ÁGORA VIRTUAL: una propuesta de entorno colaborativo y de enseñanza sobre interfaces OSID

ÁGORA VIRTUAL: A Proposal on Collaborative and Teaching Environment over OSID Interfaces

José Alfonso Accino

Resumen

La importancia de la tecnología en la formación se hace evidente en la amplia oferta de herramientas de enseñanza virtual existentes, la mayor parte de las cuales hacen hincapié en ofrecer un número siempre creciente de funciones, muchas de ellas de escasa relevancia en la práctica o redundantes con otras herramientas, mientras que queda pendiente de resolver el problema de su integración real con la infraestructura de datos y servicios de la institución en la que se implantan. En este artículo se presenta una propuesta de entorno basado en una arquitectura simple, abierta, y un ejemplo de la utilización de una capa de interfaces OSID para facilitar su adaptación a la infraestructura existente en una institución.

Palabras clave: Enseñanza virtual, framework, OSID, OKI, middleware

Summary

The wide range of e-learning tools available evidences the significance of technology in education. However, most of these tools emphasize a rising number of features, many of which are of minimal relevance to a user's daily practice, meanwhile, the problem of integration of the e-learning tool with the underlaying data and service of the institution remains unsolved. This paper proposes an e-learning environment based on an open simple architecture, and shows a use case of the OSID layer that allows it to integrate with the legacy data structure available in the institution.

Keywords: E-learning, framework, OSID, OKI, middleware

La tecnología es un factor de peso cada vez mayor en los procesos de formación

La mayor parte de las plataformas existentes vienen a coincidir en la prioridad de ofrecer el mayor número de funciones posible

1.- Introducción

La tecnología es un factor de peso cada vez mayor en los procesos de formación y esto se hace palpable en el gran número de entornos virtuales de enseñanza y aprendizaje presentes en el mercado, ya sean productos comerciales o basados en software libre. Los elementos que constituyen este tipo de software han sido descritos en otros trabajos [1] y son ampliamente conocidos: gestión de recursos, herramientas de comunicación síncrona y asíncrona, servicios de agenda, mecanismos de evaluación, etc., por lo que no nos extenderemos aquí. En la práctica, la mayor parte de las aplicaciones de esta categoría –ya sean libres o comerciales–, más que mostrar grandes diferencias de concepto o estructurales, vienen a coincidir en la prioridad de ofrecer el mayor número posible de funciones, algo a lo que no son ajenas las presiones de grupos de usuarios entusiastas o aficionados y de la propia literatura del ramo con sus múltiples tablas comparativas sobre lo que ofrece cada una de ellas.

La necesidad de destacar su producto entre muchos similares suele empujar a los desarrolladores a incrementar cada vez más el número de funciones y, en consecuencia, la complejidad visual y de uso de productos que en muchos casos no estaban inicialmente pensados para ello. Por otra parte, la creencia –muy extendida en el imaginario popular en todo cuanto se refiere a la informática– de que el número de usuarios de un producto es el indicador por excelencia de la calidad del mismo hace que la mayor parte de las plataformas existentes, forzadas por la necesidad de alcanzar a un público lo más numeroso posible, se presenten como productos de uso general, válidos para todo tipo de usuarios y situaciones educativas y, en consecuencia, como sistemas autónomos, es decir, incluyendo sus propios mecanismos y soportes de autenticación, autorización o gestión de recursos.

Sin embargo, la experiencia nos muestra:

a) que el uso real que el profesorado hace de las funcionalidades disponibles dista mucho de requerir tal sobreabundancia de herramientas [2], ya que la mayoría suele ser consciente de que la excelencia educativa no depende del último *gadget* de moda, sino de la bondad del proyecto docente, y –sobre todo en la enseñanza semipresencial– prefiere herramientas simples, ante todo sencillas de utilizar y a las que puedan llevar con facilidad unos modelos de formación costosamente elaborados, fruto de su experiencia e innovación docentes;

b) que, en cambio, en ese enfoque "generalista" quedan sin resolver aspectos tan importantes como la integración real de los entornos de formación virtual con la infraestructura de la institución en la que se implantan.

Ágora Virtual es una propuesta de plataforma elaborada siguiendo las recomendaciones proporcionadas por los docentes a partir de sus prácticas para dar la posibilidad de implementar cualquier modelo didáctico e innovador

La propuesta presentada en este artículo parte de la experiencia realizada en la Dirección de Enseñanza Virtual de la Universidad de Málaga durante los años 1994-2003, algunos de cuyos resultados se expusieron en otro lugar [3], aunque con una nueva estructura rediseñada a la luz de esa experiencia previa, con distinta metodología y desarrollando *ab initio* la totalidad del código para elaborar un entorno que satisfaga diversos requisitos: centrado en el usuario, de fácil utilización sin renunciar a la funcionalidad, modular, ampliable, adaptable a infraestructuras existentes, con una variedad de perfiles configurables de usuarios, ligera, que no consuma excesivos recursos y haciendo uso de componentes externos en todo cuanto sea posible (por ejemplo, servidor Jabber o servidor de *streaming*).

2.- Descripción

Ágora Virtual (http://www.agoravirtual.es) es una propuesta de plataforma elaborada siguiendo las recomendaciones proporcionadas por los docentes a partir de sus prácticas para dar la posibilidad de implementar cualquier modelo didáctico e innovador, ya sea trabajo colaborativo, trabajo bajo proyectos, e-tutorías, etc. Aunque incluye distintas herramientas para facilitar el desarrollo de actividades de enseñanza y aprendizaje y la productividad personal, no impone la utilización de un modelo pedagógico determinado y la elección de los módulos a utilizar queda en manos del docente. Su

Aunque incluye distintas herramientas para facilitar el desarrollo de actividades de enseñanza y aprendizaje, no impone la utilización de un modelo pedagógico determinado

FIGURA 1: INTERFAZ DE AGORA VIRTUAL

interfaz se ha mantenido deliberadamente simple a fin de no intimidar con demasiadas opciones al usuario menos experimentado y en la creencia de que restar protagonismo, incluso visual, a la herramienta constituye la mejor manera de ayudar al usuario a centrarse en lo realmente importante, al tiempo que se facilita la orientación y navegación. Por ello, dos únicos menús proporcionan inicialmente toda la funcionalidad disponible: el de áreas de trabajo, arriba a la derecha, y el menú de módulos del área actual, a la izquierda, como se muestra en la figura 1.

La organización de Ágora Virtual gira en torno al concepto de área de trabajo centrada en el usuario, independientemente de que sea alumno o docente, roles estos que pueden variar dependiendo del contexto y que, además, son configurables. Cada usuario registrado dispone de un espacio propio idéntico o Área personal dotado de diversas funciones: agenda, avisos, bloc de notas, noticias RSS, recursos para uso personal, edición de los datos de su cuenta y control de privacidad, etc., desde el cual puede acceder a las restantes áreas en las que participe: asignaturas, grupos de trabajo para proyectos... o crear otras nuevas, si tiene los permisos para ello. Las áreas de trabajo pueden ser de varios tipos, actualmente *Cursos* y *Proyectos,* para adaptarse a diferentes contextos de trabajo y roles: más orientadas a la *formación* –cursos– o al *trabajo en colaboración* –proyectos–. Ambos tipos disponen de módulos con funciones similares a los del área personal, aunque con propiedades adecuadas al contexto. Por ejemplo, en una área de tipo *curso* el módulo de Agenda mostrará sólo las actividades fijadas por el propietario o instructores de la misma; en una área de tipo *proyecto* se mostrarán las actividades introducidas por cualquier coordinador o miembro del equipo, mientras que en el *área personal* se muestran no sólo los eventos de todas las áreas en las que el usuario participa, sino también sus eventos personales, los que el usuario puede introducir, editar y borrar. Cada usuario dispone por tanto de agendas particulares para cada área o de una agenda general para todas las actividades.

FIGURA 2: MÓDULO DE RECURSOS

La organización de Ágora Virtual gira en torno al concepto de área de trabajo centrada en el usuario, independiente-mente de que sea alumno o docente

Cada usuario dispone de agendas particulares para cada área o de una agenda general para todas las actividades

Otra diferencia contextual entre los módulos atañe a las acciones que se pueden realizar en ellos y que variarán según el rol que el usuario tenga asignado en esa área. Por ejemplo, el módulo de Recursos (figura 2) ofrece al propietario o tutor de una área de cursos la posibilidad de añadir nuevos recursos (documentos de cualquier tipo y enlaces a recursos en línea, que puede organizar en subdirectorios), borrarlos, usar las funciones de *copiar* y *pegar* para moverlos entre sus áreas, renombrarlos o hacerlos invisibles; pero las acciones posibles variarán si se accede a una área de trabajo como alumno o invitado.

Las acciones habilitadas en cada módulo forman la *barra de menú* del módulo, que variará según las acciones permitidas al rol del usuario en cada área. La navegación por tanto sigue una jerarquía claramente establecida Área -> Módulo -> Acción, de manera que aunque en todo momento se puede acceder a cualquier área o a cualquier módulo dentro de ella, el usuario tiene siempre una idea clara de dónde se encuentra.

Otros módulos como Tareas, Portafolio o Foros son específicos para las áreas de trabajo, ya que atañen por definición a la actividad en grupo. En el módulo de Tareas (figura 3) el instructor puede proponer nuevas actividades a los alumnos, con varias opciones: escala de valoración, tipo de

respuesta –en línea o archivo adjunto–, y con posibilidad de generar un Aviso o una entrada en la Agenda, con lo cual el anuncio de la tarea aparecerá automáticamente en el tablón de Avisos y/o en la Agenda del alumno. Éste puede ver en todo momento qué tareas tiene pendientes, realizarlas (o repetirlas si el tutor así se lo indica) y ver la calificación recibida, con un comentario, si lo hubiera. Por su parte, el tutor dispone de una visión de conjunto de todas las tareas y de su estado –borrador, activas, completadas– y desde allí puede acceder a un listado de participantes con indicación del estado de la tarea de cada uno de ellos (figura 3).

FIGURA 3: MÓDULO DE TAREAS

El modelo de navegación es igual para todos los usuarios, aunque con diferentes subsistemas en función del tipo de área y rol

En las áreas de *proyectos* las funciones son similares, aunque los roles y funciones son los adecuados al contexto. Así, en vez de los roles de *tutor*, *asistente* y *alumno* de las áreas de cursos, se dispone de los roles de *coordinador* y de *miembro* del equipo. El funcionamiento de los módulos será, en general, más igualitario, para adaptarse al trabajo en grupo. Por ejemplo, en principio todos los miembros del equipo podrán enviar recursos o añadir y editar avisos o eventos en la agenda. No obstante, las funciones que se permiten a cada rol son configurables, por lo que cabe la posibilidad de organizarlas de otra manera.

En Ágora Virtual no existen permisos individuales, sino roles que habilitan para determinados conjuntos de tareas

Al contrario que en otros sistemas, el modelo de navegación es igual para todos los usuarios, aunque con diferentes subsistemas en función del tipo de área y rol. Por ello no es necesario "entrar como si fuera..." (lo que además resultaría incompatible con la privacidad), ya que la utilización de los módulos es siempre idéntica y la necesaria diferenciación debida a los roles sólo se refleja en la oferta de acciones disponibles. Por ejemplo, el módulo Portafolio sigue el mismo modelo que el de Recursos y ofrece similares opciones, aunque adaptadas a su finalidad que es la de servir de espacio donde el alumno construye su conocimiento personal bajo la supervisión del instructor; por tanto sólo estos dos usuarios pueden acceder al mismo para dejar documentos y únicamente el alumno propietario del portafolio está habilitado para borrarlos.

También las funciones de *Cortar*, *Copiar* y *Pegar*, que están disponibles donde resultan pertinentes para agilizar la gestión, actúan de una manera consistente permitiendo mover varios objetos –como Recursos, Portafolio, Noticias o Usuarios– con facilidad y rapidez entre módulos semejantes de distintas áreas. El sistema conoce qué "portapapeles" se usa en cada momento, por lo que se pueden utilizar al mismo tiempo sin mezclar los objetos que se mueven.

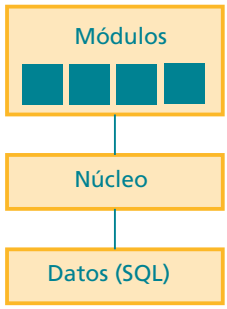
Estas semejanzas en el funcionamiento de todos los módulos permiten que el usuario se familiarice con más rapidez y redundan en una mayor facilidad de uso.

3.- Roles y permisos

En Ágora Virtual no existen permisos individuales, sino roles que habilitan para determinados conjuntos de tareas. Los roles definidos inicialmente son los ya mencionados de *propietario*, *instructor*, *asistente*, *alumno*, *invitado*, *coordinador y miembro de un proyecto*. Las funciones vienen

determinadas por las posibles acciones realizables en los módulos implementados en la aplicación, están recogidas en una tabla de funciones, y cada rol habilita para ejecutar un determinado subconjunto de ellas. La implementación de nuevos módulos llevará consigo la disponibilidad de nuevas funciones como también resulta posible establecer nuevos roles, ya que todo se reduce a definir nuevos subconjuntos, o modificar los existentes habilitándolos para funciones que no realizaran con anterioridad o restringiendo las anteriores.

FIGURA 4: ARQUITECTURA EV TÍPICA



Por ejemplo, los roles de propietario o instructor en una área de trabajo habilitan las funciones *forum_main*, *forum_new*, *forum_edit* y *forum_addpost*, mientras que el rol de alumno sólo permite al usuario acceder a *forum_main* y *forum_addpost*; es decir, puede acceder a los foros y añadir mensajes (aunque el instructor puede limitar esta función *cerrando* un foro), pero no crear un nuevo foro.

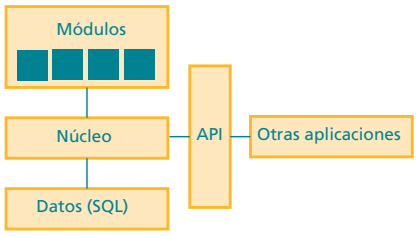
4.- Arquitectura OSID

La mayor parte de los sistemas de enseñanza virtual actuales se basa en una arquitectura constituida por diversos módulos construidos sobre un *core* o núcleo central de servicios comunes que a su vez comunica con un soporte persistente o repositorio de datos, generalmente constituido por una base de datos con lenguaje SQL (figura 4).

La mayor parte de los sistemas de enseñanza virtual actuales se basa en una arquitectura constituida por diversos módulos construidos sobre un core o núcleo central de servicios

Aunque esta solución pudiera constituir una alternativa válida en entidades que carezcan de una infraestructura de datos anterior y que, por tanto, estén en condiciones de partir de cero en la instalación de un sistema similar, la cuestión es más compleja en aquellas otras que disponen de una estructura consolidada de TI. En estos casos, la instalación de este tipo de plataformas suele plantear diversos problemas para la integración y utilización de los datos institucionales y la mayoría de las veces se hace necesario recurrir a desarrollar parches de código a medida, incluso abriendo un nuevo *fork* si es un proyecto de software libre –con lo que se pierden las posibles ventajas de estar asociados al grupo principal–, o a modificar la infraestructura existente para encajar la nueva aplicación, algo que no siempre resulta posible ni deseable.

FIGURA 5: API



La infraestructura de datos constituye, por razones económicas y de organización, el elemento de mayor persistencia en la estructura TI de cualquier institución

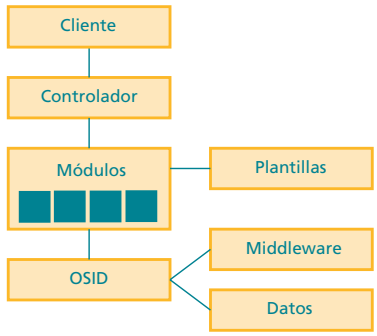
En algunos casos, los desarrolladores del entorno intentan solucionar el problema de la integración ofreciendo una API para que otras aplicaciones puedan hacer uso de las funcionalidades ofrecidas por su plataforma, según el esquema de la figura 5.

Sin embargo, aunque disponer de una API resulte útil (por ejemplo, para proporcionar servicios web), este no es el problema que se necesita resolver, sino precisamente el contrario: que la plataforma de EV pueda utilizar la infraestructura (datos) y servicios (por ejemplo, procesos de autenticación y autorización) existentes.

La infraestructura de datos constituye, por razones económicas y de organización, el elemento de mayor persistencia en la estructura TI de cualquier institución, y esa persistencia repercute en el resto

de la pirámide (*middleware*, aplicaciones y servicios al usuario final), aunque no siempre positivamente en cuanto que las necesidades de los usuarios y las tecnologías disponibles evolucionan con una rapidez no siempre asumible por la base de la pirámide. Este problema se suele afrontar de dos maneras: bien ralentizando la implantación y evolución de nuevos servicios para no dejar descolgada una costosa infraestructura heredada, o bien instalándolos sobre infraestructuras propias, con los problemas que esto representa para la consistencia de los datos y la coherencia del conjunto de la estructura institucional de TI. (Por las razones expuestas anteriormente, ésta suele ser la salida más habitual en la instalación de plataformas de enseñanza virtual). En nuestra opinión, ambos enfoques tienen más inconvenientes que ventajas.

FIGURA 6: ARQUITECTURA ABIERTA DE ÁGORA VIRTUAL



Ágora Virtual se presenta como una propuesta de arquitectura abierta

Ágora Virtual se presenta como una propuesta de arquitectura abierta con el modelo reflejado en la figura 6. El cliente puede ser cualquier navegador estándar. Todas las peticiones generadas por éste llaman al controlador que, tras realizar diversas tareas previas (como inicialización, control de sesión, comprobación de la autorización del usuario, etc.), carga el módulo correspondiente y envía el resultado al gestor de plantillas para devolverlo al cliente convenientemente formateado. El controlador constituye por tanto el único punto de acceso a la aplicación.

Los módulos se apoyan sobre una capa OSID (Open Service Interface Definitions), un conjunto de especificaciones de interfaces desarrolladas por el proyecto OKI (*Open Knowledge Initiative*) [4] que definen una serie de servicios de bajo y medio nivel presentes en la mayor parte de las aplicaciones que se utilizan en instituciones educativas y que constituyen un nivel de abstracción entre la infraestructura existente y las aplicaciones que se construyan sobre tales interfaces. Aunque escritas pensando en su uso con aplicaciones Java, las OSID son neutras en cuanto al lenguaje de implementación por lo que se pueden trasladar a otros lenguajes o incluso sería factible implantarlas en forma de servicio web si el rendimiento lo permite. (Hay, no obstante, otro proyecto de *framework* similar aunque orientado explícitamente a servicios web, el *JISC E-Learning Framework Programme* [5]).

Los módulos se apoyan sobre una capa OSID (Open Service Interface Definitions), un conjunto de especificaciones de interfaces desarrolladas por el proyecto OKI (*Open Knowledge Initiative*) [4] que definen una serie de servicios de bajo y medio nivel presentes en la mayor parte de las aplicaciones que se utilizan en instituciones educativas y que constituyen un nivel de abstracción entre la infraestructura existente y las aplicaciones que se construyan sobre tales interfaces.

Al menos dos proyectos de entornos de enseñanza virtual utilizan en alguna medida estas OSID: *Sakai*, del consorcio uPortal [6], y *Segue*, del Middlebury College [7]. Sakai es una evolución del proyecto Chef de la Universidad de Michigan. Desarrollado totalmente en Java para ejecución sobre un servidor de aplicaciones tipo Tomcat, requiere unos recursos elevados; este hecho y la complejidad de su arquitectura, componentes –Hibernate, etc.– y gestión lo ponen fuera del alcance de entidades de menor nivel como departamentos o grupos de investigación. Por su parte, el equipo del Middlebury College ha realizado una traslación de las OSID a PHP4 que nos ha servido de punto de partida para su implementación en Ágora Virtual. Sin embargo, las versiones iniciales de Segue disponían ya con anterioridad de un *framework* propio: *Harmoni*, por lo que sus desarrolladores han optado por una arquitectura mixta en la que sólo algunas funciones hacen uso de OSID, actuando ésta como una API de las correspondientes secciones del framework.

Las OSID se agrupan en *managers* según el servicio al que atiendan; por ejemplo *AuthenticationManager* o *AutorizationManager* entre las de nivel medio, o *SharedManager* para los servicios compartidos de bajo nivel como *Type*, *Iterator* y otros. Utilizar una interfaz similar tiene

Los módulos se apoyan sobre una capa OSID (Open Service Interface Definitions), un conjunto de especificaciones de interfaces desarrolladas por el proyecto OKI (Open Knowledge Initiative)

varias ventajas. En primer lugar, su modularidad permite mantener y resolver por separado cada conjunto de tareas sin entorpecer al resto de la aplicación. Además, las OSID no hacen asunciones acerca de los detalles de la implantación; tan sólo proporcionan un marco coherente en el que basar el desarrollo. No hay obligación alguna de tener implementaciones de todas las interfaces antes de empezar el desarrollo de la aplicación y tampoco las interfaces utilizadas tienen por qué incluir todos los métodos definidos, por lo que se puede aplicar una metodología de desarrollo incremental. Por ejemplo, actualmente y como prueba de concepto, Ágora Virtual hace uso de la OSID de autenticación y para ello se han implantado los métodos de la tabla I.

Void	authenticateUser(Type authenticationType)
Void	destroyAuthentication()
Void	destroyAuthenticationForType(Type authenticationType)
TypeIterator	getAuthenticationTypes()
boolean	isUserAuthenticated(Type authenticationType)

Tabla I

Se dispone en principio de dos tipos de autenticación: Login, para los accesos por medio del formulario de entrada, y HTTP-Basic, como autenticación necesaria para el acceso mediante WebDAV, actualmente en desarrollo. La necesidad de definir y operar con estos tipos de autenticación ha llevado a la implantación de los métodos del *SharedManager* que se indican en la tabla II para la gestión de *Type* y *TypeIterator*.

La principal ventaja de utilizar OSID estriba en disponer de un modelo de referencia lo bastante preciso como para constituir un marco de desarrollo práctico y efectivo, y lo bastante abierto como para servir de base a las múltiples aplicaciones que se utilizan en las instituciones académicas. El inconveniente de tener que desarrollar una implementación adaptada a la infraestructura de la institución queda compensado a medio plazo por el beneficio de disponer de un entorno coherente sobre el que basar sucesivos desarrollos, permitiendo así solventar el desfase entre una infraestructura persistente y unas aplicaciones y servicios en permanente evolución. Por ejemplo, aunque en el estado actual de la plataforma sólo se utilizan los tipos de autenticación citados anteriomente, no representaría ningún problema desarrollar el método *authenticateUser(Type authenticationType)* vinculando los tipos de autenticación a mecanismos de captura de *tokens*. Esto permitiría mantener la aplicación invariable con independencia de que los *tokens* se recojan mediante un login, una tarjeta inteligente o cualquier otro medio que se aplique en el futuro.

String	getAuthority()
String	getDescription()
String	getDomain()
String	getKeyword()
boolean	isEqual(Type type)
boolean	hasNext()
Type	next()

Tabla II

De la misma manera, la gestión de roles y permisos puede implementarse mediante el *AuthorizationManager* para hacer uso de servicios externos de autorización. Las OSID del *AuthorizationManager* giran en torno a un modelo agente-función-calificador, es decir: *quién* puede hacer *qué* y *sobre qué objeto.* Por ejemplo, los métodos indicados en la tabla III responden a las consultas más habituales en este contexto: *qué* puede hacer un determinado usuario y *quiénes* están autorizados para realizar determinada acción sobre un objeto dado.

De hecho, Ágora Virtual emplea actualmente un método equivalente para obtener la lista de acciones habilitadas para un usuario en función de su rol en una área. El menú de áreas de trabajo

La principal ventaja de utilizar OSID estriba en disponer de un modelo de referencia lo bastante preciso como para constituir un marco de desarrollo práctico y efectivo

El inconveniente de tener que desarrollar una implementación adaptada a la infraestructura de la institución queda compensado a medio plazo por el beneficio de disponer de un entorno coherente sobre el que basar sucesivos desarrollos

disponibles podría utilizar *getExplicitUserAZs()* para proporcionar la lista de áreas accesibles para un usuario; por ejemplo, a partir de los datos de matriculación en asignaturas. También el módulo de Recursos –que puede ofrecer, además de documentos, enlaces a servicios en red– podría utilizar este método para elaborar una lista de recursos reservados que sean accesibles por un usuario según los permisos que le estuvieran asignados en el directorio LDAP de la institución mediante un atributo como irisUserEntitlement del esquema IRIS [8] que con su formato URN incluyese la función y el cualificador. Alternativamente se podría recurrir al método *getAllUserAZs()* para recoger no sólo las autorizaciones explícitamente adjudicadas a un usuario, sino también las «heredadas» por su adscripción a un grupo como, por ejemplo, un departamento o servicio universitario. Las posibilidades de la interfaz OSID, como se ve, son múltiples.

La comunicación de la plataforma con el exterior se completa (aunque de manera todavía experimental) con un módulo proveedor de servicios web basado en SOAP

AuthorizationIterator	getExplicitUserAZs(id functionId, id qualifierId, boolean isActiveNow)
AuthorizationIterator	getAllUserAZs(id functionId, id qualifierId, boolean isActiveNow)
AgentIterator	getWhoCando(id functionId, Id qualifierId, boolean isActiveNow)

Tabla III

Por último, la comunicación de la plataforma con el exterior se completa (aunque de manera todavía experimental) con un módulo proveedor de servicios web basado en SOAP que habrá de proporcionar una API que permita el acceso controlado a la información propia de la plataforma.

5.- Requisitos

Ágora Virtual necesita pocos requisitos para su implantación: un servidor Apache 2.0.x y un gestor de base de datos

Ágora Virtual necesita pocos requisitos para su implantación: un servidor Apache 2.0.x y un gestor de base de datos (actualmente PostgreSQL 8.x, si bien las interfaces OSID deben permitir en un futuro utilizar cualquier otra infraestructura de datos existente con similares características). Se ha implementado en su totalidad utilizando PHP5, ya que ofrece mejor gestión de objetos que la versión 4, dispone de interfaces sin tener que recurrir al sucedáneo *extends* de la versión anterior y tiene otras mejoras que, si bien en el momento actual no han sido todavía necesarias, serán una ventaja para el desarrollo futuro. Como complemento de PHP se requieren diversos módulos PEAR y el gestor de plantillas Savant [9]. El soporte de internacionalización, indispensable en una herramienta de trabajo colaborativo en grupos internacionales (figura 7), es *Gettext* de GNU [10]. Finalmente, aunque para implementar las OSID se utilizó como punto de partida la traslación realizada por el Middlebury College de la versión 1rc6 de OSID, se optó finalmente por pasar a la versión 2.0 adaptándola a PHP5.

FIGURA 7: INTERNACIONALIZACIÓN

En el desarrollo de entornos de enseñanza virtual está relativamente extendida la tendencia a convertirlos en una especie de *superaplicación* que ofrezca todas las funciones imaginables, incluso aquellas que son satisfechas habitualmente por otras herramientas: servicios de correo, listas, mensajería, servidores de chat o editores de todo tipo. Como queda dicho, esto es explicable por la necesidad de distinguir el producto o por las presiones de grupos de usuarios entusiastas. Sin embargo, cabe otro enfoque que tome en consideración al usuario medio, menos preocupado por la novedad tecnológica que por su productividad, y que en su trabajo diario ya obtiene buen rendimiento de las herramientas con las que está familiarizado: su cliente habitual de correo, de mensajería instantánea, editores de texto o HTML, editores de ejercicios interactivos, herramientas multimedia... Es decir, se trataría, por un lado, de simplificar y aligerar el entorno y de no reinventar la rueda; por otro, de animar al usuario a ejercitar y aprovechar la "conectividad" entre sus herramientas. Al fin y al cabo, su uso será más duradero, mientras que el de la plataforma será, en la mayoría de los casos, sólo temporal. Por esta razón, en vez de incluir en la plataforma un servidor de chat propio, se ha recurrido a un servidor *jabberd2* con JRC y MUC, al que se pueda acceder con cualquier cliente Jabber/XMPP –PSI, Gaim...–, aprovechando la posibilidad de utilizar un *pipe* para dotarlo de autenticación externa y así no tener que mantener una tabla de usuarios paralela. Además, utilizar un servidor ajeno a la plataforma ofrece todas las ventajas de disponer de un producto avanzado y estándar, que se puede reemplazar con otro equivalente si las condiciones –requisitos, prestaciones– lo aconsejaran, sin afectar por ello al resto del entorno.

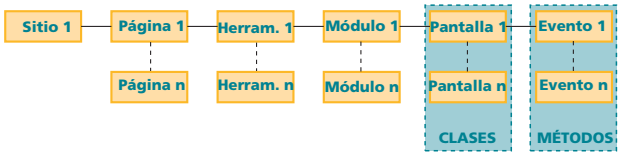
Conceptualmente, Ágora Virtual se compone de sitios web

6.- Implantación

Conceptualmente, Ágora Virtual se compone de sitios web –las áreas de los usuarios–, cada uno compuesto de una o varias páginas –los componentes de cada área–, cada una de la cuales contiene, en principio, una herramienta (aunque, con ciertas restricciones, cabría la posibilidad de hacer que incluyera varias). Una herramienta puede ser casi cualquier cosa, aunque generalmente será un componente desarrollado para funcionar dentro de este entorno. En este caso, la herramienta estará compuesta de uno o varios módulos, a su vez formado por una o varias pantallas, cada una de las cuales puede, por último, incluir uno o varios eventos. El esquema de conjunto sería el expuesto en la figura 8.

El modelo de datos está implementado en un soporte persistente –la base de datos– y la vista está representada por las distintas plantillas

FIGURA 8: ESQUEMA DE IMPLANTACIÓN



A fin de separar la lógica de la aplicación y la presentación, este modelo conceptual se implementa mediante una versión modificada del patrón *Modelo-Vista-Controlador* (MVC) propuesto en [11] que a su vez es una visión simplificada de MVC. Aunque en otros proyectos hemos utilizado una implementación MVC más convencional [12], la razón de optar por este modelo simplificado no es otra que la de utilizar una arquitectura más ligera, de fácil comprensión y utilización, en la que resulte sencillo hacer cambios en los módulos –por ejemplo, para adaptarlos paulatinamente a OSID– o incluir otros nuevos, sin afectar al conjunto de la aplicación. El modelo de datos está implementado en un soporte persistente –la base de datos– y la vista está representada por las distintas plantillas. Por último, el controlador ejecuta los diferentes módulos y muestra el resultado en el navegador del cliente.

La relación entre este modelo y el código se establece a partir de la pantalla que es la unidad básica de ejecución. Los módulos de la herramienta constituyen sólo una manera de agrupar lógicamente las distintas funciones que la forman. Por ejemplo, el módulo de la herramienta Recursos está compuesto de distintas pantallas, como la principal (*reso_main*), o la de nuevos recursos (*reso_new*). En cada una se pueden activar distintos eventos, el curso de los cuales determinará el transcurso de la aplicación.

La definición de cada herramienta especifica un modo de visualización, que por defecto será simplemente utilizar la plantilla que tiene asignada para incluir los resultados dentro de la página que se devuelve al usuario

Cada pantalla se implementa como una clase cuyos métodos se corresponden con los eventos posibles en ella. En el nivel del código hay por tanto una correspondencia directa entre pantalla-clase y evento-método, tal como se muestra en la figura 7. Por ejemplo, una clase típica para captura de datos mediante un formulario podría reducirse a los métodos de la tabla IV:

El método *init()* inicializa la clase y prepara los datos adecuados, en su caso; *display_form()* muestra el formulario utilizando la plantilla adecuada y *save_data()* hace las comprobaciones adecuadas sobre los datos y ejecuta la acción pertinente si son correctos o reenvía a *display_form()* en caso contrario.

init()
display_form()
save_data()

Tabla IV

Todas las herramientas de este tipo tienen predefinidos una pantalla y un evento iniciales. A partir de la petición recibida del cliente, y dependiendo del módulo activo y de la acción ejecutada, el controlador carga la clase correspondiente y realiza un bucle por los diferentes eventos hasta que haya resultados que mostrar (circunstancia que se señalará con la asignación de una plantilla), momento en el que el módulo termina su ejecución y se muestran los resultados en pantalla. La interacción del usuario por medio del menú de áreas, del menú de módulos o de la barra de menú del módulo en curso determinará la secuencia módulo-pantalla-evento que se ejecutará a continuación.

Pantalla aquí no debe entenderse como un modo de visualización, sino sólo como una forma de organizar los resultados de la ejecución de la cadena de eventos. La definición de cada herramienta especifica un modo de visualización, que por defecto –es decir, si la herramienta es un componente interno en PHP que sigue la arquitectura anterior– será simplemente utilizar la plantilla que tiene asignada para incluir los resultados dentro de la página que se devuelve al usuario, como es el caso de las imágenes mostradas anteriomente.

La separación entre la herramienta y el modo de visualización proporciona una gran versatilidad

Sin embargo, una herramienta puede ser también un componente externo y en tal caso el modo de visualización se ajustará para mostrarlo dentro de un iframe (figura 9) o en cualquier otro mecanismo que se habilite para ello.

FIGURA 9: HERRAMIENTAS EXTERNAS

Esta separación entre la herramienta y el modo de visualización proporciona una gran versatilidad, ya que es posible incluir fácilmente en una área de trabajo componentes proporcionados por servicios externos (aunque físicamente puedan estar localizados en el mismo servidor) con los que se dé la adecuada relación de confianza.

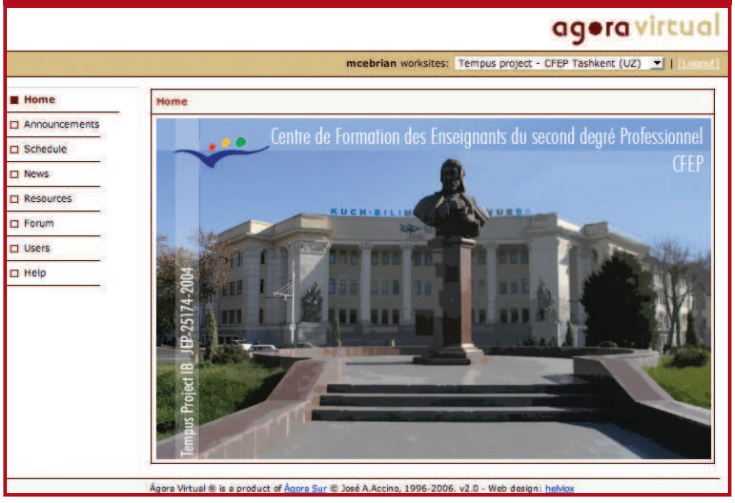
Por último, todo el diseño visual de la plataforma está recogido en una única hoja de estilo CSS, lo que facilita su modificación para adaptarla a los cambios y mejoras de funcionamiento o a otros estilos visuales.

7.- Conclusión

Ágora Virtual no se presenta como una aplicación cerrada, sino como una arquitectura abierta y ampliable. Por ello su prioridad no es el número de herramientas –aunque disponga de ellas–, ya que resulta sencillo incluir más funciones (evaluación, SCORM,...) si los usuarios las necesitan, sino su diseño estructural y la propuesta de utilización de una interfaz como OSID con una infraestructura institucional obligada a una permanente adaptación a los cambios. Por la misma razón, la interfaz visual –fácilmente configurable, por otra parte– es simple, puesto que no se trata de intimidar al usuario con un gran número de opciones, sino de permitir que vaya descubriendo la funcionalidad de la plataforma de acuerdo con sus necesidades, para que la presencia de la herramienta tecnológica no "tape" el proyecto –docente o investigador– al que sirve.

Para conocer el punto de vista del usuario sobre el entorno aquí descrito, se puso a prueba, todavía en versión beta, con un grupo de docentes especializados en el ámbito de la Tecnología Educativa procedentes de varias universidades españolas: Granada, Alicante, La Coruña, Vigo, y latinoamericanas: México (Guadalajara), Argentina, Venezuela y Brasil, y actualmente (marzo de 2006) se está empleando como soporte en varios cursos de formación y proyectos de colaboración internacional (figura 10).

FIGURA 10: UTILIZACIÓN DE ESTE ENTORNO



Desde el punto de vista de la utilización, las pruebas realizadas han puesto de relieve la amigabilidad del entorno y su neutralidad visual, la clara orientación durante la navegación y la facilidad e inmediatez de uso; mientras que en el aspecto pedagógico se han resaltado las posibilidades del *área personal* para profundizar en la capacidad de autogestión, la variedad de roles disponibles, el uso de módulos seleccionables como medio de flexibilidad y libertad pedagógica, las posibilidades de los módulos de Tareas y Portafolio de cara a la evaluación continua y al aprendizaje centrado en el alumno que demanda el EEES, el uso de la Agenda y su integración con las Tareas para la asignación de prioridades en la secuenciación, y la sustitución del habitual apartado de «contenidos» por otro de Recursos, de acuerdo con la idea de que hoy el conocimiento está en la red y por ello la *excelencia de los contenidos* puede ir dejando paso a una *excelencia del aprendizaje*.

Ágora Virtual se presenta como una aplicación abierta y ampliable. Por ello su prioridad no es el número de herramientas, sino su diseño estructural

Desde el punto de vista de la utilización, las pruebas realizadas han puesto de relieve la amigabilidad del entorno y su neutralidad visual

Referencias

(Todos los enlaces indicados están activos a 1 de marzo de 2006)

[1] ACCINO, JOSÉ ALFONSO: "Entornos integrados de enseñanza virtual". En CEBRIÁN, MANUEL (Coord.): *Enseñanza virtual para la innovación universitaria.* Madrid, 2003.

[2] CEBRIÁN, MANUEL; PÉREZ, Mª DOLORES: "¿Para qué utilizan los profesores Internet en la docencia?", en CEBRIÁN, MANUEL (Coord.). *op.cit.* También las sesiones del "Seminario Internacional sobre TIC y Calidad en la Docencia Universitaria". URV, Tarragona, 19-22 septiembre 2005, en el que han participado otras universidades españolas (UIB, UDL, UOC, UGR, EHU/UPV, UDC, US) e instituciones europeas (ICLML-International Center for Learner Managed Learning, Londres; SCIENTER-Centro di Ricerche e Servizi Avanzati per la Formazione, Bolonia; EDEN-European Distance and e-Learning Network, Budapest; FIM-NeuesLerner, Nürnberg), pendiente de publicación.

[3] ACCINO, JOSÉ ALFONSO: "Un entorno de enseñanza virtual basado en software libre". *Novática,* 2002, núm. 156, págs. 55-60. Disponible en http://www.ati.es y http://agora.edu.uma.es/documentos/novatica156-55.pdf

[4] The Open Knowledge Initiative http://www.okiproject.org/.

[5] JISC E-Learning Framework Programme http://www.elframework.org/

[6] http://sakaiproject.org/

[7] http://segue.middlebury.edu/

[8] http://www.rediris.es/ldap/esquemas/iris.schema

[9] http://www.phpsavant.com/

[10] http://www.gnu.org/software/gettext/

[11] KARAPUDA, LUKASZ: "Simplified MVC pattern for PHP Web Apps - Modularized CMSs Using Design Patterns". *PHP|Architect.* Edición electrónica. Vol. 3.12, 2004. http://www.phparch.com/

[12] PALOMARES, ROCÍO; ACCINO, JOSÉ ALFONSO: "El diseño de un sistema de información on line para la traducción profesional. La Guía de Expertos para la Traducción (GET). Actas del I congreso internacional de la Asociación Ibérica de Estudios de Traducción e Interpretación. Granada, 12-14 febrero 2003, págs. 671-679

José Alfonso Accino
(accino@uma.es)
Área de Sistemas – SCI
Universidad de Málaga