Conde, M. Á., Pozo de Dios, A., & García-Peñalvo, F. J. (2011). Moodle 2.0, Nuevas Oportunidades de Aprendizaje Basadas en Servicios eLearning. In J. L. Sierra Rodríguez & A. Sarasa Cabezuelo (Eds.), *Avances en Ingeniería del Software Aplicada al E-learning. Actas Revisadas y Extendidas del 1er Taller sobre Ingeniería del Software en e-Learning - ISELEAR'10* (pp. 71-86). Madrid: Universidad Complutense de Madrid - Área de Ciencias Exactas y de la Naturaleza.

Moodle 2.0, nuevas oportunidades de aprendizaje basadas en servicios eLearning[1]

Miguel A. Conde[2], Alberto del Pozo[2], Franciso J. García[2]

[2] Departamento de Informática y Automática, Instituto Universitario de Ciencias de la Educación (IUCE). Grupo de Investigación GRIAL. Universidad de Salamanca, España
{mconde, delpozo1988, fgarcia}@usal.es

Resumen. El imparable avance de las nuevas tecnologías ha puesto de manifiesto la necesidad de actualización de las plataformas de aprendizaje. Esa actualización se basa en la incorporación de nuevas funcionalidades para satisfacer las nuevas necesidades de los usuarios. Una de las formas en que poder llevarlo a cabo pasa por la evolución de los entornos de aprendizaje hacia las Arquitecturas Orientadas a Servicios (SOA, *Service Oriented Architecture*). Estas arquitecturas, implementadas principalmente en forma de servicios web, permitirán la creación tanto de clientes como de herramientas externas que puedan interoperar con la plataforma, ampliando sus posibilidades y proporcionando una libertad de movimiento a los usuarios que antes no tenían. Moodle 2.0 será un ejemplo de esa evolución y en el presente artículo se verán alguna de las nuevas posibles aplicaciones.

Palabras clave: *eLearning*, LMS, Moodle 2.0, servicios web, SOA.

1 Motivación

El gran avance tanto científico como tecnológico que se está produciendo en la sociedad durante los últimos años ha provocado que se deban de buscar nuevas formas de hacer llegar la enseñanza a todos los usuarios[1, 2]. El *eLearning* supone un planteamiento diferente de

[1] Este trabajo está subvencionado por el Ministerio de Industria Turismo y Comercio (proyecto TSI-020302-2009-35), por el Ministerio de Ciencia e Innovación (proyecto TIN2010-21695-C02) y por la Junta de Castilla y León a través del proyecto de excelencia GR47.

los procesos de aprendizaje y, por tanto, introduce nuevas necesidades. Éstas van a verse respaldadas por las plataformas de aprendizaje. Mientras que hace tan sólo unos años el poder usar un ordenador como herramienta de soporte al proceso enseñanza/aprendizaje era considerado un gran avance tecnológico, ahora se entiende como una actividad cotidiana y se busca poder enriquecer más aún dicho proceso con la posibilidad de tener acceso a toda la información qué se desee y cuándo se desee, es decir, que el conocimiento pueda estar disponible en todo momento. Éste podría considerarse como el principal objetivo de las plataformas de aprendizaje (LMS, *Learning Management Systems*), sin embargo no siempre son capaces de satisfacer estos requisitos.

Otro problema de las plataformas es que son demasiado genéricas, poco adaptadas a circunstancias especializadas, y escasamente escalables modularmente [3], lo que dificulta su crecimiento y sostenibilidad.

Añadido a todo esto debe considerarse que, en la mayor parte de los casos, la potencia de los LMS no se aprovecha totalmente, es decir, muchas de las funcionalidades de las plataformas de aprendizaje son ignoradas convirtiéndose en muchos casos en meros contenedores de recursos[4-6].

Ante esta situación, aparecen un conjunto de iniciativas que pretenden dotar de cierta independencia a estas plataformas de *eLearning*, para que de esta manera dejen de ser plataformas monolíticas y cerradas, es decir, de difícil evolución y con una mínima interacción con otros sistemas. Se busca concretamente aportar capacidad de evolución y crecimiento independiente de la tecnología. Esta independencia se consigue mediante la implementación de una arquitectura orientada a servicios (SOA, *Service Oriented Architecture*) que va a facilitar la interacción hacia y desde la plataforma y por tanto la incorporación de nuevas funcionalidades y la apertura a nuevos contextos como pueden ser los dispositivos móviles.

En este artículo se va a introducir dicha arquitectura y algunas de las aplicaciones elaboradas. En el próximo apartado se habla acerca de SOA y sus posibilidades, entrando también en la relación entre SOA y los servicios web. A continuación se analizará la inclusión de los servicios web dentro Moodle 2.0, y como estos permiten la integración con diferentes utilidades, se planteará un ejemplo de servicios y de su aplicación en una herramienta de *backoffice*.

2 SOA

Para poder considerar las nuevas potencialidades que pueden proporcionar los LMS es necesario definir qué es la Arquitectura Orientada a Servicios y cuáles son sus posibles aplicaciones en los contextos de *eLearning*. Se trata un paradigma o concepto de arquitectura de *software* que se basa en la creación de un conjunto de servicios, de diferente granularidad,

entre los procesos de negocio y las aplicaciones [7]. Esta arquitectura tiene como objetivos principales:

- Modelar la lógica de negocio como servicios para poder expresar la capa de negocio mediante la facilidad que ofrece la orquestación de los mismos.
- Crear una capa de servicios que ofrezca la funcionalidad de la capa de aplicación de forma independiente de la tecnología que la soporta.
- Minimizar las dependencias entre la capa de negocio y la de aplicación para desacoplar el negocio de la tecnología. De este modo se permiten los cambios en cualquiera de ellas. El objetivo sería favorecer la agilidad para el negocio.

Para comprender claramente el concepto de Arquitectura Orientada a Servicios en primer lugar debe considerarse qué significa el concepto de Arquitectura en el ámbito del *software*. Según IEEE–STD-1471-2000 ("Práctica Recomendada Para La Descripción Arquitectónica De Sistemas De *Software* Intensivos") la arquitectura es "la organización fundamental de un sistema integrado en sus componentes, la relación entre ellos y el medio, y los principios que establecen su desarrollo y evolución" [8]. También puede entenderse una arquitectura como "la estructura global del *software* y las formas en que esa estructura proporciona integridad conceptual a un sistema [9] o bien como "la estructura lógica y física de un sistema, forjada por todas las decisiones de diseño estratégicas y tácticas aplicadas durante el desarrollo"[10].

En cualquiera de las definiciones aportadas se considera la arquitectura como la organización estructural de los componentes de un sistema, este tipo de organizaciones van a evolucionar vinculadas al planteamiento del modelo de negocio, que a su vez se verá condicionado por la evolución de la tecnología. Como ejemplo de estos cambios, puede observarse como las estructuras de negocio evolucionan de un modelo de jerarquización vertical a uno horizontal y multidisciplinar, para posteriormente pasar un modelo de tipo ecosistema, en el que las áreas implicadas no se comunican sólo entre las áreas más próximas.

Una vez explicado el término arquitectura, se tiene que conocer también el de servicio. Existen diferentes planteamientos acerca de lo que son los servicios. Pueden considerarse desde un punto de vista de negocio como "una funcionalidad construida como un componente reusable para ser utilizado en un proceso de negocio" [11] o desde un punto de vista técnico como "elementos autodescritos e independientes de plataforma que soportan la composición rápida, barata y distribuida de aplicaciones" [12].

En cualquiera de los casos, los servicios serán provistos por aplicaciones o proveedores de servicios de cara a proporcionar una funcionalidad sin desvelar su implementación. Esto supone la definición de una interfaz para su publicación. Teniendo en cuenta todo esto, las características deseables para un servicio serían:

- Tecnológicamente neutral. Los Servicios no solamente deben ser independientes de cualquier tipo de implementación sino que deben de poder ser invocados de una forma lo más estándar posible.

- Bajo acoplamiento. No debe requerirse ningún conocimiento de la estructura interna o externa del servicio en ningún contexto del mismo (ni en el lado del cliente ni en el del servidor).
- Descubrimiento transparente. La funcionalidad expresada por los servicios debe estar disponible en diferentes repositorios para su localización y difusión.
- Calidad de servicios. Los servicios deben ser capaces de ofrecer la funcionalidad expresada con un nivel cualitativo adecuado de cara a que su reutilización sea posible en diferentes contextos.
- Debe existir la posibilidad de trabajar con un servicio de forma individual o formando parte de una composición de estos en busca de una funcionalidad más completa.

La idea principal, por tanto, de esta arquitectura es que conviene ordenar la forma en que se comunican las distintas partes de un sistema. Para conseguir este objetivo se define una capa de servicios con la que los sistemas interactúan, evitando así el trabajar de manera directa entre ellos, favoreciendo la interoperabilidad y la escalabilidad. De esta manera, si dos sistemas desean comunicarse, no necesitan conocer el funcionamiento del otro, sino que utilizarán esta capa de servicios como intermediaria, la cual sí conoce como funcionan estos sistemas. Si en algún momento se desea sustituir o realizar algún cambio en alguno de los dos sistemas, la acción sería independiente del otro, ya que se han desarrollado de manera que no dependan del otro sistema, sino que sólo dependen de los datos que devuelven[13]. Esta forma de relacionar componentes aporta las siguientes ventajas: i) permite sustituir componentes individuales sin que eso afecte a otros componentes; ii) todos los sistemas se conectan al bus de la misma forma, con lo que se gana en homogeneidad; iii) facilidad en la operación y mantenimiento; y iv) arquitectura sencilla, robusta y escalable.

2.1 SOA y los servicios web

Es importante conocer que SOA no es un sinónimo de servicios web. Mientras que SOA es un paradigma de desarrollo (y estrategia de Tecnología de la Información, TI), los servicios web son una de las posibles tecnologías que se pueden utilizar para implementar SOA, aunque hay que destacar que la implantación de SOA está siendo mucho más rápida gracias a los servicios web y que éstos se están convirtiendo en el estándar de facto para la implementación de estas arquitecturas.

Una vez establecida la diferencia entre SOA y los servicios web, se va a tratar de describir estos últimos. Cabe destacar que existen varias posibles definiciones acerca de lo que son los servicios web, lo que muestra su complejidad a la hora de dar una explicación adecuada de todo lo que son e implican. Podrían considerarse como un conjunto de aplicaciones o de 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 esos procedimientos a través de la web [14]. Estos servicios proporcionan

mecanismos de comunicación estándares entre diferentes aplicaciones, que interactúan entre sí para presentar información dinámica al usuario. El uso de los servicios web proporciona una serie de ventajas (muchas de las cuales derivan de las ventajas de implementar una Arquitectura Orientada a Servicios) como son:

- Promueven la interoperabilidad ya que la interacción entre un proveedor y un solicitante de servicio está diseñada para que sea completamente independiente de la plataforma y del lenguaje.
- Fomentan los estándares y protocolos basados en texto, que hacen más fácil acceder a su contenido y entender su funcionamiento.
- Al apoyarse en HTTP, los servicios web pueden aprovecharse de los sistemas de seguridad *firewall* sin necesidad de cambiar las reglas de filtrado.
- Reducen la complejidad por medio del encapsulamiento, ya que los solicitantes y los proveedores del servicio se preocupan por las interfaces necesarias para interactuar. Como resultado, un solicitantes de servicio no sabe cómo fue implementando el servicio por parte del proveedor, y éste, a su vez, no sabe cómo utiliza el cliente el servicio.
- Permiten la interoperabilidad entre plataformas de distintos fabricantes por medio de protocolos estándar y abiertos ya que las especificaciones son gestionadas por una organización abierta, la W3C.

2.2 Arquitectura SOA en *eLearning*

De cara a incorporar interoperabilidad en las plataformas de aprendizaje y hacer a estas flexibles y escalables es necesaria la definición de una nueva generación de plataformas de aprendizaje. Este tipo de plataformas se van a asentar en las Arquitecturas Orientadas a Servicios. Este tipo de solución proporcionará una separación entre la interfaz de un servicio y su implementación subyacente. No será relevante que una aplicación que se quiera conectar a una plataforma esté implementada en una tecnología diferente del *core* del LMS. SOA aporta independencia en la evolución del *software*, esto permite la incorporación de nuevas funcionalidades independientemente de la versión del LMS subyacente.

Dentro de las posibles aplicaciones de las arquitecturas SOA deberían considerarse diferentes enfoques:

- Uso de las arquitecturas SOA para proporcionar información a contextos externos. Como podría ser el uso de arquitecturas orientadas a servicios para búsquedas semánticas de información sobre la plataforma de aprendizaje, como el proyecto LUISA [15].
- Adaptación ligera de plataformas de aprendizaje a otras aplicaciones, como puede ser con servicios de autenticación o herramientas de comunicación administrativa y *backoffice* [16]. Generalmente son adaptaciones parciales y no necesariamente tienen que integrarse transparentemente.

- Vinculaciones completas entre las plataformas y las aplicaciones en las que la integración es totalmente transparente para los usuarios, lo que posibilita una comunicación bidireccional y aporta un modo de presentación totalmente adaptado al LMS. Para esto se proponen varias especificaciones entre las que destacan IMS LTI (*Learning Tools for Interoperability*), para integración transparente de aplicaciones en las plataformas o los OSIDs (*Open Service Interface Definitons*) del proyecto OKI (*Open Knowledge Initiative*)[17], que describen medios de comunicación entre la plataforma y otras herramientas.
- Adaptaciones mixtas, en las que las aplicaciones requieren de comunicación con el LMS para extraer información e incorporar información al mismo, pero sin que tengan que estar incorporados en ellos, como podrían ser clientes móviles para las plataformas de aprendizaje. En este sentido cabe destacar Moodbile [18], como ejemplo de integración.

En cualquiera de los casos lo que se pretende es proporcionar nuevas funcionalidades a las plataformas de aprendizaje, haciéndolas evolucionar de un modelo monolítico, que tiende a quedarse obsoleto, a un modelo evolutivo, escalable y flexible.

3 Servicios web en Moodle 2.0

A lo largo del 2010 está previsto el lanzamiento de la nueva versión de Moodle. Esta nueva versión, la 2.0, que sustituirá a la actual 1.9, se ha visto como una oportunidad de realizar las cosas de manera distinta, de darle un cambio radical a la plataforma y adaptarla a las a las tecnologías que están inundando el mercado de las telecomunicaciones. Muchos de estos cambios son el soporte para repositorios externos (Picasa, Youtube, Flickr, Wikimedia, etc.), nuevos módulos y bloques, cambios en el código del *core* de Moodle (aunque se podrá actualizar sin problemas de la versión 1.9 a la 2.0), etc., pero uno de los cambios más significativos es el de soporte para servicios web. Estos servicios web van a ampliar enormemente las posibilidades de Moodle, pasando de ser una plataforma monolítica a una aplicación escalable y con la cual se puede interoperar. De esta manera se podrían aportar soluciones a nuevas necesidades de los usuarios, como serían las siguientes:

- La aparición de dispositivos móviles con conexión a Internet, con diferentes interfaces y características diferentes en cuanto a compatibilidad. Muchos de estos dispositivos permiten navegar por la aplicación, pero debido a sus limitaciones físicas, la navegación puede llegar a ser realmente compleja. Por tanto, sabiendo que el número de estos dispositivos está creciendo, es conveniente que Moodle facilite la creación de interfaces alternativas adaptadas a estas plataformas.
- El gran número de organizaciones que confían en Moodle como su plataforma de *eLearning* ha aumentado en los últimos años, lo que provoca cambios en los requisitos del sistema (escalabilidad) y diversificación de los mismos, ya que continuamente surgen

nuevas necesidades a satisfacer, como son nuevas funcionalidades (siempre y cuando no se corrompan los principios pedagógicos de Moodle). Aún así, hay que adaptar Moodle a los sistemas de información de las organizaciones donde se implementa.

- Integración con *Backoffice*. Se ofrece la posibilidad de, por ejemplo, crear una aplicación que pueda interactuar con varios sistemas a la vez, ejecutando una misma acción en varios lugares, haciendo que todo funcione de forma conjunta, evitando tener que realizar las operaciones varias veces. Por ejemplo, podría ser necesario dar de alta a un usuario en la plataforma y en la base de datos de la organización (Figura 1).

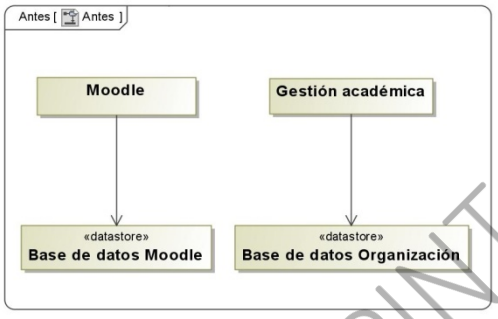

Fig. 1. Gestión de Moodle y una base de datos de una organización antes de la utilización de los servicios web.

Utilizando los servicios web se pasaría a realizarlas tan sólo una vez, dentro de una herramienta de *backoffice*, la cual se encargará de realizar las operaciones pertinentes tanto en la base de datos de la organización como en Moodle, mediante los servicios web, sin necesidad de modificar o acceder al código de esta plataforma (Figura 2).

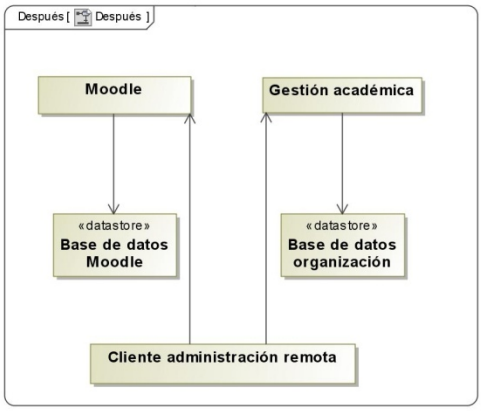

Fig. 2. Gestión de Moodle y una base de datos de una organización después de la implementación de los servicios web, utilizando una herramienta de *backoffice*.

3.1 Planteamiento arquitectónico

El primer paso dentro del desarrollo de los servicios web ha sido definir una arquitectura que permitiera garantizar la interoperabilidad que se les presupone. Por ello, dichos servicios web debían de cumplir una serie de requisitos establecidos por el equipo de desarrollo de Moodle [19]:

- Los servicios web deben ser accesibles desde cualquier sistema de conexión, tanto actual como futura, y deben de poder ser invocados independientemente del lenguaje utilizado para ello (interoperabilidad).
- La estructura de los servicios web debe de desarrollarse de tal manera que aunque se realicen cambios dentro del *core* de Moodle, sea necesario realizar pocas o ninguna modificación en la API.
- Las funciones que conforman la API deben ser ampliables para favorecer las contribuciones.
- El servicio web debe adaptarse al sistema de privilegios de Moodle (*capabilities*) para garantizar la seguridad.

Atendiendo a esta serie de requisitos, los servicios web de Moodle 2.0 están divididos en tres capas fundamentales [20] que pueden observarse en la Figura 3. Los clientes externos se conectan a la plataforma a gracias a los conectores, que reciben las peticiones, las interpretan y llaman a las funciones correspondientes de los servicios web, definidas dentro de los ficheros externallib.php repartidos por los directorios correspondientes de Moodle. Estas

funciones se encargarán ya del acceso a la base de datos o de la invocación de funciones propias de Moodle para llevar a cabo las tareas correspondientes. A continuación se describen cada una de esas capas:

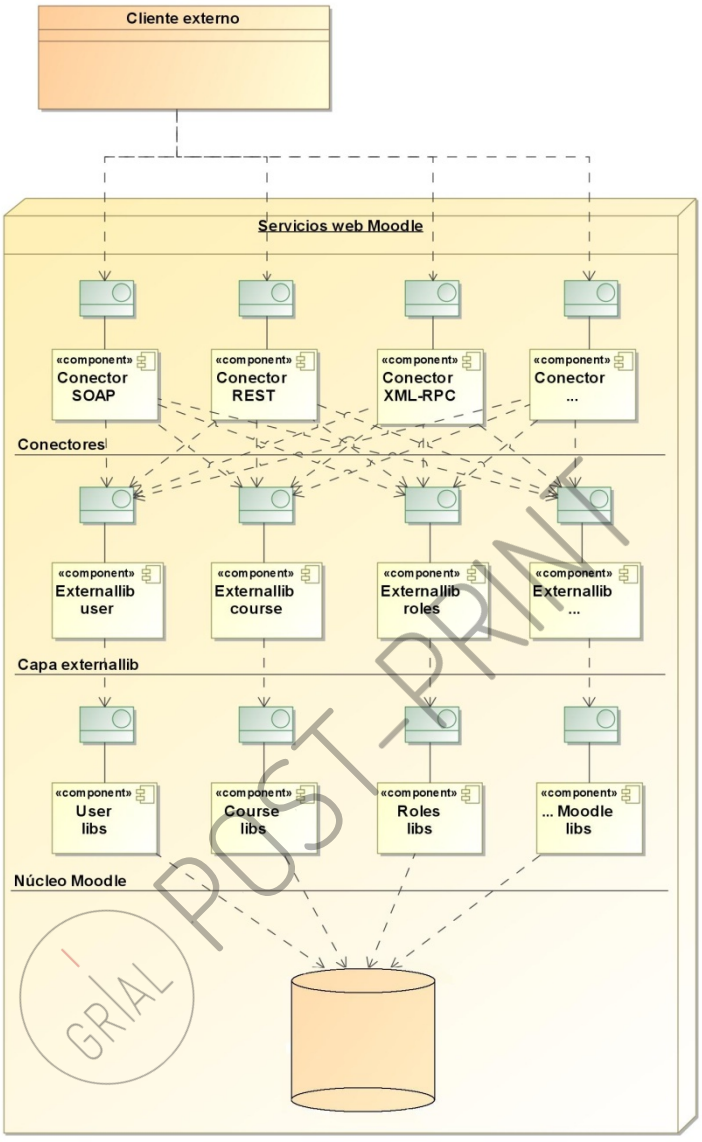

Fig. 3. Capas de los servicios web de Moodle 2.0.

1. Conectores. Hasta el momento se puede conectar con la plataforma a través de 4 protocolos, éstos son: REST, SOAP, XML-RPC y AMF (Flash). Además, se está trabajando en un quinto conector, JSON, el cual es bastante más rápido que los cuatro ya nombrados en lo que al intercambio de datos se refiere. Cada uno de estos protocolos posee su propio conector, los cuales se encargarán de recibir la petición desde el exterior, comprobar si la función a la que se desea acceder existe, comprobar los permisos del usuario que la invoca (y en caso de que sea necesario, generar o devolverle el *token* que le va a identificar durante la sesión, aunque también se puede conectar con la plataforma enviando el nombre de usuario y la contraseña en cada invocación). Una vez hecho esto, los conectores hacen el análisis sintáctico de los datos (*parsering*) y llamarán a la función correspondiente, esto es, se da paso a la capa externa. Los conectores admiten *plugins*, así que se podrán añadir fácilmente nuevos conectores para que los sistemas externos puedan conectar con Moodle mediante el uso de protocolos distintos a los que la plataforma trae de serie.
2. Externallib. Esta capa está formada por un conjunto de ficheros denominados externallib.php, que se encuentran expandidos por todo el árbol de directorios de Moodle. Dichos ficheros se llaman desde los conectores y en ellos se encuentran todas las funciones que se ofrecen en la API de los servicios web. Es decir, resumen todas las funcionalidades de Moodle para ofrecerlas al exterior, intentando utilizar para ellos el mayor código posible existente dentro de la plataforma para no crear código innecesario o duplicado. De esta manera, el externallib.php de users se encuentra dentro de la carpeta user de Moodle, y contiene una serie de funciones que permiten gestionar los aspectos relacionados con los usuarios de la plataforma (como ya se explicará más adelante). Como es lógico, antes de comenzar a realizar cualquiera de las acciones, se chequen los permisos del usuario con relación a la acción a realizar, y se comprueban además los parámetros recibidos y que se van a devolver. Esto último se consigue gracias a que en estos ficheros se encuentran además una serie de funciones que indican los parámetros que acepta y que debe devolver cada función de la API. Por último, indicar un problema con el que se han encontrado los desarrolladores: el tratamiento de errores. La solución actual consiste en el lanzamiento de excepciones cada vez que haya un error, cambiando de esa manera la idea inicial de devolver cadenas de texto. Este tratamiento de errores ha sido posible finalmente gracias a los requisitos mínimos de Moodle 2.0, que necesita de PHP 5 para funcionar (las excepción entraron en PHP en esta versión), por lo que si se desean utilizar los servicios web en versiones anteriores (hay un proyecto que está trabajando en el *backporting* de los servicios web) será necesario actualizar la versión de PHP si se tiene versiones anteriores a la 5.
3. Núcleo de Moodle. La capa de núcleo de Moodle está formada por todas las librerías que contienen funciones que puedan interesar dentro de la capa de los externallib, es decir,

funciones relaciones con los usuarios, los cursos, los grupos, etc. Esta capa ha sido mejorada en Moodle 2.0 porque muchas de estas funciones imprimían mensajes de error en pantalla cuando había algún problema, por lo que se han reescrito parte de estas funciones del núcleo para que en caso de error devuelvan excepciones (hasta ahora Moodle no poseía una API y gracias a estos cambios se está generando una).

3.2 Ejemplo de descripción y uso de un servicio web

Dentro de este apartado se va a explicar la estructura interna de un servicio perteneciente a los servicios web de Moodle, más concretamente la parte correspondiente a la capa externa del servicio *logging*, encargado de la gestión de la actividad (*logs*) dentro de la plataforma. Los diagramas utilizados para la explicación del servicio son diagramas creados mediante SoaML, un proyecto de especificación *open source* que describe un perfil y metamodelado UML para el diseño y modelado de las arquitecturas orientadas a servicios.

3.2.1 Arquitectura y funciones del servicio

En primer lugar se representa la arquitectura del servicio, que indica la relación entre el consumidor del servicio y su proveedor, en este caso el paquete *logging* con su correspondiente externallib.php, de forma que el contrato del servicio hace de mediador. El diagrama que se utiliza para la representación de esta información es el diagrama de arquitectura del servicio (*service architecture diagram*), del que se tiene un ejemplo en la Figura 4. Mientras que ese diagrama no ofrece una información demasiado valiosa dentro de la documentación de los servicios web, el diagrama de servicio (*service diagram*, Figura 5) representa de una manera clara las funciones que la API de *logging* va a ofrecer como servicio web.

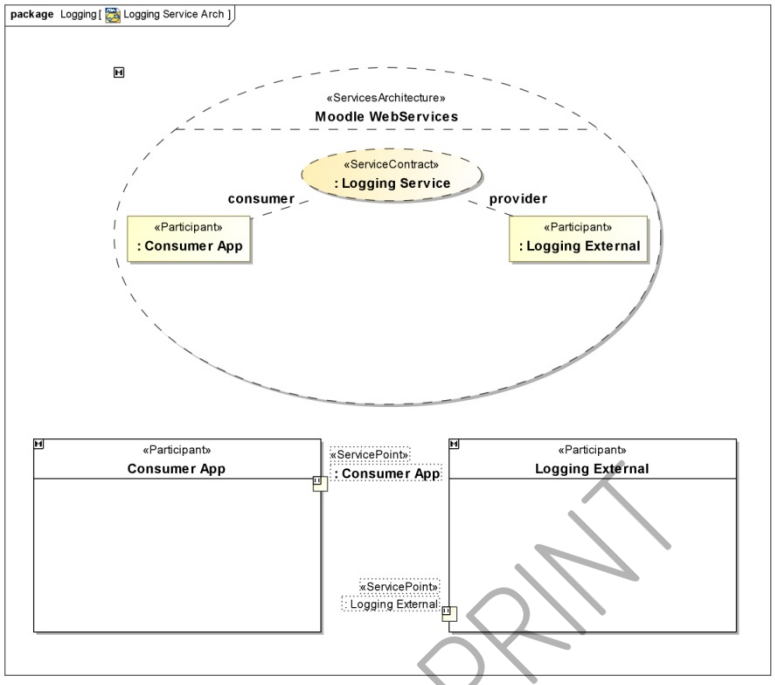

Fig. 4. Diagrama de arquitectura de servicio (*logging*)

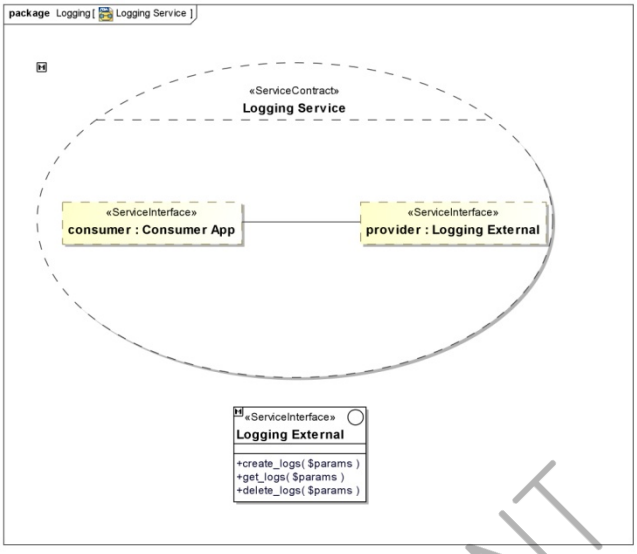

Fig. 5. Diagrama del servicio (*logging*).

En el diagrama de la Figura 5 cabe destacar la representación de las tres funciones que la API de los servicios web va a ofrecer en relación a la parte de *logging*, éstas son:

i) create_logs, que permitirá crear *logs* dentro de la base de datos de la plataforma y, por tanto, ofrecer la posibilidad de registrar acciones importantes que se realicen desde fuera de la plataforma (realmente lo que haría es ofrecer una información más concreta del trabajo realizado, ya que los servicios web ya registran ellos mismos si han sido utilizados).
ii) get_logs, función que permitirá recuperar una serie de *logs* de la plataforma. Esta función permite recuperar la actividad de Moodle en función del tiempo, del usuario, del curso o del id dentro de la base de datos. En algunas plataformas demasiado grandes, el número de *logs* es demasiado elevado, por lo que se está trabajando en una función que cuente los *logs* existentes para que en caso de que sea un número muy elevado éstos se puedan devolver por bloques, evitando así problemas de memoria.
iii) delete_logs, ésta es una función que no debería ser muy utilizada ya que sirve para borrar cierta actividad dentro de la plataforma.

3.2.2 Parámetros de las funciones

El tercer diagrama que documenta un servicio web es el diagrama de mensajes (*message diagram*) que complementa la información ofrecida en el diagrama anterior. Indica los

parámetros que cada una de las funciones del servicio acepta y devuelve en cada ejecución. Un ejemplo de este diagrama se puede ver en la Figura 6. En ella se observan los mensajes que los consumidores van a intercambiar con el servicio. En la mayor parte de los casos, las funciones deben recibir un *array* de *arrays*, donde cada uno de estos segundos *arrays* contiene la información necesaria para una ejecución de la función. De esta manera, se hace posible realizar varias acciones, como dar de alta cien usuarios por ejemplo, utilizando sólo una invocación a la función create_users, pero enviándole un *array* con cien *arrays*. Dicho esto, en el diagrama se puede observar que la función get_logs debe recibir un *array* de *arrays* con los parámetros criteria (curso, usuario, id, fecha), data1 y data2, mientras que devolverá toda la información existente acerca de los *logs* dentro de la base de datos. Cabe destacar que para conocer el uso de los parámetros será necesario leer la documentación de Moodle donde aparece una pequeña descripción de cada uno de ellos con sus posibles valores.

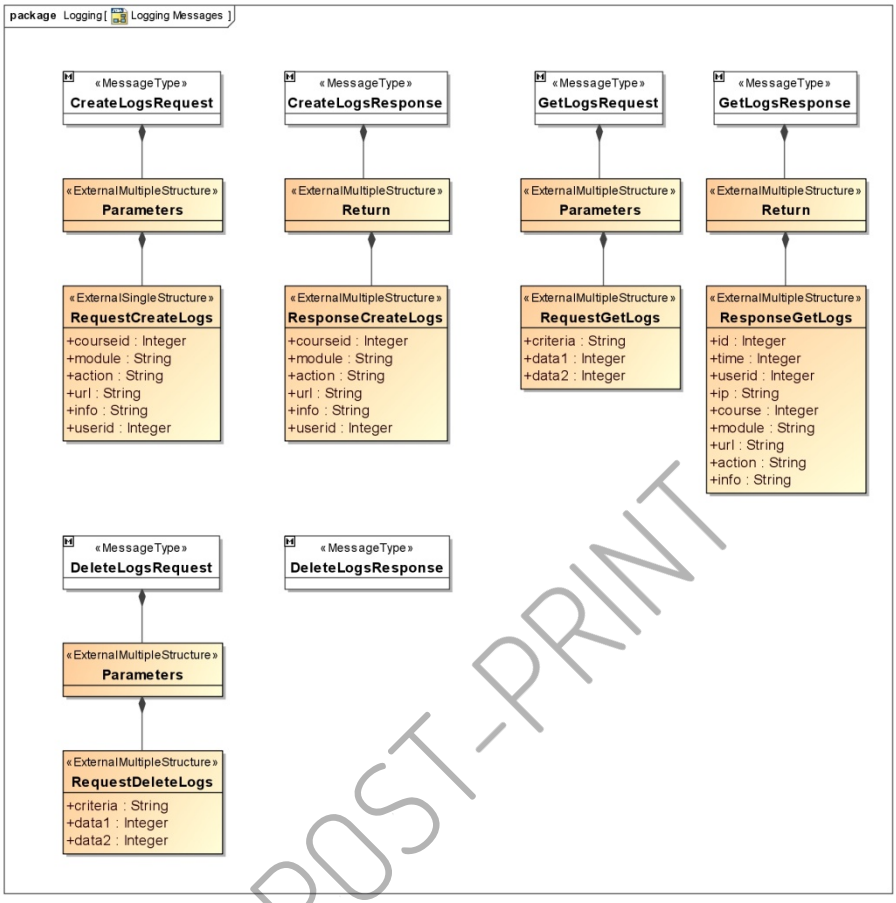

Fig. 6. Diagrama de mensajes del servicio (*logging*).

3.3 Aplicaciones de los servicios web de Moodle

Son diversas las aplicaciones que los servicios web tienen y pueden tener en Moodle. A continuación se muestran algunas de ellas.

3.3.1 Herramienta de Backoffice

Durante el desarrollo de la capa externa de los servicios web, se decidió realizar una herramienta de prueba que permitiera probar el correcto funcionamiento de las funciones de la API que se iban creando. A la vez, esta herramienta podría permitir observar (desde el punto de vista de los futuros desarrolladores de clientes para Moodle) las posibles necesidades que éstos pudieran tener, para de esta manera poder realizar una API lo más sencilla y usable posible. Se quería abarcar el mayor número de funciones de la API posibles, así pues se decidió crear una herramienta de *backoffice* (a pequeña escala) que permitiera a un administrador realizar las gestiones básicas de una plataforma Moodle. Dichas gestiones básicas serían:

- **Administración.** La opción más interesante dentro de la administración de la herramienta es la de poder cambiar el protocolo mediante el cual se quiere conectar con la plataforma Moodle, ya que el cliente creado permite la conexión con tres de los cuatro protocolos que soporta Moodle, éstos son: REST, SOAP y XML-RPC. Además, se está trabajando para adaptarla al protocolo JSON, que como se ha dicho anteriormente se encuentra actualmente en desarrollo. El resto de opciones son el envío masivo de mensajes a los usuarios inactivos (se le indica el número de días mínimo que un usuario debe llevar inactivo y automática se manda un mensaje a todos esos usuarios) y la opción de testear directamente las funciones de la API de los servicios web, tan sólo indicándole la función que se desea probar y los parámetros que se le van a enviar.
- **Usuarios.** Permite el acceso a todos los usuarios de la plataforma, además de permitir gestiones tales como la creación de nuevos usuarios y la actualización y eliminación de los ya existentes.
- **Cursos.** De la misma forma que con los usuarios, permite la gestión de la configuración de los cursos, así como de alguna de sus actividades. Es decir, se pueden crear, eliminar, modificar y acceder a cursos, así como a las categorías de cursos. Ya dentro de ellos, se puede acceder a sus foros, calificar a los alumnos y gestionar los grupos.
- **Roles.** Una parte importante dentro de Moodle es el sistema de roles, por lo que la herramienta permite la gestión de los roles en los dos contextos más utilizados: sistema y cursos.
- ***Logs*.** La parte más interesante llega con la gestión de los *logs*. Además de poder acceder a los *logs* de la plataforma ya sea mediante usuario, curso o fecha, se integra una herramientas de visualización de *logs* [21] que será comentada más adelante.

3.3.2 Otras aplicaciones

A la vez que se estaba trabajando en el desarrollo de los servicios web, y por lo tanto en la herramienta de *backoffice* que permitía su prueba, apareció la posibilidad de explotar las nuevas potencialidades de Moodle en otros contextos, algunos de los posibles serían:

- **Herramienta de visualización.** Adaptación a los servicios web de Moodle de una herramienta de visualización de *logs*. Esta herramienta tiene con fin trabajar con la información de la actividad del usuario en las plataformas de aprendizaje (hasta ahora para Moodle), ofreciendo por ejemplo, datos que permitan observar los mejores momentos para realizar el manteniendo de la plataforma, hacer estudios sociales (permite la visualización de las conexiones entre personas que hay entre la plataforma, nubes de palabras para visualizar los términos más usados en la misma, etc.). Mediante el uso de los servicios web se facilitaba el acceso a los *logs* de la plataforma. De esta manera se aporta independencia de la tecnología y la versión de la plataforma Moodle subyacente.
- **Elementos educativos portables.** Otro de los posibles aspectos a considerar como posible explotación de la nueva arquitectura de servicios va a ser la posibilidad de exportar funcionalidad de la plataforma a otros contextos, como podrían ser entornos de aprendizaje informal, redes sociales, dispositivos móviles, etc. Exportar este tipo de funcionalidad requiere una forma de representación y está serán los *widgets*. Los *widgets* son elementos portables pequeños que pueden funcionar en cualquier contexto HTML, proporcionando interacción, contenido o funcionalidad de otro contexto web [22]. Existen diferentes tipos de *widgets*, así como motores para la generación de éstos. En este momento se ha definido un *widget* para la exportación de los foros haciendo uso de un motor de *widgets* cómo es Apache Wookie [23]. Un ejemplo del trabajo que se está realizando con los *widgets* se puede observar en la Figura 7. En este widget puede observarse la funcionalidad del foro extraída a un widget definido según la especificación de la W3C utilizando Apache Wookie. Ese elemento portable podría incluirse en diferentes contextos web y por tanto extraerse del entorno de aprendizaje institucional, abriendo las puertas hacia contextos informales de formación.

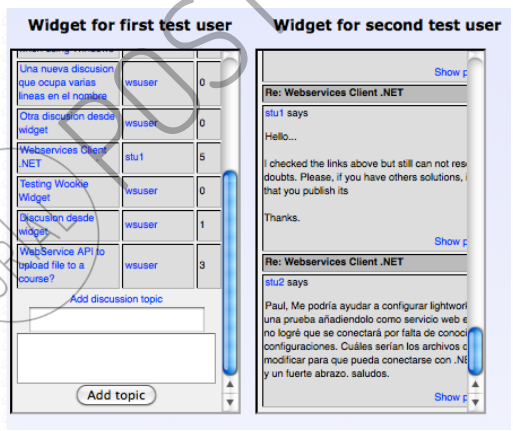

Fig. 7. Ejemplo de *widget* del foro creado en Wookie

4 Conclusiones

Los procesos de aprendizaje están cambiando, no sólo por la evolución de las nuevas tecnologías, sino también a cambios sociológicos como pueden ser la irrupción de las tendencias sociales favorecidas por las herramientas 2.0; las nuevas necesidades formativas cada vez más orientadas hacia el estudiante; los nuevos contextos y situaciones en los que los estudiantes aprenden; el hecho de que los receptores de la información han nacido en la era digital y, por tanto, utilizan la tecnología a un nivel antes no contemplado; los nuevos paradigmas formativos derivados de propuestas como la normativa de Bolonia que aboga por el reconocimiento de aprendizaje tanto formal como no formal; etc.

Ante esta situación las plataformas de aprendizaje, que han sido las herramientas más extendidas y utilizadas de los últimos años están tienden a quedarse obsoletas. Cuando estas herramientas han alcanzado un mayor grado de madurez, y aún a pesar de que su potencial no se aproveche, surgen nuevas necesidades y éstas suponen la evolución, que conduce a nuevos contextos, paradigmas y orientaciones en lo que se refiere al aprendizaje, y para ello son necesarias las arquitecturas SOA.

Las Arquitecturas Orientadas a Servicios serán una puerta que permite incrementar la funcionalidad de las plataformas de aprendizaje, proporcionando, por tanto, un medio para evitar su estancamiento, lo que permite abrirse camino hacia las nuevas tendencias como pueden ser los entornos de aprendizaje personalizados (*Personal Learning Environments, PLEs*) [24].

Moodle 2.0 y las herramientas que se han definido para su prueba son un claro ejemplo de cómo una plataforma puede ir más allá y puede realmente ser escalable y flexible gracias a la base que aportan los servicios web. Utilizando las Arquitecturas Orientadas a Servicios sobre este conocido LMS se ha conseguido proporcionar un medio para enriquecer las funcionalidades, existían ya trabajos en este sentido pero en cualquiera de los casos suponía tener que acceder directamente al código de Moodle y se generan dependencias tecnológicas y de versión. De este modo, como demuestran los ejemplos descritos, se abre un nuevo mundo para este LMS. Lo que se consigue con este tipo de aplicaciones será por tanto la evolución de la "especie", que no es otra que los medios que el estudiante utiliza para aprender, como son las plataformas de aprendizaje y los contextos a los que éstas pueden aplicarse.

Referencias

1. García Peñalvo, F.J.: Estado Actual de los Sistemas E-Learning. Teoría de la Educación. Educación y Cultura en la Sociedad de la Información 6, (2005)

2. García Peñalvo, F.J.: Preface of Advances in E-Learning: Experiences and Methodologies. Information Science Reference, Hershey, PA, USA (2008)
3. Fontenla, J., Caeiro, M., Llamas, M.: Una Arquitectura SOA para sistemas de e-Learning a través de la integración de Web Services. In: Conference Una Arquitectura SOA para sistemas de e-Learning a través de la integración de Web Services, pp. 22-29. (Year)
4. Carolina, U.o.N.: Sakai Pilot Evaluation Final Report. (2009)
5. Cuban, L.: Oversold and underused: Computers in the classroom. Harvard University Press, Cambridge, MA (2001)
6. Milligan, C.: The Road to the Personal Learning Environment? CETIS (2006)
7. http://www.theserverlabs.com/folletos/Folleto%20SOA.pdf
8. Maier, M.W., Emery, D., Hilliard, R.: IEEE 1471 and systems engineering. Syst. Eng. 7, 257-270 (2004)
9. Shaw, M., Garlan, D.: Software architecture: perspectives on an emerging discipline. Prentice-Hall, Inc. (1996)
10. Booch, G.: Object Oriented Analysis and Design with Applications. The Benjamin/Cummings Publishing Company (1994)
11. Keen, M., Adamski, D., Basu, I., Chilcott, P., Eames, M., Endrei, M., Fagalde, B., Raszka, R., Seabury, S.D.: Implementing Technology to Support SOA Governance and Management. IBM Redbooks publication (2007)
12. Papazoglou, M.P.: Service -Oriented Computing: Concepts, Characteristics and Directions. Proceedings of the Fourth International Conference on Web Information Systems Engineering, pp. 3. IEEE Computer Society (2003)
13. Alba, J.: ¿Qué es SOA - Arquitectura Orientada al Servicio. bit, vol. 167, pp. 52-53 (2008)
14. W3C: Guía Breve de Servicios Web. (2010)
15. LUISA: Learning Content Management System Using Innovative Semantic Web Services Architecture. . (2009)
16. Pätzold, S., Rathmayer, S., Graf, S.: Proposal for the Design and Implementation of a Modern System Architecture and integration infrastructure in context of e-learning and exchange of relevant data. In: E-Learning, E.I.F. (ed.) ILearning Forum 2008., pp. 82-90 (2008)
17. http://www.okiproject.org/
18. Alier, M., Casany, M.J.: Moodbile: Extending Moodle to the Mobile on/offline Scenario. IADIS International Conference Mobile Learning, Algarve (2008)
19. http://blogs.dfwikilabs.org/moodle_ws/2008/04/10/arquitectura-de-los-webservices-de-moodle/
20. http://blogs.dfwikilabs.org/pigui/2009/04/27/moodle-20-web-services-architecture/
21. Gómez, D.A., Sánchez, R.T., García, F.J.: Semantic Spiral Timeline como apoyo al e-learning. Journal of Universal Computer Sience (j-jucs) (2008)
22. W3C: Widgets 1.0 Packaging and Configuration. In: Draft, W.C.W. (ed.), (2008)
23. http://incubator.apache.org/projects/wookie.html
24. Wilson, S., Liber, O., Johnson, M., Beauvoir, P., Sharples, P., Milligan, C.: Personal Learning Environments: Challenging the dominant design of educational systems Journal of e-Learning and Knowledge Society 3, 27-38 (2007)