UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

PROYECTO FINAL DE CARRERA

"Diseño e Implementación de Módulos Docentes en Joomla"

DISCA-59

Autor:
Vicente Reolid Serrano

Dirigido por:
Dr. Lenin G. Lemus Zúñiga
Profesor Titular de la Universidad Politécnica de Valencia

ÍNDICE

RELACIÓN DE FIGURAS.

1. INTRODUCCIÓN.

En el informe [1] “Experiencias en el uso de las TIC’s para facilitar la impartición de la asignatura de grado Fundamentos de Computadores” se avanzó un nuevo modelo de evaluación en el proceso de implantación de las asignaturas de Grado en la Universidad Politécnica de Valencia. Para la evaluación continuada de los alumnos se creó una herramienta que permite la realización de controles, apoyada en un Servidor Web.

El modelo basado en web se ha desarrollado utilizando un Sistema de Gestión de Contenidos basado en *Joomla* e implementado como extensiones al mismo, extensiones denominadas componentes.

Dada la aceptación de la herramienta y sus posibilidades se considera útil la evolución de la misma como continuidad al ciclo iniciado.

Con el fin de avanzar en la evolución y mejora de la herramienta de evaluación de los controles realizados por los alumnos de la asignatura de Grado, “Fundamentos de Computadores”, se plantea la posibilidad por un lado, de facilitar al alumno un informe detallado de la evaluación realizada y por otro dar la posibilidad del envío automatizado de las notas obtenidas por los alumnos a la plataforma de la Universidad “Poliformat”.

Se considera de utilidad, además, la posibilidad de mantener tanto la Base de Datos de alumnos como el registro de los mismos, para que el acceso a la herramienta lo realicen como usuarios registrados, aprovechando para ello las características de la plataforma. Partiendo de un listado de alumnos extraído de la plataforma de la Universidad, facilitar su importación y registro automatizado.

2. OBJETIVO.

El objetivo de este PFC es desarrollar e implementar un componente que permita gestionar las diferentes tareas que el alumno ha de realizar como parte de la evaluación de la asignatura. El profesor podrá dar de alta la tarea, especificando el tipo de tarea a realizar, un identificador de tarea y la fecha de inicio, finalización y publicación de la misma. Junto a la gestión de tareas se implementará la opción que permita agregar los alumnos a la aplicación, siguiendo un procedimiento automático y dirigido.

Se desarrollará además, en cada una de los diferentes cuestionarios ya existentes la utilidad de obtención de un informe en formato PDF que contendrá una relación de aciertos y errores junto con las notas obtenidas por el alumno, una vez alcanzada la fecha de publicación del mismo.

Se implementará también una utilidad para la publicación de las notas, donde el profesor introducirá en la tarea, su referencia a la tarea de Poliformat, que obtendrá después de haber creado la misma en la aplicación de la Universidad. Esto permitirá relacionar la tarea del sistema de evaluación con la tarea de Poliformat, utilizando un servicio web para la transferencia de las notas del alumno de la Aplicación de Evaluación a la aplicación de Poliformat.

3. METODOLOGÍA.

El procedimiento a seguir para la consecución de los objetivos, parte del conocimiento de las herramientas y tecnologías necesitadas en la implementación, así como la descripción de los componentes y su funcionamiento en el sistema. Ello se aborda siguiendo el siguiente orden metodológico:

- Descripción del CMS JOOMLA.
    - Introducción.
    - MVC.
    - Creación de un componente.
- Descripción sucinta de las TECNOLOGÍAS.
    - JQuery.
    - Json.
    - JTable.
    - Web Services.
    - TCPDF.
- DESARROLLO DEL COMPONENTE.
    - Estructura del Componente.
    - Adecuaciones y cambios en los componentes de Cuestionarios.
    - Módulo de Instalación del Componente
- INSTALACIÓN, CONFIGURACION Y VALIDACIÓN DE LA APLICACIÓN.
    - Joomla, Componentes y Utilidades.

4 EL CMS JOOMLA.

En este capítulo se presenta el Sistema de Gestión de Contenidos de *Joomla*, dando una pequeña introducción de *Joomla* como un Servicio para Web, cual es su arquitectura de desarrollo, que son los componentes, como parte fundamental para la extensión de *Jooma*, su implementación y un ejemplo de desarrollo de un componente. [2][3][4].

4.1 Introducción.

Un sistema de Gestión de Contenidos *CMS* (*Content Management System*) es una aplicación que permite mantener una infraestructura Web. Con la interfaz se mantiene de forma independiente tanto el contenido como el diseño. Lo que permite en cualquier momento variar el diseño y la presentación sin necesidad de reformatear todo el contenido.

Joomla es un sistema de Gestión de Contenidos *CMS* de código abierto (gratuito tanto para su uso como para su desarrollo) que permite la edición de contenido de un sitio Web de una forma sencilla. Está programado en *PHP* y requiere de una base de datos *MySQL*, así como de un servidor *HTTP* sobre el que funciona, normalmente Apache. Sus principales características incluyen flash con noticias, blog, foros, encuestas, calendarios, múltiples idiomas, variedad de presentaciones y un sinfín de opciones. Todas estas características crecen de manera considerable, gracias a la posibilidad de incorporar diferentes extensiones. Existen diferentes tipos de extensiones, plantillas, idiomas, componentes, etc.

4.2 El Patrón MVC.

El patrón de diseño de software que utiliza Joomla es el patrón Modelo, Vista, Controlador *MVC* (*Model View Controller*).

El Modelo, Vista, Controlador *MVC*, es un patrón de arquitectura del software similar al patrón de desarrollo de tres capas (persistencia, lógica de negocio, presentación) en el que se separan los datos que requiere la aplicación, de la lógica de control y esta a su vez de la presentación (interfaz de usuario) en tres componentes distintos. Este patrón es muy común en las aplicaciones web.

- *El Modelo*: Contiene el código necesario para suministrar los datos que le sean solicitados, independientemente de que estos provengan de una base de datos, ficheros *XML* o cualquier otro soporte con el fin de generar la vista solicitada.

- *La Vista*: En ella se incluye el código correspondiente a la presentación del modelo, posibilitando además la interacción del usuario con el controlador, se corresponde con la interfaz de usuario.

- *El Controlador*: Es el punto de entrada de la aplicación y el que ejecuta la mayor parte de la lógica de la aplicación. Permanece en ciclo continuo, atendiendo las peticiones recibidas a través de la interfaz, e interactúa con el modelo, actualizándolo en función de la petición recibida, con el fin de generar la vista solicitada por el usuario.

4.3 Creación de un Componente.

Entre las diferentes extensiones existentes en *Joomla* se encuentran los componentes, que son en realidad aplicaciones independientes, que haciendo uso del núcleo y del patrón de diseño MVC de *Joomla*, pueden ser agregadas al entorno de trabajo de *Joomla* y convertirse en parte de él.

Los componentes son fundamentales en *Joomla*, todo lo que se muestra en la interfaz utilizando *Joomla* se realiza utilizando componentes. El componente se encarga de generar el contenido que se visualiza.

En *Joomla* podemos observar dos lados claramente diferenciados: Presentación y Administración.

Presentación ó *Front-End*. Es la parte visible para el usuario.

Administración ó *Back-End*. Es la parte que se hace visible cuando se accede como administrador para gestionar el sitio.

Esta forma de trabajar en *Joomla* afecta al desarrollo de los componentes, puesto que el componente una vez desarrollado tendrá una componente de visualización para el usuario y otra de administración.

No es obligatoria la existencia de las dos partes de un componente, puede que existan componentes que solamente existan en el lado de presentación, otros en el lado de administración y otros en ambos. Todo depende de la acción y la funcionalidad del componente.

Dentro de la estructura de directorios de *Joomla* los componentes se ubican en sitios diferentes en función de si se trata de la parte de presentación o la de administración. La parte de presentación se ubica partiendo de la raíz de la instalación de *Joomla* en el directorio *components*, mientras que la parte de administración se ubica en *administrator/components*.

Una vez dentro del directorio *components* (cualquiera de los dos) se puede observar la existencia de un conjunto de subdirectorios, cada uno de estos subdirectorios es un componente, que ha sido instalado por defecto durante el proceso de instalación de *Joomla*. Así mismo se puede comprobar que todos siguen un patrón y es que todos tienen el prefijo *com_* y a continuación el nombre del componente. Esto nos lleva a la primera norma que pone *Joomla*.

Dado que se trata de la creación de un componente, vamos a crear un componente de ejemplo y le denominaremos “prueba”. El componente que se cree se debe ubicar dentro de un directorio que se denomine *com_nombredelcomponente*, por ello, hemos de crear un directorio denominado *com_prueba*.

Una vez creado el directorio creamos la estructura del componente, es decir, todos aquellos ficheros y directorios que componen el despliegue del mismo, dado que hemos de seguir el modelo de desarrollo *MVC*. Los ficheros los creamos en blanco y posteriormente los iremos completando.

Para ello crearemos en el directorio raíz de nuestro componente el fichero *prueba.php.* El fichero tiene que llamarse igual que el nombre del componente, dado que *Joomla*, cuando intente cargar el componente lo que hará será buscar en el directorio de nuestro componente el nombre de archivo con extensión php (el desarrollo en *Joomla* se realiza con PHP) que coincida con nuestro componente, en nuestro caso, *prueba.php*. Este será el punto de entrada de nuestra aplicación.

En el mismo directorio crearemos además el fichero controller.php, este será llamado por el fichero de entrada a la aplicación y será el encargado de atender todas las peticiones de usuario y realizar las diferentes tareas que éste requiera de la aplicación. Contendrá la parte de desarrollo correspondiente al apartado Controlador del patrón *MVC*. Podrían existir diferentes controladores que fueran llamados desde el punto de entrada, en función del tipo de tareas a realizar o funcionalidad, por ejemplo, si en la lógica de negocio se utilizara la tecnología *AJAX*, se podría tener un controlador diferente, p. ej. *controllerajax.php* que tendría su lógica propia, diferenciada del otro. En nuestro caso, vamos a utilizar solamente el primero.

A continuación, para completar el patrón, y partiendo del raíz de nuestro componente se crear los directorios *models* y *views* que contendrán la parte correspondiente al Modelo y la Vista del patrón *MVC*.

Nos situamos dentro del directorio *models*, que será donde se encontrarán los posibles modelos de acceso a datos con el fin de recuperar y/o modificar información y preparar los datos para entregárselos a la vista. Crearemos un fichero denominado *prueba.php* siguiendo la nomenclatura de *Joomla* de poner en cada una de las tres partes del patrón MVC, el mismo nombre de fichero que el del componente.

Si además de conexión con base de datos, fuera necesario el acceso a otro tipo de tablas o ficheros de datos (XML, texto, etc.) adicionales de la aplicación podríamos crear un directorio que los contuviera.

Tras la creación del modelo y el controlador, pasamos a crear la parte de la estructura correspondiente a la vista. Nos situamos en el directorio *views* y creamos un directorio con el mismo nombre que el componente *prueba*, dentro del cual a su vez, creamos un fichero denominado *view.html.php* que obtiene la información del modelo, quien ha obtenido la información previamente de la Base de Datos, y una vez obtenida dicha información pudiendo incluso modificarla, la pone a disposición de la vista, a través de la plantilla (layout) que será la encargada de transformar la vista en el formato solicitado por el usuario(*HTML*, *PDF*, …). Dentro del directorio de prueba, se crea un directorio *tmpl* (template) que contiene las diferentes plantillas que se pueden utilizar para generar la vista solicitada por el usuario. Cuando solamente existe una plantilla, esta tiene un nombre por defecto para joomla. En nuestro caso como solo existirá una plantilla, se crea dentro del directorio tmpl, el fichero *default.php*.

Tras realizar la secuencia de pasos descrita hasta ahora ya tenemos establecida la estructura básica en la creación del componente. La imagen siguiente muestra gráficamente dicha estructura.

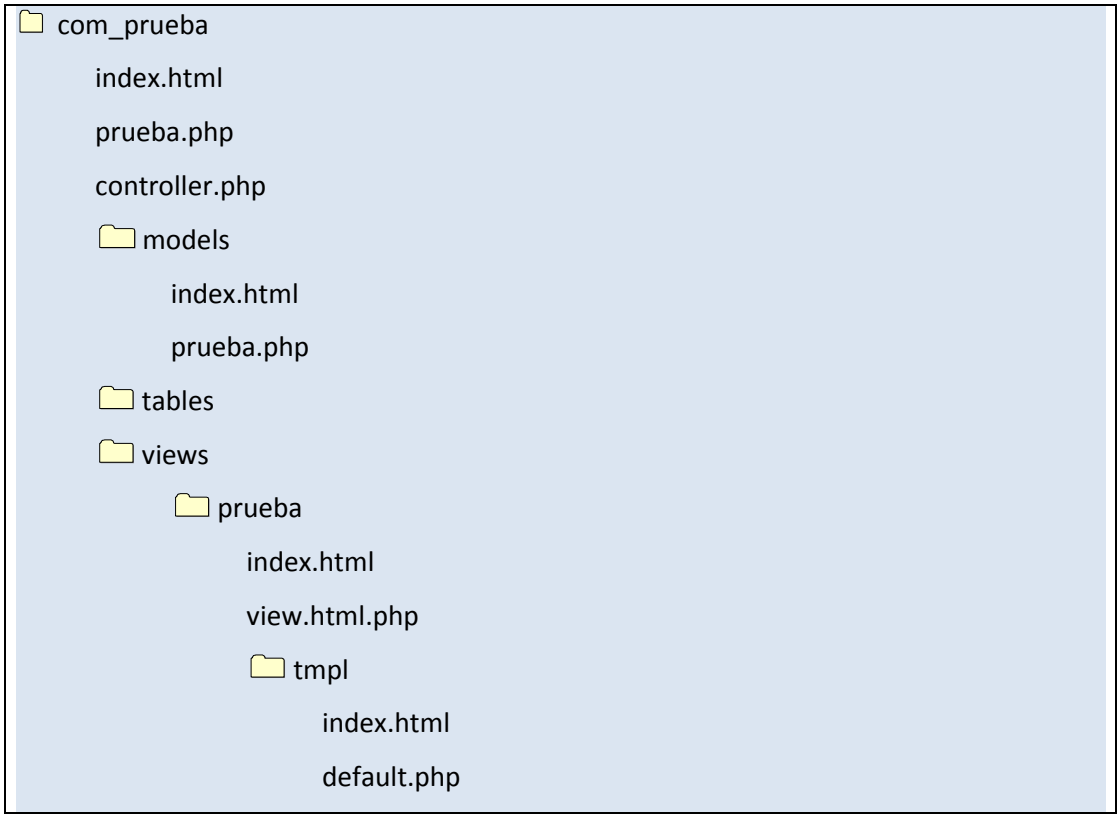


Figura 1. Estructura de un componente en Joomla.

5. TECNOLOGÍAS COMPLEMENTARIAS.

Con el fin de facilitar el desarrollo e implementación de código en entornos Web, han ido apareciendo en la red, una serie de tecnologías; herramientas, bibliotecas y utilidades, que de un lado, facilitan tanto el intercambio de objetos, como el acceso a los mismos y de otro han posibilitado la interconexión entre entornos y sistemas diferentes. A continuación se relacionan y describen aquellas que se usan en el desarrollo de nuestra aplicación.

5.1 JQuery.

Es una librería de funciones escritas en *JavaScript*, que posibilita interactuar con los documentos *HTML*, el árbol del *DOM* y la tecnología *AJAX* que implementan los diferentes navegadores web existentes, de manera transparente a las características especificas de cada uno de ellos. Es decir, hace independiente el uso de los recursos de los navegadores Web de la forma en los que el fabricante haya definido como han de ser utilizados estos. [5] [6].

Aunque el estándar *W3C* ha definido estándares tanto para el uso de *HTML* como del *DOM* y de *AJAX*, la forma de acceso a los recursos de cada agente de usuario (navegador) ha sido establecida por sus creadores y no siempre se corresponde con el estándar. Es por ello que desarrollar código en la parte cliente se hace complejo dadas las peculiaridades y especificidades de cada navegador.

Con la finalidad de evitar esta serie de restricciones, *JQuery* se desarrolla como un conjunto de funciones con una sintaxis propia, que implementa toda la funcionalidad necesaria para acceder a las tecnologías que se utilizan en los clientes de navegación, adaptando su ejecución a la especifica del agente de usuario donde se vaya a utilizar, por lo que la hace ser independiente de donde se utilice y simplificando además la inserción de código *JavaScript* para el desarrollador.

Su finalidad es la de facilitar con una sintaxis sencilla y un conjunto de funciones comunes la forma en gestionar los diferentes eventos y funcionalidades de los agentes de usuario, tanto eventos, estilos, propiedades y características.

Las librerías *JQuery* se obtienen fácilmente de internet en nuestro caso las hemos descargado de la página oficial [7] la versión 1.4.4.2 y se han situado en un directorio auxiliar que parte del directorio público Web */jscripts/jquery/jquery.1.4.4.min.js*

5.2 Json.

En la actualidad para el intercambio de información entre el cliente y el servidor en entornos Web se utiliza la tecnología *AJAX* la cual permite que haciendo uso de *JavaScript* en la parte del Cliente del objeto *XMLHTTPRequest* permite la transferencia asíncrona de información entre cliente y servidor. Si la información requerida desde el objeto de *AJAX* está en formato *TXT*, esta puede ser procesada sin más, sin embargo, si está en formato *XML*, el objeto *XMLHTTPRequest* ha de realizar un análisis del documento devuelto y construir con él un árbol *DOM* en memoria del *XML*. Una vez construido se pueden utilizar los métodos estándar de acceso al *DOM*, navegar por el árbol y recuperar los elementos, atributos y el resto

de información que contiene el árbol. Esto supone un elevado coste procesamiento en la parte del cliente [7].

A pesar de que *XML* sea la manera estándar de intercambiar datos, no siempre es la mejor, debido a su descripción excesivamente detallada y la necesidad de un analizador no trivial para su validación (construir el árbol *DOM* y posteriormente construir los objetos *JavaScript*).

Con el fin de mejorar el rendimiento en la parte del cliente aparece *JSON* (*JavaScript Object Notation*). Su principal importancia proviene del hecho de que es un subconjunto del lenguaje *JavaScript* en sí mismo. Tiene muchas características en común con *XML* como el poseer una sintaxis con anidamiento de datos al igual que XML, aunque más ligera. También está basado en texto, al igual que *XML*. Utiliza *UNICODE* y es igual de legible como *XML*, pero a diferencia de este es más claro y menos redundante. Mientras que *XML* es apropiado para marcar documentos, *JSON* es ideal para el intercambio de datos. Cada bloque o instancia de *JSON* describe un objeto con objetos anidados, matrices, cadenas, números, y cualquier otro tipo de datos.

Aunque el ahorro en el tamaño del documento no llegue a ser significativo, de hecho en algunos casos, el tamaño de un documento *JSON* es incluso superior al mismo en *XML* en casos de tamaños muy pequeños y en formato comprimido, su ventaja está en el análisis. Mientras que para un objeto XML es necesario construir su árbol *DOM* y posteriormente con llamadas de *JavaScript* acceder a los elementos para construir el objeto, en el formato de *JSON* al ser un subconjunto de *JavaScript*, directamente en una sola llamada es analizado por el compilador realizando una llamada a *eval*, tras lo cual, automáticamente se obtiene ya un objeto *JavaScript* al que es más fácil acceder, que el hecho de tener que navegar por el árbol de *DOM*.

Para poder hacer uso de la biblioteca de funciones de json se ha descargado esta las mismas de la página oficial de *JSON* [8] y se han ubicado en el mismo directorio auxiliar que parte de la raíz directorio público Web */jscripts/json/json.js*

5.3 JTable.

Se trata de una clase abstracta que forma la base para la construcción de objetos de la clase tabla en *Joomla*. Para cada tabla que se desee implementar es necesario crear una nueva clase extendiendo *JTable*. Se trata de la implementación del patrón de diseño de registro activo y permite la construcción de una tabla física en *Joomla*, con el fin de leer, actualizar y eliminar registros de tabla de base de datos [3] [10].

Al extender la clase se han de tener en cuenta algunas convenciones:

Los objetos de tipo *table* (tablas) que se creen, han de residir en un directorio determinado partiendo de la raíz de administración del componente que la utiliza, este es el directorio *tables.*

El nombre de la clase ha de ser igual al nombre de la tabla a crear prefijada con el prefijo *Table.*

Ha de tener un campo identificador que constituya la clave primaria y este ha de ser numérico y autoincremental. En el momento de la inserción de datos el campo estará a *null*, y no podrá estar afectado en las operaciones de actualización. Solamente existirá dicha clave, por lo demás, el resto de los campos de la tabla pueden ser de cualquier tipo.

Es de gran utilidad cuando se trata de mantener registros en memoria o en una estructura física del servidor que requiera del uso de tablas de trabajo, sin necesidad de hacer uso de ningún gestor de base de datos.

5.4 Servicios Web.

Un servicio web es una aplicación web con capacidad para interoperar con otro servicio o usuario en la red. Esta aplicación permite el intercambio de datos entre sí con objeto de ofrecer unos servicios. El proveedor ofrece sus servicios como un procedimiento remoto y el usuario solicita el servicio llamando a este procedimiento a través de la Web. Se utiliza para proporcionar un mecanismo de comunicación entre aplicaciones que interactúan con el fin de presentar información dinámica. Para que esto se pueda realizar es necesario que la tecnología que se utilice sea estándar [11].

Para que dicha intercomunicación se pueda llevar a cabo, es necesario el uso de diferentes tecnologías que lo posibilitan.

En primer lugar se encuentra el ***Protocolo Simple de Acceso a Objetos***, *SOAP*. Basado en *XML*, permite la interacción entre dispositivos y tiene capacidad para transmitir información compleja. Sirve para especificar el formato de los mensajes; compuestos por un sobre *envelope* que contiene, una cabecera *header* y un cuerpo *body*.

Por otro lado se encuentra el lenguaje de Descripción de Servicios Web *WSDL*, que posibilita el acuerdo entre el cliente y servicio web en lo referente al transporte del mensaje y su contenido. Especifica la sintaxis y los mecanismos de intercambio de mensajes.

Existen otras tecnologías complementarias a *SOAP* que permiten optimizar el rendimiento de los servicios web, agilizando el envío de mensajes y los recursos que se transmiten en esos mensajes.

NuSOAP [10] es una biblioteca de clases escritas en *PHP* que permiten crear y consumir servicios WEB a los desarrolladores que utilizan *SOAP* con *WSDL*, con el fin de crear dichos servicios Web y utilizarlos.

Para implementar el cliente del Servicio Web en la aplicación utilizaremos la última versión de *nuSOAP* disponible en *SourceForge* [13], que permite el uso de *SOAP 1.1* y corrige defectos de las versiones anteriores. La versión *SOAP* para php la incluiremos dentro del componente de *tareasfco*, en el directorio *libs/nusoap.php.* Con el fin de probar el correcto funcionamiento del cliente Web, implementaremos un servicio Web con *.NET* que montaremos sobre *IIS*.

5.4 TCPDF.

Se trata de una biblioteca [14] de funciones de Código Abierto, para ser utilizada con *PHP* que permite la generación dinámica de documentos *PDF* y donde su potencialidad se encuentra en la facilidad de uso en la generación de documentos y en la posibilidad de interpretar *HTML* convirtiéndolo a contenido formateado en el documento. La librería se puede ser descargada de su página oficial [15]. En ella se encuentra además de una descripción detallada del cometido de cada una de las funciones que contiene, una amplia gama de ejemplos en los que se nos muestra, cómo aprovechar la potencialidad de la herramienta.

Las bibliotecas que permiten la generación de los documentos en formato PDF se encuentran partiendo en el directorio raíz de la aplicación en *libraries/tcpdf_5_9_063/config/lang/spa.php* y *libraries/tcpdf_5_9_063/tcpdf.php*.

6. DESARROLLO DEL COMPONENTE.

En este capítulo se describe tanto la estructura física, es decir, su distribución en el sistema de archivos de la aplicación, como la lógica con el contenido y las funcionalidades implementadas en cada uno de los diferentes ficheros que comprende el componente.

Por último se detalla la estructura y contenido del fichero de instalación del componente.

6.1 Estructura del Componente.

El componente principal de la aplicación se denomina ***tareasfco*** y comprende el siguiente conjunto de directorios y ficheros:

```
com_tareasfco
    index.html
    tareasfco.php
    controller.php
    controllerajax.php
    models
        index.html
        alumno.php
        mntoalumnos.php
        notas.php
        tareasfco.php
    tables
        tareasfco.php
    views
        alumno
            index.html
            view.html.php
            view.raw.php
            tmpl
                index.html
                default.php
        notas
            index.html
            view.html.php
            tmpl
                index.html
                default.php
        mntoalumnos
            index.html
            view.html.php
            tmpl
                index.html
                default.php
                listausuarios.php
        tareasfco
            index.html
            view.html.php
```

```
                tmpl
                    index.html
                    default.php
                    tareas.php
                    chgpassword.php
            tareasfcoagregar
                index.html
                view.html.php
                tmpl
                    index.html
                    default.php
            tareasfcoeditar
                index.html
                view.html.php
                tmpl
                    index.html
                    default.php
        media
            Ficheros de imágenes
        admin
            index.html
            install.tareasfco.php
            uninstall.tareasfco.php
```

Destacar la existencia en todos los directorios del fichero index.html con el siguiente contenido:

```
<html><body bgcolor="#FFFFFF"></body></html>
```

Su finalidad es la de evitar que cuando se referencie el directorio, éste pueda mostrar su contenido. Con ello, se trata de mejorar la seguridad del sistema.

6.2 Descripción de Funciones.

El siguiente módulo es el de tareasfco.php. Contiene el punto de entrada al componente y en él se selecciona cual es el controlador del componente que se encarga de llevar a cabo las tareas que se especifican. Si no se especifica ninguno, por defecto, será controller.php, en caso de que el parámetro *controller* que se obtiene de la *query string* sea *AJAX*, el controlador que se cargará será *controllerajax.php*.

```
$clase=JRequest::getVar('controller');
switch($clase){
        case 'ajax':
                require_once (JPATH_COMPONENT.DS.'controllerAjax.php');
                $controller = new tareasfcoControllerAjax();
                break;
        default:
                require_once (JPATH_COMPONENT.DS.'controller.php');
                $controller = new tareasfcoController();
                break;
```

```
}
$controller->execute( JRequest::getCmd('task'));
$controller->redirect();
```

Además de la entrada del componente, en este directorio se encuentran también los controladores, que se encargan de realizar las tareas que le son requeridas desde la vista. En el caso de *controller.php* se ejecutarán las siguientes tareas:

6.2.1 Controladores.

taskLeerFicheroUsuarios: Se encarga de solicitar al modelo *mntoalumnos* una relación con los alumnos en función del contenido del fichero especificado y entregar esa lista de alumnos al modelo para poder gestionarla

taskupdPassword: Solicita del modelo *mntoalumnos* la actualización del password, pasándole como parámetros el usuario joomla y la contraseña especificada para el alumno. Tras ello, se redirecciona la salida a la vista correspondiente en función de si el password modificado es el del administrador o el del usuario.

taskaddUsuario: Se comprueban los parámetros con los datos del usuario y a continuación se carga el modelo *mntoalumnos* y se ejecuta el método del mismo que permite realizar el alta de usuario.

taskdelUsuario: Se comprueba el dni que identifica el usuario a suprimir y a continuación se obtiene el modelo *mntalumnos* que implementa la función de suprimir el alumno. Una vez suprimido se redirecciona la salida a la tarea a la vista de mantenimiento de alumnos de nuestro componente.

taskBorrarRegistroTareas: Se carga el modelo de *tareasfco* donde se implementa las funcionalidades de acceso a datos y se requiere de la función de borrarRegistrodetareas

taskAlmacenarRegistroTareas: Esta tarea se encarga de grabar los datos introducidos en el formulario de tareas; tanto una nueva tarea y como las modificaciones realizadas a una existente.

El controlador de *AJAX* implementa las funciones requeridas por las vistas y que entrega de manera asíncrona al agente del cliente que le ha solicitado la petición. La tarea no retorna ningún estado devuelve un objeto con contenido, que será utilizado por el agente del cliente, para mostrar el resultado de la acción realizada.

taskgetUsuario: Haciendo uso del modelo *mntoalumnos*, se llama a la función del mismo *getusuario* que devuelve los datos del usuario si este se encuentra registrado o en blanco si no existiera este.

taskRegistroUsuario: Esta tarea obtiene de la llamada, los datos de un alumno que son pasados a través de la query string, junto con el tipo de operación a

realizar (1->alta, 0->baja) y carga el modelo *mntoalumnos* que implementa la operación a realizar con una llamada a *altadeusuario* o *bajadeusuario* en función de la acción, retornando al agente del cliente, un objeto que incorpora una referencia a la función *Javascript* que implementa la petición de cambio de estado del objeto.

taskComentarios: Tarea que hace uso de la función *listadecomentarios* implementada en el modelo notas y que devuelve al cliente una relación de comentarios introducidos por los alumnos en la encuesta que se les plantea .

taskServicioPoliformat: Tarea que es solicitada desde la vista y que se encarga de implementar un cliente web que realiza una petición de servicio a un servidor web externo, con el que se establece una conexión y haciendo uso del servicio montado en el servidor de la Universidad 'Poliformat', se le envía a la tarea creada a través del servicio, el alumno y la nota obtenida en el cuestionario.

En el caso de que la nota se haya podido entregar al servicio de manera correcta, se marcará en el registro del alumno dentro de la tabla correspondiente a la tarea realizada, como publicada, retornando al cliente el mensaje de *true*, como indicador de que la operación se ha realizado bien. En caso de que el envío no se haya podido realizar, se devuelve *false* al usuario como indicador de que el cliente no ha podido enviar la nota junto con el error que lo motivó.

Tras la descripción del funcionamiento básico de los controladores, pasamos a describir la funcionalidad implementada en los modelos.

6.2.2 Modelos.

El modelo alumno.php solamente incorpora una función, esta es la de *getRegistrosTareasAlumno*, que permite obtener una relación con las tareas disponibles para el alumno, y que será solicitada directamente desde la vista *alumno* y será mostrada por defecto en el agente del cliente una vez que el alumno se registre.

El modelo mntoalumnosincorpora una serie de funciones interaccionan tanto con los controladores como con la vista e implementan las peticiones de estos sobre los procesos de creación y mantenimiento de alumnos/usuarios del sistema.

getListaAlumnos: Devuelve una relación de los alumnos registrados en el sistema para su gestión en el lado del cliente, con la posibilidad de que cada uno pueda ser suprimido del sistema y/o restaurar su contraseña, volviéndolo a registrar.

getFicheroDatos: Haciendo uso del fichero pasado al servidor desde el cliente en el proceso de importación de alumnos, se analiza el contenido del fichero, descartando líneas y caracteres no útiles, se genera una tabla con todos los alumnos contenidos en el fichero, separando los apellidos, el nombre y el *DNI* que será el identificador del usuario para *Joomla*, y se

entrega una lista al cliente, para que pueda realizar el alta de los mismos en el sistema.

getUsuario: Dado un usuario/alumno identificado por su *DNI*, se comprueba su existencia en el sistema y se retorna el mismo si existe y nulo en caso de que no exista.

altaUsuario: Cogiendo los valores de *DNI*, apellidos y nombre, pasados como parámetros a la función, se registra el alumno en el sistema, dándolo de alta en la tabla de usuarios del sistema con el *DNI* como identificador de usuario y *password*. Se inserta además en las tablas necesarias del sistema que permitan al usuario hacer uso del entorno y por último se comprueba su existencia en la tabla *alumnosfco*, donde se almacena en el caso que no lo estuviera. Dado que el alta del usuario supone la inserción en diferentes tablas de la base de datos consideraremos el conjunto de operaciones como una transacción única para evitar que en el caso de que se produzca un error en la grabación de una de las tablas estas queden en modo inconsistente.

updPassword: Permite establecer a un usuario pasado como parámetro el valor de la contraseña introducido en el formulario habilitado para el cambio de contraseña. Este cambio puede ser efectuado tanto por el administrador como por el cliente.

bajaUsuario: A diferencia del alta del usuario, para efectuar la baja, solamente es necesario el identificador del usuario, que se corresponde con el dni. La supresión del alumno/usuario supone la baja en diferentes tablas de la BD como: #__user, #__core_acl_aro, #__core_acl_groups_aro_map y #__alumnofco, al igual que ocurre en el alta, éste conjunto de operaciones sobre la base de datos, se realizan integradas en una transacción.

El modelo notas, implementa la funcionalidad necesaria para obtener de la base de datos los resultados de los test realizados por los alumnos.

genTotales: Función que dado el tipo de test y el identificador de tarea asignado al mismo, se obtiene una relación de las notas medias obtenidas para cada pregunta del cuestionario. El profesor podrá analizar la dificultad de las preguntas en función de los resultados medios obtenidos por los alumnos. Si el resumen se corresponde con el de representación de datos, junto a la relación mostrada, se acompaña un enlace que conduce a la muestra de los resultados de la encuesta de satisfacción que realiza el alumno.

listaEncuesta: Función que accede a los datos de una encuesta realizada por los alumnos y muestra el resultado estadístico de la misma. Incorporando un objeto adicional que permite mostrar los comentarios realizados por los alumnos durante la evaluación.

listaComentarios: Con esta función se obtiene la lista de los comentarios efectuados por los alumnos en la encuesta correspondiente a una tarea. Esta relación se solicita directamente desde el agente de usuario como una petición

ajax que el modelo generará y entregará directamente al cliente.

listaAlumnos: Del mismo modo que se solicita por parte de la vista las notas medias globales obtenidas en un cuestionario, en esta función se obtiene una relación de los alumnos junto con las notas obtenidas, y enlaces para cada alumno, en los que en uno permite generar un informe en formato PDF, y en el otro permite el envío de la nota a poliformat, haciendo uso de la implementación del servicio web.

updNotaPublicada: Función que dada la tarea, el alumno y el test realizado se modifica la base de datos, estableciendo el atributo publicada a uno, indicando con ello, que la nota ha sido enviada de forma correcta a poliformat.

El archivo de modelo tareasfco implementa las funcionalidades básicas a realizar sobre la tabla de tareasfco. Estas son:

getRegistrosTareas: Accede a la tabla de *tareasfco* y devuelve una relación con las tareas existentes y la posibilidad al cliente de interactuar con ellas, modificarlas y/o suprimirlas. Junto a ellas se devuelven dos enlaces; uno permite ver las notas medias obtenidas en una determinada tarea y otro que permite ver los resultados obtenidos por los alumnos en una tarea en concreto, posibilitando interactuar sobre los mismos para generar un informe de notas y/o realizar la publicación de las mismas.

borrarRegistroTareas: Obtenido de la query string el identificador de la tarea, se obtiene una instancia del objeto de tabla *tareasfco* implementado en *tables* y se borra el registro en cuestión.

almacenarRegistroTareas: Función que se utiliza tanto para el alta como para la modificación de una tarea. Lee la lista de parámetros del formulario de la tarea enviados con put, leyendo la query string. Se realiza una comprobación de los valores pasados y que estos cumplen los requerimientos previstos, como campos nulos, condiciones de fechas, formato de fechas, etc. Si la comprobación no retorna mensaje de error entonces se produce el enlace con el objeto tabla de tareasfco definida en tables y se realiza la operación, si el id es nulo se trata de un alta, en caso contrario se trata de una modificación. La función retorna un mensaje con el error producido si lo hubiere y nulo si la operación de grabación se ha producido correctamente.

6.2.3 Tablas.

En la carpeta tables, esta definido el fichero tareasfco que identifica el objeto de la base de datos que será utilizado en las tareas de persistencia. Se crea un objeto que extiende la clase JTable y que se utilizará en el modelo para combinar los datos del formulario con el registro que corresponda en la tabla y realizar la operación necesaria.

6.2.4 Vistas.

En cuanto a las vistas del componente, hemos de destacar que existen vistas diferentes en función de las tareas a realizar.

La vista de alumno contiene en primer lugar una comprobación del usuario que realiza una petición de dicha vista, si se trata de un usuario no registrado o del administrador, se abandona la vista y se redirige el flujo de la aplicación a la vista de *tareasfco* para que muestre el *template* por defecto.

En el caso de que el usuario sea un usuario registrado, se comprueba el parámetro adicional correspondiente al DNI definido para los usuarios alumnos que se han de conectar para realizar los cuestionarios.

Una vez pasados los filtros de comprobación de usuarios, se llamará haciendo uso del modelo de la lista de tareas disponibles para el alumno. La tabla obtenida se pasará por referencia al *layout* por defecto *default* que se encargará de mostrar la lista de tareas en el contenedor del componente.

Con la vista de *mntoalumnos* mostramos en el *layout* por defecto, las diferentes opciones disponibles en el mantenimiento de alumnos/usuarios donde se muestran los formularios para la carga de los alumnos del curso, el alta o baja individualizada de un alumno/usuario y la opción de listar los usuarios del sistema.

Tanto el listado de usuarios del sistema, como el contenido del fichero de alumnos para su importación se muestran haciendo uso del *layout listausuarios.php* disponible en la carpeta *tmpl*.

Para que el cliente pueda efectuar las operaciones de alta y baja de alumnos/usuarios en el sistema, se han implementado las funciones en javascript que haciendo uso de ajax, interactúan con el servidor para ejecutar la funcionalidad requerida. Estas funciones son:

recorrerUsuarios: Recorre la lista de usuarios y para todos aquellos que tengan marcado el indicador de cambio de estado se ejecutará la opción de updUsuario. En lugar de hacerlo individualmente sobre cada fila que se seleccione, cabe la posibilidad de marcar un amplio grupo de usuarios y realizar la operación de forma conjunta pulsando sobre el botón de *Procesar Marcados* que se encuentra disponible en el *layout* de *listausuarios.*

chgestado: Recorre el listado de usuarios y cambia el indicador de estado del usuario para realizar sobre él la operación de actualización.

updUsuario: Realizando una petición AJAX al servidor se realizará la operación de alta o baja del usuario sobre el que se pulsa, cambiándolo a la situación opuesta. Es decir, si el alumno se encuentra registrado, se suprime del registro, si no estuviera registrado se registra.

getusuario: Petición que se realiza cuando cambie de DNI, se produce una petición AJAX del cliente al servidor solicitando los datos de identificación del usuario. El cual retornará el nombre y apellidos o en blanco si el

alumno no está registrado.

clearForm: Limpia el formulario de mantenimiento de usuarios del sistema restaurando los valores de las variables.

A la vista de notas se accede desde la vista de tareas, donde aparece el listado de tareas disponibles y pulsando sobre el enlace notas disponible en cada tarea. Es una vista que se utiliza para mostrar los diferentes listados estadísticos que se generan partiendo de los test realizados.

Para identificar el tipo de informe que se va a mostrar definimos un nuevo atributo *vista* que servirá para seleccionar que tipo de operación del modelo se va a solicitar, la referencia del listado obtenido se pasa al *layout* que en este caso es uno y que sirve para mostrar tanto las notas medias del cuestionario, el desglose por alumno con la posibilidad de generar el informe PDF y publicar su nota, como el estadístico generado en base a las encuestas realizadas, pudiendo ver los comentarios efectuados por los alumnos.

Para que el cliente pueda realizar estas últimas acciones, estas se implementan como peticiones del cliente que haciendo uso de AJAX, interactúan con el servidor para realizar la acción solicitada. Todas las peticiones javascript que hacen uso de ajax, se les pasa como parámetro en la tarea el atributo *no_html=1* ó *format=raw* con el fin de que solamente retorne el objeto que genera la tarea llamada, excluyendo cualquier otro texto adicional, como cabeceras y formato estándar del *template* que implementa el componente.

Las funciones *Javascript* del cliente que implementan las peticiones son:

servicioNotas: Recorre la lista de notas de un cuestionario y para aquellas que tienen activa la marca se realizará la llamada a *enviarNotasAlumno*

enviarNotasAlumno: Para cada alumno seleccionado se realizará una petición asíncrona que actuará como cliente de un servicio web enviando a Poliformat la tarea, la nota y el alumno. Si se ha llevado a cabo de manera correcta queda marcada como publicada en la tarea del test correspondiente en la BD. En caso contrario se mostrará un error.

recorrerNotas: Función con un comportamiento similar al de servicio Notas, pero en este caso actúa sobre la posibilidad de generar informe PDF para el alumno, dentro de la lista de alumnos, para aquellos que tengan su indicador de la izquierda activo, se generará el informe.

genPDF: Haciendo uso de AJAX, realiza la petición al componente adecuado que implementa la tarea de generar el informe dado el DNI del alumno y el identificador de la tarea. Es decir, en función de que la tarea sea de representación de datos o flipflop la función de generar informe está definida en cada uno de ellos, dado que la estructura de la encuesta y el informe que se genera partiendo de cada encuesta es diferente.

chgEstado: Permite cambiar el estado del indicador que acompaña a cada una de las operaciones marcándolo o desmarcándolos, para poder

realizar la operación sobre el conjunto activo, y no tener que realizarla sobre cada uno de ellos.

verComentarios: Se podrá activar cuando se hayan mostrado las estadísticas de la encuesta que realiza el usuario junto al test de representación de datos. Al final de los resultados aparece la opción de mostrar los comentarios de los alumnos sobre la asignatura. Con esta función podemos mostrar u ocultar dichos comentarios. La generación del contenido es dinámica, se realiza haciendo uso de una petición ajax, por lo que en cada petición puede mostrar información diferente si entre petición y petición se ha introducido nueva información.

La vista *tareasfco* es la que se carga por defecto cuando se llama al componente desde la entrada al sistema. Para trabajar con la aplicación ha de ser un usuario registrado, tanto el alumno como el administrador, por lo que para cualquier otro usuario y/o invitado no podrá realizar ninguna tarea al sistema. Lo que se visualizará en la vista será el logo con un enlace de la UPV.

Si el usuario se ha validado como administrador se cargará el *layout* por defecto con una lista de las funciones que puede realizar el administrador.

En el caso de que el *layout* sea *tareas*, se enlazará con el modelo de tareasfco y se realizará una llamada a la función que carga una relación con el conjunto de tareas disponibles y la posibilidad de realizar un mantenimiento sobre las mismas, agregando, modificando o consultando resultados sobre cada una de las tareas.

Cuando el *layout* especificado sea el de *chgpassword* se mostrará un formulario, para introducir la nueva contraseña. Este formulario será llamado tanto por el Administrador como por los alumnos por ello en el *layout* la aceptación del formulario supone la carga de la vista *alumno* cuando el usuario es un alumno y la vista *tareasfco* cuando se trata del administrador.

```
<?php
$user =& JFactory::getUser();
if ( $user->name == "Administrator" ) {
        echo "<input type='hidden' name='usuario' value='admin' />";
        echo "<input type='hidden' name='view' value='tareasfco' />";
}
else {
        $dni = $user->getparam('dni');
        echo "<input type='hidden' name='usuario' value='$dni' />";
        echo "<input type='hidden' name='view' value='alumno' />";
}
?>
```

Tanto la vista de *tareasagregar* como la de *tareaseditar* únicamente comprueba que el usuario es el administrador, en cuyo caso mostrará el formulario por defecto que tiene definido cada vista, de introducción de una nueva tarea en uno y de modificación de la tarea en el otro, tras haber accedido al registro a modificar y haber mostrado sus valores en el *layout*.

6.2.5 Imágenes.

En la carpeta media se incluyen todas las imágenes que requiere el componente para su correcto funcionamiento. También se incluyen la imagen que se utilizará como logo del sistema.

6.2.6 Instalación.

En la carpeta *admin*, se incluyen los ficheros que se utilizan en el proceso de instalación y desinstalación del componente con el fin de crear y suprimir según el caso, las tablas necesarias para el funcionamiento del componente.

El fichero *install.tareasfco.php* contiene la creación de la tabla de tareas.

```
DROP TABLE IF EXISTS `#__tareasfco`;

CREATE TABLE `#__tareasfco` (
	`id` int(11) NOT NULL AUTO_INCREMENT,
	`test` varchar(100) NOT NULL,
	`poliformat` varchar(100) DEFAULT NULL,
	`descripcion` varchar(100) NOT NULL,
	`fechaInicio` date DEFAULT NULL,
	`fechaCierre` date DEFAULT NULL,
	`fechaPublicacion` date DEFAULT NULL,
	PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8 COLLATE=utf8_spanish_ci;


DROP TABLE IF EXISTS `#__alumnofco`;

CREATE TABLE `#__alumnofco` (
	`id` int(11) NOT NULL AUTO_INCREMENT,
	`dni` varchar(16) COLLATE utf8_spanish_ci NOT NULL,
	`nombres` varchar(64) COLLATE utf8_spanish_ci NOT NULL,
	`apellidos` varchar(64) COLLATE utf8_spanish_ci NOT NULL,
	PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8 COLLATE=utf8_spanish_ci;
```

En el proceso de desinstalación del componente se ejecutará el contenido del fichero *uninstall.tareasfco.sql*.

```
DROP TABLE IF EXISTS `#__tareasfco`;
DROP TABLE IF EXISTS `#__alumnofco`;
```

6.3 Cambios en los componentes de Cuestionarios.

Los componentes de evaluación ya se encontraban implementados, tanto el de biestables como el de representación numérica, pero la evolución constante de los navegadores (versiones de IE, Firefox, Chrome, etc) y su necesidad de integración con el nuevo

componente ha hecho necesario que se tenga que revisar su implementación y corrección de algunos errores existentes, por la incorporación de nuevos parámetros y funcionalidades.

A los componentes se les ha agregado un vector con un comentario para cada opción posible de las diferentes preguntas, que se mostrará en el informe de notas del alumno, explicando la acción realizada en aquellas cuestiones cuya respuesta sea contestada incorrectamente.

Se han resuelto una serie de errores que se producían en el funcionamiento de los mismos; unos debidos a errores en el código de los *layouts* de y otros en el código de la librerías adicionales que utiliza *joomla* y que permiten el uso de ventanas emergentes.

En el código de los *layouts* de las preguntas se ha cambiado la función que obtenía los parámetros de la *query string* y generaba un par; variable y valor para uso de javascript , equivalente a los parámetros pasados de la *query string*. El uso de la anterior función suponía un bloqueo en la carga de las páginas cuando era visualizado en las nuevas versiones de los navegadores *IExplorer* y *Mozilla Firefox*.

```
function getQueryParams(qs) {
	qs = qs.split("+").join(" ");
	var params = {};
	var tokens;
	if (location.search) {
		qs = qs.substring(1).split('&');
		for (var i = 0; i < qs.length; i++) {
			tokens = qs[i].split('=');
		if (!tokens[0]) continue;
	params[decodeURIComponent(tokens[0])] = decodeURIComponent(tokens[1]) || true;
	}
	return params;
}
```

Otro error existente era la reinterpretación de la cadena $merge en la función de Javascript Squeeze, que era interpretada como una variable php y sustituida por su contenido, cuando en realidad, se trata de un texto que contenido en la definición de una función JQuery para su uso en el navegador del cliente. Esto se ha resuelto enlazando las dos partes de la composición de la función junto al texto que se reinterpretaba y se ha añadido a la declaración de funciones *Javascript* del documento.

```
	$document->addScriptDeclaration ( $squeeze0 . '$merge' . $squeeze1 );
```

En los cuestionarios de *flip-flop* en la función de *Javascript gradeQuestion* que almacena los valores de los diferentes vectores del cuestionario haciendo uso de *Json* para la generación de los valores que se almacenan en la BD. Se ha detectado que la función *toJSONString* daba problemas de funcionamiento, para resolver este problema se ha sustituido esta función por otra que tiene el mismo comportamiento pero funciona de manera correcta:

```
function gradeQuestion(name,valuations){
	var grandTotal=0;
	for (i=0;i<valuations;i++){
		var aux=name+i;
		grandTotal=grandTotal + grade(aux,i);
	}
	grandTotal = grandTotal.toFixed(2);
	alert( 'nota Eval : ' + grandTotal );

	jQuery('#Q1Note').text(grandTotal);
	jQuery('#notaEval').attr('value',grandTotal);
	jQuery('#respuesta').attr('value', JSON.stringify(solutionQ1) );
	jQuery('#solucion').attr('value', JSON.stringify(solutionQ1Student) );
	jQuery('#comentario').attr('value', JSON.stringify(commentsQ1) );
	document.formEvaluate.submit();

	var divQ = parent.document.getElementById("divQ3");
	divQ.innerHTML = '<p style="color:red;">Q3. Latch NAND /S /R =' + grandTotal + '</p>';

	cerrarVentana();
}
```

Además de estos errores en el código de las vistas, se han detectado otros errores en las librerías de funciones *Javascript* que acompañan al *Joomla* utilizadas para las ventanas modal que se utilizan en la visualización de los test.

El primero se localiza en la librería modal.js ubicada en el directorio *media/system/js* y el error estriba en la definición de los parámetros ancho y alto de la ventana que se carga con la función *SqueezeBox*, definida en esta librería. El código que a continuación se detalla es el resultante de corregir la función.

```
    reposition: function(sizes) {
        sizes = sizes || window.getSize();
        this.overlay.setStyles({
            //'left': sizes.scroll.x, 'top': sizes.scroll.y,
            // FROM:
            //   width: sizes.size.x,
            //   height: sizes.size.y
            //TO:
            width: sizes.x,
            height: sizes.y
        });
```

Por otro lado, existe otro error detectado en la visualización de las ventanas modal cuando se realiza su carga. Este error es exclusivo de los navegadores de Microsoft, no así en el resto de navegadores chrome, safari, firefox, etc.

En el momento de la carga de las ventanas modal aparece el mensaje; error en la línea 31 en el carácter 1.

Este error se debe al intento de asignación de un atributo de estilo que no ha sido definido y dada la incompatibilidad de IExplorer en la ejecución del código JavaScript de la librería, se bloquea la ejecución normal. Aplicando una pequeña corrección al código de la librería se resuelve y la ejecución posterior de JavaScript sobre la ventana modal no sufre ninguna alteración.

El código al que se hace referencia y que se encuentra al inicio de la página 31 a corregir es:

```
this.sytle[property]=value;
```

y su versión corregida es:

```
try {this.style[property] = value;}catch(e){}
```

Una vez sustituido el código se nos resolverá los problemas de ejecución en las diferentes versiones del Agente de Usuario de Microsoft Internet Explorer.

Se ha incorporado a los módulos la identificación de la tarea por la que ha accedido el alumno, esto permite que puedan existir grupos de alumnos diferentes que hayan realizado un mismo test, diferenciándolos por la tarea y periodo asignado de evaluación.

Cuando el alumno ha contestado la encuesta o ha realizado el test, además de mostrarle la nota obtenida, se impedirá que el test pueda ser accedido de nuevo. Al mismo tiempo se controlan los periodos de realización de las tareas y de la posibilidad de visualización de los resultados. Por otra parte se permite que el propio alumno pueda generar, visualizar y descargar su propio informe de notas, una vez abierto el periodo de publicación de las mismas.

El directorio *PDFAlumnos*, se crea de manera dinámica a medida que se van generando los informes de notas en formato *PDF* de los alumnos. Se crea en la raíz del sitio web y se distribuye en forma de un nodo por alumno/usuario dentro del cual se encontraran los informes de notas de los diferentes test realizado por el alumno en cuestión. El nombre del documento se conforma con el identificador de tipo de tarea, el DNI y el identificador de la tarea.

Dentro de las plantillas de los test, se ha introducido una matriz de comentarios para cada test diferente de tal forma que cuando se graba el registro de notas de dicho alumno, también se graban los comentarios a cada una de las cuestiones que se le plantean. Esto permitirá su posterior visualización en el informe, en el caso de que el alumno haya errado la pregunta.

6.4 Módulo de Instalación del Componente.

Una vez desarrollado el componente principal y los cambios efectuados a cada uno de los componentes de cuestionario, se procede a generar el instalable del componente para poder ser agregado como extensión desde el *back-end* de *Joomla*. Se crean para ello el fichero XML que detalla el procedimiento de instalación del componente y se crea un fichero

comprimido que contendrá el conjunto de directorios y ficheros que comprenden el componente.

El fichero de especificación de la instalación del componente principal es tareasfco.xml y su contenido:

```
<?xml version="1.0" encoding="utf-8"?>

<!DOCTYPE install SYSTEM "http://dev.joomla.org/xml/1.5/component-install.dtd">

<!-- http://docs.joomla.org/Components:xml_installfile -->
<install type="component" version="1.5.0" method="upgrade">

	<name>TareasFCO</name>
	<!-- Los siguientes elementos son opcionales y no tienen restricción de formato -->
	<creationDate>Julio 2011</creationDate>
	<author>Vicente Reolid</author>
	<authorEmail>vireoser@ei.upv.es</authorEmail>
	<authorUrl>http://www.upv.es</authorUrl>
	<copyright>Derechos reservados para el autor</copyright>
	<license>GPL</license>
	<!--  La cadena con la versión se graba en la tabla de componentes -->
	<version>version 1.0</version>
	<!-- La descripción es opcional y por defecto es el contenido de la etiqueta name -->
	<description>Tareas para Evaluación de Alumnos</description>

	<!-- Sección de instalación de tablas -->
	<install type="component">
		<sql>
			<file driver="mysql" charset="utf8">install.tareasfco.sql</file>
		</sql>
	</install>

	 <!-- Sección de desinstalación de tablas de la BD -->
	<uninstall>
		<sql>
			<file driver="mysql" charset="utf8">uninstall.tareasfco.sql</file>
		</sql>
	</uninstall>

	<files>
		<filename component="com_tareasfco">tareasfco.php</filename>
		<filename>controller.php</filename>
		<filename>controllerajax.php</filename>
		<filename>index.html</filename>
		<filename>tareasfco.xml</filename>
		<folder>lib</folder>
		<folder>media</folder>
		<folder>models</folder>
		<folder>tables</folder>
		<folder>views</folder>
	</files>

	<administration>
		<menu>Tareas FCO</menu>
```

```
            <files folder="admin">
                <filename>index.html</filename>
                <filename>install.tareasfco.sql</filename>
                <filename>uninstall.tareasfco.sql</filename>
            </files>
        </administration>
</install>
```

Con el proceso de instalación se registrará el componente en el sistema, creando una entrada en el menú de componentes y que posibilitará ser referenciado. En el mismo proceso se crearán las tablas definidas en *install.tareasfco.sql*, necesarias para el funcionamiento del componente.

De igual manera que se ha desarrollado el fichero de instalación del componente que implementa las tareas, se ha generado un paquete de instalación para cada uno de los cuestionarios de evaluación existentes. Uno para el componente de evaluación de Representación de Datos y otro para el de Biestables. Aunque no necesitan gestión en desde el *back-end* de Joomla, su instalación desde Joomla permite la creación automática de las tablas MySQL que necesitan para almacenar el resultado de las respuestas de los alumnos.

Tras la instalación del componente de tareas, se instalaran los componentes correspondientes a cada uno de los cuestionarios planteados. En el proceso de instalación se crearán las tablas necesarias para almacenar los resultados.

6.4 Creación de un Servicio Web.

Dado que hemos de implementar un cliente de Web que transmita las notas obtenidas por los alumnos en los diferentes test al servidor de la plataforma *Poliformat* utilizada por la Universidad, y con la finalidad de comprobar la correcta implementación del cliente, se ha desarrollado un Servicio Web que simula el servicio que suministra poliformat. Esto nos permitirá realizar pruebas de consumo del cliente y de utilización del servicio, sin conocer ni atacar al servicio de Poliformat. Una vez comprobado su funcionamiento solamente será necesario, cambiar la dirección web que nos da el servicio, así como el nombre del servicio en cuestión.

En primer lugar se instala el conector .NET de MySQL para poder acceder desde los servicios creados a la base de datos.

En Microsoft Visual Studio se crea un nuevo proyecto Web, utilizando para ello, la ***plantilla de Aplicación de Servicio Web de ASP*** [14].

Dentro del proyecto web se crean los servicios como funciones de C# tal y como se especifica en la plantilla que crea Visual Estudio, y se agrega la conexión con la Base de Datos de MySQL utilizando los parámetros de conexión necesarios, para crear la cadena de conexión a la BD.

Para que los servicios Web que creemos en .NET puedan funcionar, ha de estar instalado IIS, de no estarlo, habrá que agregarlo desde Programas, en el apartado de Activar o desactivar características de Windows. Una vez instalado se accede al Administrador de IIS

desde las Herramientas Administrativas del Panel de Control, para poder iniciar el servicio y establecer su configuración.

Hay dos posibilidades para montar los servicios sobre el directorio público de IIS:

La primera manera es agregando la aplicación en el administrador de IIS. Para ello, se le da un alias sobre el punto de montaje del directorio público raíz de IIS y se indica el directorio donde se encuentra el servicio de nuestro proyecto desarrollado previamente.

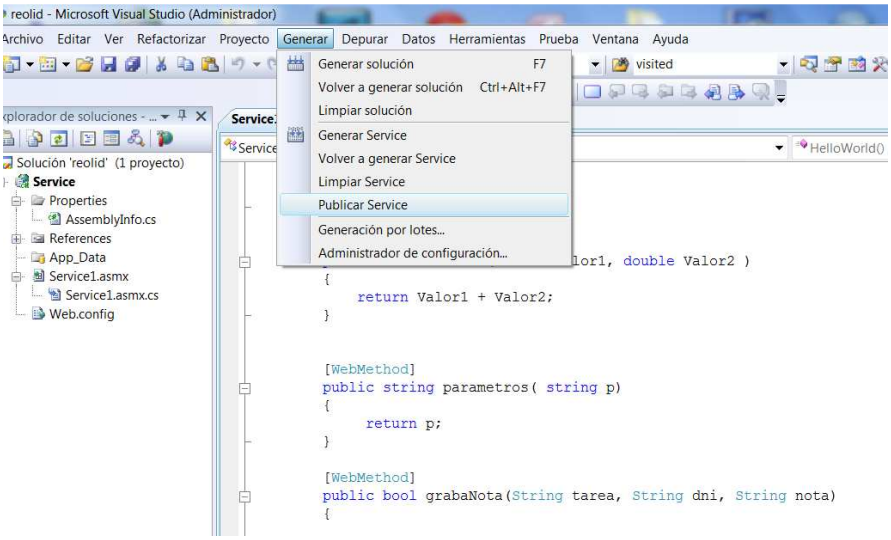


Figura 2. Generar Servicios con .NET.

La otra posibilidad consiste en crear el directorio y desde nuestro proyecto de Visual Studio haciendo uso de las opciones del menú Generar, generamos primero el servicio y posteriormente lo publicamos indicando el directorio de publicación que hemos creado previamente en IIS para alojar nuestro servicio. Bastará dentro del Administrador de IIS situarnos sobre la carpeta creada previamente y que alojará el servicio, y con el botón derecho del ratón nos aparecerán las diferentes acciones que podemos realizar sobre la misma. Entre las opciones disponibles seleccionamos la de convertir en aplicación.

Con el fin de que pueda coexistir el IIS con el Servidor Web de Apache montado para la aplicación, es preciso cambiar el puerto desde el que se atienden las peticiones HTTP en uno de los dos servidores, en nuestro caso, cambiaremos el puerto por el que atenderá IIS.

Para establecer el puerto en IIS, es necesario abrir el Administrador de IIS, seleccionar la opción de Examinar que se encuentra en el apartado de Acciones y modificar el puerto de escucha del protocolo HTTP al puerto 81.

Una vez creado y publicado el servicio, podemos comprobar su funcionamiento, para ello indicamos sobre el navegador la dirección http://localhost:81/servicio/Service1.asmx donde nos debe aparecer una relación de los servicios implementados en el Servicio y que se encuentran disponibles.

Figura 3. Servicios Montados en IIS.

Si en la petición web le añadimos el parámetro *?WSDL*, nos mostrará los diferentes parámetros de entrada y salida que compone cada servicio y que necesitaremos para enlazar desde el cliente que ha de consumirlos.

```
<?xml version="1.0" encoding="utf-8" ?>
<wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/" xmlns:tns="h
  xmlns:s="http://www.w3.org/2001/XMLSchema" xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/" xmlns:http="http://sc
  targetNamespace="http://dsic.upv.es/" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
- <wsdl:types>
  - <s:schema elementFormDefault="qualified" targetNamespace="http://dsic.upv.es/">
    - <s:element name="HelloWorld">
        <s:complexType />
      </s:element>
    - <s:element name="HelloWorldResponse">
      - <s:complexType>
        - <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="HelloWorldResult" type="s:string" />
          </s:sequence>
        </s:complexType>
      </s:element>
    - <s:element name="getNota">
      - <s:complexType>
        - <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="tarea" type="s:string" />
            <s:element minOccurs="0" maxOccurs="1" name="dni" type="s:string" />
          </s:sequence>
        </s:complexType>
      </s:element>
    - <s:element name="getNotaResponse">
      - <s:complexType>
        - <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="getNotaResult" type="s:string" />
          </s:sequence>
        </s:complexType>
      </s:element>
    - <s:element name="Sumar">
```

Figura 4. Detalles de los Servicios.

Entre los diferentes servicios de ejemplo implementados se encuentran:

HelloWorld: No tiene parámetros. La invocación del servicio devuelve un objeto XML con un elemento de tipo string que contiene el resultado de retorno de la función.

Sumar: Dados dos parametros Valor1 y Valor2 los interpreta como numéricos de tipo doble y retorna un objeto XML con el resultado de la suma.

actualizarNota: Dados tres parámetros; tarea, dni y nota se actualiza la nota de un alumno en una determinada tarea. Retornando un objeto con un elemento booleano que indicará si la transacción se ha realizado correctamente o nó.

getNota: Utilizando dos parametros de entrada que identifican la tarea y el DNI del alumno, retorna la nota obtenida por dicho alumno o un elemento vacío en caso de que no exista nota para el alumno y la tarea especificada.

grabaNota: Utiliza tres parámetros para almacenar en una tabla la nota obtenida por el alumno en una determinada tarea. Es el servicio que utilizaremos para comprobar la correcta implementación de nuestro cliente WEB, para ello se utilizan tres parámetros: tarea, que se corresponde con el identificador de tarea de Poliformat como destino de la nota, DNI que identifica al alumno que ha realizado la tarea y nota que contiene la nota obtenida por el alumno que será enviada a Poliformat.

Una vez introducidos los parámetros se realiza la grabación en la tabla de Notas para comprobar que esta se realiza correctamente.

```
<?xml version="1.0" encoding="utf-8"?>

<grabaNotaResponse xmlns="http://dsic.upv.es/">
        <grabaNotaResult>boolean</grabaNotaResult>
</grabaNotaResponse>
```

El servicio retorna un objeto que contiene el resultado de la ejecución del servicio indicando en el elemento *grabaNotaResult* si ha sido correcta *true* o incorrecta *false*.

A continuación se describe el código que implementa el servicio.

```
[WebMethod]
public bool grabaNota(String tarea, String dni, double nota){
        if (tarea.Equals("") || dni.Equals("") || nota < 0 || nota > 10)
        return false;

        string path = "server=localhost;User Id=root;database=webfco";

        try {
                MySqlConnection con = new MySqlConnection(path);
                con.Open();
                MySqlCommand cm = new MySqlCommand("INSERT INTO NOTAS
                ( tarea, dni, nota ) VALUES ( '" + tarea + "', '" + dni + "','" + nota + "');");

                cm.Connection = con;
                cm.ExecuteNonQuery();
                con.Close();
        } catch (Exception) { return false; }

        return true;
}
```

7. INSTALACION BÁSICA DE JOOMLA.

El desarrollo de los servicios relacionados se ha realizado tomando como base *Joomla* como motor de portales dinámicos en su versión 1.5, montado sobre un sistema *XAMPP* (*Servidor Apache/MySQL/PhP/Python*) y se han implementado como un complemento.

Partiendo de la base ya realizada de dos componentes operativos desde el lado del cliente *front-end,* que implementan los test de los temas 2, 3 y 4 de la asignatura de Fundamentos de Computadores, se han integrado en el entorno del nuevo complemento y se han completado incorporándoles nuevas funcionalidades como: identificador de tarea, comentarios a respuestas incorrectas, fechas de gestión en uno de los controles (inicio, cierre y publicación) y la generación de informes de resultados de la tarea realizada.

Junto a la integración de nuevas funcionalidades y depuración de los componentes que implementan los cuestionarios, se ha desarrollado un nuevo componente. Dicho componente será el eje central sobre el que están integrados tanto los componentes que evalúan los controles ya existentes, como los nuevos que se puedan desarrollar en el futuro y completen el sistema de evaluación. Desde este componente se gestiona tanto la generación y especificación de tareas seleccionadas por el profesor y que estarán disponibles para los alumnos, como la entrada centralizada del alumno, para realizar todos los controles de evaluación que tenga disponible en la lista de tareas abiertas y que le hayan sido propuestos durante el periodo de realización de la asignatura.

Así mismo, se le ha añadido la posibilidad de gestionar los alumnos de la asignatura, para que estos puedan realizar los controles. Posibilitando dar de alta, suprimir e importar en bloque los mismos.

Para facilitar la administración al profesor de la asignatura, encargado de generar las tareas y acceder a los resultados de las mismas, se le facilita la posibilidad de establecer su contraseña, sin necesidad de acceder al *backend* de Joomla.

Así mismo, dado que tanto para administrar las tareas como para realizar el test se ha de estar registrado, se ha implementado una nueva funcionalidad, la de modificación de contraseña que podrán realizar ambos, alumno y profesor, lo que permite mejorar la confidencialidad en la conexión al sistema.

7.1 Instalación del Entorno.

Aunque en la actualidad Joomla tiene disponible la versión 1.6 con cambios importantes sobre la versión 1.5. Se ha considerado conveniente realizar la implementación en la versión 1.5, al tratarse de una versión estable y muy extendida de este CMS ya que a pesar de encontrarese operativa la versión 1.6 desde principios de año, esta supone grandes cambios con respecto a la anterior y no podemos considerar esta versión como definitiva puesto que está prevista su sustitución por la versión 1.7 en fechas próximaspara mediados de Julio del mismo año, lo que le da a la versión 1.6 una gran inestabilidad puesto que dejará de tener servicio en breve.

Una vez instalado el CMS la instalación del sistema de control de evaluación de Fundamentos de Computadores puede realizarse de dos maneras diferentes:

1. Sustituir el arbol de directorios del CMS instalado por el que se acompaña en el proyecto e incluir la base de datos con el contenido del backup de mysql que se acompaña junto al proyecto. Lo que dejará una instalación límpia.

2. Instalar el CMS de Joomla en su versión 1.5 y posteriormente haciendo uso del backend de la instalación, instalar los componentes que componen la Aplicación y configurar el entorno.

El primer método es quizás el método más sencillo pues no es necesaria una instalación propia de joomla, solamente es necesario tener instalado el XAMPP y acceso a MySQL. Una vez copiado el arbol de directorios e importada la base de datos en MySQL probando la conexión con http://host//webfco estará disponible el Sistema de Control de FCO.

En el caso de utilizar el segundo método de instalación, es necesario tener algunos conocimientos de Joomla, para poder instalar los productos (módulos y complementos) así como crear las entradas de menú correspondientes y asignar acciones a los módulos. A continuación se describe los pasos a seguir en el caso de que se opte por el segundo método.

7.2 Instalación y Puesta en Marcha de la Aplicación.

En este apartado se describen las diferentes tareas que se han de realizar para instalar el componente de tareas desde el *back-end* de *Joomla* hasta que este se encuentre disponible para poder ser utilizado desde el *front-end*.

7.2.1 Instalación del Componente.

Para realizar la instalación del componente com_tareasfco es necesario conectar como administrador en el *back-end* deJoomla y a traves del Gestor de Extensiones se realiza la instalación del componente creado. [15].

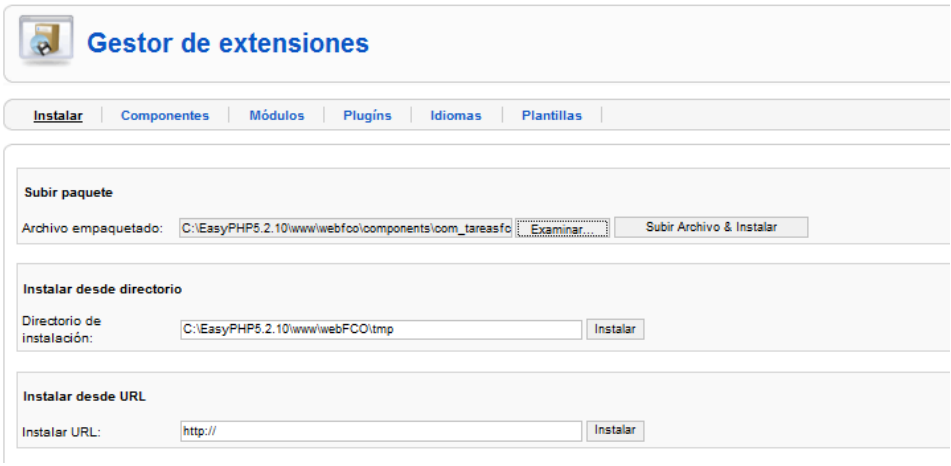


Figura 5. Instalación de Extensiones Joomla.

Una vez instalado el componente nos aparecerá una nueva entrada en el menu de de componentes disponibles en el sistema, con el nombre dado en el fichero de instalación.

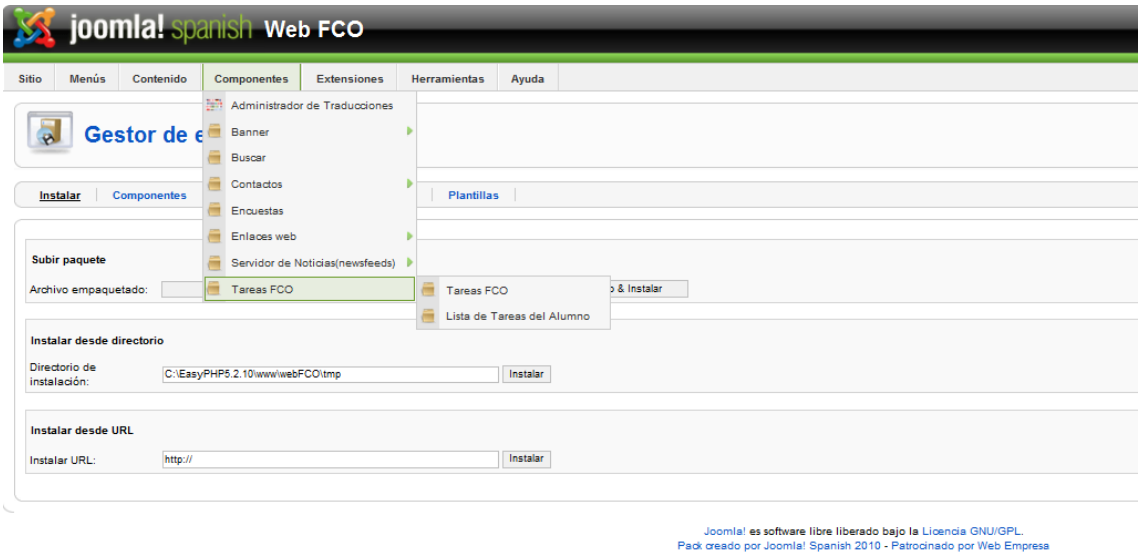


Figura 6. Menú del Componente ya instalado.

Junto con la instalación del componente se crean las tablas necesarias para su funcionamiento.

Las tablas de notas correspondientes a cada cuestionario, se instalan cuando se instale el componente que implementa el test en cuestión.

Estos componentes son com_flipflop que implementa los test referentes a los temas de biestables y com_encuesta que contiene tanto el test relativo a la representación de datos como la encuesta de aceptación y valoración de la asignatura.

7.2.2 Creación de los menús de inicio y desconexión.

Una vez disponible el componente, creamos dos entradas de menú. [16].

La primera contendrá un enlace a las tareas disponibles para el alumno y que será, la página de entrada una vez el usuario se ha registrado.

La segunda será el enlace a la entrada genérica del componente, que será la que estará visible siempre que no se trate de un usuario registrado.

Figura 7. Gestor de Entradas del menú.

En primer lugar, editamos la entrada inicio para ello dentro de la edición de la entrada inicio del menú, elegiremos la opción cambiar tipo y ahí seleccionaremos la entrada de web FCO donde seleccionamos la entrada de alumno. Esto nos crea el enlace index.php?option=com_tareasfco&view=alumno. Esta es la entrada por defecto al componente cuando el usuario se valida. En caso de que el usuario sea el administrador del

sistema, el componente lo identifica y redirecciona su entrada al menú de tareas del mismo, con el fin de que pueda realizar las tareas que le corresponden como gestor.

Figura 8. Parámetros de la entrada de Inicio.

Del mismo modo que hemos creado la entrada de menú para inicio creamos la contrapartida a esta. La de desconexión, el proceso de creación es identico al anterior pero en este caso seleccionamos la página por defecto de entrada al componente. Index.php?option=com_tareasfco.

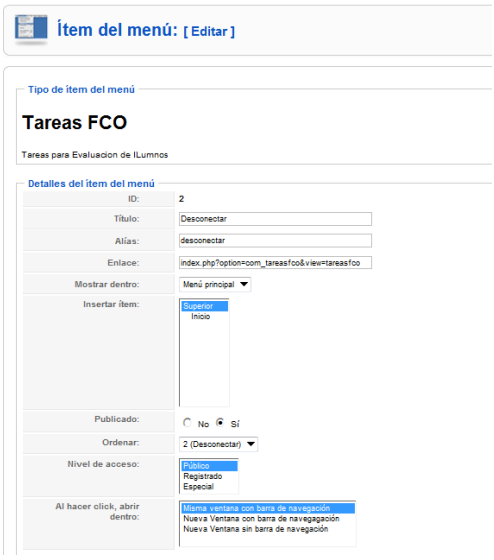


Figura 9. Parámetros de la entrada de desconectar.

7.2.3 Enlazar los menús con el plugin de Conexión.

Tras la creación de las entradas de menú con los enlaces básicos para conectar y desconectar, nos situamos en la gestión de modulos y activamos el módulo referente al login. Una vez activado el módulo, entramos en la edición del mismo y establecemos la URL de redireccionamiento del inicio de sesión a la entrada de menú Inicio y la URL de redireccionamiento de finalización de sesión a la entrada del menu Desconectar).

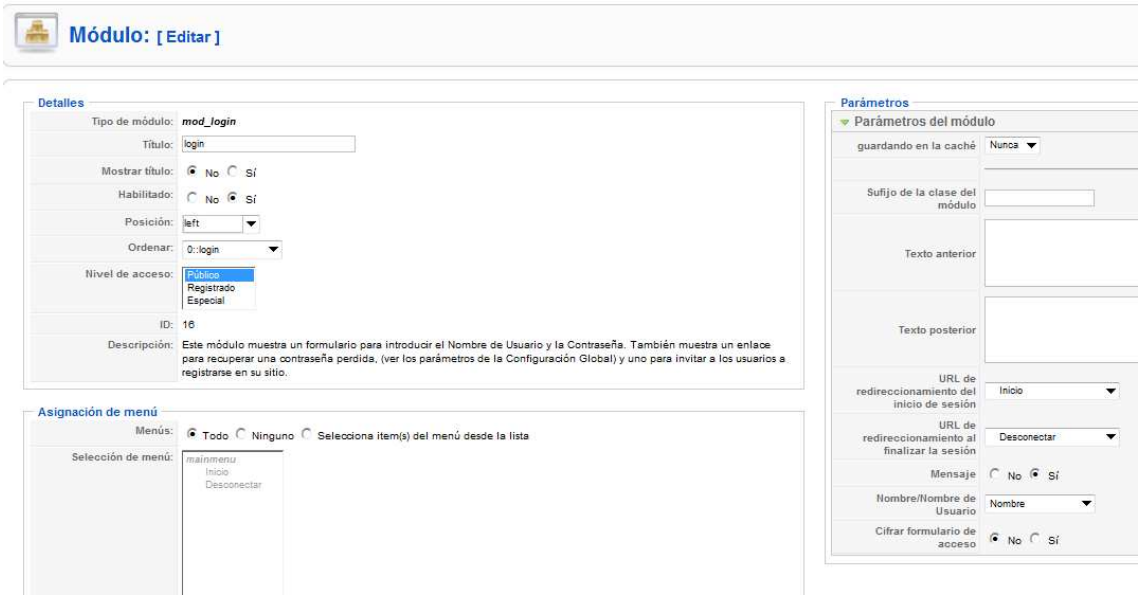


Figura 10. Configuración del módulo de conexión.

7.3 Incorporación de parámetros adicionales al usuario.

Con el fin de tener relacionados los usuarios de Joomla con los alumnos que van a utilizar la aplicación, hacemos uso de un plugin adicional para Joomla que extiende las propiedades de los usuarios, permitiendo la adición de parámetros adicionales en las propiedades de los usuarios. Se trata del plugin usermeta.

La instalación del plugin la realizaremos conectándonos al backend como administrador y a través de la Entrada de Menú de Extensiones seleccionando la opción de Instalar/Desinstalar.

Localizamos el fichero del plugin usermeta.zip con examinar y pulsamos la opción de subir archivo e instalar. Una vez instalado nos aparece el mensaje “Pluging instalado con éxito”.

Tras la instalación nos vamos al menú de extensiones y seleccionamos la entrada de Gestor de Plugins. Dado que pueden haber una gran cantidad de plugins podemos pulsar en la parte inferior de la página en mostar todos y nos aparecerá la relación completa de plugins existentes donde localizaremos el plugin instalado “System – UserMeta”, para que esté operativo tendremos que publicarlo.

No obstante, una vez instalado el plugin, no hemos definido todavía los parámetros adicionales que incorporaremos al usuario. Para realizar esto se han de realizar algún trabajo adicional a la instalación del plugin.

Dentro del fichero de instalación del plugin, existe un Readme.txt que explica el funcionamiento del mismo y sus posibilidades. Cabe destacar, que se pueden definir parámetros adicionales específicos de usuario y de grupo haciendo uso de los fichero user.xml y/o nombre_del_grupo.xml (p.e. si el grupo es Registred el fichero será registred.xml). Por otro lado, cabe la posibilidad de que los parámetros sean añadidos directamente en el atributo params de la tabla 'jos_user' o en una tabla adicional que complemente a esta y cuyo script de creación es el siguiente:

```
CREATE TABLE `jos_usermeta` (
        `user_FK` INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
        `example` VARCHAR(125) NOT NULL,
    PRIMARY KEY (`user_FK`)
);
```

Para definir los parámetros se ha de editar el fichero que los contiene y adecuarlo a nuestros intereses, el fichero de especificación de parámetros se encuentra instalado y su ubicación dentro del sitio es:

```
plugins/system/usermeta/user.xml
```

Para ello, primero realizamos una copia del archivo user.xml, por cuestion de seguridad y lo editamos. Dentro nos encontramos bloques de etiquetas params, en la que dentro de cada bloque se definen un conjunto de parámetros, identificado cada uno por la etiqueta param. Existen dos grupos de etiquetas params, una sin atributo y otra con atributo. El grupo de params que no posee el atributo group será la que se almacena junto con la tabla de usuarios del sistema. El conjunto de parámetros del grupo params con atributo de group, se almacenan en la tabla asociada de usermeta descrita con anterioridad, por lo que si no está creada, aunque los parámetros pudieran aparecer en el modulo de datos de usuario, no se grabaran.

En nuestro caso, utilizaremos una sola etiqueta que será el dni y que asociará el usuario de joomla con el alumno de la tabla del componente jos_alumnosfco.

```
<params>
    <param name="dni" type="text" client="administrator" default=""
        label="Dni : " description="Introduce aquí el dni del Alumno" />
</params>
```

Cada parámetro que se añada aquí aparecerá como un conjunto de valores de la forma label=valor_asignado. Esto permite que posteriormente en php se puedan establecer un array asociativo de claves y valores donde a clave es el label y el valor el asignado.

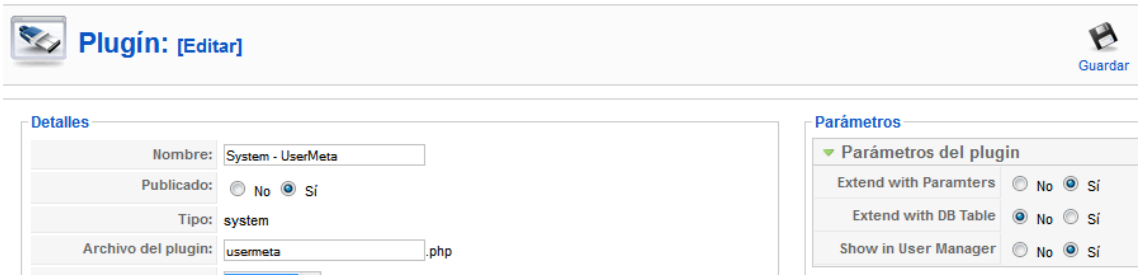


Figura 11. Configuración del plugin UserMeta.

Dentro de la edición del plugin, en los parámetros del mismo especificaremos si se utilizarán los parámetros extendidos, es decir, los que definamos para *usermeta*, si se visualizan los de la tabla extendida (aquellos que tienen el atributo *group* en *params*) y si deseamos que se muestren o no en el módulo de edición de usuarios.

En la que cada atributo de la relación es equivalente a cada parámetro que se almacena en la tabla jos_user. Si bien cabe la posibilidad de especificar parámetros tanto en la tabla jos_user como en la tabla jos_usermeta.

7.4 Alta de Usuarios desde el Back-End.

Los usuarios pueden ser dados de alta de forma manual desde el *back-end* como administrador a través del Gestor de Usuarios, para ello, iremos agregando nuevos usuarios, introduciendo su nombre genérico, su identificador de usuario, password, email, especificando el grupo y colocando en parámetros el dni que se corresponderá con el del alumno. Esto le permitirá conectarse al sitio y poder realizar los test.

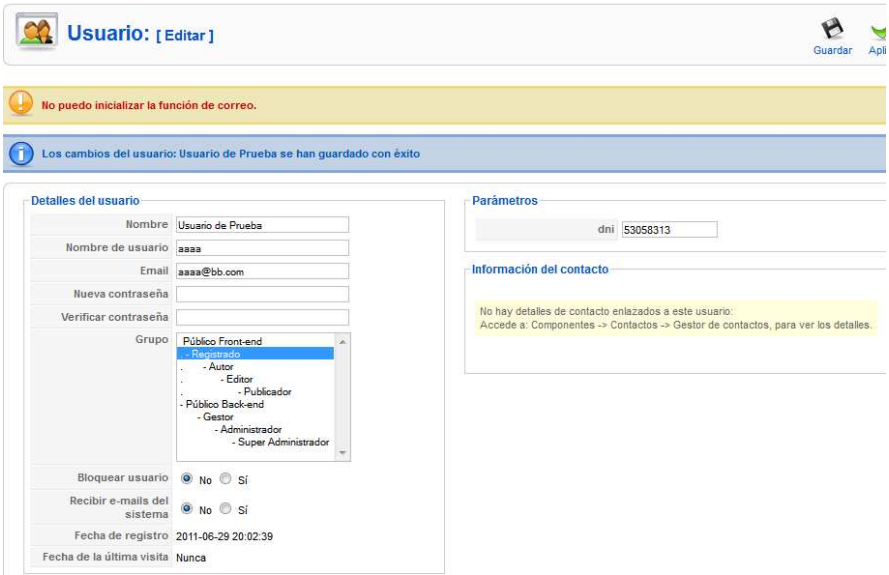


Figura 12. Alta de usuarios desde el BackEnd.

Estos valores podrán ser leídos posteriormente haciendo uso del método getUser de la clase JFactory.

```
$user = & JFactory::getUser();
```

```
if (!$user->guest) {
                echo 'You are logged in as:<br />';
                echo 'User name: ' . $user->username . '<br />';
                echo 'Real name: ' . $user->name . '<br />';
                echo 'User ID  : ' . $user->id . '<br />';
                echo 'User Params  : ' . $user->params . '<br />';
}
```

Donde con la llamada:

```
                $dni = $user-> getparam('dni');
```

obtendremos el valor del parámetro dni especificado en el modulo de usuarios.

Figura 13. Ubicación de la conexión a la aplicación.

7.5 Selección y adaptación de la plantilla.

Una vez instalado Joomla accedemos al backend del sistema registrándonos como admin y la contraseña dada en el proceso de instalación y a continuación seleccionamos en el menú de extensiones la opción de Gestor de Plantilla.

En el mismo aparecen una lista con las plantillas que se instalan por defecto junto con Joomla, utilizaremos la plantilla rhuk_milkyway y la adecuaremos a nuestro proyecto. Para ello en primer lugar la activaremos como plantilla predeterminada pulsando sobre predeterminada.

A continuacion, entramos a editar la plantilla y en su configuración editamos los parametros de visualización indicando como variantes de color y fondo el blanco y el ancho medio de pantalla.

Tras la definición de los parametros le daremos a aplicar para que los cambios queden registrados y tras ello, pulsamos el botón de editar HTML para configurar el pié de pagina de la plantilla, con el fin de que aparezca el autor del proyecto. Esto hará que se edite el contenido

del archivo de nuestro sitio creado *templates\rhuk_milkyway\index.php* que contiene la plantilla que se visualizará en el front end del cliente.

En el fichero se ha de sustituir el apartado correspondiente a la etiqueta power_by del pié de pagina por:

```
<p id="power_by">
    <p align='right'>Autor : <b>Vicente Reolid Serrano - </b><i>(07/2011)<i></b></p>
</p>
```

Tras el cambio se pulsará el botón de guardar. Esto grabará los cambios y regresará al menú de edición de la plantilla. Pulsaremos en Editar CSS con el fin de modificar el logo de la plantilla para adaptarlo a nuestro diseño. Los parámetros de configuración del logo de la plantilla se encuentran en el fichero de la plantilla template.css ubicado en *templates\rhuk_milkyway\css,* el estilo definido para el logo quedará como sigue:

```
div#logo {
    position: absolute;
    left: 0;
    top: 0;
    float: left;
    width: 798px;
    height: 75px;
    background: url(../../../components/com_tareasfco/media/logo_proyecto.png) 0 0 no-
repeat;
    margin-left: 30px;
    margin-top: 30px;
}
```

con esto conseguiremos sustituir el logo por defecto de la plantilla por el de nuestro proyecto. En la propiedad *background* se encuentra definida la ubicación del fichero que contiene el logo. Esto debe hacerse una vez instalado el componente de tareas que es el que incorpora la imagen del logo en su carpeta de imágenes denominada media.

Por último, con el fin de evitar que aparezcan en la conexión del usuario los apartados de olvido de contraseña, olvido de usuario y registrarse, se ha de editar el layout del modulo de login ubicado en la carpeta modules\mod_login\tmpl\default.php y comentar los apartados del formulario que no deseamos que aparezcan, utilizando:

```
<!-- y -->.
```

8. FUNCIONAMIENTO DE LA APLICACIÓN.

Con el fin de facilitar el uso de la aplicación desarrollada, a continuación se describe de manera detallada el funcionamiento de la misma, tanto desde la perspectiva del profesor como gestor o administrador de la misma, como desde el alumno como usuario de la aplicación, detallando además aquellas funcionalidades comunes a ambos.

8.1 Punto de Entrada al Aplicativo.

Una vez configurado el sistema y situados sobre el agente de usuario deseado; firefox, explorer, chrome, etc. Nos conectaremos a nuestro sistema usando la dirección donde se encuentre instalada la aplicación, p.ej: http://host/webfco el cual nos cargará por defecto la página:

Figura 14. Punto de entrada a la aplicación.

Donde en función del tipo de usuario (profesor/administrador y alumno) que se conecte aparecerá posteriormente un conjunto u otro diferente de funcionalidades disponibles.

8.2 Conexión como Profesor/Administrador.

El responsable de creación de las tareas se ha de validar como administrador. Una vez registrado como administrador, nos aparecerán dos opciones que se corresponden con dos enlaces. Cada uno de ellos representa una funcionalidad distinta para el administrador.

Figura 15. Menú de conexión del Administrador/Profesor.

A través del Mantenimiento de Tareas y notas, el administrador podrá definir nuevas tareas, mantener las existentes y o suprimirlas. Así mismo podrá tener acceso al tanto estadístico de las notas, como a la relación de notas de cada alumno en la tarea especificada, permitiendo la generación del documento PDF con las notas del alumno, poner el mismo a disposición del alumno y publicar las notas de la tarea en *poliformat* haciendo uso de un servicio web habilitado a tal efecto.

Con el mantenimiento de usuarios, se le facilita al administrador la tarea de la creación de cuentas para los alumnos, de tal forma que quede integrada en Joomla. Esto facilita la confidencialidad en la ejecución de los test por parte del alumno.

8.2.1 Gestión de Tareas.

Las tareas son la base de trabajo para los alumnos, para que ellos puedan realizar cualquier cuestionario, este ha de haber sido previamente creado como tarea por parte del profesor/administrador a través del mantenimiento de tareas y habérsele asignado una fecha de inicio de tarea.

8.2.1.1 Mantenimiento de tareas.

Una vez conectados como administrador y seleccionada la opción del mantenimiento de tareas nos aparecerá una relación con las tareas que tenemos dadas de alta en el sistema, junto con la opción de Agregar, editar una tarea existente y/o suprimir una tarea.

Figura 16. Lista y mantenimiento de Tareas definidas.

Para dar de alta una tarea pulsaremos sobre agregar tarea y nos aparecerá una nueva página donde se podrán introducir los datos referentes a la nueva tarea estos son:

Figura 17. Formulario de alta de tareas..

Tipo de Test: De un desplegable con los diferentes test existentes, donde cada test se corresponderá con un componente que lo implementa y que servirá de referencia para la extracción de resultados y la edición y publicación de notas. En la actualidad se incorporan dos tests diferentes; uno para la evaluación de conocimientos sobre la representación de datos y otro para la evaluación de los biestables.

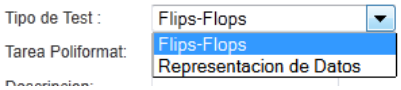


Figura 18. Tipos de Test.

Tarea Poliformat: Puesto que la aplicación no está integrada dentro de Poliformat, sistema de evaluación y seguimiento del alumnado, para la publicación de las notas obtenidas en cada tarea por los alumnos se hace necesario un enlace con el sistema Poliformat. En este atributo se almacenará el identificador de tarea asignado al crear la misma en Poliformat. Su introducción no es obligatoria puesto que no es obligatoria la necesidad de publicar las notas, si lo es en el caso de que se deseara utilizar el servicio de publicación de notas.

Descripción: Contiene el texto descriptivo de la tarea, es el nombre que aparecerá en la lista de tareas disponibles para el alumnado.

Fecha Inicio: Contendrá el valor de inicio de la tarea, la tarea solamente aparecerá como disponible en la lista de tareas para el usuario, cuando la fecha del sistema igual o superior a la fecha de inicio de la tarea. Esto permite que se tenga una tarea definida con anterioridad a la realización de la misma y no sea necesario activarla en el momento en que el usuario la vaya a realizar.

Fecha Cierre: Ultimo día hasta el que se tiene la posibilidad de realizar el test. Una vez sobrepasada la fecha de cierre, la opción de realizar el examen en la tarea correspondiente no estará disponible para el alumno.

Fecha Publicación: Para evitar el uso indebido de las soluciones por parte de los alumnos, no se facilita la publicación de los resultados hasta que no se haya completado la fecha de publicación de la tarea, que en todo caso ha de ser superior o igual a la fecha de cierre de la misma.

Una vez introducidos los datos que identifican a una tarea, pulsando sobre agregar tarea, se dará de alta la misma incorporándose a la relación de tareas y volvemos a la lista de tareas donde ya nos aparecerá como disponible.

En el formulario de introducción de tareas, para facilitar la introducción de fechas, se ha adjuntado un calendario junto a cada una de las entradas de fecha. Para ello, hemos de incluir el siguiente código en nuestras vistas, tanto de agregar tarea, como la de editar tarea existente.

En el fichero de funcionalidades *javascript* de *joomla* disponible en '*includes/js/joomla/javascript.js*' se encuentra definida la función *showCalendar*. Para poder hacer uso de dicha función hemos de incluir la librería en la definición del documento donde la queremos incluir.

```
$document= JFactory::getDocument();
$document-> addScript(JURI::base().'/includes/js/joomla.javascript.js');
```

Otra opción de referenciar el calendario es, utilzando la funcionalidad de joomla:

```
        JHTML::_('behavior.calendar');
```

Una vez incluido en la vista el código de la función que permite utilizar el calendario, en los campos del formulario, incorporaremos la imagen que permitirá seleccionar el formulario, indicándole en la función de *javascript* que se ejecutará en el cliente y encargada de mostrar el calendario. En los parámetros de la función del calendario se indicará el id del objeto de retorno donde se incluirá la fecha que se selecciona y el formato en el que será devuelta la fecha.

```
<input id="fechaInicio" name="fechaInicio" type="text" />

<img class="calendar" onclick="return showCalendar('fechaInicio',
'%d/%m/%Y');" src="templates/system/images/calendar.png"
alt="calendar" />
```

Figura 19. Selección de fecha sobre calendario.

8.2.1.2 Modificación de Tareas.

Para realizar la modificación de una determinada tarea, en la lista de tareas que aparecen hemos de seleccionar una pulsando sobre tarea deseada en la columna de descripción que en realidad es un enlace al mantenimiento de la tarea. Una vez seleccionada nos aparece la tarea con todos sus valores y la posibilidad de realizar la edición de la misma, asignando código de tarea de *Poliformat*, cambiando las diferentes fechas, etc. Tras realizar los cambios pertinentes en el formulario de edición, pulsaremos en Modificar datos de la tarea y estos se registraran en la BD regresando de nuevo al Mantenimiento de Tareas.

Figura 20. Edición de una tarea.

8.2.1.3 Supresión de Tareas

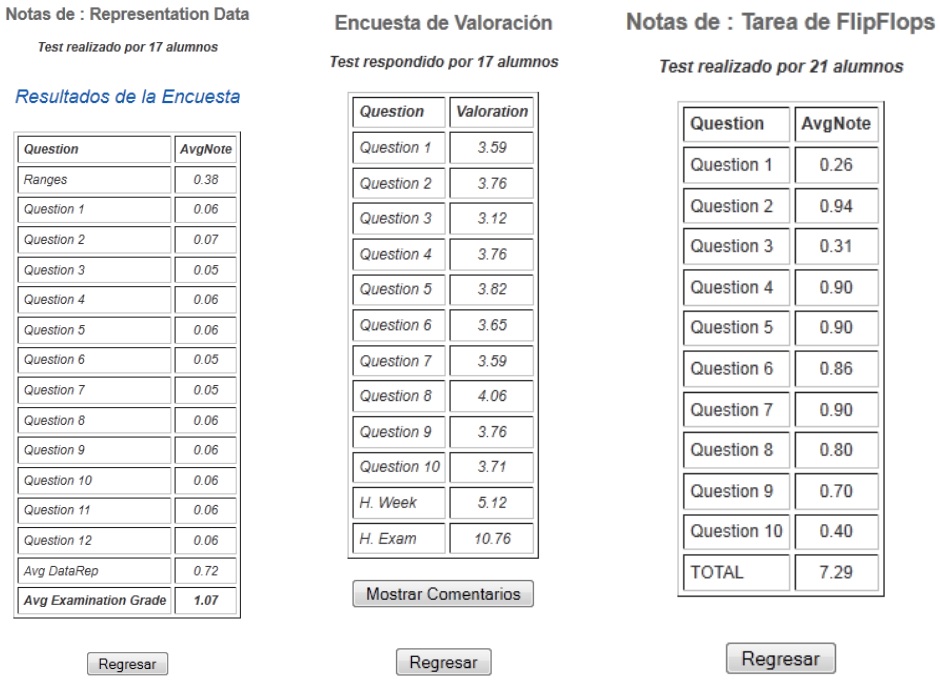
Para suprimir una determinada tarea de la lista de tareas bastará con pulsar en la columna de suprimir, sobre el icono correspondiente a la tarea que deseamos borrar, tras lo cual la tarea desaparecerá de la lista. La supresión de la tarea no implica la supresión de los datos de los test que identifican la tarea. Estos quedan en sus correspondientes tablas como históricos de evaluaciones.

8.2.1.4 Visualización de resultados de una tarea.

En la lista de tareas existentes se encuentra la columna de Resultados. En esta columna se puede acceder a los resultados medios obtenidos por los alumnos en cada una de las diferentes tareas creadas. Esta información permite al profesor realizar una estimación de

la dificultad de las preguntas y/o de su nivel de comprensión por parte del alumnado. Así mismo, permite obtener una valoración media del test.

Cada tarea se corresponde con un tipo de test y dado que este tiene un conjunto de preguntas y una valoración diferente, se genera un listado específico para cada tarea.



Notas de : Representation Data

Test realizado por 17 alumnos

Resultados de la Encuesta

Question	AvgNote
Ranges	0.38
Question 1	0.06
Question 2	0.07
Question 3	0.05
Question 4	0.06
Question 5	0.06
Question 6	0.05
Question 7	0.05
Question 8	0.06
Question 9	0.06
Question 10	0.06
Question 11	0.06
Question 12	0.06
Avg DataRep	0.72
Avg Examination Grade	**1.07**

Regresar

Encuesta de Valoración

Test respondido por 17 alumnos

Question	Valoration
Question 1	3.59
Question 2	3.76
Question 3	3.12
Question 4	3.76
Question 5	3.82
Question 6	3.65
Question 7	3.59
Question 8	4.06
Question 9	3.76
Question 10	3.71
H. Week	5.12
H. Exam	10.76

Mostrar Comentarios

Regresar

Notas de : Tarea de FlipFlops

Test realizado por 21 alumnos

Question	AvgNote
Question 1	0.26
Question 2	0.94
Question 3	0.31
Question 4	0.90
Question 5	0.90
Question 6	0.86
Question 7	0.90
Question 8	0.80
Question 9	0.70
Question 10	0.40
TOTAL	7.29

Regresar

Figura 21. Estadísticos de los diferentes cuestionarios.

En el apartado de visualización del informe de resultados del test de Representación de Datos, los alumnos han tenido la posibilidad de responder a una encuesta de valoración de la docencia en la asignatura de fundamentos de los computadores. Por ello, en los resultados del test aparece un enlace que permite conocer los Resultados de la Encuesta, tal y como aparece en la imagen superior.

Una vez visualizados los resultados estadísticos de la encuesta con la valoración media del alumno y tras la visualización del número de horas semanales medio y el número de horas de estudio en caso de exámenes, aparece un botón con la opción de poder mostrar los comentarios realizados por los alumnos sobre la asignatura.

Una pulsación del botón nos mostrará los comentarios de los alumnos y una nueva pulsación hará que se oculten los comentarios.

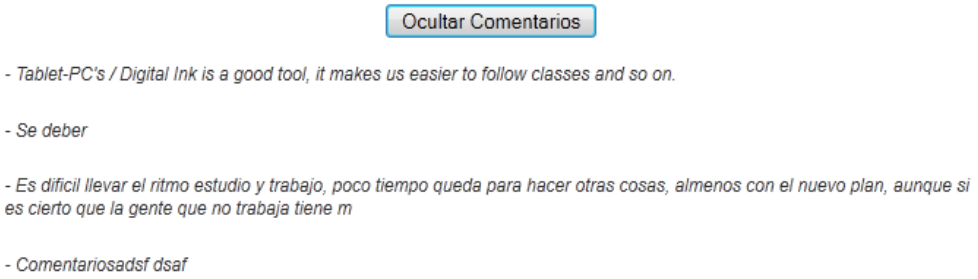
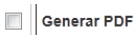
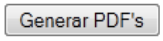


Figura 22. Comentarios de la Encuesta.

8.2.1.5 Evaluación de Resultados.

Esta operación dentro del mantenimiento de tareas nos permite acceder al resultado individualizado obtenido por cada alumno en la tarea seleccionada. Junto a la nota del alumno aparece la posibilidad de generar un documento PDF en el que se extrae un informe con el resultado del test realizado, en el que aparecen tanto las respuestas correctas como las incorrectas con su corrección pertinente y un comentario sobre la misma. Junto a las respuestas aparece la nota por apartados y la nota general obtenida por el alumno en el test. Dicho informe puede ser generado desde aquí por el profesor como por el propio alumno desde la tarea donde realizó el test, una vez alcanzada la fecha de publicación.

En la columna de la izquierda, junto a Generar PDF, seleccionamos todos aquellos alumnos de los cuales queremos generar las notas, pulsando sobre la cabecera 

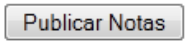
seleccionaremos todos o ninguno alternativamente. Una vez determinados aquellos alumnos de los que deseamos generar el PDF pulsaremos sobre 

. Si se genera desde aquí cuando el alumno accede directamente a la página desde donde realizó el test. El documento estará disponible para su descarga.

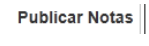
Figura 23. Listado de Resultados de un test.

La generación del PDF se realiza de manera asíncrona en el servidor por lo que el orden de respuesta del servidor es diferente para cada petición, el icono  representa la operación en curso sobre el usuario cuyo documento se está generando, mientras que cuando aparece el icono  se indica que el documento ya ha sido generado y está disponible para su visualización y/o descarga.

De igual forma que se genera el informe del test de cada alumno, el profesor tiene la posibilidad de publicar las notas del alumno en una tarea abierta a tal efecto en *poliformat*. Para poder publicar las notas primero se ha de introducir dentro del mantenimiento de tareas, en la tarea correspondiente, el identificador de la tarea de *poliformat* obtenido tras su creación.

Tras pulsar el botón 

se accederá al servicio asíncrono de publicación de notas, donde la aplicación se conecta al servicio web de *poliformat* habilitado a tal efecto y se le pasan los parámetros correspondientes a la tarea, el alumno y la nota. Durante el proceso de envío de la nota aparecerá el indicador  de que el proceso se está realizando. Si ha finalizado de manera correcta aparecerá el icono  o un mensaje de error en caso contrario.

Al igual que en el servicio de publicación de PDF el servicio de publicación de notas se realiza pulsando el botón de Publicar Notas y se enviarán aquellas que se encuentren seleccionadas en el indicador de la derecha de la publicación de notas. El indicador para cada alumno se activará y desactivará pulsando sobre el mismo en la fila seleccionada y en caso de querer accionarlos todos, pulsaremos sobre el indicador 

disponible en la cabecera.

Una vez la nota ha sido publicada, esta se marca en el test correspondiente al alumno, para que el envío solamente se realice una sola vez. Por ello cada vez que se muestra el listado de notas, solamente se encontrarán activo para su envío aquellas que no se hubieran enviado con anterioridad. No obstante, la selección personaliza sobre la celda de un determinado alumno nos permite el reenvío de una determinada nota.

La funcionalidad del servicio de publicación de notas está implementada en la tarea del controlador *AJAX* denominado *taskServicioPoliformat* y que establecida así:

```
require_once("lib/nusoap.php");
$wdsl="http://localhost:81/servicioweb/Service1.asmx?wsdl";
$client=new soapclient($wdsl, true);
$param=array('tarea'=>$poliformat, 'dni'=>$dni, 'nota'=>$nota);
$result= $client->call('grabaNota', $param);
```

Donde:

En primer lugar cargamos las bibliotecas correspondientes al consumo de servicios web disponibles para php.

$wsdl Identifica al servidor y el servicio montado sobre el que se realizará el envío de las notas.

$client: Es el objeto con un enlace al servicio web del servidor.

$param: Es un array con los parámetros y sus correspondientes valores que se enviarán al servidor.

En la llamada *call* se efectúa el envío de los parámetros haciendo uso del método que atiende el servicio web configurado en el servidor.

En todas las unidades de interacción, aparece un enlace que nos posibilita navegar a la página inicial de administración. Dicho enlace está implementado en el botón 

.

8.2.2 Gestión de Alumnos.

Figura 24. Pantalla de Gestión de alumnos/usuarios.

Para que el alumno pueda realizar el test correspondiente, este ha de validarse en el sistema. Y para que esta validación se pueda realizar el administrador ha de dar de alta con antelación al alumno en el sistema. El alta en el sistema puede realizarse de dos formas diferentes, utilizando un fichero que contenga un listado con los datos de los alumnos, o la introducción individualizada de los alumnos. Con posterioridad podremos acceder a la relación de alumnos dados de alta en el sistema, pudiendo suprimir aquellos que deseemos.

8.2.2.1 Importar Alumnos.

Para la importación de alumnos, se parte de un fichero de texto, generado desde *Poliformat*, donde desde la página de la asignatura se puede generar el mismo, el cual contiene los apellidos y el nombre del alumno y su identificador.

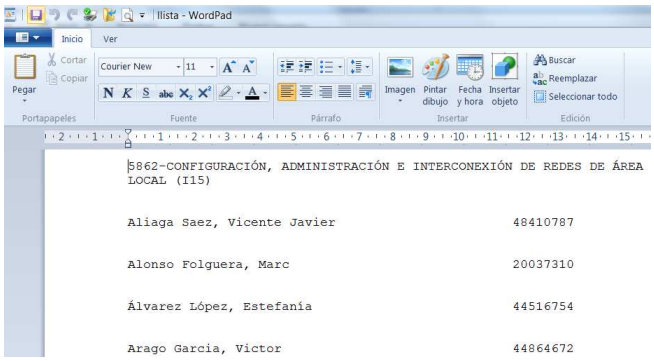


Figura 25. Lista de alumnos obtenida de poliformat.

Con estos datos se puede obtener la lista para su grabación en el sistema. En el sistema se utilizará el DNI, tanto para establecer el identificador de usuario como su contraseña. Los apellidos y el nombre del alumno posibilitan el reconocimiento del usuario una vez registrado en el sistema.

Figura 26. Entrada y selección del fichero de alumnos.

Para leer el archivo se ha de pulsar en examinar, tras lo cual nos aparecerá un cuadro de dialogo del sistema donde podremos localizar el fichero de texto que hemos generado con anterioridad.

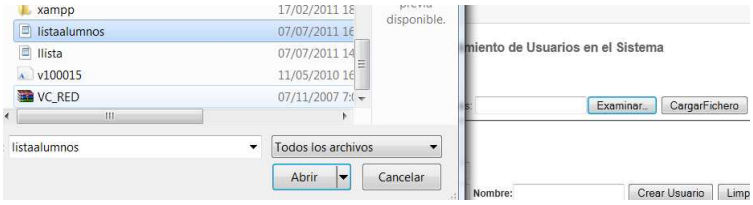


Figura 27. Examinar disco para importar archivo de datos.

Una vez seleccionado el fichero este estará disponible para la carga en el sistema. Para efectuar la carga, hemos de pulsar sobre la opción *CargarFichero*.

Figura 28. Carga del Fichero de alumnos.

Esto lanzará la tarea que se encargara de leer el mismo y generar una lista de usuarios que nos mostrará por pantalla. En la relación generada se genera una fila por alumno del fichero y junto a él un botón en el que podremos dar de alta el alumno y/o de baja en el caso de que ya estuviera dado de alta en el sistema. Por lo que de manera individualizada podremos manejar el alta y/o baja de los usuarios que contiene el fichero. Así mismo, tenemos la oportunidad de realizar este proceso en bloque pulsando sobre procesar marcados. Proceso que registrará y/o suprimirá aquellos usuarios que hayamos especificado con el indicador que

aparece a la derecha del botón de creación/supresión del alumno. Pulsando sobre el marcador de la cabecera, podremos seleccionar o deseleccionar todos los alumnos del listado.

Figura 29. Relación de alumnos leída del fichero de carga.

8.2.2.2 Mantenimiento de Alumnos.

Para la creación y supresión individualizada de alumnos en el sistema disponemos del formulario de mantenimiento de alumnos.

Figura 30. Mantenimiento individualizado de alumnos.

Introduciendo el DNI del alumno y haciendo uso de javascript, se comprueba la existencia del mismo en el sistema. Si está registrado da la posibilidad de suprimirlo directamente pulsando sobre eliminar alumno.

Figura 31. Muestra de un alumno.ya registrado.

En caso de que no exista el alumno y tras la introducción de sus datos, pulsando sobre crear usuario, el alumno quedará automaticamente registrado, utilizando como contraseña el mismo valor que el DNI. En todo momento, siempre esta disponible la opción de limpiar el formulario.

Figura 32. Alta individualizada de un nuevo alumno.

8.2.2.3. Listado de Alumnos.

En todo momento, podemos obtener una lista de alumnos registrados.

Figura 33. Opción de Listar Alumnos Registrados

Con esta opción tenemos la oportunidad de operar sobre ellos, suprimiendolos si se desea. El sistema opera del mismo modo que para el resto de listados que permiten acciones, o bien pulsamos sobre el botón de la fila correspondiente al usuario que deseamos suprimir  o agregar  O bien, marcamos aquellos usuarios, sobre los que deseamos operar y posteriormente pulsaremos sobre el botón de **Procesar Marcados**.

Figura 34. Lista de Alumnos registrados en el Sistema.

Dado que las operaciones de supresión y alta se realizan desde la interfaz de usuario aprovechando la tecnología de Javascript, mientras no abandonemos la relación podremos alternar entre la creación y/o supresión del usuario. Quedando siempre reflejada la última operación que hayamos realizado sobre él.

Cuando el usuario se conecta, le aparecen una lista de tareas. Dichas tareas han sido definidas con anterioridad por el administrador. De las que haya definidas aparecerán solamente aquellas cuya fecha de inicio de la tarea sea menor que la fecha actual. Las tareas se corresponderán con los diferentes tipos de test disponibles en el sistema. En la actualidad, solamente se encuentran disponibles los test de Representación de datos y de Biestables.

8.3 Conexión como Alumno/Usuario.

Cuando el alumno tiene que contestar a un test propuesto por el profesor, este ha de conectarse a la aplicación. Para ello el alumno ha de introducir su DNI como identificador de usuario y como contraseña, pues el sistema está implementado de tal modo que al dar de alta el alumno, se le asigna como contraseña el mismo valor que el del usuario. Con posterioridad, el alumno podrá cambiarse su contraseña.

Una vez conectado le aparece una relación con las tareas disponibles, bien ya finalizadas o bien en curso. Así mismo, le aparece la posibilidad de realizar el cambio de su contraseña de conexión, dado el carácter público de su generación.

Figura 35. Conexión de un alumno al sistema.

Las tareas disponibles se pueden encontrar en diferentes estados:

La tarea se encuentra abierta y disponible para realizarla.

El tiempo asignado a la tarea ha finalizado y la tarea ya no se puede realizar.

Los resultados de la tarea estan disponibles para ser consultados y/o descargados.

Cuando el usuario pulsa sobre uno de los test disponibles, por ejemplo, el relativo a la representación de datos, el alumno tiene la posibilidad de cumplimentar una encuesta que será de consumo estadístico para el profesor de la asignatura y de la posibilidad de realizar el examen correspondiente al test, siempre y cuando la fecha del sistema este comprendida entre la fecha de apertura de la tarea y la fecha de cierre.

Figura 36. Selección de un test por parte del Alumno.

Una vez realizado el examen, se graban los resultados del test y aunque evaluado, el alumno no podrá tener acceso al resultado del mismo, mientras no se haya cumplido la fecha de publicación, hecho que evitará que el informe de notas de un alumno, pueda ser utilizado por otro alumno.

La opción logout , nos permite regresar a la Relación de Tareas disponibles.

En el caso del test de biestables, este se encuentra dividido en diez preguntas que se contestan de manera independiente, en formularios específicos, uno por pregunta. Una vez contestada una cuestión ya no es posible volver al mismo, mostrando el resultado de la parte correspondiente a dicho cuestionario.

Figura 37. Estado del test de biestables de un alumno.

Las cuestiones de cada test se contestan tras mostrar una ventana superpuesta sobre el mismo, tal y como se muestra en la figura.

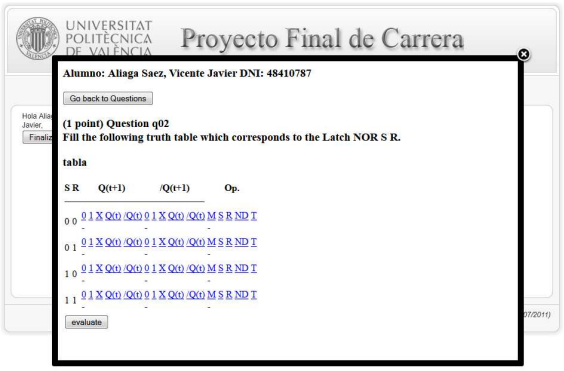


Figura 38. Cuestionario de una pregunta de biestables.

Esto evita que el alumno pueda visualizar el código fuente de la ventana del cuestionario, donde podría tener acceso a las respuestas de las cuestiones propuestas. Para abandonar la ventana modal lo haremos incluyendo el codigo javascript siguiente en cada una de las ventanas y que será:

```
function cerrarVentana(){
            parent.SqueezeBox.close();
            return false;
}
```

Con el código html

```
<input name="Questions" type="submit"
onClick="javascript:cerrarVentana()" value="Go Back to Questions"/>
```

Activaremos la función que nos permite abandonar el formulario. Tambien podemos abandonar el mismo al pulsar sobre Evaluate, si bien, en este caso se graban antes de regresar los resultados del test realizado.

Una vez alcanzada la fecha de publicación de una tarea concreta, aparecerá para aquellos usuarios que hayan contestado el test, la opción de ver el informe de notas del test en cuestion, tanto en pantalla, pulsando la opción notas, como generar un documento PDF, si este no estuviera ya disponible y una vez generado, poder descargarlo.

Figura 39. Generación del informe de examen por el alumno.

La visualización de las encuestas se puede realizar tras pulsar uno de los dos botones de visualización de resultados. Accionando Ver Notas se obtendrá una visualización en una ventana html del informe de notas.

Una vez mostradas las notas podemos abandonar el informe pulsando sobre Go Back o bién podemos imprimir el informe de notas pulsando sobre Print, en este caso nos aparece el cuadro de dialogo de selección de impresora en el equipo del alumno, y tras su selección y aceptación obtendrá una copia impresa del mismo.

Figura 40. Visualización de notas por parte de un alumno.

En el caso que el usuario desee obtener una copia personalizada del informe de resultados podrá pulsar sobre el icono que representa el documento, y si este no estuviera visible, pulsando sobre Generar PDF se generará en el instante el informe y lo tendrá a su disposición para su descarga.

La opción de Finalizar sesión ubicada en el lado izquierdo de la interfaz de usuario, nos posibilita en todo momento desconectarnos de la aplicación.

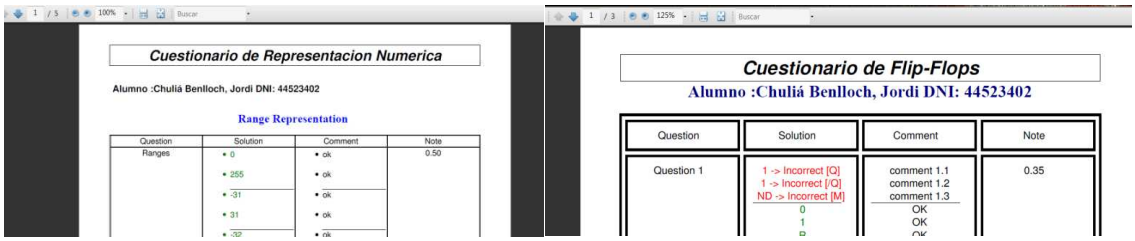


Figura 41. Informe de notas de cuestionarios en formato PDF.

8.4 Cambio de Contraseña.

Con el fin de mejorar el nivel de seguridad de la contraseña en el proceso de validación en el sistema, se ha implementado un formulario disponible tanto para el administrador (profesor) como para el usuario (alumno) para que puedan realizar la modificación de la misma.

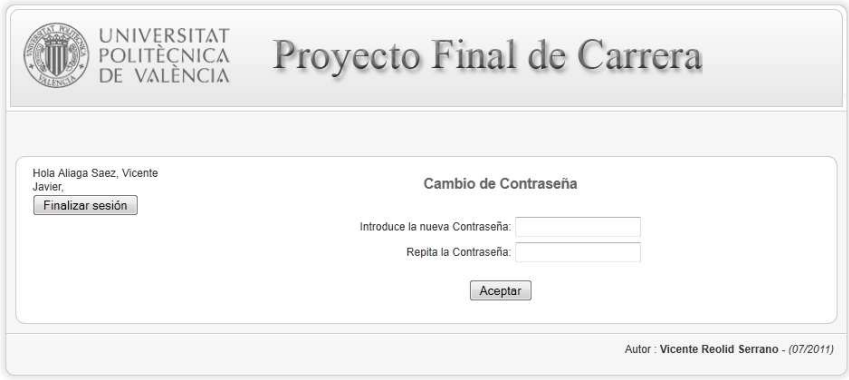


Figura 42. Formulario de Cambio de Contraseña.

Para ello, hay un icono visible  en la parte superior derecha disponible tras iniciar la sesión y que da acceso al formulario del cambio de contraseña. Para establecer la nueva contraseña es necesario introducir la misma dos veces, y ha de cumplir los siguientes requisitos; no debe exceder de quince caracteres ni ser inferior a cuatro. Una vez pulsado el botón de Aceptar nos modificará la contraseña y esta será válida para la siguiente conexión al sistema. Si queremos cancelar el proceso bastará con pulsar el botón de regresar.

9. CONCLUSIONES.

El desarrollo de una aplicación Web exige el conocimiento de varias tecnologías (*CMS*, *PHP*, *HTML*, *CSS*, *JAVASCRIPT*, *JSON*, *JQUERY*, *AJAX*, *MySQL*, *APACHE*) que forman un conjunto muy diverso en las que unas en el lado del cliente y otras en el lado del servidor y combinadas de manera adecuada convergen y permiten un amplio abanico de posibilidades en el desarrollo Web. El hecho de tener que utilizar tantas tecnologías nos permite adquirir un amplio conocimiento del espectro de las nuevas tecnologías y herramientas que se utilizan para el desarrollo de webs dinámicas, conocimiento que acompañado del desarrollo continuado, nos permite alcanzar un alto grado de competitividad en la implantación y uso de las nuevas tecnologías y servicios para la Web.

La mejora constante en el conocimiento de la herramienta, el API de desarrollo y las diferentes tecnologías que se utilizan, mejoran las posibilidades de elaborar un buen producto y un aprovechamiento adecuado de las mismas.

En este Proyecto Final de Carrera, se ha desarrollado un componente con las siguientes características:

- Integración los diferentes cuestionarios de evaluación que comprende la asignatura de primero de Grado de “Fundamentos de Computadores”.

- Mejora y ampliación de los cuestionarios existentes, agregándoles nuevas funcionalidades; la como la de facilitar al alumno la obtención de un informe del test realizado, que pueda servirle de comprensión y estudio de los temas que comprende el test de evaluación.

- Facilitar al profesor las tareas tanto de registrar los alumnos inscritos en la asignatura como la de automatizar el proceso de publicación de las notas obtenidas por los alumnos.

- Obtención de resultados estadísticos para cada uno de los test realizados, con el resultado global de cada cuestionario y la visualización de cada cuestionario realizado por un determinado alumno. En el cuestionario de valoración de la docencia, se permite al profesor la posibilidad de observar, en línea, los comentarios al sistema realizados por los alumnos.

Tras la implementación de esta aplicación se ha conseguido, que de forma sencilla, el profesor pueda plantear tareas de control y seguimiento del alumnado, automatizando las tareas de alta de los alumnos en el sistema, evaluación, obtención de resultados y publicación automatizada de notas en el sistema poliformat. Así mismo, permite que los alumnos tengan a su disposición una herramienta de fácil uso, que les permite evaluar su progreso, teniendo acceso de forma automatizada a los resultados obtenidos, con las correcciones pertinentes a las respuestas incorrectas y que le servirán para mejorar sus conocimientos en la materia y como soporte adicional para utilizar en sus estudios. Con ello, logramos obtener los objetivos propuestos.

10. TRABAJO FUTURO.

Para el futuro queda por desarrollar e integrar nuevos test que completen la evaluación de todos los temas de la asignatura. Dichos test se podrán ir integrando en el componente de tareas aquí desarrollado.

En la actualidad en la tabla de tareas están definidas de manera estática (al haber solamente dos componentes existentes) los tipos de tarea, así como de manera interna se identifican las tablas de la base de datos. Para que esto sea más dinámico, sería conveniente la creación de un mantenimiento nuevo en el que se generalicen los tipos de tareas introduciendo el nombre del componente, la tabla de la base de datos que contiene los resultados del test, etc. Con la finalidad de tener más parametrizada y automatizada la gestión de tareas.

Los comentarios referentes a cada una de las cuestiones se encuentran almacenados en el *layout* del cuestionario correspondiente, por lo que serán los mismos comentarios para todos los profesores que utilicen la herramienta. Para poder personalizar estos comentarios, sería conveniente que estos pudieran ser almacenados en la BD pudiendo ser redefinidos y readaptados para cada instalación de la aplicación.

Evaluar la migración de la herramienta de la versión 1.5 de *Joomla* a la versión 1.7 lo que implicará el conocimiento del nuevo API de *Joomla* y el posible aprovechamiento y utilización de nuevas funcionalidades.

11. BIBLIOGRAFÍA.

1. Lemus Zúñiga Lenin G, Benlloch Dualde J. Vicente. Experiencias en el uso de las TICs para facilitar la impartición de la asignatura de grado Fundamentos de Computadores. Valencia

2. http://es.wikipedia.org/wiki/Joomla

Definición y descripción de Joomla.

3. http://www.nosolocodigo.com/programacion-de-componentes-bajo-joomla-15

Tutorial en castellano de procedimiento de creación de componentes en Joomla.

4. http://api.joomla.org/.

Referencia del API de Joomla 1.5.

5. http://es.wikipedia.org/wiki/JQuery

Definición de JQuery.

6. http://www.javascriptya.com.ar/jquery/.

Tutorial de Introducción al uso de JQuery.

7. http://jquery.com/

Página oficial de JQuery.

8. http://www.JSON.org

Página oficial de JSON.

9.http://tecnologiasjava.blogspot.com/2010/02/json-javascript-object-notation.html.

Cómo funciona JSON.

10. http://docs.joomla.org/How_to_use_the_JTable_class.

Como utilizar las clases JTable en Joomla.

11. http://www.w3c.es/divulgacion/guiasbreves/ServiciosWeb

Guía Breve de Servicios Web.

12. http://www.ferdychristant.com/blog/articles/DOMM-6J2QFF.

Ejemplo de instalación y uso de un servicio web en PHP.

13. http://sourceforge.net/projects/nusoap/

Librería de NuSOAP.

14. http://msdn.microsoft.com/es-es/library/8wbhsy70%28v=vs.80%29.aspx#Y4560

Tutorial: Crear y usar un servicio Web ASP.NET en Visual Web Developer.

15. Juan Félix Mateos Barardo. Guía Práctica Joomla! 1.5.x .Anaya Multimedia.

16. Rahmel Dam. Profesional Joomla! . Anaya Multimedia.