Desarrollo del portal web de la E.U. de Ingeniería Técnica en Informática de Oviedo

Daniel Gayo Avello, Benjamín López Pérez, José E. Labra Gayo
Departamento de Informática
Universidad de Oviedo
C/Calvo Sotelo s/n 33007 Oviedo
e-mail: {dani, benja, labra}@lsi.uniovi.es

Resumen

La Escuela Universitaria de Ingeniería Técnica en Informática de Oviedo (EUITIO) viene manteniendo desde hace unos años un sitio web con el fin de proporcionar todo tipo de información a la comunidad educativa del Centro; la información generada (exámenes, conferencias, cursos, becas, ofertas de empleo, etc.) es enorme y hace totalmente inviable un mantenimiento manual de la misma, exigiendo una gestión más próxima a la de un portal temático.

En este artículo se pretenden esbozar las distintas posibilidades existentes para la implantación de herramientas que faciliten la publicación de información en un Centro universitario; para ello se describirá el trabajo que se viene llevando a cabo en la EUITIO desde hace un año para dotarla de un *website* renovado que permita la oferta de información permanentemente actualizada de la forma más sencilla posible.

1. Aproximación al problema

1.1. Un poco de historia

El sitio web de la EUITIO se inauguró a comienzos del curso 96/97 y durante los últimos tres años de funcionamiento se ha convertido en el referente fundamental en cuanto a temas académicos, formativos, profesionales e incluso lúdicos relacionados con el Centro y sus miembros.

Sin embargo, este sitio padecía una dolencia que, por desgracia, es aún frecuente en muchas de las organizaciones con presencia en la web: la limitada mantenibilidad del mismo debido a una gestión totalmente manual de la información. Este problema suponía que los servicios ofrecidos por el Centro en la web eran muy limitados y, aún así, requerían un esfuerzo considerable por parte de los encargados de administrar y mantener el sitio.

A principios del curso 1999/2000 se toma la decisión de reconstruir enteramente el sitio, debía abandonarse el enfoque seguido hasta ese momento que lo reducía a un mero repositorio de documentos y adoptar otro que implicaba el desarrollo de una serie de aplicaciones web que proporcionasen las herramientas necesarias para poder administrar un portal con publicación *on-line* que, en definitiva, era en lo que se había convertido el sitio de la Escuela.

Además, ese planteamiento del sitio web como aplicación Internet favorecería la futura prestación de nuevos servicios; por ejemplo: la diversificación de personas que pudieran "publicar" aparte de los administradores (dirección, profesores, etc.), la posibilidad de ofrecer un interfaz web más adecuado para los alumnos (personalizable según sus asignaturas y preferencias, posibilidad de realizar ciertos trámites *on-line*, etc.), así como las ventajas para implementar ciertas aplicaciones "ubicuas".

1.2. Forma de trabajo

A la hora de construir un sitio web es necesario determinar qué servicios se van a prestar y a quién se desea prestar dichos servicios. Para ello se parte de la información que maneja la organización y se procede a especificar la forma de manipularla y consultarla en la web, así como quién y de qué modo operará sobre dicha información.

En la Figura 1 se muestra el modelo de datos de la información que el Centro deseaba tener

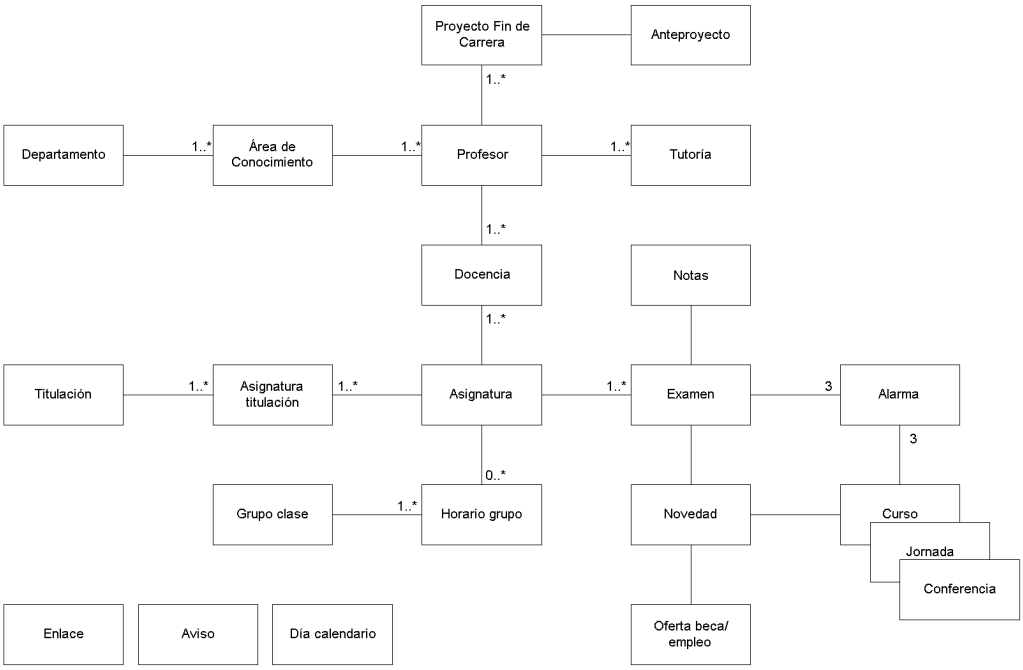

Figura 1. Modelo de datos para la información de la EUITIO

accesible a través de la web. A partir de dicho modelo de datos se procede a determinar los servicios básicos del sitio en función del elemento de información y del tipo de usuario.

Una vez se dispone de una lista completa de aquellos servicios relacionados con la manipulación y consulta de la información se hace necesario especificar otros nuevos que faciliten la prestación de los primeros.

Así, en el caso del sitio de la EUITIO, se decide proporcionar cuatro servicios "avanzados"; para el público un sistema de búsqueda, y para los administradores un sistema automático para la generación de alarmas y novedades, un interfaz para la importación de grandes volúmenes de datos y un interfaz para la carga de ficheros hacia el sitio. Con ello se pretenden lograr dos cosas: facilitar al público general la obtención de información y a los administradores la posibilidad de trabajar sobre el sitio desde cualquier ubicación física con sólo disponer de un navegador web.

Es decir, el sitio web de la Escuela debería permitir:

- La consulta de información pública (asignaturas, profesores, exámenes, cursos, etc.) a todo el público.
- La consulta de información privada (notas de exámenes) solamente a alumnos, profesores y administradores.
- La introducción, modificación y eliminación de cualquier tipo de información por parte de los administradores.
- La modificación de información sobre su persona y asignaturas a los profesores.
- La gestión de anteproyectos a profesores y alumnos: el alta podrá realizarla tanto el alumno como el profesor, baja y modificación sólo podrán ser llevadas a cabo por el profesor.
- La introducción y manipulación de avisos y enlaces por parte de los administradores y los profesores.

Además de todo lo anterior, se primaría la futura extensibilidad del sistema, a fin de poder

incluir nuevos tipos de usuario, actividades y otras funcionalidades.

1.3. Solución propuesta

Conocidos ya los servicios a prestar se plantea el esquema básico de la solución; consistirá ésta en una aplicación Internet que permita gestionar toda la información pública del Centro; para ello, dicha información deberá estar almacenada de forma conveniente en una base de datos que será consultada y actualizada mediante un interfaz web.

Esta solución presenta múltiples ventajas, algunas de ellas por simple oposición al sitio antiguo:

- Facilidad de mantenimiento de la información.
- Estilo homogéneo en todos los documentos.
- Control total sobre el crecimiento del sitio.
- Publicación inmediata de cualquier tipo de información.
- Escalabilidad (incorporación de nuevos servicios de forma sencilla y paulatina).
- Discriminación de servicios según el tipo de usuario; por ejemplo, cada profesor podría controlar su información y la de sus asignaturas pero únicamente ésa, sólo los alumnos tendrían acceso a las notas de los exámenes, etc.
- Control sobre la información publicada por parte del autor al eliminar intermediarios (administrador o *webmaster*).

Por otra parte, este esquema de solución es totalmente extrapolable a cualquier tipo de sitio que deba llevar a cabo alguna forma de publicación *on-line*, ya se trate de una organización que desee informar puntualmente sobre los eventos que lleva a cabo o de una publicación electrónica.

En todos los casos deberá determinarse la estructura de la información que maneja la organización, los distintos tipos de usuarios a servir y los servicios a prestar a cada uno de ellos; una vez hecho esto, la información se almacenará siempre en una base de datos, los servicios serán implementados mediante un lenguaje de programación capaz de trabajar con la misma y generar documentos *HTML* y el acceso a dichos servicios se hará a través de un interfaz web.

2. Tecnologías necesarias

2.1. Elementos del sistema

A la hora de implementar una aplicación Internet como la descrita, y dejando a un lado el *hardware*, es necesario tomar una serie de decisiones sobre la elección de cuatro elementos diferentes para la construcción del sistema, a saber:

- Sistema operativo.
- Servidor web.
- Sistema de Gestión de Bases de Datos (SGBD).
- Lenguaje de desarrollo.

Existen múltiples artículos que pretenden analizar objetivamente sistemas operativos [5], SGBD's [12], servidores web [13] o lenguajes [4][3]; un término bien conocido en el campo de las comparativas es el de "guerra cuasi-religiosa" lo cual dice mucho acerca del modo en que finalmente y de forma "objetiva" se escoge el operativo, el servidor o el lenguaje de desarrollo.

En el caso del nuevo sitio web de la EUITIO el proceso de selección de los componentes primó considerablemente la gratuidad de los mismos con lo cual los elementos elegidos fueron los siguientes:

- *Linux* como sistema operativo.
- *Apache* como servidor web [1].
- *PostgreSQL* como SGBD [10].
- *PHP* como lenguaje de desarrollo [9].

2.2. El motor de base de datos

La selección tanto del operativo como del servidor web eran prácticamente obligadas al partir de un requisito como la gratuidad del software pero no sucedía así en el caso del SGBD ni del lenguaje de desarrollo, ¿qué fue entonces lo que llevó a esa elección, máxime cuando uno de los autores era fiel devoto de *Perl* [8] y *mySQL* [6]?

Por lo que se refiere al motor de la base de datos existían tres claras opciones: mySQL (la preferida inicialmente), *PostgreSQL* y *Oracle* [7]

(se decidió estudiarla someramente a pesar de no ser gratuita). Para la selección del SGBD (como para la del lenguaje) no se llevó a cabo una prueba exhaustiva sino que se recurrió a la documentación disponible en Internet en busca de análisis y comparativas entre los tres motores; la opción elegida a priori, *mySQL*, resultó ser inferior respecto a *PostgreSQL* y a *Oracle* (menor grado de compatibilidad con el estándar ANSI SQL, carencia de transacciones, etc.)

Eliminada *mySQL*, las opciones eran claras: *Oracle* era un sistema superior pero no gratuito frente a *PostgreSQL* que estaba disponible sin coste alguno para cualquier plataforma, proporcionaba el código fuente, con las ventajas de adaptación que esto supone y, aún disponiendo de menos prestaciones podría servir perfectamente a las necesidades del nuevo sitio. Se eligió así *PostgreSQL* como motor para la base de datos del sistema no pudiendo manifestarse queja alguna sobre el rendimiento de la misma durante todo este tiempo.

2.3. El lenguaje de programación

Por lo que respecta a la implementación de las distintas herramientas de la aplicación se abrían varias posibilidades tecnológicas:

- *CGI*
- *Servlets*
- *ASP*
- *PHP*

A continuación se describirá someramente en qué consiste cada una de ellas:

CGI es una especificación para la transferencia de información entre un servidor web y un programa que puede estar escrito en cualquier lenguaje siendo *Perl* el más habitual. Un problema que se plantea es el de la lentitud de ejecución debido a la necesidad de iniciar un nuevo proceso cada vez que se invoca un *script CGI*; proceso que, en el caso de *Perl*, es interpretado. Se suelen mencionar los *servlets* como una alternativa al *CGI*, sin embargo, existe una extensión para este interfaz, conocida como *FastCGI*, que permite *scripts* persistentes con lo cual el problema de crear un nuevo proceso por cada invocación desaparece; *FastCGI* es una opción más que recomendable para aquellos desarrolladores con gran cantidad de código *CGI* heredado.

Los *servlets* son el planteamiento ofrecido por la plataforma *Java* para extender servidores web; los *servlets* son independientes de la plataforma y del servidor y pueden acceder a todos los *APIs* de *Java* posibilitando el desarrollo de verdaderas aplicaciones en la web. La principal ventaja que se suele esgrimir al comparar *servlets* con *CGI* es el mayor rendimiento.

ASP es un entorno de programación basado en tecnologías de *script* para el servidor web *MS-Internet Information Server*; *ASP* permite combinar *HTML* y *scripts* en diversos lenguajes (*VBScript* y *Jscript*) para crear páginas dinámicas.

Por su parte, *PHP* es comparado a menudo con *ASP* por tratarse de un lenguaje que se incrusta en las páginas *HTML* y es interpretado por el servidor; *PHP* puede llevar a cabo cualquier tarea posible mediante *CGI* (y, por tanto, mediante *servlets*), además, proporciona una gran cantidad de interfaces para distintas bases de datos así como la posibilidad de emplear diversos protocolos de red (*HTTP*, *IMAP*, *POP3*, etc.)

Como se mencionó con anterioridad, la idea original era implementar el sistema mediante *Perl* y *mySQL* pero, una vez desechada la base de datos inicialmente elegida, la fe había sido puesta a prueba, así pues, ¿qué tecnología/lenguaje sería el más adecuado?

Para tratar de determinarlo se realizó una búsqueda lo más exhaustiva posible de artículos, comentarios y casos de uso de los distintos lenguajes en lid (a saber, *Perl*, *Java* y *PHP*); se recomienda al lector interesado consulte algunas de las reseñas bibliográficas empleadas [11][14][2].

Tras la consulta de la bibliografía sólo se podía llegar a una conclusión: en esencia, todos los lenguajes podían llevar a cabo las mismas tareas y con prestaciones muy similares.

Uno de los estribillos más oídos fue el de "la lentitud de *CGI* por la necesidad de lanzar un nuevo proceso con cada invocación del *script*" olvidando hábilmente la existencia de *FastCGI* o *mod_perl* que permiten la ejecución de *scripts CGI* de forma mucho más ágil.

Por otra parte, los incondicionales del *Perl* se aferraban a dichas posibilidades para asegurar una larga vida a *CGI* frente a los *servlets* y a cantar las "facilidades" del lenguaje que habían elegido.

De la misma forma, los usuarios de *PHP* recordaban continuamente que el lenguaje se incrusta en los documentos *HTML* así como la gran cantidad de interfaces para bases de datos que soporta (no mencionaban que tanto *Java* como *Perl* tienen un soporte de SGBD's tan bueno como *PHP*).

A la vista de esto, ¿qué fue lo que nos decantó por *PHP* para el desarrollo del nuevo sitio de la EUITIO? Se disponía de tres lenguajes, *Perl*, *Java* y *PHP*, todos ellos podían servir perfectamente para las necesidades del proyecto a desarrollar, sin embargo, existía una peculiaridad respecto a dicho proyecto que no ha sido mencionada hasta el momento.

La peculiaridad en cuestión se refiere al personal encargado del futuro mantenimiento de este nuevo sitio web; dicho personal está constituido por becarios (alumnos de último curso con pocas asignaturas pendientes) que, además de otras muchas tareas, deberán dedicar algo de tiempo al servidor web. Un becario podría, teóricamente, prestar sus servicios durante dos años académicos pero, en general, dichos servicios se prolongan durante unos nueve meses.

Cada lenguaje pasó entonces a considerarse a la luz de dicho punto:

- *Perl*: los *scripts* escritos en *Perl* son crípticos y difíciles de mantener incluso por el autor de los mismos, pretender que un proyecto como un sitio web académico fuera desarrollado en *Perl* por un becario distinto cada año y se mantuviera comprensible y mantenible parecía difícil de creer, así que se decidió desechar este lenguaje.
- *Java*: un lenguaje muy potente y una de las elecciones más adecuadas para un desarrollo como el descrito. Sin embargo, tampoco fue el lenguaje elegido; la razón fue coyuntural: los becarios encargados del mantenimiento y futuro desarrollo del sitio web de la EUITIO son alumnos de los últimos cursos que no tienen por qué conocer nada de tecnologías Internet; un proyecto como éste desarrollado integramente en *Java* podía resultar demasiado ambicioso para un solo alumno trabajando de forma discontinua. No obstante, en el futuro próximo los alumnos adquirirán de forma cada vez más temprana conocimientos sobre *Java* y desarrollo en Internet, momento en el cual habrá que plantearse la posibilidad de migrar el sistema a un entorno basado en *servlets*.
- *PHP*: la última esperanza, para ser de utilidad con los requisitos planteados dicho lenguaje tendría que tener una curva de aprendizaje poco pronunciada y un período de entrenamiento corto; además, tendría que tener una sintaxis fácilmente comprensible y facilitar en cierta medida el mantenimiento. ¿Era esto así? Por suerte sí, la sintaxis de *PHP* se asemeja a la del lenguaje C y a los típicos lenguajes de *script* como *Tcl*, resulta sencillo de comprender y de mantener; por otra parte, la gran calidad de la documentación disponible y el enorme número de funciones que proporciona hacen realmente sencillo su empleo. Para comprobar la validez de esta hipótesis se planteó un período de aprendizaje de un par de semanas al término del cual se decidiría si el sistema se implementaría en dicho lenguaje, al finalizar este período la respuesta fue afirmativa y comenzó todo el proceso de desarrollo.

2.4. Integración en un sistema completo

Por último, la integración final de todos los elementos es relativamente sencilla. Tanto *Apache* como *PostreSQL* proporcionan dos servidores que serán los encargados de atender las peticiones recibidas de los clientes web y las consultas SQL recibidas a través de la red, respectivamente.

Por su parte, el código *PHP* será interpretado por un módulo que se integra en *Apache* de tal forma que, una vez configurado el servidor web, los documentos con una extensión determinada serán preprocesados por dicho módulo, que interpretará el código *PHP* que haya en los mismos, para generar un documento *HTML* que será enviado por el servidor hacia el cliente. Por lo que respecta a las consultas SQL que ejecutará el módulo de *PHP* contra la base de datos, éstas serán llevadas a cabo mediante una comunicación *TCP/IP* que establecerá el módulo *PHP* con el servidor del SGBD.

Figura 2. A la izquierda, presentación de la información en el portal; a la derecha, interfaz de administración.

3. Conclusiones

En la actualidad existen gran cantidad de tecnologías disponibles para el desarrollo de sitios proveedores de información dinámica; tecnologías que varían enormemente en términos de coste, calidad y dificultad de desarrollo. A lo largo de este artículo se ha venido hablando de una serie de paquetes de software y lenguajes de programación que aúnan en un solo producto tres grandes virtudes: coste razonable, buenas prestaciones y facilidad de uso.

Creemos que sistemas construidos sobre el conjunto *Linux-Apache-postgreSQL-PHP* pueden resultar perfectamente satisfactorios para organizaciones pequeñas y medianas con presencia en la web o para proveedores de servicios avanzados en Internet que deseen implantar soluciones de bajo coste.

Referencias

[1] Apache Software Foundation. *Apache.* http://www.apache.org

[2] Paul Ferris. *PHP 4.0: Dynamic Content for the Web Warrior.* http://www.linuxplanet.com/linuxplanet/print/1891/ (2000)

[3] Craig Knudsen. *Servlets or CGI/Perl.* http://www.sunworld.com/swol-12-1998/swol-12-servlets.html (1998)

[4] Cameron Laird, Kathryn Soraiz. *Choosing a scripting language.* http://www.sunworld.com/swol-10-1997/swo-l-10-scripting.html (1997)

[5] Cameron Laird. *Linux versus NT. Are you getting the most from your OS?* http://www.sunworld.com/sunworldonline/swol-08-1998/swol-08-linuxvnt.html (2000)

[6] MySQL AB. *MySQL*. http://www.mysql.com

[7] Oracle Corp. *Oracle.* http://www.oracle.com

[8] O'Reilly & Associates, Inc. *Perl.* http://www.perl.org

[9] PHP Development Team. *PHP: Hypertext Preprocessor.* http://www.php.net

[10] PostgreSQL Global Development Group. *PostgreSQL*. http://www.postgresql.org

[11] Jamie Scheinblum. *Intro to Mod_Perl.* http://hotwired.lycos.com/webmonkey/98/38/index2a.html (1998)

[12] Matthias Warkus. *Apples and Oranges: A Linux DBMS Comparison.* http://www.linuxplanet.com/linuxplanet/print/1224/, 1251/ y 1282/ (1999)

[13] Gregory Yerxa, Ahmad Abualsamid. *The Best Bets for Web Development.* http://www.nwc.com/1020/1020f1.html (1999)

[14] Richard M. Yumul. *Getting Started with Java Servlets using Apache Jserv.* http://www.devshed.com/Server_Side/JServ/Started (1999)