Componente Web 2.0 para administración de la información sobre investigación

HOYOS PINEDA. JORGE GABRIEL

Grupo GIBRANT, Facultad de Ingeniería de Sistemas
Universidad Santo Tomas Seccional Tunja Boyacá, Colombia

Resumen— En este documento se realiza una descripción del proceso de desarrollo de un componente para Joomla 1.5 que servirá como herramienta para la administración seguimiento y divulgación del proceso de investigación llevado a cabo al interior de la Universidad Santo Tomás Seccional Tunja, y que será instalado en el sitio Web del Centro de Investigaciones de la USTA Tunja (CIUSTA). El componente permite ingresar, actualizar y consultar la información relativa a los grupos de investigación y las líneas, proyectos, productos, semilleros e investigadores vinculados a los mismos. El componente fue desarrollado siguiendo el patrón de diseño Modelo-Vista-Controlador (MVC) e hizo uso de las clases proporcionadas por el API de Joomla para ese modelo.

Palabras clave— Componente, MVC, Desarrollo Web.

Abstract— This document is a description of the development process of a component for Joomla 1.5 that will serve as a management tool for monitoring and dissemination of the research process undertaken within the Santo Tomas University Sectional Tunja, and that will be installed in the Web site of the Research Center at USTA Tunja. The component allows to enter, update and review information for research groups and lines, projects, products, assistants and researchers related to them. The component was developed following a design pattern Model-View-Controller (MVC) and made use of the classes provided by the Joomla API for that model.

Keywords— Component, MVC, Web Development.

I. INTRODUCCIÓN

El desarrollo que han tenido en los últimos años las Tecnologías de la Información y las Comunicaciones, y la disminución de los costos de acceso a Internet y de adquisición de equipos y dispositivos personales, ha conducido a que un cada vez mayor porcentaje de la población se encuentre conectado. Esta nueva sociedad conectada ha generado nuevos escenarios y nuevas formas de relación entre los individuos. Uno de estos nuevos escenarios es la denominada Web 2.0, que incorpora elementos que otorgan a los usuarios un mayor protagonismo y posibilita el desarrollo de nuevos mecanismos de comunicación y colaboración entre usuarios unidos por intereses comunes (Hoyos Pineda, 2009).

Aunque existen diferentes tecnologías para la construcción de sitios Web con las anteriores características, una de las más utilizadas es la que toma como base los llamados Sistemas de Gestión de Contenidos (Content Manager System, CMS), cuya tarea principal es la creación, administración, distribución, publicación y descubrimiento de información (Robertson, 2003). Adicionalmente, un CMS debe permitir la incorporación de nuevos componentes para obtener funcionalidades específicas.

El desarrollo de un componente para administración de la información de investigación tiene como propósito integrar en un único portal, a los grupos de investigación, junto con sus proyectos y productos sobre una plataforma web. El sistema está enfocado a crear un ambiente colaborativo y a facilitar el seguimiento de los proyectos de investigación de una forma sencilla y veraz. Adicionalmente el componente aporta a la superación de una de las mayores falencias del proceso investigativo llevado a cabo al interior de la instituciones de educación superior, como es la falta de divulgación de sus actividades y resultados. Es de suma importancia que los estudiantes, docentes y el público en general dispongan de un medio de acceso libre, como un portal web, para informarse de la actividad investigativa desarrollada al interior de las instituciones de educación superior.

II. DESARROLLO

En la primera parte del documento se relacionan los antecedentes encontrados en relación a la administración de información de investigación en las instituciones de educación superior. En la segunda parte se describe la metodología de desarrollo aplicada a la solución propuesta, dónde se resaltan el modelo vista controlador (MVC), que sirve de base para el esquema de desarrollo de componentes propuesto por la comunidad Joomla, y los elementos principales del API que posibilitan el desarrollo de componentes compatibles con este framework. En el resto del documento se describe la forma como el modelo es aplicado a la creación de un componente para la administración de la información sobre investigación, y su incorporación al portal del Centro de Investigaciones de la USTA Tunja.

A. Antecedentes

La mayoría de instituciones de educación superior cuentan con un espacio o una sección en las páginas Web institucionales, dedicado a presentar información general sobre los procesos de investigación que se llevan a cabo al interior de las mismas. Normalmente estas páginas son alimentadas sólo por el administrador de la página ocasionándose con frecuencia la desactualización e incompletitud de la información publicada (UPTC, 2010; Universidad de Boyaca, 2010; Fundacion Universitaria Juan de Castellanos, 2010; USTA, 2010). Esta situación ha generado una dispersión de la información ya que los grupos de investigación se han dado a la tarea de crear y administrar sus propios sitios en procura de superar estas limitaciones (GICOGE, 2010).

Algunas instituciones cuentan también con sistemas de información internos que les permiten tener una visión más completa, pero de igual forma la información no se hace visible para el conjunto de la comunidad académica que pudiera estar interesada en los trabajados desarrollados por otras instituciones y grupos de investigación.

Otra fuente importante de información acerca de la investigación llevada a cabo en el país, la constituye la plataforma ScienTI (Colciencias, 2010). Esta plataforma brinda información sobre la historia investigativa tanto de los grupos de investigación como de los mismos investigadores, pero no permite ofrecer una visión global sobre el estado de la investigación en una institución particular.

B. El Modelo Vista Controlador

El Modelo Vista Controlador (MVC) es un patrón de diseño que consiste en la división del desarrollo en tres capas de aplicación: el modelo, la vista y el controlador.

En la Figura 1 se muestra un esquema de la relación existente entre las tres capas de aplicación.

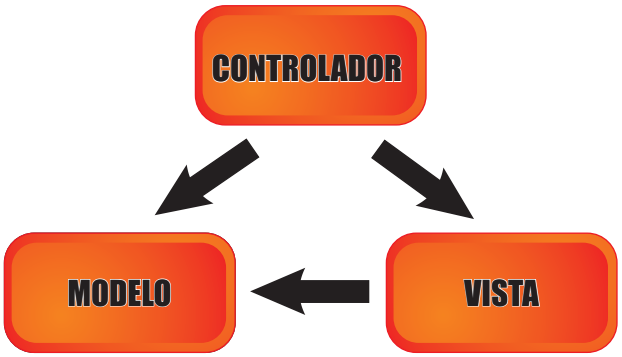

FIGURA 1. ESQUEMA DEL PATRÓN DE DISEÑO MODELO-VISTA-CONTROLADOR
Fuente: Autor

La primera capa conocida como el modelo, es la encargada de la lógica de negocio y la administración de los recursos de datos. El modelo representa objetos reales que hacen parte del ámbito de aplicación. Por ejemplo para el caso de estudio el modelo contiene la representación de las entidades propias de la estructura de investigación como son grupo, línea, proyecto, semillero e investigador.

La segunda capa se conoce como Vista, y tiene como función proporcionar una interfaz que le permita al usuario interactuar con el sistema. Las vistas se encargan de presentarle al usuario la información proporcionada por el modelo. Una vista está asociada a un modelo, y varias vistas pueden estar asociadas a un mismo modelo. La vista se actualiza en respuesta a cambios en el modelo gracias a un sistema de notificaciones generadas por el mismo (Deacon, 2009).

La ultima capa referida como Controlador, se ocupa de recibir las peticiones realizadas por el usuario a través de una interfaz gráfica, y reorientarlas hacia las unidades específicas del modelo encargadas de su procesamiento. Como su nombre lo dice, el controlador debe administrar el flujo de eventos provenientes de la interfaz de usuario, y de convertir esos eventos en operaciones que son solicitadas a y procesadas por el modelo. Además tiene la responsabilidad de administrar las vistas, es decir puede decidir que vista utilizar y cuando las mismas de cierran o se abren.

A. El API de Joomla

Joomla está construido sobre un API comúnmente conocido como el "Joomla-Framework" (Framework - Joomla! Documentation, 2010). Esta API está compuesta por un conjunto de clases agrupadas en paquetes que hacen posible la integración de componentes, módulos y plantillas. Así mismo, su utilización adecuada hace posible que una gran cantidad de componentes desarrollados para atender necesidades específicas, puedan ser integrados en forma transparente con el núcleo central.

Adicionalmente, el framework proporciona un conjunto de funcionalidades, como por ejemplo un esquema de seguridad que protege la aplicación de accesos indebidos, razón por la cual no es necesario que el desarrollador se preocupe por ese aspecto cuando está diseñando un nuevo componente.

Para el caso de los componentes el API proporciona cuatro clases que facilitan la implementación del modelo MVC, son estas JModel, JView, JController y JTable.

- JModel. Funciona como fábrica de los objetos requeridos por la aplicación y proporciona un amplio conjunto de funciones de soporte. Sirve como clase base en la implementación de las clases que representan el modelo.
- Jview. Provee los métodos necesarios para la visualización de la información sobre una interfaz gráfica. Sirve como clase base para la implementación de las clases que representan la vista.
- JController. Proporciona la funcionalidad básica y permite la administración de las vistas. Sirve como clase base para la implementación de las clases que representan el controlador.
- JTable. Proporciona la funcionalidad necesaria para crear, leer, actualizar y borrar registros sobre una tabla de la base de datos. Sirve como clase base para implementar las clases que representan las tablas de la base de datos.

B. Metodología

Para el desarrollo del componente se utilizo una metodología de tipo incremental con sucesivas iteraciones en cada fase hasta llegar a la versión final del sistema, característica importante fomentada por el Proceso Unificado (UP) (Larman, 2003). Esta metodología permitió que el software fuera probado por los usuarios en etapas refinadas sucesivamente a lo largo del proyecto, de manera que se introdujeron mejoras o correcciones al sistema en un menor tiempo.

1) Análisis de la solución propuesta

La solución propuesta consiste en la construcción de un Componente para Joomla 1.5, que facilite las labores de administración de la información relacionada con el proceso investigativo llevado a cabo al interior de una institución. Adicionalmente, el componente debería permitir el acceso a la información de investigación, en diferentes niveles de acuerdo a la definición de varios tipos de usuarios.

2) Desarrollo del Componente

Los usuarios de Joomla tienen la posibilidad de interactuar con el portal mediante dos interfaces diferentes. La primera conocida como "front end", corresponde a la interfaz principal del sitio web sobre la que puede navegar cualquier usuario anónimo o registrado. La segunda interfaz se conoce como "back end", y le permite a un usuario administrador ejecutar tareas de configuración, mantenimiento, generación de contenidos, y otras propias de dicho rol (Graf, 2008).

Cuando se construye un componente es usual dividirlo en las dos partes, "front end" y "back end". Las definiciones correspondientes a cada una de las dos partes, se incluyen en una carpeta del mismo nombre, y sólo difieren en su ubicación. Mientras que en el caso del "front end" la carpeta estará ubicada dentro de la carpeta components del sitio web, la carpeta correspondiente al "back end" estará ubicada dentro de la carpeta administrator/components del sitio.

En el proceso de construcción de componentes para Joomla 1.5 es necesario observar una serie de convencio-

nes para la denominación de las clases, los archivos, las carpetas y el mismo componente. Para el caso de la denominación del componente se usa el sufijo "com_" seguido del nombre distintivo del componente. Para el caso de estudio, el componente se llamará "com_investigacion". Este mismo nombre será utilizado como nombre de la carpeta que va a contener los archivos del componente, tanto en el "front end", como en el "back end".

a) Desarrollo del Back End del componente

El "back end" puede incluir una completa interfaz de administración del sitio, en caso de requerirse diferentes opciones de configuración aplicables a todo el componente. El "back end" es también el lugar adecuado para definir la estructura de clases que va a permitir el acceso a la base de datos que almacena la información utilizada por el componente, dichas definiciones quedarán en la carpeta "tables" del componente

Aunque la creación de las tablas en la base de datos es una tarea previa, es necesario observar algunas convenciones en ese proceso. La convención utilizada para denominar las tablas en la base de datos consiste en anteponer al nombre de la tabla el prefijo de la base de datos (jos_ por defecto en joomla) y el nombre del componente. Para el caso de estudio si se va a crear la tabla encargada de almacenar la información de los investigadores, el nombre de la tabla debería ser "jos_investigacion_investigador".

Una vez de la tabla ha sido creada físicamente en la base de datos, el siguiente paso consiste en definir una clase derivada de la clase JTable que permita el acceso a los datos. En la definición de esta nueva clase se debe garantizar la correspondencia entre el nombre de los atributos y el nombre de los campos en la tabla. Para la denominación de la clase se aplica la convención consistente en anteponer al nombre de la tabla, el prefijo "Table". Igualmente se debe observar que el archivo que contiene la definición de la clase quede ubicado en la carpeta "tables". Para el caso de estudio se debe definir la clase "TableInvestigador", que quedará almacenada en un archivo de nombre "investigador.php" dentro de la carpeta /administrator/components/com_investigacion/tables, como se muestra a continuación:

Script de creación de la tabla	Definición de la clase

CREATE TABLE
`ciu_investigacion_investigador`
(
`identificacion` int(11) NOT NULL,
`nombre` varchar(50) NOT NULL,
`facultad` varchar(50) NOT NULL,
`email` varchar(50) NOT NULL,
`cvlac` varchar(255) NOT NULL,
PRIMARY KEY (`iden`)
);
	<?php
	defined('_JEXEC') or die('Restricted access');
	class TableInvestigador extends JTable
	{
	var $identificacion = null;
	var $nombre = null;
	var $facultad = null;
	var $email = null;
	var $cvlac = null;
	function __construct(&$db)
	{
	parent::__construct('#__investigacion_investigadores', 'id', $db);
	}
	}
	?>

FIGURA 2. SCRIPT DE CREACIÓN DE UNA TABLA DE LA BASE DE DATOS Y CÓDIGO DE DEFINICIÓN DE LA CLASE CORRESPONDIENTE
Fuente: Autor

b) Desarrollo del Front End del componente

A partir de la carpeta principal del componente se deben crear las carpetas "models" y "views", para contener las clases representativas del modelo y la vista respectivamente.

Un modelo de aplicación del patrón MVC es presentado en (KENNARD, 2007). El modelo utiliza las clases del API de Joomla, JView, JModel y JController, y contempla algunas recomendaciones para la definición de las clases derivadas que harán parte del componente, como las siguientes:

- Para el caso del modelo, la convención establece que el nombre de la clase debe estar formado por el nombre del componente, la palabra Model y el nombre de la entidad de datos. Para el caso de estudio si la entidad de datos de llama Grupo, el modelo se llamará InvestigacionModelGrupo. A su vez la definición de clase debe estar almacenada en un archivo cuyo nombre se corresponda con el de la entidad y que debe estar ubicado en la carpeta "models", para el caso de estudio la clase estará almacenada en un archivo de nombre grupo.php en la carpeta "models" del componente.

```
<?php
defined('_JEXEC') or die("Invalid access");
jimport('joomla.application.component.model');

class InvestigacionModelGrupo extends JModel
{
 var $datos;

 . . .

function crear($post){
  $row =& JTable::getInstance('grupo', 'Table');
  if (!$row->bind($post)) {
    return JError::raiseWarning(500, $row->getError());
  }
  if (!$row->store()) {
    return JError::raiseWarning(500, $row->getError());
  }
 return 1;
  }
}
?>
```

FIGURA 3. FRAGMENTO DEL CÓDIGO Y ESTRUCTURA DE CARPETAS UTILIZADAS EN EL MODELO
FUENTE: AUTOR

- Para el caso de las vistas se requiere crear una carpeta por cada vista diferente dentro de la carpeta *"views"* del componente. La carpeta correspondiente a cada vista puede contener un archivo diferente por cada tipo de documento soportado por la vista. La convención especial usada para las vistas contempla que el nombre de la clase debe estar formado por el nombre del componente, la palabra View y el nombre de la vista. Esta definición debe ser almacenada en un archivo de nombre *view.documentType.php,* donde *documentType* corresponde al tipo de documento que puede ser FEED, HTML, PDF o RAW. Para el caso de estudio si la vista tiene por objeto mostrar la información de un grupo, la vista

se llamará *InvestigacionViewMostrarGrupo*, y estará almacenada en un archivo de nombre *view.html.php*, que estará ubicado en la carpeta *"mostrargrupo"* que a su vez estará contenida en la carpeta *views* del componente. Adicionalmente si la vista soporta el tipo de documento *HTML* se debe incluir una carpeta con nombre *"tmpl"*, donde se ubican los archivos que contienen el código *html* necesario para desplegar la vista, para el caso de estudio se llamará *default.html.*

```
<?php
defined( '_JEXEC' ) or die( 'Restricted access' );
jimport( 'joomla.application.component.view');

class InvestigacionViewMostrarGrupo extends JView
{
   function display($tpl = null)
   {
                $option = JRequest::getCmd('option');
          $nombre = JRequest::getVar('id', 0);
          $model =& $this->getModel();
          $datos = $model->consultar($nombre);
          $this->assignRef( 'datos',   $datos );
          $lineas = $model->consultarLineas($nombre);
          $this->assignRef( 'lineas',   $lineas );
         parent::display($tpl);
   }
}
?>
```

FIGURA 4. FRAGMENTO DEL CÓDIGO Y ESTRUCTURA DE CARPETAS UTILIZADA PARA LAS VISTAS
FUENTE: AUTOR

- Para el caso del controlador, es usual la definición de una sola clase que se almacenará en un archivo de nombre *controller.php*, aunque es posible definir varios controladores. Los controladores usan un mecanismo que asocia el nombre de una tarea o *task*, con la ejecución del método del controlador que lleva el mismo nombre. El valor de la tarea se puede establecer mediante la variable *task,* cuando se realiza una operación de tipo POST.

```
<?php
defined( '_JEXEC' ) or die('Restricted access' );
jimport('joomla.application.component.controller');

class InvestigacionController extends JController
{
          function display()
          {
                parent::display();
          }

function crearInvestigador(){
          $option = JRequest::getCmd('option');
          $modelo = $this->getModel('investigador');
   echo $modelo->getMensaje();
           $res = $modelo->crear(JRequest::get('post'));
           if($res == 1){
                $this->setMessage('El investigador ha sido creado!');
          $this->setRedirect('index.php?option=' $option. '&view=investigador');
           }else{
                $this->setMessage('Error al introducir!');
           }
}
. . .
}
```

FIGURA 5. FRAGMENTO DE CÓDIGO DEL ARCHIVO CONTROLLER.PHP
FUENTE: AUTOR

Por último es necesario crear un archivo que sirve como puerta de entrada al componente. Este archivo debe llevar el mismo nombre del componente (sin el prefijo com_) y debe estar ubicado en la carpeta raíz del mismo. Para el caso de estudio este archivo se llamará investigación.php.

```
<?php
defined( '_JEXEC' ) or die( 'Restricted access' );
require_once( JPATH_COMPONENT.DS.'controller.php' );
 if($controller = JRequest::getWord('controller')) {
    $path = JPATH_COMPONENT.DS.'controllers'.DS.$controller.'.php';
    if (file_exists($path)) {
       require_once $path;
    } else {
       $controller = '';
    }
}
 $classname    = 'InvestigacionController'.$controller;
$controller   = new $classname( );
 $controller->execute( JRequest::getVar( 'task' ) );
 $controller->redirect();
?>
```

FIGURA 6. FRAGMENTO DE CÓDIGO DEL ARCHIVO INVESTIGACIÓN.PHP
FUENTE: AUTOR

En la figura 6 se puede observar que este archivo es el encargado de crear una instancia del controlador e invocar el método "redirect" responsable de cargar la vista correspondiente.

En la Figura 7 se muestra la estructura general del front end del componente aplicando el modelo mencionado.

FIGURA 7. ESQUEMA DE LA ESTRUCTURA DE ARCHIVOS UTILIZADAS PARA EL COMPONENTE EN EL FRONT END
FUENTE: AUTOR

3) Integración el componente con el portal

La integración del componente al portal resulta un procedimiento muy sencillo, si el componente ha sido desarrollado siguiendo el modelo propuesto. Joomla proprociona en el modulo de administración del sitio, una utilidad que instala y registra el nuevo componente a partir de un archivo empaquetado, tal como se muestra en la Figura 8

Una vez registrado el componente, lo único que resta es vincularlo a la opción de menú que servirá como puerta de entrada al mismo.

E. Resultados

Una vez instalado el componente se tiene acceso a una serie de funcionalidades que facilitan las tareas de administración de la información sobre los procesos de investigación llevados a cabo al interior de la institución. Es así como el usuario con rol de coordinador puede ingresar información acerca de los grupos, las líneas, los proyectos, los productos y semilleros de investigación. La información ingresada es utilizada para generar consultas dinámicas, que están disponibles para todo tipo de usuario, donde se muestra la relación existente entre grupos y líneas, líneas y proyectos, proyectos y productos. Adicionalmente, la información presentada en dichas consultas contiene hipervínculos que le permiten al usuario acceder a un mayor nivel de detalle, en cuanto a aspectos como información sobre los investigadores y productos. Un ejemplo de la interfaz es mostrado en la Figura 9.

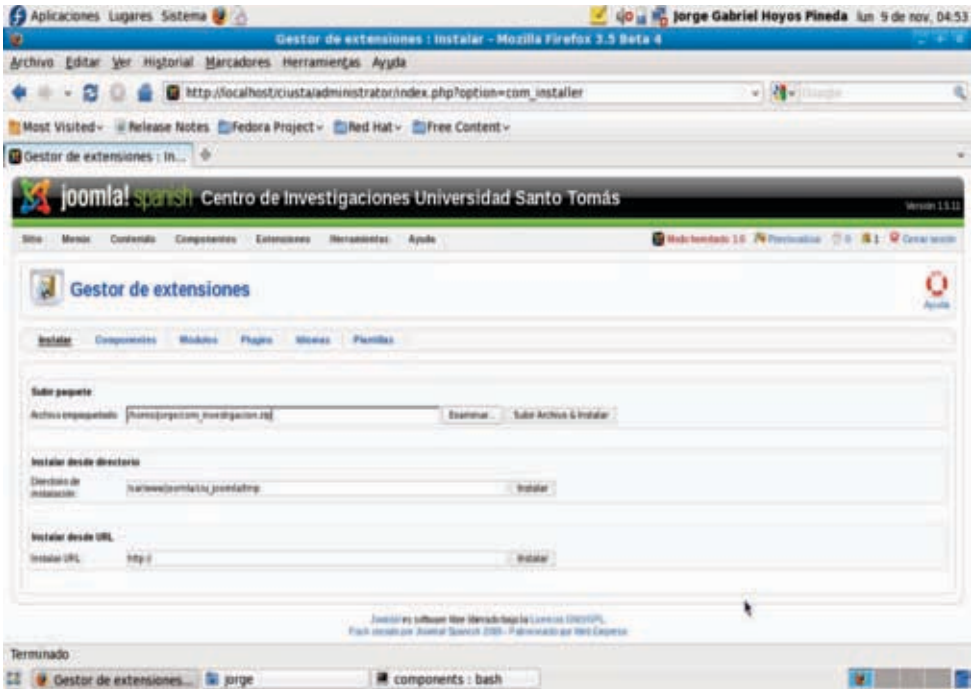

FIGURA 8. INTERFAZ DEL BACK END PARA LA INSTALACIÓN DEL COMPONENTE
FUENTE: AUTOR

FIGURA 9. INTERFAZ DEL FRONT END PARA VISUALIZACIÓN DE INFORMACIÓN
FUENTE: AUTOR

III. CONCLUSIONES

La divulgación de las actividades y resultados obtenidos por parte de la estructura investigativa de las instituciones de educación superior resulta de suma importancia en la construcción de una cultura y una comunidad investigativa. En procura de superar las limitaciones en el trabajo interdisciplinario e interinstitucional se hace necesario hacer público el trabajo desarrollado por los diferentes grupos de investigación. Un recurso que se tiene a la mano y que se puede utilizar en forma efectiva son los sitios Web institucionales, aprovechando el uso cada día más extenso de la Web por parte de todo tipo de usuarios y el surgimiento de nuevas formas de relacionarse con otros a través de Internet.

La utilización de componentes permite un desarrollo incremental, ya que es posible agregar nuevas funcionalidades, o crear nuevos componentes que complementen al primero, atendiendo temas relacionados. El desarrollador, además puede hacer uso de su creatividad para utilizar y combinar componentes ya desarrollados, y hacerlos parte de sus aplicaciones.

En el mundo del software libre, existe una gran variedad de tecnologías y herramientas para la construcción de componentes, sólo se requiere que el desarrollador profundice en el conocimiento de las mismas. Esto se evidencia en el caso de Joomla, el cuál proporciona un modelo de desarrollo de componentes, las clases base necesarias para su implementación y un conjunto de reglas de nombrado que garantizan en gran medida la adecuada integración del componente con el portal Web.

REFERENCIAS

Bascon, E. (2004). El patron de diseño Modelo-Vista-Controlador (MVC) y su implementacion en Java Swing. http://www.ucbcba.edu.bo/Publicaciones/revistas/actanova/documentos/v2n4/v2.n4.bascon.pdf .

Colciencias - ScienTI. (2010). http://www.colciencias.gov.co/scienti.

Deacon, J. (2009). Model-View-Controller (MVC) Architecture. Retrieved November 6, 2009, from http://www.jdl.co.uk/briefings/mvc.html .

Framework - Joomla! Documentation. (2010). http://docs.joomla.org/Framework.

Fundacion Universitaria Juan de Castellanos. (2010). http://www.jdc.edu.co/index.php?option=com_content&view=article&id=47&Itemid=58.

GICOGE. (2010). http://gicoge.udistrital.edu.co/.

Graf, H. (2008). Building Websites with Joomla! 1.5. Birmingham: Packt.

Hoyos Pineda, J. G. (2009). Herramientas de desarrollo para la Web 2.0. Investigium Ire .

Kennard, J. (2007). Mastering Joomla! 1.5 Extension and Framework Development. Birmingham: Packt Publishing Ltd.

Larman, C. (2003). UML y Patrones. Madrid: Prentice Hall.

LeBlanc, J. (2007). Learning Joomla! 1.5 Extension Development. Birmingham: Packt Publishing Ltd.

Merlino, H. (2009). Patron de Diseño de Vistas Adaptables.

Robertson, J. S. (2003). What is a CMS? Retrieved Jul 3, 2010, from http://www.steptwo.com.au/papers/kmc_what/index.html .

Universidad de Boyaca. (2010). http://www.uniboyaca.edu.co/index.php?option=com_content&view=category&layout=blog&id=65&Itemid=163.

UPTC Investigación. (2010). http://www.uptc.edu.co/.

USTA. (2010). http://www.usta.edu.co/index.php?option=com_content&view=article&id=133.