

espiral de valores: técnicas avanzadas de programación PHP

Diego Arranz
Interactors

Madrid
España
diego@lawebespiral.org

Javier Linares
Interactors

Sevilla
España
javier@lawebespiral.org

espiral de valores (<http://espiraldevalores.org>) es un lugar en Internet donde se debaten de forma organizada y colaborativa posicionamientos ante problemáticas sociales, políticas, medioambientales, etc. Cada uno de estos posicionamientos se denomina “valor”, y consiste en un texto con el planteamiento de un problema concreto y una propuesta de solución. El presente artículo explica las ideas y procesos colaborativos que hay detrás de esta web e introduce las metodologías y los correspondientes paquetes de software libre utilizados en el desarrollo. También se esbozan algunas futuras líneas de desarrollo que ampliarían en gran medida las posibles aplicaciones de la herramienta.

1. Introducción

Internet está llena de foros de debate sobre sociedad, política, religión, etc. Millones de lugares virtuales donde tienen lugar trillones de debates y discusiones. Se trata sin duda de una de las mejores aplicaciones a nivel social de Internet.

Sin embargo, detrás de la espiral de valores está la convicción de que es necesario dar un paso más allá de este tipo de foros. Al fin y al cabo, un foro virtual no es más que una gran cantidad de datos, argumentos y opiniones en un formato que hace prácticamente imposible la consulta de algo determinado pasado un tiempo.

La espiral de valores es un esfuerzo que pretende dar con un modo más productivo de canalizar todas esas energías desparramadas a lo largo de tanto debate y discusión prácticamente estéril. Se trata de un piso más, edificado sobre un foro virtual. Aporta un proceso de debate orientado hacia la consecución de consensos, y una herramienta que permite referenciar, organizar y reutilizar los resultados de los debates, ya sean definitivos o parciales.

1.1. Un poco de historia

espiral (<http://lawebespiral.org>) nace en mayo de 2001 con el objetivo de introducirse legalmente en entornos políticos, mediáticos y corporativos promocionando posibles futuros más sostenibles y humanos. Se trata de una organización abierta, transparente y descentralizada, que utiliza Internet como herramienta principal de comunicación.

espiral de valores surge en abril de 2002 como proyecto de espiral, tomando como inspiración el proyecto Open Directory Project (<http://dmoz.org>), directorio de páginas de Internet descentralizado, mantenido por voluntarios. La idea inicial era crear un directorio de posicionamientos, a la vez que un espacio participativo de reflexión y diálogo, y así afrontar mejor los debates teóricos sobre un futurible programa electoral del partido espiral, ya reconocido legalmente por aquellas fechas.

1.2. Objetivos

Aunque no contemplados desde el principio, con el tiempo se fueron haciendo evidentes una serie de objetivos que podía intentar cumplir el proyecto:

1. Servir de ayuda a los debates y acciones de personas y colectivos dentro y fuera de espiral.
2. Ofrecer una visión condensada y organizada de los muchos esfuerzos que llevan a cabo multitud de personas y organizaciones en pos de un mundo mejor.
3. Proveer de una base teórica al desarrollo de denuncias, es decir, anuncios que no venden nada sino que denuncian un determinado problema.
4. Ofrecer a la comunidad de software libre una herramienta que puede servir como base para proyectos software más ambiciosos en campos como la gestión de contenido o la democracia participativa.
5. Demostrar que el compromiso social o el uso de software libre no son incompatibles con el uso de avanzadas metodologías y herramientas software.

2. Valores y debates

Los valores son las unidades de información básicas. Son como los ladrillos que forman un edificio. Cada valor debe contener en su texto el planteamiento de un problema más o menos concreto y una propuesta de solución. Todos los valores son el resultado de un proceso de debate abierto que tiene lugar en un foro virtual enlazado con la web.

2.1. Composición y organización de los valores

Formalmente, cada valor se compone de un título, una descripción de uno o dos párrafos y una ampliación más detallada con argumentos a favor, puntos a profundizar, argumentaciones típicas en contra de la propuesta, etc. Es recomendable que incluya también enlaces con información adicional, organizaciones relacionadas con el valor, etc. Lo normal es que todo lo anterior se pueda imprimir en una sola hoja de papel.

A modo de ejemplo, algunos títulos de valores que pueden encontrarse en la web son: “Los ciudadanos tienen derecho a decidir a qué se destinan sus impuestos”, “Vidas sencillas” o “La ciencia y la tecnología están en nuestras manos”.

Los valores se organizan por categorías y subcategorías y también según unos niveles de importancia. Cualquier persona puede participar en los debates que conducen a estos valores, así como proponer nuevos debates de

valores. La gestión y actualización del contenido se realiza de forma descentralizada, por medio de una serie de editores que, conectándose a la web mediante una clave, se ocupan de mantener la categoría que libremente han elegido.

2.2. El proceso de debate

Aunque cualquiera puede participar en los debates de la forma que desee, hay unas normas en lo que se refiere a la presentación, discusión y aprobación de los valores. Este proceso está inspirado en la motivación principal que guía este proyecto: intentar siempre que sea posible llegar a consensos, no dejar los debates a medias.

De forma resumida el proceso es el siguiente:

Una vez formulado el valor cualquiera puede criticar, apoyar, corregir o añadir los aspectos que considere oportunos. El valor se debate el tiempo que sea necesario hasta llegar a un consenso. Para que sea oficialmente aprobado, cuatro o más participantes deben dar su apoyo explícito a la redacción del valor. Más adelante, es posible reabrir el debate y modificar el texto si alguien lo cree necesario.

Cada editor es el responsable de dinamizar la categoría que eligió, presentando valores a debate, procurando que los propuestos por otras personas lleguen a buen puerto y actualizando la información de la web en consecuencia.

3. Metodología Modelo-Vista-Controlador

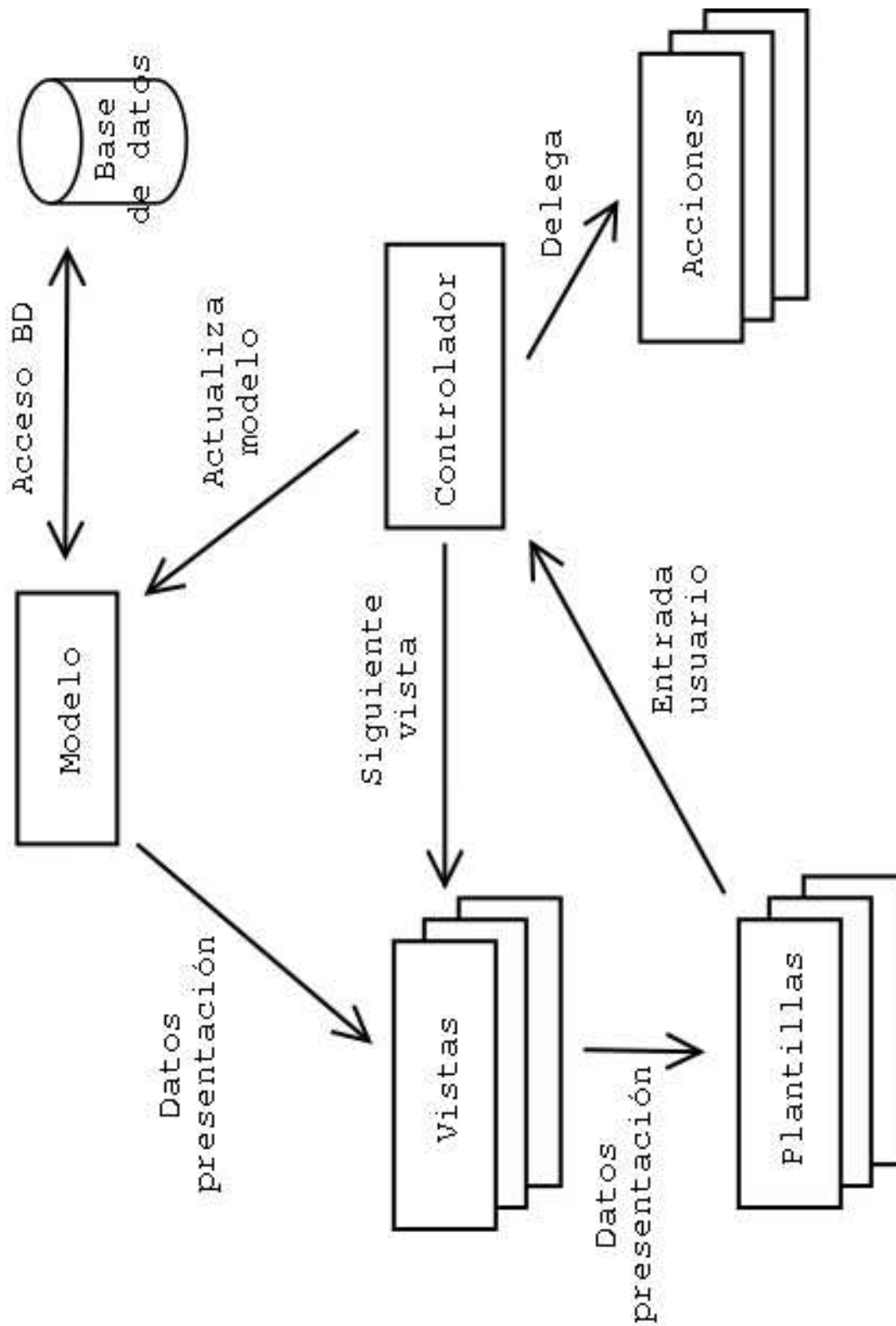
Las aplicaciones web se programan según algún lenguaje de programación para servidor, por ejemplo PHP, JSP o ASP. Estos lenguajes permiten mezclar en cada fichero código de presentación (normalmente HTML) y código de programación de cualquier tipo (construcciones simples, uso de objetos, comunicación de bases de datos), de forma sencilla y flexible. Sin embargo, cuando las aplicaciones adquieren un tamaño considerable, si no se ha seguido alguna metodología de desarrollo puede resultar una pesadilla revisar el código para solucionar errores o añadir nuevas funcionalidades.

Lo que requieren la mayoría de aplicaciones web es una forma correcta de organizar el código en componentes que tengan funciones claras, como alternativa al habitual código “spaghetti” en el que todo está mezclado. La idea en la que se basa la metodología MVC (http://www.ulpgc.es/otros/tutoriales/java/Apendice/arq_mvc.html) es que los aspectos visuales de un sistema deberían estar aislados del funcionamiento interno, que a su vez debería estar separado del mecanismo de coordinación global de la aplicación.

A continuación se expone la arquitectura de componentes, con un esquema de esta metodología y una explicación de cada parte, y más adelante se hace un recorrido por el código, siguiendo el flujo del programa.

3.1. Arquitectura de componentes

Figura 1. Esquema metodología MVC



3.1.1. El modelo

Es el código que administra el estado interno del sistema. Se compone de una serie de clases o funciones que gestionan por una parte la lógica de proceso (o reglas de negocio) y por otra el acceso a la base de datos. Es recomendable que ambos aspectos estén en módulos separados. El modelo no contiene ningún componente visual. Funciona como un interfaz de programación (API) que encapsula los detalles internos del sistema.

3.1.2. Las vistas

Constituyen la capa de presentación del sistema, la parte visible de cara al usuario. No deben acceder directamente a bases de datos ni contener lógica de proceso (quizá si un poco de lógica para la presentación). Cualquier dato que necesiten mostrar al usuario se recupera mediante llamadas a la API que ofrece el modelo.

3.1.3. El controlador

Es el componente que coordina el funcionamiento global de la aplicación. Toda acción realizada por el usuario es gestionada por él. En función de la vista actual, la acción concreta y el estado del modelo: Manipula el modelo invocando la API de éste y selecciona la siguiente vista a mostrar.

3.2. Plantillas

El mecanismo de plantillas sirve para separar el código de presentación del resto del código de una aplicación web. Consiste en codificar todo lo que tenga que ver con la presentación en una serie de plantillas de código HTML (u otro lenguaje de presentación), con expresiones sencillas intercaladas para comunicarse con el resto de la aplicación y poder mostrar datos dinámicos. Un motor de plantillas es el que se encarga de hacer la traducción a HTML.

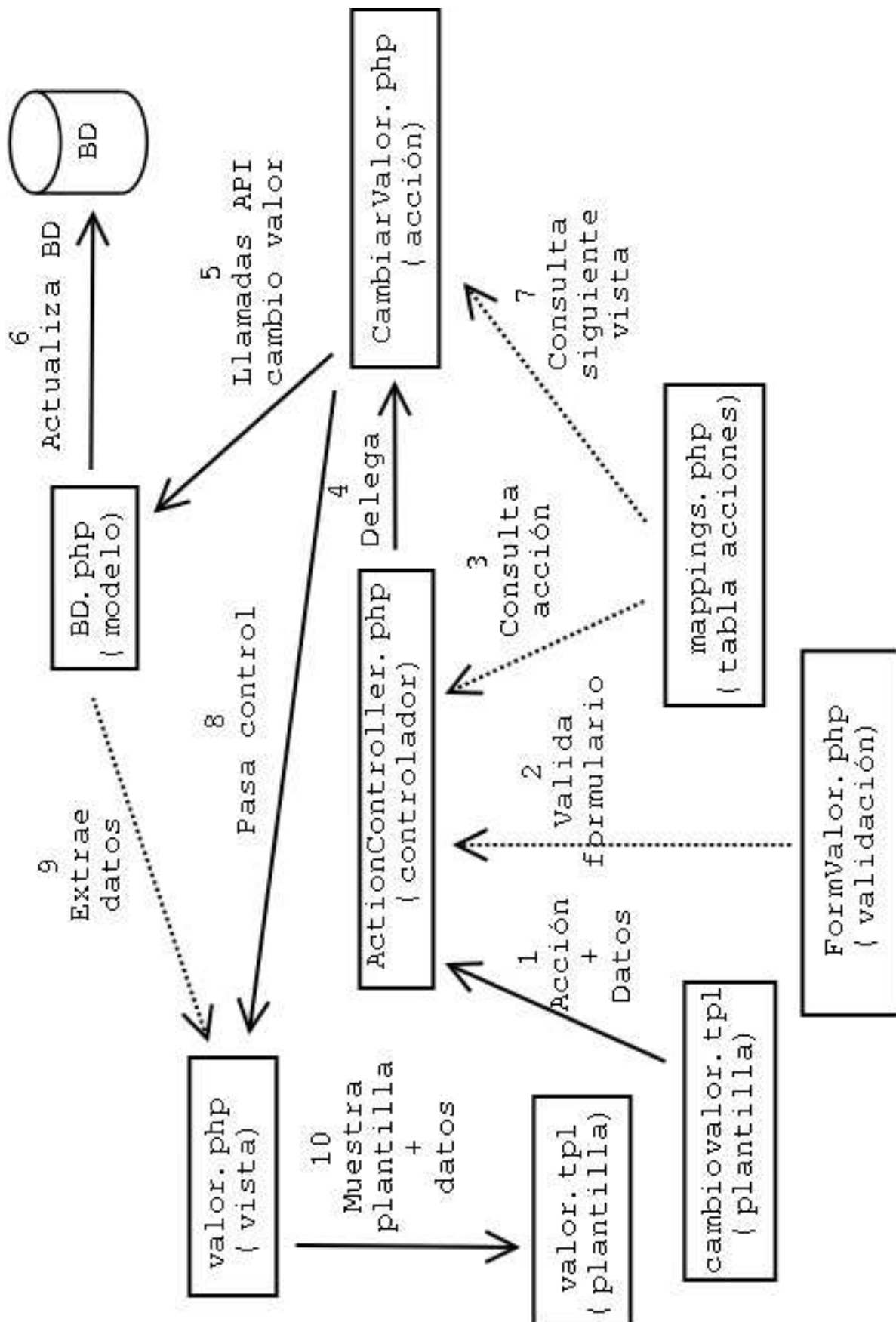
Este mecanismo es independiente de la metodología MVC y puede utilizarse sin ella, pero combinando las dos cosas se consigue una estructura de código muy organizada y elegante.

De esta forma, las vistas simplemente se ocupan de extraer la información necesaria del modelo y comunicársela a las plantillas, y estas simplemente de dar un formato visual a esa información, añadiendo la información estática pertinente.

Aunque así se complica algo la construcción del sistema al introducir un nuevo nivel, en las plantillas el código de presentación queda mucho mejor aislado que en las vistas de una aplicación MVC clásica (sin plantillas), de forma que se facilita la tarea de los diseñadores web.

3.3. Un recorrido por el código

Figura 2. Ejemplo de flujo



A continuación explicamos cómo funciona la arquitectura MVC mediante un recorrido por los distintos componentes que colaboran para resolver una acción concreta del usuario sobre la aplicación, ilustrando el ejemplo con algo de código representativo de cada componente. El número de los pasos se corresponde con los pasos del esquema de flujo.

Paso 1: Partimos de la plantilla que muestra un formulario a través del cual se pueden cambiar los datos de un valor. Los datos introducidos se envían al controlador junto con el código propio de esta acción.

```
<form action="{\$control}" method="post">
<input type="text" name="titulo" size="50" value="{\$titulo}">
<input type="radio" name="nivel" value="2" {if \$nivel==2}checked{/if}>Esencial
<input type="radio" name="nivel" value="1" {if \$nivel==1}checked{/if}>De categoría
<input type="radio" name="nivel" value="0" {if \$nivel==0}checked{/if}>Normal
<input type="hidden" name="accion" value="{\$accion}"/>
</form>
```

Paso 2: Los datos del formulario son validados por un componente específico para tal fin.

```
function validate()
{
if (!$this->get('titulo')) {
trigger_error('Introduzca título');
$isValid = FALSE;
}
else if (!$this->get('descripcion')) {
trigger_error('Introduzca descripción');
$isValid = FALSE;
}
...
return $isValid;
}
```

Pasos 3 y 4: El controlador consulta en la tabla de acciones qué componente debe atender la acción y le delega el control.

```
'CambiarValor' => array(
  _TYPE => 'CambiarValor',
  _NAME => 'FormValor',
  _INPUT => 'index.php?view=views/cambiovalor.php',
  _VALIDATE => 1,
  _ACTION_FORWARDS => array(
    'OK' => array(
      _PATH => 'index.php?view=views/valor.php',
      _REDIRECT => 0
    ),
    'Error' => array(
      _PATH => 'index.php?view=views/cambiovalor.php',
      _REDIRECT => 0
    )
  )
),
```

Pasos 5 y 6: La acción realiza las modificaciones oportunas en el modelo invocando las funciones adecuadas de la API del mismo.

```
if (($SESSION[_ERRORS])) {
$actionForward = $actionMapping->get('Error');
}
```

```
else {
bd_cambiarValor ($valorid, $titulo, $nivel, $estado, $temaforo,
$descripcion, $ampliacion);
//fechas
bd_guardarFechaAprob($valorid, $diaapr, $mesapr, $anyoapr);
bd_guardarFechaInicio($valorid, $diaini, $mesini, $anyoini);
...
$actionForward = $actionMapping->get('OK');
}
return $actionForward;
```

Pasos 7 y 8: La acción consulta qué vista se debe mostrar a continuación y le pasa el control (ver mappings.php).

Paso 9: La vista obtiene a través del modelo los datos necesarios del valor y se los envía a la plantilla correspondiente.

```
$titulo = bd_extraerTituloValor($valorid);
$nivel = bd_extraerNivelValor($valorid);
$estado = bd_extraerEstadoValor($valorid);
$s->assign('titulo', $titulo);
$s->assign('nivel', $nivel);
$s->assign('estado', $estado);
$s->display('valor.tpl');
```

Paso 10: La plantilla es quien finalmente muestra los datos del valor al usuario de la aplicación.

```
<h1>{$titulo}
{if $nivel == 2}
- Esencial
{elseif $nivel == 1}
- De categoría
{/if}
</h1>
<p class="dates">{if $estado==1}Aprobado: {$fechaaprob} - {/if}Iniciado: {$fechainicio}</p>
<h3>Descripción</h3>
<p>{$descripcion}</p>
<h3>Ampliación</h3>
<p>{$ampliacion}</pC>
```

3.4. Software Libre

El compromiso de la espiral de valores con el software libre es triple:

1. La herramienta está escrita en el lenguaje PHP y utiliza MySQL como servidor de base de datos, ambos libres. Además se utilizan como código base dos paquetes de software libre, Phrame y Smarty.
2. Todo el código de la herramienta se ofrece como software libre a cualquier particular, organización o empresa interesada en realizar de él cualquier uso recogido en la licencia GPL.
3. Aunque no es un requisito general para la herramienta, la instalación realizada en espiraldevalores.org está soportada por el software de servidor Apache, ejecutándose en un sistema GNU/Linux, concretamente la distribución Debian GNU/Linux 3.0.

3.4.1. Software básico de servidor

Los programas que se ejecutan en el servidor, y que en definitiva componen esta aplicación web, están escritos en el lenguaje PHP, desarrollado por una gran comunidad de personas a través de Internet, a lo largo de casi diez años, de forma similar a como se desarrolló Linux, o el sistema operativo Debian.

En cuanto a las principales ventajas de PHP frente a otros lenguajes de servidor, son las siguientes:

1. Su rapidez de ejecución.
2. Es un lenguaje específicamente diseñado para realizar aplicaciones web, mientras que otros lenguajes son adaptaciones de lenguajes preexistentes, no pensados para la web.
3. El software necesario para ejecutar aplicaciones es software libre.

Los datos con los que trabaja la espiral de valores se almacenan utilizando MySQL, un sistema de gestión de bases de datos relacional licenciado como software libre. Se trata del gestor más usado en el mundo del software libre, debido a su gran rapidez, facilidad de instalación y de uso, y a que existen infinidad de librerías y otras herramientas que permiten utilizarlo desde gran cantidad de lenguajes de programación.

Enumeramos brevemente los puntos fuertes de este gestor:

1. La velocidad a la hora de realizar operaciones, manteniendo un bajo consumo de recursos de máquina.
2. La proliferación y facilidad de uso de las utilidades de administración.
3. Gran seguridad, muy poca probabilidad de corromper los datos.

3.4.2. Software para MVC y plantillas

Además de PHP y MySQL, se han utilizado los siguientes paquetes de software libre, como base para el desarrollo de la espiral de valores, según las técnicas explicadas:

1. Phrame (<http://phrame.sourceforge.net/>): Software que proporciona un marco de trabajo para la metodología MVC. Contiene un componente controlador, a completar con el código de las distintas acciones que tengan lugar en la aplicación web. Permite trabajar con múltiples lenguajes de presentación (HTML, XSLT, etc.). Ofrece otras utilidades, como por ejemplo la posibilidad de separar el código de validación de formularios hacia ficheros específicos para ese fin.
2. Smarty (<http://smarty.php.net/>): Software que proporciona un motor de plantillas. Internamente trabaja con una caché que reduce al mínimo la pérdida de eficiencia que todo mecanismo de traducción introduce. Ofrece un lenguaje para construir expresiones complejas, lógica condicional y de iteración, funciones predefinidas, etc.

Por supuesto existen otras herramientas dentro del mundo del PHP que permiten trabajar con las técnicas expuestas. Hay ya un considerable número de motores de plantillas libres. No hay muchos paquetes libres para trabajar con MVC en PHP, quizá porque PHP ha sido considerado como un lenguaje sencillo de desarrollo rápido y no se ha prestado la suficiente atención a una buena organización del código de las aplicaciones. Pero poco a poco aparecen desarrollos en este sentido, como por ejemplo php.MVC (<http://phpmvc.net>).

El lenguaje PHP todavía queda lejos en estas materias de otros lenguajes, como Java, en el que hablar de MVC es hablar de Struts (<http://jakarta.apache.org/struts/index.html>), prácticamente un “estándar de facto”, un software maduro que sirve como base para desarrollar aplicaciones web según la metodología MVC. Incluso existen herramientas visuales que facilitan dichos desarrollos. A modo de curiosidad, decir que Phrame es prácticamente una copia de Struts para PHP.

4. Desarrollos futuros

La espiral de valores es una herramienta completa y funcional que ha resuelto un problema de una organización, y que con ligeras modificaciones puede servir para otros propósitos similares. Sin embargo existen algunas posibles líneas de desarrollo que podrían llegar a una herramienta verdaderamente potente, útil para gran variedad de fines distintos.

Una primera posibilidad es elevar el nivel de abstracción de la herramienta y llegar a obtener una metaherramienta que sirva como base para crear webs de conocimiento organizado, sin necesidad de modificar el código, simplemente especificando las características particulares de la web que queremos construir por medio de un interfaz de usuario. A partir de dicha metaherramienta podríamos obtener fácilmente una espiral de valores, pero también una web de recopilación y debate de obras de cultura, o de textos con procedimientos operativos de una empresa. Con características de participación como comentarios y puntuaciones, y un sistema de edición descentralizado (aspectos éstos ya implementados en la actual web).

Otra posibilidad es orientar el desarrollo futuro hacia una herramienta de democracia participativa orientada al consenso, integrando y automatizando todo lo relacionado con las votaciones, plazos, histórico de propuestas, etc., con flexibilidad para adaptar las normas de debate según las peculiaridades y deseos de una organización en cuestión. Dicha herramienta podría resultar muy útil a todo tipo de empresas, organismos públicos y asociaciones con necesidades de organizar distintos debates organizados por temas y de poder elaborar y posteriormente acceder a consensos parciales o definitivos.

Es difícil saber a ciencia cierta hacia dónde se dirigirá el desarrollo de la herramienta. Lo único seguro es que el seguimiento de unas técnicas orientadas a conseguir una buena organización del código hará más sencilla cualquier futura evolución que si la aplicación consistiera en una serie de ficheros PHP de tipo “spaghetti”.

5. Conclusión

En este trabajo hemos presentado una herramienta de software libre construida para resolver las necesidades de organización y participación de un colectivo concreto. También se ha esbozado cómo elevando el nivel abstracción o automatizando una serie de cuestiones, la herramienta podría ser útil para muchos más colectivos.

Se han introducido las distintas metodologías utilizadas en el desarrollo, que tienen como fin potenciar la organización, la facilidad de ampliación y mantenimiento, una separación clara de papeles en el equipo de trabajo, etc. Quizá la única desventaja a mencionar es la complejidad asociada a un desarrollo tan modular y con tantos niveles, pero esto realmente sólo es una desventaja para aplicaciones sencillas sin requisitos especiales de mantenimiento ni posibles ampliaciones en previsión. En casos así, es posible que no valga la pena complicarse tanto, pero en los demás casos (que son clara mayoría) esta complejidad inicial se rentabiliza con creces según pasa el tiempo de desarrollo y mantenimiento.