje de Ontologías Web (OWL) del W3C. [En línea] 10 de febrero de 2004. http://www.w3c.es/Traducciones/es/SW/2005/owlfaq.

15. *Software Engineering Ontology – the Instance Knowledge (Part I).* **Wongthongtham P, Chang E, Dillon T y Sommerville I.** 2, s.l. : IJCSNS International Journal of Computer Science and Network Security, 2007, Vol. 7.

16. **Wongthongtham, Dr. P.** Ontology-based Software Engineering. *SEOntology: The Software Engineering Ontology.* [En línea] 13 de febrero de 2007. http://www.seontology.org/doc/Onto-basedSE.pdf.

17. **SEOntology.** Introduction to the Software Engineering Ontology. *SEOntology: The Software Engineering Ontology.* [En línea] http://www.seontology.org/index.php?option=com_content&task=view&id=15&Itemid=31#s.

18. *Software Engineering Ontology for Software Engineering Knowledge.* **Wongthongtham, Pornpit, Chang, Elizabeth y Sommerville, Ian.** Budapest, Hungría : 10th International Protégé Conference, 2007.

19. **García Peñalvo, Francisco José.** Estado actual de los sistemas de e-Learning. *Universidad de Salamanca.* [En línea] http://www.usal.es/~teoriaeducacion/rev_numero_06_2/n6_02_art_garcia_penalvo.htm.

20. **García Peñalvo, Francisco José y García, Joaquín.** Los espacios virtuales educativos en el ámbito de Internet: Un refuerzo a la formación tradicional, Teoría de la Educación. Educación y Cultura en la Sociedad de la Información. *Universidad de Salamanca.* [En línea] 2001. http://www3.usal.es/~teoriaeducacion/rev_numero_03/n3_art_garcia-garcia.htm.

21. **Rosenberg, M. J.** *E-learning strategies for delivering knowledge in the digital age.* s.l. : McGraw-Hill, 2001.

22. **Lozano Galera, J.** El triángulo del e-learning. [En línea] 2004. http://www.noticias.com.

23. **Rengarajan, R.** LCMS and LMS: Taking advantage of tight integration. Click 2 Learn. . [En línea] 2001. http://www.e-learn.cz/soubory/lcms_and_lms.pdf.

24. **Moodle Community.** Filosofía Moodle. [En línea] http://docs.moodle.org/es/Filosofía.

25. **Lozano, Juan Carlos.** Técnicas y Herramientas de Evaluación On-Line. *Vertice Learning.* [En línea] "http://www.verticelearning.com/articulos/tecnicas_y_herramientas_de_evaluacion_on_line.html.

26. **Vertice Learning.** Herramienta de Autor Vértice 2.1. *Vertice Learning.* [En línea] http://www.verticelearning.com/e-learning_Herramienta_autor_online.html.

27. **Scantron.** Software de Evaluación Online Perception. *Scantron.* [En línea] http://espanol.scantron.com/software/perception.htm.

28. **creartest.com.** Crear Test. [En línea] http://www.creartest.com.

29. **Grupo HEOL.** Herramientas de Evaluación OnLine . *Grupo HEOL - DOSI.* [En línea] http://dosi.itis.cesfelipesegundo.com/heol/index.php.

30. **Grupo Exanet.** Exanet: Exámenes en Red. *Sourceforge.net.* [En línea] http://sourceforge.net/projects/exanet/.

31. **Joglar Prieto, Nuria et al.** EVALUACIÓN ONLINE EN EDUCACIÓN SECUNDARIA Y UNIVERSITARIA: Una experiencia de enseñanza-aprendizaje. *Grupo HEOL - DOSI.* [En línea] http://dosi.itis.cesfelipesegundo.com/heol/archivos/Joglar_et_al_Evaluacion_Online_Anales_ITIS.pdf.

32. **Advanced Distributed Learning.** SCOORM. *Advanced Distributed Learning.* [En línea] http://www.adlnet.gov/scorm/.

33. **IMS Global Learning Consortium.** IMS Global Learning Consortium. [En línea] http://www.imsglobal.org/.

34. —. IMS Question & Test Interoperability Specification. *IMS Global Learning Consortium.* [En línea] http://www.imsglobal.org/question/index.html.

35. **Sun Microsystems.** Java EE at a Glance. [En línea] http://java.sun.com/javaee/.

36. **No Magic, Inc.** Magic Draw. [En línea] http://www.magicdraw.com.

37. **AndroMDA.org.** AndroMDA. [En línea] http://www.andromda.org/.

38. **Red Hat Middleware.** Hibernate. [En línea] http://www.hibernate.org/.

39. **Sun Microsystems.** MySQL:: The world's most popular open source database. [En línea] http://www.mysql.com/.

40. **Potix Corporation.** ZK - Direct RIA. [En línea] http://www.zkoss.org/.

41. **Apache Software Foundation.** Apache Struts. [En línea] http://struts.apache.org/.

42. **Sun Microsystems.** jaxb: JAXB Reference Implementation. [En línea] https://jaxb.dev.java.net/.

43. **Jena Team.** Jena Semantic Web Framework. [En línea] http://jena.sourceforge.net/.

44. **Spring Community.** Spring Source. [En línea] http://www.springsource.org/.

45. **The Eclipse Foundation.** Eclipse.org. [En línea] http://www.eclipse.org/.

46. **CollabNet.** Subversion. [En línea] http://subversion.tigris.org/.

47. **University of Southampton.** JQTI. [En línea] http://jqti.qtitools.org/.

48. **Apache Software Foundation.** Apache log4j 1.2. [En línea] http://logging.apache.org/log4j/1.2/.

49. **CollabNet.** ROME: RSS/Atom syndication and publishing tools. [En línea] https://rome.dev.java.net/.

50. **Sun Microsystems.** GlassFish. [En línea] https://glassfish.dev.java.net/.

51. **CollabNet.** Subclipse. [En línea] http://subclipse.tigris.org/.

52. —. Tortoise SVN. [En línea] http://tortoisesvn.tigris.org/.

53. **Slackhat.** dotProject - Project management software. [En línea] http://www.dotproject.net/.

54. **Consejo Superior de Administración Electrónica.** MÉTRICA. VERSIÓN 3. *Ministerio de Administraciones Públicas.* [En línea] http://www.csi.map.es/csi/metrica3/index.html.

55. **Comment ça marche.** MERISE - Initiation à la conception de systèmes d'information. *Comment ça marche.* [En línea] 14 de octubre de 2008.

56. **European Space Agency.** Software Engineering and Standardisation. *European Space Agency.* [En línea] 29 de marzo de 2007. http://www.esa.int/TEC/Software_engineering_and_standardisation/TECP5EUXBQE_1.html.

57. **Beck, Kent, y otros.** Manifesto for Agile Software Development. [En línea] 2001. http://www.agilemanifesto.org/.

58. **Wells, Don.** Extreme Programming: A gentle introduction. . [En línea] 17 de febrero de 2006. http://www.extremeprogramming.org/.

59. **Letelier, Patricio y Penadés, Mª Carmen.** Métodologías Ágiles para el desarrollo de software: eXtreme Programming (XP). [En línea] 2004. http://www.willydev.net/descargas/masyxp.pdf.