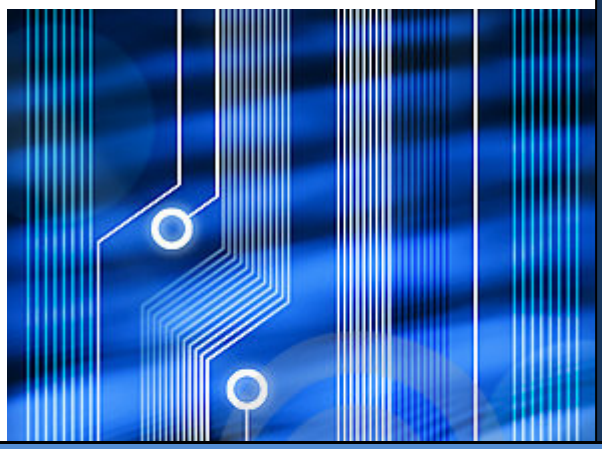
UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática etsinf



PROYECTO FIN DE CARRERA

Ingeniería Informática

ESTUDIO E IMPLANTACIÓN DE UNA PLATAFORMA DE EXPLOTACIÓN DE UN GESTOR DE CONTENIDO MULTIENTIDAD, PARA LOS PORTALES WEB MUNICIPALES DE LA PROVINCIA DE VALENCIA

ALEJANDRO MIRAGALL ARNAL

CURSO

2010-2011

ÍNDICE DE CONTENIDO

I. CONTEXTO DEL PROYECTO

I.1 PROYECTO PORTALES MUNICIPALES

El 22 de junio de 2007, se pone en vigor la ley 11/2007. En ella se reconoce el derecho del ciudadano a relacionarse con las Administraciones Públicas por medios electrónicos y regula aspectos básicos de la utilización de las tecnologías de la información en la actividad administrativa, en las relaciones entre las Administraciones Públicas, así como en las relaciones de los ciudadanos con las mismas con la finalidad de garantizar sus derechos. Así, esta ley exige que las distintas Administraciones Públicas garanticen la disponibilidad, el acceso, la integridad, la autenticidad, la confidencialidad y la conversación de los datos, informaciones y servicios que gestionen.

El adecuarse a esta nueva ley, incluye un esfuerzo económico de los que muchos municipios con menos recursos no son capaces de hacer frente. Para ello existe actualmente el sistema Pista Administración Local, en el que la mayoría de entidades locales (EELL) basan sus portales. Este sistema permite a estos municipios tener presencia en la red por un coste reducido. Ahora mismo este sistema ha sido superado en funcionalidad y estética por los desarrollos más actuales.

Así mismo, la Generalitat y las tres Diputaciones Valencianas, han desarrollado una Carpeta Ciudadana que desarrolla el derecho de los ciudadanos para relacionarse con las administraciones públicas municipales a través de medios electrónicos. Esta plataforma se apoya en el proyecto de administración local SOA de la diputación de Valencia, y en un buen número de otros desarrollos de e-Administración desarrollados por distintas administraciones, como el sellado de tiempo y verificación de firma electrónica de la Autoritat de Certificació de la Comunitat Valenciana, la plataforma unificada del perfil del contratante, el registro electrónico cv-Registro, etc.

La Administración General del Estado, a través del Centro de Transferencia de Tecnología, proporciona el mecanismo de @firma, así como un servicio de portafirmas y valija electrónica. También proporciona a través del plan Avanza, el sistema gestor de contenidos LocalWeb y el sistema de información geográfica LocalGIS.

La diputación de Valencia convoca un concurso para la implantación y puesta en marcha de todas estas tecnologías para las Administraciones Locales de la Provincia de Valencia, desarrollando para cada una de ellas su Sede Electrónica tal como se establece en la ley 11/2009 y se desarrolla en su reglamento.

En respuesta a este concurso, nace el **proyecto Portales Municipales** (ilus. 1) que pretende, como ya se ha comentado, la implantación y puesta en marcha de una plataforma de portales municipales y que incluye la migración y la actualización de todos los portales Pista de la provincia de Valencia.

Ilustración 1 - Logo Proyecto Portales Municipales

Esta plataforma va a contener cientos de sitios de los distintos municipios que van a tener en común todo un sistema base, el código del gestor de contenidos de nuestra plataforma debe ser multientidad. Por el peso de este proyecto y la necesidad de que estos servicios estén disponibles las 24 horas del día, la plataforma necesitará también un sistema de tolerancia a fallos.

Participamos en este proyecto y vamos a intentar garantizar tanto la multientidad como la estabilidad de la plataforma. Para ello vamos a buscar las herramientas necesarias para validar la plataforma y testearla ante distintas cantidades de tráfico, probando a su vez distintas configuraciones para así encontrar la más óptima.

La presente memoria del proyecto está estructurada en tres partes. En primer lugar, explicaremos tanto el contexto como los objetivos del proyecto, así como una descripción del concepto de gestor de contenido. En este caso especificaremos las

características de Drupal. En una segunda parte, definiremos la plataforma de explotación, para lo que definiremos el concepto de multientidad y estudiaremos las distintas alternativas posibles para su implementación en esta plataforma, seleccionando la más adecuada. En último lugar, realizaremos el análisis de rendimiento (profiling) de la plataforma, con el fin de encontrar la configuración hardware y software, óptima de la misma.

I.2 OBJETIVOS

Planteamos cuatro objetivos:

- **Estudiar, seleccionar e implantar un sistema de administración para la plataforma "Portales Municipales".**
- **Realizar manuales de usuario adecuados para el sistema de administración.**
- **Diseño de pruebas de rendimiento bajo varias configuraciones sobre el entorno de producción.**
- **Trabajar en el entorno profesional de la empresa.**

Además, el sistema multientidad debe ser accesible remotamente a través de una interfaz web y debe permitir administrar los portales de la plataforma de forma centralizada, permitiendo como mínimo las siguientes operaciones:

- **Creación o eliminación de portales.**
- **Backup de portales.**
- **Actualización de los portales (Control de versiones).**
- **Validación de Portales.**

Los portales de la plataforma están construidos en base al gestor de contenido (o CMS) Drupal. Este CMS (Content Management System) es de código libre, está

desarrollado en PHP y es ampliamente usado en importantes webs de la administración pública de Estados Unidos.

Las pruebas de rendimiento, deben obtener datos del rendimiento del servidor web bajo varias configuraciones hardware y software. Ejemplos:

- **1,2 o 4 cores de procesamiento,**
- **con o sin acelerador PHP,**
- **con o sin diferentes tipos de sistemas de caché.**

En todas las pruebas usaremos, un servidor web Apache con base de datos MySQL, instaladoen una máquina virtual. El sistema de virtualización utilizado es VMware debido a su implantación en los ordenadores de la Diputación de Valencia.

I.3 GESTORES DE CONTENIDO Y DRUPAL

Para la realización de un proyecto tan extenso es necesario el uso de herramientas que faciliten la creación, la gestión, la administración y en general el mantenimiento de todos los sitios. A su vez deben ser amigables y fáciles de usar por personas que en la mayoría de los casos no van a tener demasiados conocimientos informáticos.

Para esto, hacemos uso de aplicaciones que nos facilitan todas estas funcionalidades, los gestores de contenido. En concreto, en el proyecto Portales Municipales utilizamos un gestor de contenido de software libre llamado Drupal ampliamente utilizado, debido a su gran cantidad de módulos y a la seguridad de su código.

A continuación comentamos el concepto de gestión de contenidos y detallamos el gestor utilizado en nuestra plataforma, Drupal (ilus.2).

Ilustración 2 - Logo Drupal

I.3.1 GESTORES DE CONTENIDO

La tarea de realizar una web es un trabajo complicado y muy laborioso si no se dispone de las herramientas apropiadas. En el pasado se utilizaban herramientas que facilitaban la tarea de creación de la web pero no estaban enfocadas a su mantenimiento. En los últimos años ha surgido el concepto de gestión de contenido. Estos programas nos ayudan a la hora de crear, pero sobre todo a la hora de mantener nuestra web y ahorran mucho tiempo a los administradores web.

Un gestor de contenido o CMS es un programa que nos suministra una estructura de soporte (Framework) para la creación y administración de contenidos, en este caso, contenido web.

Consiste principalmente en una interfaz gráfica de usuario que controla una o varias bases de datos donde se aloja el contenido de nuestro sitio. El CMS permite aislar el contenido del diseño. Esto nos da la posibilidad de modificar o crear diseño y contenido de forma independiente, haciéndonos el trabajo más rápido y sencillo.

James Robertson propone una división de la funcionalidad de los sistemas de gestión de contenidos en cuatro categorías:

- **Creación de contenido.**
- **Gestión de contenido.**

- **Publicación.**

- **Presentación.**

Antes de utilizar un CMS deberíamos preguntarnos si realmente es necesario para nuestro proyecto pues va a acarrear el uso de código innecesario, y si no vamos a aprovechar las funcionalidades extra que nos proporciona sólo nos va a perjudicar en rendimiento.

En cualquier caso, la flexibilidad y escalabilidad que permiten estos sistemas, justifican su uso en la mayoría de los casos.

Estos son los factores más importantes que hacen necesario el uso de un CMS:

- **Inclusión de nuevas funcionalidades en la web.** Esta opción puede implicar la revisión de multitud de páginas y la generación del código que aporta las funcionalidades. Con un CMS eso puede ser tan simple como incluir un módulo realizado por terceros, sin que eso suponga muchos cambios en la web.

- **Mantenimiento de gran cantidad de páginas.** En una web con muchas páginas necesitamos un sistema para distribuir los trabajos de creación, edición y mantenimiento con permisos de acceso a las diferentes áreas. También tenemos que gestionar los metadatos de cada documento, las versiones, la publicación y caducidad de páginas y los enlaces rotos, entre otros aspectos.

- **Reutilización de objetos o componentes.** Un CMS nos permite la recuperación y reutilización de páginas, documentos, y en general de cualquier objeto publicado o almacenado.

- **Páginas interactivas.** Las páginas estáticas llegan al usuario exactamente como están almacenadas en el servidor web. En cambio, las páginas dinámicas no existen en el servidor tal como se reciben en los navegadores, sino que se generan según las peticiones de los usuarios. De esta manera cuando por ejemplo utilizamos un buscador, el sistema genera una página con los resultados que no existían antes de la petición. Para conseguir esta interacción, los CMS

conectan con una base de datos que hace de repositorio central de todos los datos de la web.

- **Cambios del aspecto de la web.** Si no hay una buena separación entre contenido y presentación, un cambio de diseño puede comportar la revisión de muchas páginas para su adaptación. Los CMS facilitan los cambios con la utilización, por ejemplo, del estándar CSS (Cascading Style Sheets u hojas de estilo en cascada) con lo que conseguimos la independencia de presentación y contenido.

- **Consistencia de la web.** La consistencia en un web no quiere decir que todas las páginas sean iguales, sino que hay un orden (visual) en vez de caos. Un usuario percibe enseguida cuándo una página no es igual que el resto de las de la misma web por su aspecto, la disposición de los objetos o por los cambios en la forma de navegar. Estas diferencias provocan sensación de desorden y dan a entender que el sitio web no lo han diseñado profesionales. Los CMS pueden aplicar un mismo estilo en todas las páginas con el mencionado CSS, y aplicar una misma estructura mediante patrones de páginas.

- **Control de acceso**. Controlar el acceso a un sitio web no consiste simplemente en permitir la entrada al mismo, sino que comporta gestionar los diferentes permisos a cada área aplicados a grupos o individuos.

I.3.2 INTRODUCCIÓN A DRUPAL

Drupal es un sistema de gestión de contenido modular multipropósito y muy configurable que nos permite publicar artículos, imágenes, u otros archivos y servicios añadidos como foros, encuestas, votaciones, blogs. Además permite administrar usuarios y permisos. Drupal es un sistema dinámico: en lugar de almacenar sus contenidos en archivos estáticos en el sistema de ficheros del servidor de forma fija, el contenido textual de las páginas y otras configuraciones son almacenados en una base de datos y se editan utilizando un entorno Web.

Es un programa libre, con licencia GNU/GPL, escrito en PHP, desarrollado y mantenido por una activa comunidad de usuarios. Destaca por la calidad de su código y de las páginas generadas, el respeto de los estándares de la web, y un énfasis especial en la usabilidad y consistencia de todo el sistema.

El diseño de Drupal es especialmente idóneo para construir y gestionar comunidades en Internet. No obstante, su flexibilidad y adaptabilidad, así como la gran cantidad de módulos adicionales disponibles, hace que sea adecuado para realizar muchos tipos diferentes de sitio web.

Drupal es un gestor de contenido multipropósito que podemos usar para aplicaciones como por ejemplo:

- Portales comunitarios
- Foros de discusión
- Sitios web corporativos
- Aplicaciones de Intranet
- Sitios personales o blogs
- Aplicaciones de comercio electrónico
- Directorio de recursos
- Sitios de redes sociales

I.3.3 CARACTERÍSTICAS DE DRUPAL

Características Generales

- **Ayuda on-line.** Un robusto sistema de ayuda online y páginas de ayuda para los módulos del 'núcleo', tanto para usuarios como para administradores.

- **Búsqueda.** Todo el contenido en Drupal es totalmente indexado en tiempo real y se puede consultar en cualquier momento.

- **Código abierto.** El código fuente de Drupal está libremente disponible bajo los términos de la licencia GNU/GPL. Al contrario que otros sistemas de 'blogs' o de gestión de contenido propietarios, es posible extender o adaptar Drupal según las necesidades.

- **Módulos.** La comunidad de Drupal ha contribuido muchos módulos que proporcionan funcionalidades como 'página de categorías', autenticación mediante jabber, mensajes privados, bookmarks, etc.

- **Personalización.** Un robusto entorno de personalización está implementado en el núcleo de Drupal. Tanto el contenido como la presentación pueden ser individualizados de acuerdo las preferencias definidas por el usuario.

- **URLs amigables.** Drupal usa el mod_rewrite de Apache para crear URLs que son manejables por los usuarios y los motores de búsqueda.

Gestión de usuarios

- **Autenticación de usuarios.** Los usuarios se pueden registrar e iniciar sesión de forma local o utilizando un sistema de autenticación externo como Jabber, Blogger, LiveJournalo otro sitio Drupal. Para su uso en una intranet, Drupal se puede integrar con un servidor LDAP.

- **Permisos basados en roles.** Los administradores de Drupal no tienen que establecer permisos para cada usuario. En lugar de eso, pueden asignar permisos a un 'rol' y agrupar los usuarios por roles.

Gestión de contenido

- **Control de versiones.** El sistema de control de versiones de Drupal permite seguir y auditar totalmente las sucesivas actualizaciones del contenido: qué se ha cambiado, la hora y la fecha, quién lo ha cambiado, y más.

También permite mantener comentarios sobre los sucesivos cambios o deshacer los cambios recuperando una versión anterior.

- **Enlaces permanentes. (Permalinks)** Todo el contenido creado en Drupal tiene un enlace permanente asociado a él para que pueda ser enlazado externamente sin temor de que el enlace falle en el futuro.

- **Objetos de Contenido. (Nodos)** El contenido creado en Drupal es, funcionalmente, un objeto (Nodo). Esto permite un tratamiento uniforme de la información, como una misma cola de moderación para envíos de diferentes tipos, promocionar cualquiera de estos objetos a la página principal o permitir comentarios -o no- sobre cada objeto.

- **Plantillas. (Templates)** El sistema de temas de Drupal separa el contenido de la presentación permitiendo controlar o cambiar facilmente el aspecto del sitio web. Se pueden crear plantillas con HTML y/o con PHP.

- **Sindicación del contenido.** Drupal exporta el contenido en formato RDF/RSS para ser utilizado por otros sitios web. Esto permite que cualquiera con un 'Agregador de Noticias', tal como NetNewsWire o Radio UserLand visualice el contenido publicado en la web desde el escritorio.

Blogging

- **Agregador de noticias.** Drupal incluye un potente Agregador de Noticas para leer y publicar enlaces a noticias de otros sitios web. Incorpora un sistema de cache en la base de datos, con temporización configurable.

- **Soporte de Blogger.** API La API de Blogger permite que un sitio Drupal sea actualizado utilizando diversas herramientas, que pueden ser 'herramientas web' o 'herramientas de escritorio' que proporcionen un entorno de edición más manejable.

Plataforma

- **Independencia de la base de datos.** Aunque la mayor parte de las instalaciones de Drupal utilizan MySQL, existen otras opciones. Drupal incorpora una 'capa de abstracción de base de datos' que actualmente está implementada y mantenida para MySQL y PostgresSQL.

- **Multiplataforma.** Drupal ha sido diseñado desde el principio para ser multi-plataforma. Puede funcionar con Apache o Microsoft IIS como servidor web y en sistemas como Linux, BSD, Solaris, Windows y Mac OS X. Por otro lado, al estar implementado en PHP, es totalmente portable.

- **Múltiples idiomas y Localización.** Drupal está pensado para una audiencia internacional y proporciona opciones para crear un portal multilingüe. Todo el texto puede ser fácilmente traducido utilizando una interfaz web, importando traducciones existentes o integrando otras herramientas de traducción como GNU ettext.

Administración y Análisis

- **Administración via Web**. La administración y configuración del sistema se puede realizar enteramente con un navegador y no precisa de ningún software adicional.

- **Análisis, Seguimiento y Estadísticas.** Drupal puede mostrar en las páginas web de administración informes sobre referrals (enlaces entrantes), popularidad del contenido, o de cómo los usuarios navegan por el sitio.

- **Registros e Informes.** Toda la actividad y los sucesos del sistema son capturados en un 'registro de eventos', que puede ser visualizado por un administrador.

Características de comunidad

- **Comentarios enlazados.** Drupal proporciona un potente modelo de comentarios enlazados que posibilita seguir y participar fácilmente en la discusión sobre el comentario publicado. Los comentarios son jerárquicos, como en un grupo de noticias o un foro.

- **Encuestas.** Drupal incluye un módulo que permite a los administradores y/o usuarios crear encuestas on-line totalmente configurables.

- **Foros de discusión.** Drupal incorpora foros de discusión para crear sitios comunitarios vivos y dinámicos.

- **Libro Colaborativo.** Esta característica es única de Drupal y permite crear un proyecto o "libro" a ser escrito y que otros usuarios contribuyan contenido. El contenido se organiza en páginas cómodamente navegables.

Rendimiento y escalabilidad

- **Control de congestión.** Drupal incorpora un mecanismo de control de congestión que permite habilitar y deshabilitar determinados módulos o bloques dependiendo de la carga del servidor. Este mecanismo es totalmente configurable y ajustable.

- **Sistema de Cache.** El mecanismo de cache elimina consultas a la base de datos incrementando el rendimiento y reduciendo la carga del servidor.

I.3.4 ESTRUCTURA DE DRUPAL

Core (Núcleo)

El *core* de Drupal es la instalación básica de Drupal la cual puede opcionalmente ser extendida. Por defecto, el contenido de una página web Drupal puede ser modificado por un usuario registrado o por uno anónimo (a elección del administrador).

El *core* de Drupal también incluye un sistema de taxonomías que nos permite categorizar y etiquetar el contenido mediante el uso de palabras clave para un fácil acceso.

El *core* incluye también módulos que pueden ser activados por el administrador para extender sus funcionalidades.

Módulos

En Drupal podemos ampliar sus funcionalidades mediante extensiones llamadas módulos programados por su comunidad de usuarios. Entre los más importantes que no están incluidos en la distribución oficial, podemos destacar:

- **Views.** Esencialmente, es un constructor de consultas a base de datos inteligente. Permite generar las consultas, ejecutarlas y mostrar los resultados en vistas personalizadas.
- **Content Construction Kit (CCK).** Permite al usuario generar nuevos tipos de contenido personalizados.
- **Pathauto.** De forma automática genera rutas amigables a los contenidos de Drupal.
- **FileField.** Módulo que mejora el sistema de subida de archivos del *core* de Drupal. Entre otras funciones permite la subida masiva de ficheros.
- **Administration menu.** Añade a la interfaz gráfica de administración una barra de menú con accesos directos a las principales funcionalidades de Drupal.
- **ImageField.** Módulo similar a filefield pero centrado en la subida a la web de ficheros de imagen.

- **ImageAPI.** Mejoras para el tratamiento de imágenes en Drupal. Varios módulos importantes de Drupal son dependientes de este.

- **ImageCache.** Mejora el sistema de cache de imágenes en disco.

Temas

Los temas, nos van a permitir modificar la apariencia de un sitio Drupal. Por defecto ya viene con varios temas pero en la práctica se suelen utilizar otros ya sea o bien desarrollados para el proyecto en cuestión o descargados de la red y creados por terceros. Están formados por un conjunto de hojas de estilo y fichero php.

II. SISTEMA DE EXPLOTACIÓN GESTOR DE CONTENIDOS MULTIENTIDAD

En esta parte, vamos a ver primero cómo funciona una aplicación multientidad y el porque necesitamos de esta característica en nuestra plataforma. Después nos vamos a centrar en las distintas formas de conseguir la multientidad en el CMS que hemos usado, Drupal.

En segundo lugar, veremos que es un sistema de explotación y nos centraremos en el que hemos utilizado, Aegir.

Hemos realizado también los manuales de instalación y de uso de este sistema.

II.1 MULTIENTIDAD

Una aplicación multientidad es aquella que con una única copia o instalación nos va a permitir atender las necesidades de distintas entidades, manteniendo un nivel de aislamiento suficiente entre cada entidad que es atendida (ilus. 3). En caso contrario, tendríamos que tener una instalación de la aplicación por cada entidad de la misma que necesitemos. (ilus. 4)

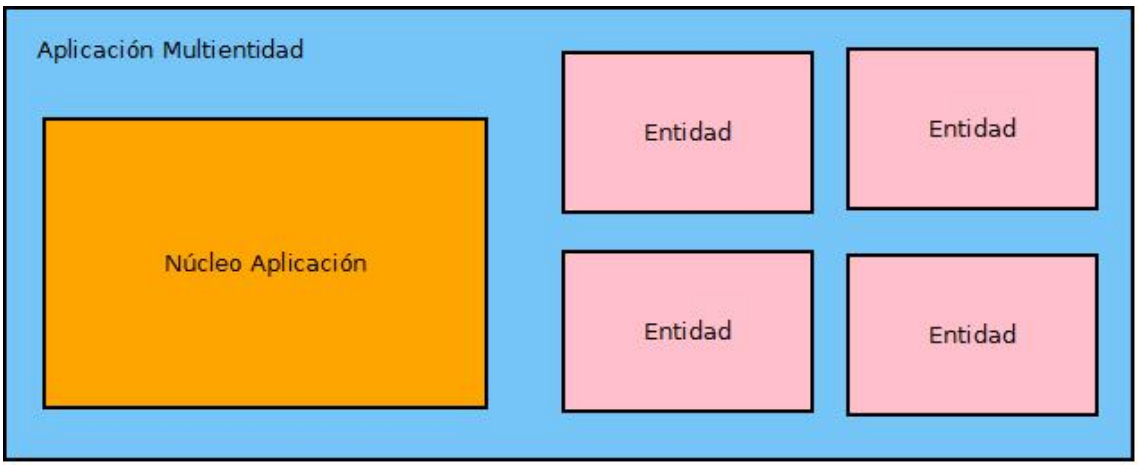


Ilustración 3 - Aplicación Multientidad

En el caso que nos ocupa estamos buscando una configuración del gestor de contenido Drupal multientidad que permita sobre una única instalación o plataforma servir los portales web de distintos municipios, cada uno de ellos con sus propios usuarios y permisos, contenidos independientes, y completamente aislados entre ellos.

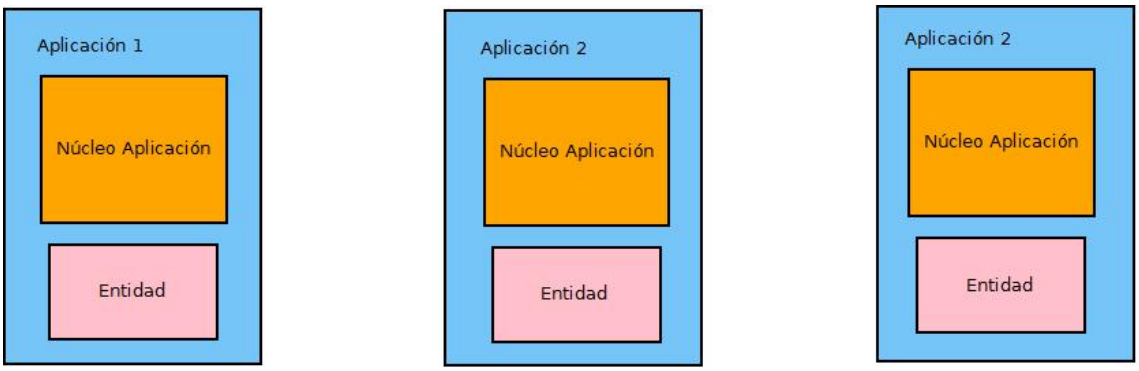


Ilustración 4 - Aplicación sin multientidad.

Vamos a ver las ventajas y desventajas que nos puede brindar una aplicación multientidad y que han hecho que busquemos esta característica para nuestra plataforma.

Ventajas:

- **Tiempo.**
- **Economía de licencias, y equipamiento.** El hecho de usar una sola instalación de los programas en algunos casos debería suponer un ahorro económico.
- **Centralización de las actividades de explotación.** Todas las entidades se configuran de forma centralizada.
- **Ahorro de recursos.**

Desventajas

- **Aumento de la complejidad.** Esta desventaja se puede presentar pero no es necesaria.

II.2 MULTIENTIDAD EN DRUPAL

II.2.1 INTRODUCCIÓN

En este apartado, veremos las diferentes alternativas que nos ofrece Drupal como plataforma multientidad y también analizaremos cuáles son los diferentes elementos que podemos compartir o no.

El propio *core* de este gestor de contenidos en su versión 6 ya permite la multientidad.

En la carpeta “sites” podemos crear carpetas con el nombre de dominio de nuestro nuevo sitio y Drupal las reconoce como instalaciones diferentes del mismo.

Sin embargo, hay algunas funcionalidades que el *core* por sí solo no nos centraliza: actualizaciones de los distintos módulos (en el caso de que impliquen actualizar la/s base/s de datos), la instalación de las diferentes entidades.

Es por esto que también se han desarrollado algunos módulos que facilitan estas labores. Aunque dependerá de las necesidades que tengamos el utilizarlos o no. Principalmente estos módulos los vamos a utilizar cuando tengamos una gran cantidad de entidades que administrar, como es el caso que nos ocupa.

También tendremos que analizar qué es lo que realmente queremos compartir, ya que hay que tener en cuenta que compartimos el código del *core* de Drupal entre todas las entidades que queramos pero también podemos compartir los temas, los módulos y/o la base de datos, según las necesidades.

En principio, lo que siempre vamos a querer compartir por motivos de espacio y de organización es el código base de Drupal, el cual va a ser idéntico para cualquiera de nuestros sitios. Pero también se puede dar el caso de que queramos compartir nuestra base de datos, ya sean todas las tablas o sólo algunas.

II.2.2 RESUMEN DE ALTERNATIVAS

En esta sección vamos a comentar los principales métodos para crear un multisitio con Drupal.

- Multientidad sólo usando el *core* de Drupal

- **Bases de datos independientes.** Para cada sitio web vamos a disponer de una base de datos distinta.

- **Base de datos compartida.** Se comparte una base de datos entre varios sitios web. Vamos a ver más adelante que hay tres formas diferentes de base de datos compartida.

- **Base de datos mixta.** Un hibrido entre las dos alternativas anteriores. Nos permite compartir datos entre sitios usando una base de datos común y a su vez mantener la independencia para otras tablas que queremos mantener aisladas.

- Multientidad con ayuda de módulos de terceros

- **Content-Manager.** Nos permite compartir contenido entre distintos módulos Drupal. Es decir vamos a poder crear una noticia en uno de nuestros sitios y tenerla accesible desde los demás. Tendremos una sola base de datos y habrá uso de prefijos

- **Multisite Manager.** Nos permite evitar tener que seguir el formulario de instalación cada vez que queramos crear una nueva base de datos. Este módulo va a poder trabajar tanto utilizando una sola base de datos, como utilizando bases de datos independientes para cada sitio.

- **Aegir.** Es el módulo más completo de todos. Nos permite crear varias plataformas Drupal con sus respectivos sitios, hacer migraciones…Va a requerir que tengamos permisos ssh en el servidor porque necesitamos acceso al terminal.

II.2.3 MULTIENTIDAD NATIVA EN DRUPAL

En esta sección, vamos a ver cómo es posible conseguir que Drupal pueda alojar en una misma instalación distintos sitios webs, es decir podemos hablar de instalación de Drupal multientidad o multisitio (*multisite*).

Para ello vamos a observar cual es la estructura de ficheros de una instalación de Drupal, pues es necesario para entender su funcionamiento (ilus. 5 y 6).

Podemos observar que del directorio de instalación cuelgan varias carpetas. Entre ellas tenemos las carpetas “modules” y “themes”, que son el código del núcleo de Drupal y sus temas por defecto.

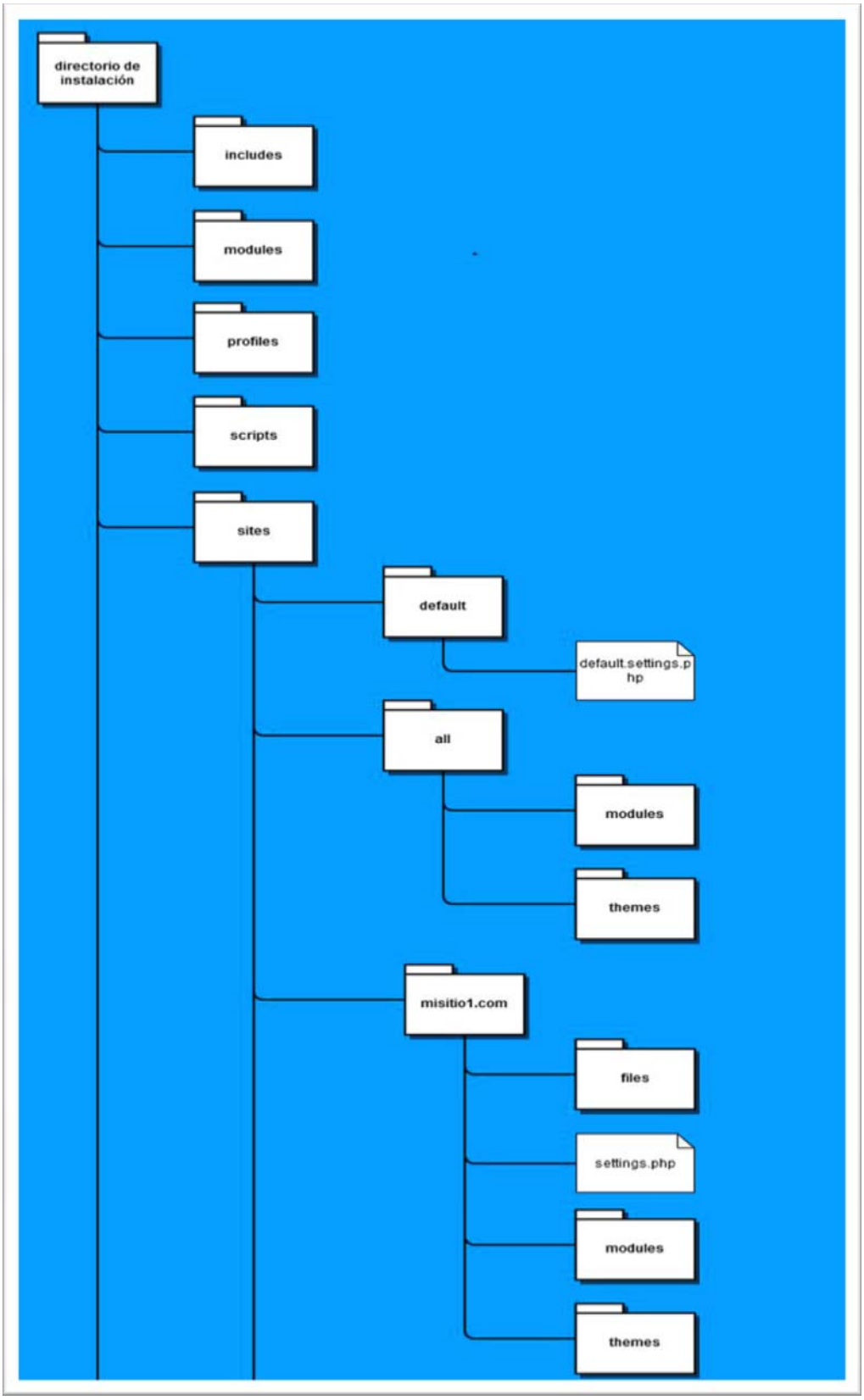


Ilustración 5 - Estructura directorios de Drupal 6 (1)

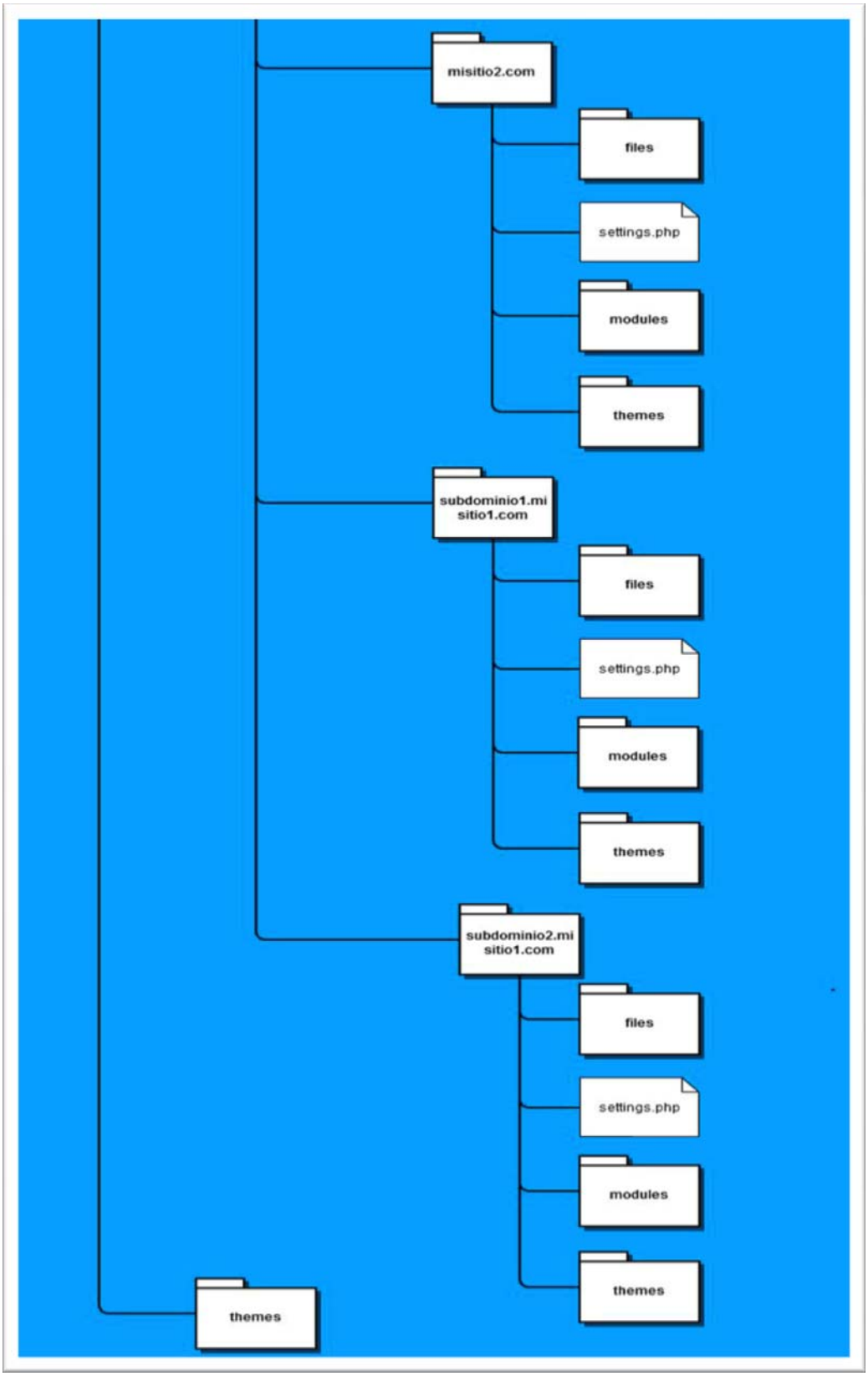


Ilustración 6 - Estructura de directorios de Drupal 6 (2)

Centrémonos ahora en la carpeta “sites” que es dónde vamos a tener que realizar modificaciones para alojar varios sitios web.

De esta carpeta, cuelgan la carpeta “all” y la carpeta “default”. En “default” es donde instalaríamos los módulos y temas del sitio en caso de que quisiésemos tener una solo sitio web, por lo que vamos a ignorarla.

En nuestro caso, tenemos que usar la carpeta “all”. En ella tenemos que instalar todos los módulos y temas que queremos compartir entre nuestras distintas webs. Es decir todo lo que instalemos en esta carpeta va a ser visible desde todos nuestros sitios webs.

El siguiente paso va a ser crear una carpeta por cada uno de los sitios que queramos crear, esta la renombraremos con el nombre de dominio o subdominio del mismo. Dentro de cada una de estas carpetas van a estar todos los archivos exclusivos a ese sitio en concreto, por ejemplo dentro de “misitio1.com” vamos a tener los módulos, los temas y los archivos (carpeta files) pertenecientes a ese sitio y estos no van a ser accesibles desde los demás.

También tenemos por cada carpeta un fichero “settings.php”, fichero de configuración del sitio que entre otras cosas nos va a permitir elegir la ruta de la base de datos que vayamos a usar.

Hasta aquí todos los tipos de instalación son similares y es ahora cuando debemos decidir cómo vamos a tratar el tema de la base de datos. Podemos decidir compartir todas las tablas de la base de datos, sólo algunas o mantener bases de datos independientes para cada uno de nuestros sitios.

Veamos qué pasos deberíamos seguir según la alternativa que elijamos.

Bases de datos distintas

En este caso, cada sitio web va a hacer uso de una base de datos distinta. (ilus. 7) Esto garantiza el aislamiento entre nuestros sitios y con según que configuraciones (varios)

permite un rendimiento superior para los casos en que nuestros sitios vayan a hacer un uso intensivo de base de datos.

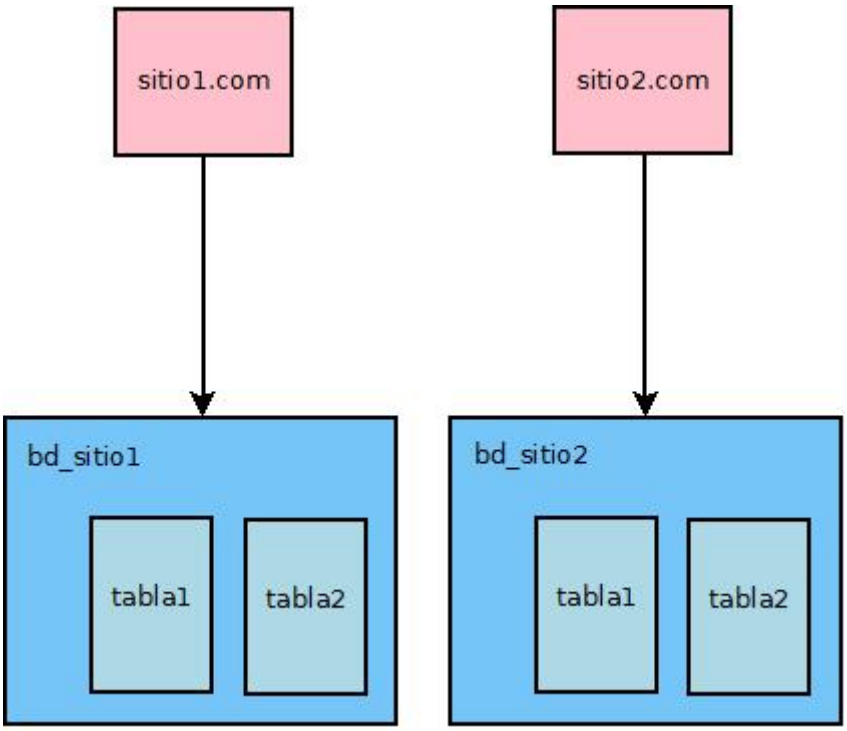


Ilustración 7 - Bases de datos independientes

Veamos que tenemos que modificar para conseguir esta configuración. Pongamos que tenemos para la base de datos del sitio1 el nombre de usuario “user_sitio1”, la contraseña “pwd_sitio1” y el nombre de la base de datos “bd_sitio1”. Para la base de datos del sitio2 tenemos el nombre de usuario “user_sitio2”, la contraseña “pwd_sitio2” y el nombre “bd_sitio2”.

El archivo “settings.php” de la carpeta “misitio1.com” tendrá las líneas:

```
$db_url = 'mysql://user_sitio1:pwd_sitio1@localhost/bd_sitio1';

$base_url = 'http://sitio1.com';
```

Y el de la carpeta “misitio2.com”

```
$db_url = 'mysql://user_sitio2:pwd_sitio2@localhost/bd_sitio2';
```

```
$base_url = 'http://sitio2.com';
```

Base de datos compartida

En este caso, elegimos usar una base de datos común para todos nuestros sitios (o para algunos). Vamos a tener tres opciones, la primera va a ser no compartir ninguna tabla mediante el uso de prefijos.

La segunda sería compartir todas las tablas entre los distintos sitios aunque cómo ya veremos no es recomendable.

Y la tercera nos va a permitir compartir sólo algunas tablas que queremos que sean comunes como por ejemplo las relativas a los usuarios o a las taxonomías (tipos de documento). Así conseguiremos por ejemplo que si un usuario se registra en uno de nuestros sitios, tenga acceso a todo el resto o que si creamos un tipo de documento en un sitio, no haya que repetir el proceso en otro sitio.

Vamos a aunar la segunda y tercera opción en un mismo apartado pues son muy similares.

Tablas distintas para cada sitio

Este método de instalación es el idóneo en caso de que dispongamos únicamente de una base de datos pero queramos tener un nivel de aislamiento alto.

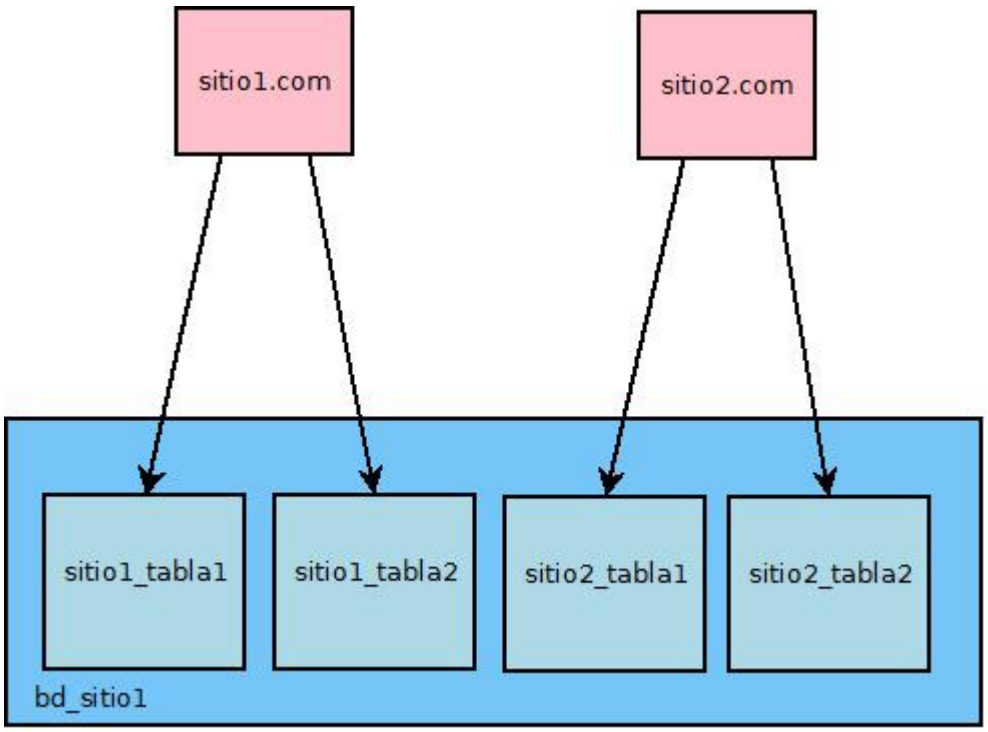


Ilustración 8 - Tablas independientes

En este caso, vamos a estructurar todo como antes pero a la hora de instalar cada sitio vamos a poner la misma base de datos para todos. Además un prefijo diferente para nuestras tablas en cada uno de las instalaciones. Por ejemplo en el sitio1 ponemos el prefijo "sitio1_" y en el sitio2 ponemos "sitio2_".

Con esto conseguimos que todas las tablas del sitio1 comiencen por un prefijo y las del sitio2 por otro, por lo tanto no se va a compartir información entre las 2 instalaciones pese a estar en la misma base de datos.

Los ficheros "settings.php" van a ser en este caso muy similares:

Para el sitio1 tendremos:

```
// en /sites/misitio1.com/settings.php

$db_url = 'mysqli://user1:contraseña@localhost/base_de_datos';

$db_prefix = array(
```

```
‘default’ =>  ‘sitio1_’);
```

Y para el sitio 2:

```
// en /sites/misitio2.com/settings.php

$db_url = ‘mysqli://user1:contraseña@localhost/base_de_datos’;

$db_prefix = array(

‘default’ =>  ‘sitio2_’);
```

Compartir algunas tablas entre los distintos sitios

Este tipo de instalación suele resultar útil cuando nuestros sitios pertenecen a la misma persona o están muy relacionados entre ellos. Utilizando este método podemos por ejemplo compartir la tabla de usuarios entre varios sitios, manteniendo el resto de tablas aisladas. (ilus. 9) De esta forma, un usuario que entre en el sitio1 y se registre, podría acceder con los mismo credenciales al sitio2.

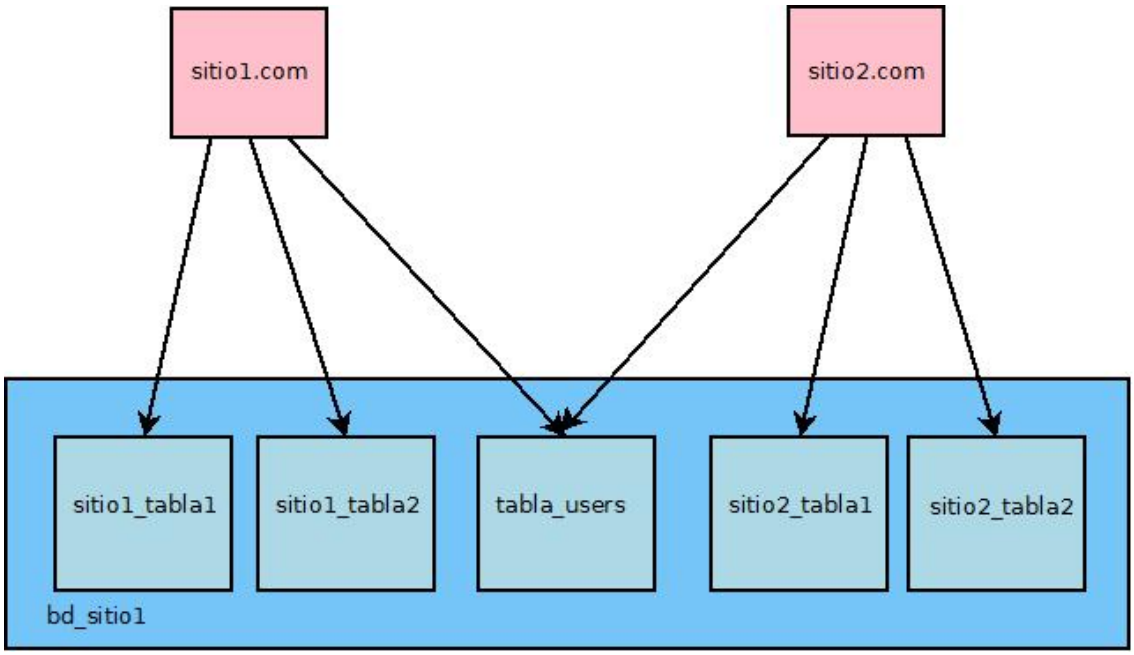


Ilustración 9 - Tablas compartidas

Va a ser muy similar al anterior pero en este caso vamos a tener que modificar antes de instalar los sitios “settings.php".

Para el sitio1 tendremos:

```
// en /sites/misitio1.com/settings.php

$db_url = ‘mysqli://user1:contraseña@localhost/base_de_datos’;

$db_prefix = array(

‘default’ =>  ‘sitio1_’,

‘users’ => ‘shared_’,

‘sessions’ => ‘shared_’,

     ‘role’ => ‘shared_’,

     ‘authmap’ => ‘shared_’,

);
```

Y para el sitio2:

```
// en /sites/misitio2.com/settings.php

$db_url = ‘mysqli://user1:contraseña@localhost/base_de_datos’;

$db_prefix = array(

‘default’ =>  ‘sitio2_’,

‘users’ => ‘shared_’,

‘sessions’ => ‘shared_’,

     ‘role’ => ‘shared_’,

     ‘authmap’ => ‘shared_’,

);
```

En este ejemplo como hemos comentado anteriormente, estamos compartiendo las tablas necesarias para tener usuarios comunes a los diferentes sitios, pero podríamos compartir otras tablas (incluso todas) si fuese necesario[1].

Base de datos mixta

Este método es una combinación de las alternativas que hemos visto.

Consiste en tener bases de datos distintas para cada sitio y a su vez tener otra base de datos más para compartir los usuarios de los sitios (u otras tablas). (ilus. 10)

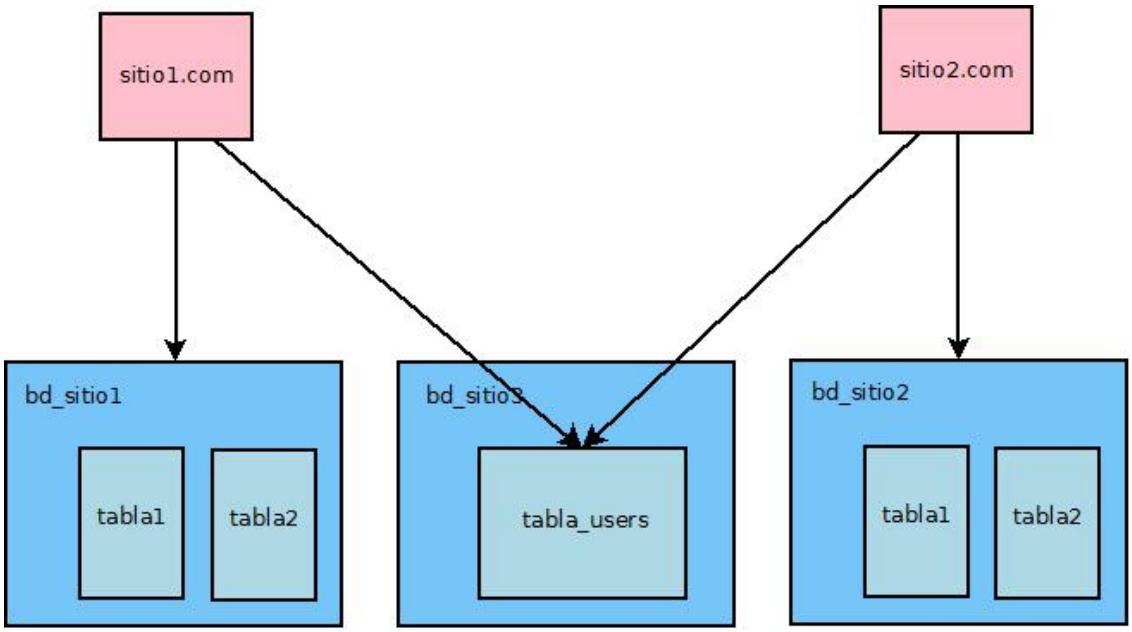


Ilustración 10 - Base de datos mixta

Por ejemplo tenemos la "base_de_datos1" que contiene las tablas de misitio1.com, la "base_de_datos2" que contiene las tablas de misitio2.com y una tercera base de datos ("base_de_datos_users") donde vamos a meter las tablas comunes a ambos sitios.

Otra vez tenemos que modificar los ficheros "settings.php":

[1] Por motivos de seguridad se recomienda compartir el mínimo número de tablas.

Para el sitio1 tendremos:

```
// en /sites/misitio1.com/settings.php

$db_url =
‘mysqli://user1:contraseña1@localhost/base_de_datos1’;

$db_prefix = array(

‘default’ =>  ‘’,

‘users’ => ‘base_de_datos_users.shared_’,

‘sessions’ => ‘base_de_datos_users.shared_’,

     ‘role’ => ‘base_de_datos_users.shared_’,

     ‘authmap’ => ‘base_de_datos_users.shared_’,

);
```

Y en el sitio2:

```
// en /sites/misitio2.com/settings.php

$db_url =
‘mysqli://user2:contraseña2@localhost/base_de_datos2’;

$db_prefix = array(

‘default’ =>  ‘’,

‘users’ => ‘base_de_datos_users.shared_’,

‘sessions’ => ‘base_de_datos_users.shared_’,

     ‘role’ => ‘base_de_datos_users.shared_’,

     ‘authmap’ => ‘base_de_datos_users.shared_’,
```

II.3 SISTEMA DE EXPLOTACIÓN

Hemos visto que Drupal en su propio núcleo da soporte a una instalación multientidad pero en nuestro caso esto no va a ser suficiente ya que este CMS por sí solo no dispone de un sistema integrado de explotación, un sistema que en la fase de producción nos facilite todas las tareas de administración y mantenimiento.

Buscamos un sistema capaz de permitirnos distintas tareas sobre todas nuestras entidades (sitios webs) desde una interfaz centralizada. Esto va a permitirnos mejorar la eficiencia y rentabilidad de nuestra plataforma.

Tareas requeridas

Las tareas que queremos que facilite el sistema de explotación son:

- **Creación de un nuevo sitio.**
- **Modificación de un sitio.**
- **Actualización de un sitio.**
- **Comprobación del estado de un sitio.**

Explorando por la web oficial vimos que existe un sistema de explotación para este CMS llamado Aegir que cumplía todas nuestras expectativas. Este sistema aporta todo lo que necesitamos en nuestra plataforma.

II.4 AEGIR

Aegir es un sistema distribuido administrador de instancias Drupal. (ilus.12) Nos va a permitir administrar de forma centralizada todas las plataformas Drupal que queramos las cuales a su vez van a poder alojar cientos de sitios cada una.

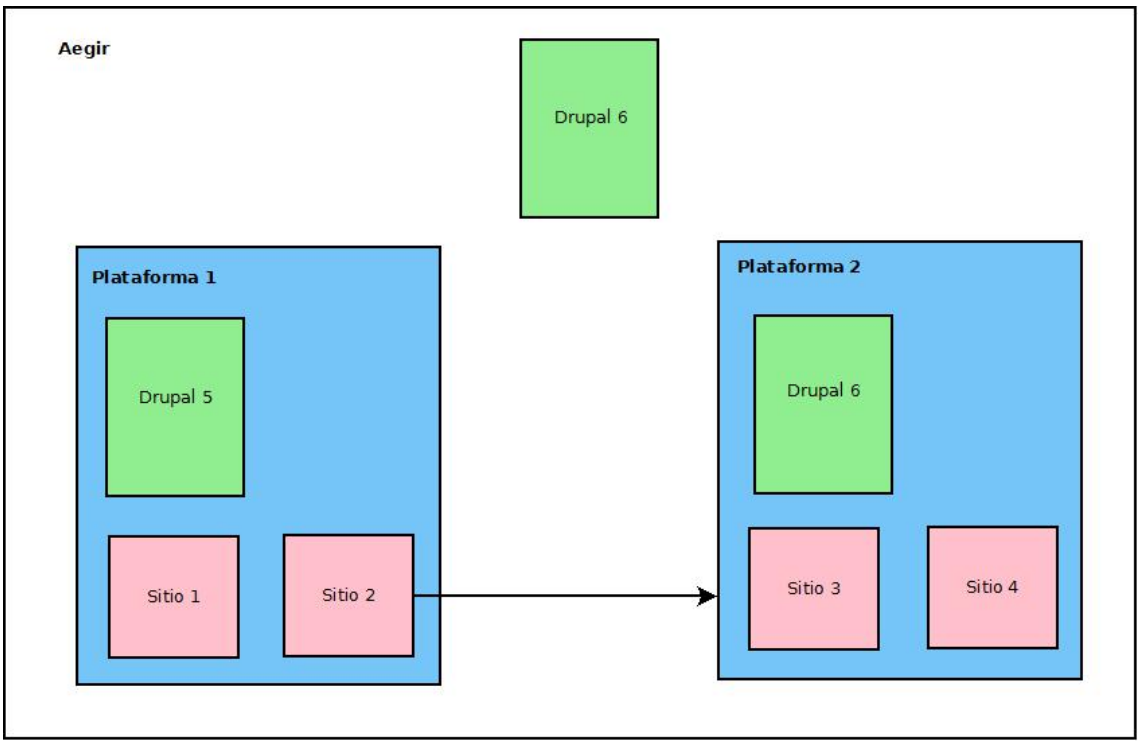


Ilustración 11 - Diagrama Aegir

Además en las últimas versiones, Aegir maneja plataformas alojadas en servidores distintos por lo que podemos hablar de sistema distribuido. Toda la administración se va a hacer desde un portal basado en Drupal. (ilus. 11)

Ilustración 12 - Logo de Aegir

II.4.1 UN POCO DE HISTORIA

Aegir es versión del original Hostmaster desarrollado por AdrianRossouw. La primera versión ha estado funcionando durante casi 4 años en http://brygh t.com durante casi 4 años, alojando miles de sitios, incluidos los de grandes clientes.

Originalmente, el sistema sólo conseguía una única instancia de Drupal, en un único servidor web con un único servidor de bases de datos y como los requisitos en funcionalidad crecieron (ejecutar al mismo tiempo varias versiones de Drupal con diferentes módulos cada uno y más tarde tener que administrar varios servidores) también lo hizo la complejidad del sistema.

Hubieron varias decisiones que acarrearon algunos problemas, tales como la elección de Python y PostgreSQL para el back-end y el no plantearlo como un proyecto puramente GPL , limitaron el número de desarrolladores que se sentían atraídos por el mismo. Otro fallo también podría ser la dificultad para instalarlo por los usuarios noveles.

El nuevo sistema fue diseñado con todos estos problemas en mente y se pueden ver los objetivos que tiene en el próximo apartado.

Aegir a día de hoy ya supera en funcionalidad ampliamente al antiguo HostMaster y está siendo utilizado para multitud de proyectos del mundo real.

II.4.2 OBJETIVOS DE AEGIR

- **Facilidad.** Tanto para el usuario como para el desarrollador.
    - Para el usuario: intentar que el sistema sea fácilmente instalable y que sea también de fácil manejo a la hora de administrarlo. El sistema busca auto-documentarse, por ejemplo proporcionando una ayuda ante fallos.
    - Para los desarrolladores se intenta mantener una implementación clara, simple y concisa y utilizar los mínimos recursos externos a Drupal. También se busca crear una documentación extensa para el programador.

- **Usabilidad.** El sistema busca servir para mantener sitios Drupal no solo en el momento de instalación sino durante todo su ciclo de vida.

- **Seguridad.** Seguridad entre los diversos sitios Drupal, por ejemplo el administrador de un sitio no debe poder modificar el contenido de otro sitio ajeno al suyo.

- **Sistema Distribuido.**

  - Como extensión del sistema de backup/restauración el sistema va a permitir mover sitios entre distintos servidores

  - También será capaz de cambiar de servidores de bases de datos, sencillamente desde el front-end de usuario.

- **Diagnósticos.**

  - El sistema va a generar completos logs de todo lo que ocurra en el mismo así como reportar cualquier error que ocurra mediante una notificación. Va a hacer verificaciones antes y después de realizar cualquier tarea. Antes, para comprobar que es capaz de realizarla y después para comprobar que ha sido bien realizada.

  - Cuando ocurre algún error en una tarea va a ser capaz de volver atrás dejando el sistema en un estado anterior al error, con el fin de proteger nuestros sitios.

- **Flexible.** El sistema busca ser tan flexible como el propio Drupal.

II.4.3 ESTRUCTURA DE AEGIR

Aegir principalmente está formado por 2 módulos (*hosting* y *provision*) y un perfil de instalación.

También podemos decir que Aegir tiene 2 componentes, el front-end y el back-end. En diseño de software el front-end es la parte del software que interactúa con el o

los usuarios y el back-end es la parte que procesa la entrada desde el front-end. En esta sección se va a describir de qué entidades están formadas estos componentes y su utilidad. (ilus. 13)

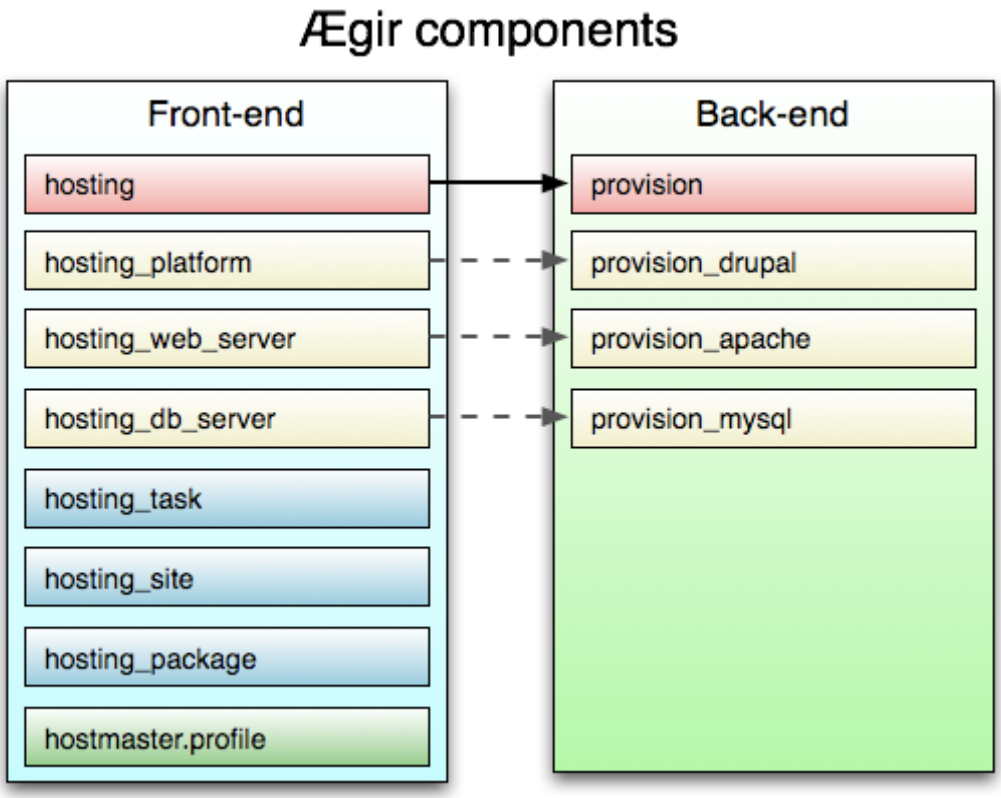


Ilustración 13 - Componentes de Aegir

Front-end

El Front-end es la interfaz de usuario que se usa para administrar nuestros sitios. Está constituido por el perfil de instalación hostmaster y el módulo hosting. Define un modelo de datos Drupal completo para manejar varios aspectos de la instalación.

- **Entidades.** Las entidades se utilizan principalmente en el Front-end para administrar la configuración y los datos. El sistema llama luego al back-end con las propiedades de estas entidades para que pasen a la línea de comandos.

  - **Client.** La persona o grupo que administra el sitio.

- **Site.** Una instancia de un sitio alojado. Contiene información relativa al sitio como por ejemplo el nombre de dominio, el servidor de base de datos, y la plataforma donde ha sido publicado. Un sitio además puede tener varios alias para el caso en que tenga que poder ser accedido desde varios nombres de dominio.

- **Task.** Es el mecanismo por el que el perfil hostmaster actualiza todos los cambios que ocurren en el sistema. Cada tarea actúa como un comando para el back-end, y contiene un log de los cambios que se realizan.

- **Platform.** Una plataforma no es más que un código base Drupal o similar donde desplegar nuestras webs. Podría ser cualquiera de las versiones estándar de Drupal u otras más específicas como Open Atrium, PressFlow, ProsePoint… Normalmente, una plataforma de otra se distingue por su perfil de instalación como podría ser el caso de Open Atrium pero también podría por módulos, librerías o temas extras.

- **Web server.** La máquina física donde nuestros sitios alojan sus ficheros. Cada servidor web aloja una o más plataformas. Si no piensas utilizar aegir de una forma distribuida no necesitas crear servidores web adicionales para tus propósitos.

- **Database server.** El servidor de base de datos donde nuestros sitios almacenan sus datos. Muchos servidores de base de datos y servidores web están alojados en la misma máquina pero por motivos de rendimiento puede requerirse servidores de bases de datos externos. Si no se pretende utilizar un servidor web externo esta entidad no es necesario utilizarla.

- **Package.** Un componente instalable de nuestro sitio web. Hostmaster tiene constancia de todos los módulos, temas y perfiles de instalación instalados en todo nuestro sistema aegir. Los usuarios de Aegir no pueden crear este tipo de nodos pero son generados automáticamente en las tareas de sincronización como pueden ser la tarea de validación de plataformas y sitios.

- **PackageRelease.** La versión de un paquete específico. Cada paquete de nuestro sistema puede
- **PackageInstance.** Es el sitio donde un packagerelease ha sido instalado. Un packagerelease puede haber sido instalado varios sitios de una misma plataforma. Y nuestros sitios pueden tener accesos a múltiples versiones de ese paquete. Esta entidad también registra que sitios tienen activados estos paquetes y que versión de ellos.

- **Tipos de tarea.**
  - **Validar Plataforma.** Verifica que cada plataforma esta bien instalada. Se crea automáticamente cuando añadimos una nueva plataforma para asegurar que es capaz de tener instalados nuevos sitios en ella. Opera en “Platforms”.
  - **Importar Sitios.** Importa sitios existentes a una plataforma. Opera en “Platforms”.
  - **Instalar Sitio.** Instala un nuevo sitio. Esta tarea se genera automáticamente cuando se crea un nodo de tipo sitio. Opera en “Sites”
  - **Backup Sitio.** Genera un backup de un sitio existente. Opera en “SItes”.
  - **Restaurar Backup.** Restaura un sitio al estado en el que estaba en su último backup.

Back-end

El Back-end es una interfaz de línea de comandos. Está diseñada para funcionar independientemente del front-end por lo que podemos tener múltiples back-ends tanto en un solo servidor como en varios.

El Back-end es capaz de crear nuevas bases de datos, configurar virtual hosts y resetear el servidor apache con el fin de que nuestras nuevas instalaciones tengan efecto.

Cuando las colas de tareas son procesadas por el Front-end estas son transformadas en comandos para el back-end. La comunicación del Back-end con el

Front-end es bastante sencilla y se base en el envío de strings que contienen información de estados y códigos de error.

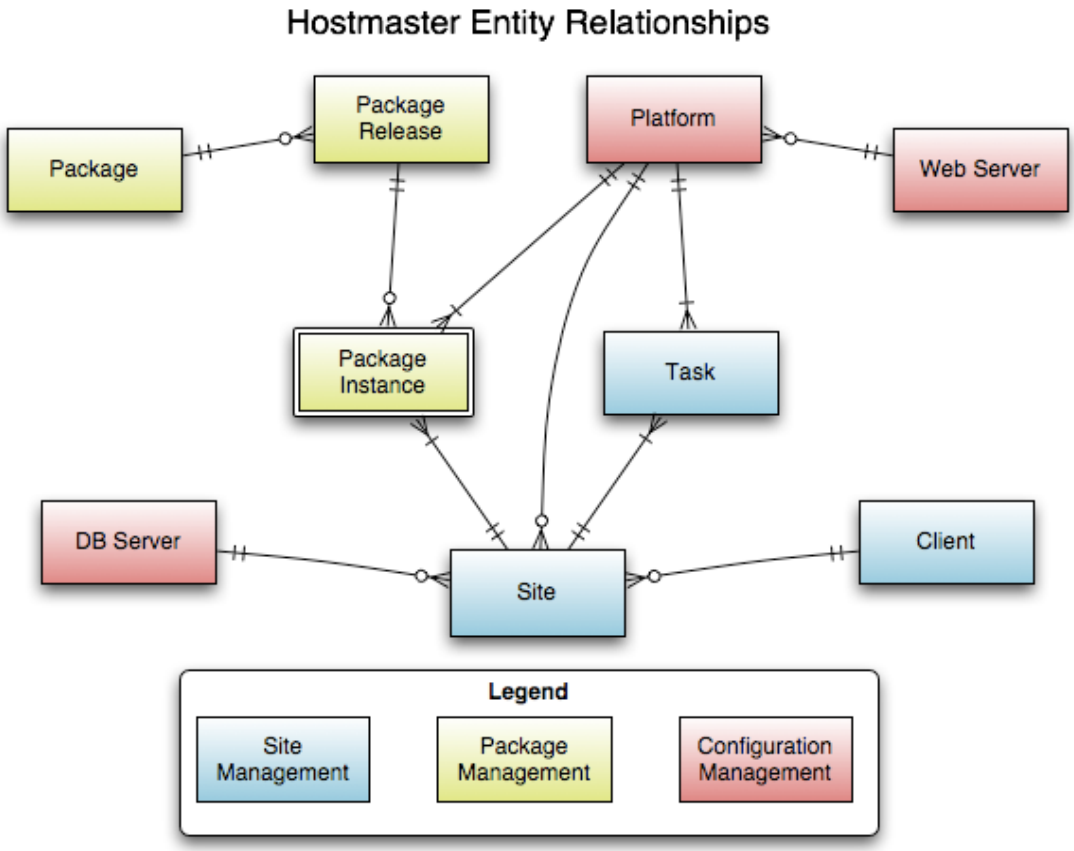


Ilustración 14 - Entidades presentes en Aegir

II.5. INSTALACIÓN, USO, Y AMPLIACIÓN DE LAS FUNCIONES DE EXPLOTACIÓN

Seguidamente vamos a describir tanto la instalación de Aegir cómo los pasos a seguir para realizar las funciones que necesitamos en nuestra plataforma de explotación.

II.5.1 INSTALACIÓN

Vamos a instalar la última versión de Aegir (Aegir 0.4 alpha8).

La instalación la realizaremos sobre un sistema operativo Linux, en este caso vamos a utilizar Debian 5. Cómo servidor web utilizamos Apache2 con php5.2[2] y como servidor de bases de datos, MySQL. No hay que olvidarse tampoco de instalar algún gestor de correo, como en nuestro caso postfix.

Descargar algunos paquetes necesarios

Lo primero que vamos a hacer es preparar nuestro servidor, para ello tendremos que descargar algunas aplicaciones mediante el comando apt.

```
apt-get install apache2 php5 php5-cli php5-mysql mysql-server
postfix
```

Esto nos va a instalar apache2, php5, MySQL Server y Postfix, ya que necesitamos un agente de transporte de correo (MTA[3])

[2] muy importante que a día de hoy no usemos PHP 5.3 porque vamos a tener problemas de compatibilidad tanto con Aegir alpha8 como con drupal 6.17 por lo tanto descartamos esta opción
[3] Mail Transfer Agent

Seguidamente tendremos que instalar otras utilidades para diversas tareas a lo largo de la instalación.

```
apt-get install sudogit-core unzip
```

Crear un nuevo usuario

Aegir requiere que sus scripts corran como un usuario no-root. Para asegurarse de que los permisos de los archivos son lo más seguros posibles y especialmente para asegurarse de que el servidor web no tiene la capacidad de modificar el código del sitio. Tendremos entonces que crear un nuevo usuario. Este, tendrá que ser miembro del grupo del servidor web, para poder configurar correctamente los permisos de los ficheros.

En nuestro caso, vamos a llamar a nuestro usuario “aegir” aunque podría ser cualquier otro. Su directorio home va a ser el directorio donde vamos a instalar aegir “/var/aegir” y el grupo del servidor web es “www-data”

```
adduser --system --group --home /var/aegiraegir

adduseraegirwww-data
```

Configurar nuestro servidor Apache

Vamos a tener que configurar ahora nuestro servidor web Apache. En un sistema Debian como en el que estamos tenemos que activar si no lo está el módulo mod_rewrite.

```
a2enmod rewrite
```

Vamos a crear un enlace lógico del archivo de configuración de Aegir en la carpeta /etc/apache/conf.d de Apache.

```
ln –s /var/aegir/config/apache.conf
/etc/apache2/conf.d/aegir.conf
```

Modificar archivo: /etc/sudoers:

```
aegir ALL=NOPASSWD: /usr/sbin/apache2ctl
```

Configurar nuestra Base de Datos

Vamos a realizar la configuración básica de nuestro servidor MySQL. Todos los comandos los vamos a tener que ejecutar como superusuario (root) de este. Es decir lo primero sería entrar en el gestor MySQL con el siguiente comando.

```
mysql –u root –p
```

Los nombres tanto de la base de datos y de los usuarios que vamos a crear se pueden cambiar por los que le convengan al administrador siempre que luego se tenga en cuenta cuando estemos rellenando el formulario web de instalación de Aegir.

Creamos una base de datos, en nuestro caso vamos a llamarla “aegir”.

```
CREATE DATABASE aegir;
```

Creamos un usuario con los privilegios suficientes para administrar esa base de datos, en nuestro caso vamos a llamarlo también “aegir”.

```
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, INDEX,
ALTER, \  CREATE TEMPORARY TABLES, LOCK TABLES ON aegir.* TO \

  'aegir'@'localhost' IDENTIFIED BY 'XXXXXXXX';
```

Aegir va a necesitar crear bases de datos nuevas por lo que necesitamos un usuario con permisos para ello. Creamos este usuario y lo llamamos “aegir_root”.

```
GRANT ALL PRIVILEGES ON *.* TO 'aegir_root'@'localhost'
IDENTIFIED \  BY 'XXXXXXXX' WITH GRANT OPTION;
```

Descargar ficheros de Aegir y ejecutar script

Vamos a descargar ahora de la siguiente dirección un script que nos permitirá instalar fácilmente Aegir. El enlace que mostramos aquí es el de la versión 0.4 alpha8 de Aegir. Lo vamos a descargar a la carpeta /var/aegir de nuestro servidor.

```
http://git.aegirproject.org/?p=provision.git;a=blob_plain;f=install.sh.txt;hb=provision-0.4-alpha8
```

Abriendo el script veremos que disponemos de varias variables que podemos modificar. Podemos por ejemplo descargar otra versión de aegir, otra versión de drush, cambiar la ruta de instalación (por defecto /var/aegir) o cambiar el dominio donde va a estar disponible nuestra página web de administración (aegir.example.com).

En este caso vamos a dejar intactas las variables por defecto y pasaremos directamente a la ejecución del script.

El script debe ser ejecutado por el usuario UNIX que creamos anteriormente llamado “aegir” y bajo el shell sh así que puesto que ahora mismo estamos identificados como supersusuarios del sistema tendremos que ejecutarlo con la siguiente orden.

```
su -s /bin/shaegir -c "sh install.sh.txt"
```

Este proceso de instalación hace modificaciones en la configuración de los vhosts de Apache por lo que después de ejecutarlo vamos a reiniciarlo.

```
/etc/init.d/apache2 restart
```

Instalación en la web

Abrimos nuestro explorador web y entramos en nuestra web de administración. Por defecto tendremos que entrar en

```
http://aegir.example.com /install.php
```

Aquí, lo primero que tendremos que hacer es elegir el perfil de instalación hostmaster y seguir paso por paso los formularios. Tendremos que introducir el nombre de la base de

datos los 2 usuarios que hemos creado para administrar nuestro servidor MySQL con sus respectivas contraseñas.

El propio formulario en un momento dado también nos pedirá que introduzcamos un par de instrucciones en la línea de comandos para completar la instalación.

Después de seguir estos formularios tendremos listo nuestro sistema Aegir para empezar a administrarlo.

II.5.2 GUÍA DE USO

Esta parte pretende informar de los pasos necesarios para administrar correctamente Aegir. Éste dispone de un menú bastante intuitivo para administrar nuestros sitios pero algunas de las operaciones las tendremos que hacer manualmente.

Explicaremos cómo realizar funciones tales como crear un sitio, migrar, renombrar (su dominio principal y sus alias), hacer un backup y restaurarlo (volver al estado del último backup), validar el sitio (comprobar el sitio).

Crear nueva plataforma

Como hemos visto previamente, una plataforma no es más que un código base de Drupal o similar con algun/os perfil/es de instalación, temas y módulos. Para instalar una vamos a tener que hacerlo en dos pasos, el primero es a través del Shell de Linux y el segundo ya de forma gráfica desde nuestro portal Aegir.

Preparar plataforma desde el servidor

1) Lo primero que debemos hacer es descargar este código base. Para ello disponemos de los módulos de Drush[4] que nos facilitan esta tarea, eso si debemos obligatoriamente hacerlo vía shell de linux.

[4] una "interfaz de linea comandos"(CLI) que nos permite realizar tareas rutinas de mantenimiento del sitio

2) Para tener una buena organización de nuestro sistema estaría bien crear en nuestra carpeta raíz de Aegir una carpeta “platforms” o “plataformas” donde vamos a tener todas las plataformas que creemos. Esto permitirá tener siempre localizados los ficheros de nuestros sitios en un futuro.

Todo este proceso se debería hacer identificado como el usuario aegir y no como root.

3) Por lo tanto nos logueamos con shell sh:

```
su –s /bin/shaegir
```

Creamos la carpeta “platforms” en /var/aegir.

```
mkdir /var/aegir/platforms
```

4) Ahora dentro de la carpeta platforms vamos a descargar el código base drupal, Drush nos facilita el trabajo.

Por ejemplo para descargar la versión 6.17 de drupal utilizaríamos el siguiente comando:

```
/var/aegir/drush/drush.php dl drupal-6.17
```

También podemos usar otros métodos propios de Linux como wget o CVS por ejemplo.

5) Lo siguiente ya es configurar este código base desde nuestra web de administración, por lo tanto entramos en nuestro portal Aegir y seguimos los siguientes pasos.

Configurar la plataforma desde Aegir

En el menú principal hacemos clic sobre Content Management >> Create Content >> Platform. (ilus. 15)

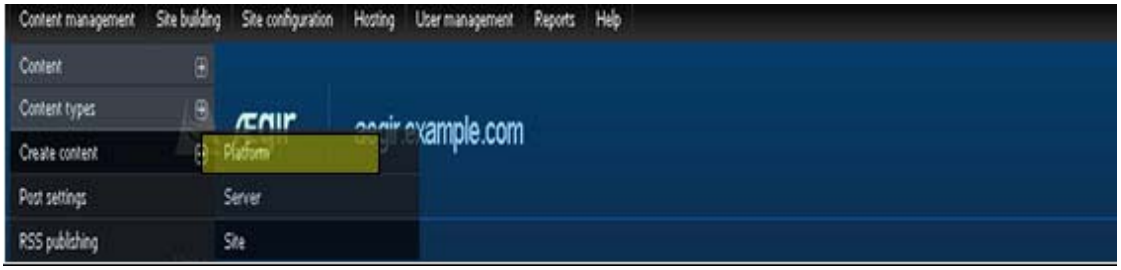


Ilustración 15 - Menú superior Aegir

Ahora nos aparece una nueva pantalla tenemos que darle un nombre a nuestra plataforma, y la ruta dentro de nuestro servidor dónde está el código base que acabamos de instalar. (si hemos seguido todos los pasos anteriores será "/var/aegir/platforms/drupal-6.17")

Podemos seleccionar si queremos que sea nuestra plataforma por defecto. Si hacemos clic sobre esta opción todos los sitios que creemos a partir de ahora de "forma rápida" (ver sección Crear nuevo Sitio) serán instalados en esta plataforma.

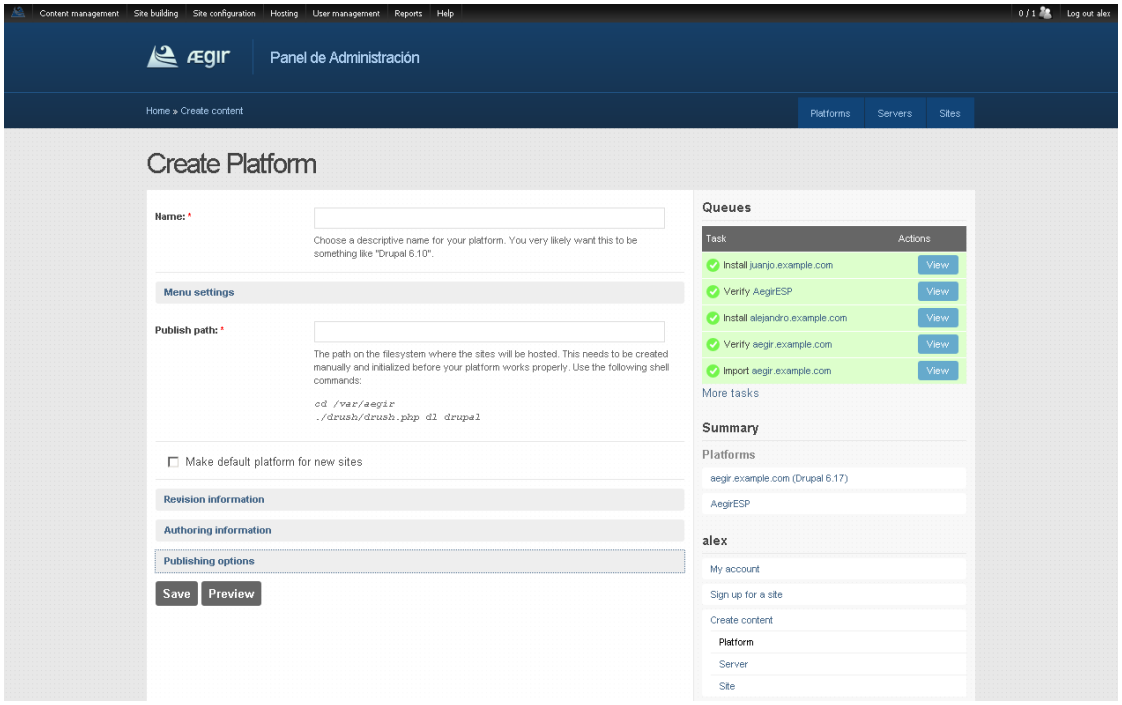


Ilustración 16 - Pantalla "Crear plataforma"

Luego también podemos modificar parámetros propios de todos los contenidos de Drupal. (Información de revisión, información de autoría u opciones de publicación)

Crear nuevo sitio

Tenemos 2 formas de crear un nuevo sitio, tenemos una forma rápida, en la que no podemos elegir la plataforma ni el perfil de instalación pero podemos asignar como cliente un cliente inexistente hasta el momento y otra forma más lenta en la que podremos personalizar más parámetros pero a cambio tendremos que haber creado antes el cliente.

Forma Rápida

1) Hacemos clic en “Sign-up for a Site” disponible en el menú lateral derecho. (ilus. 17)

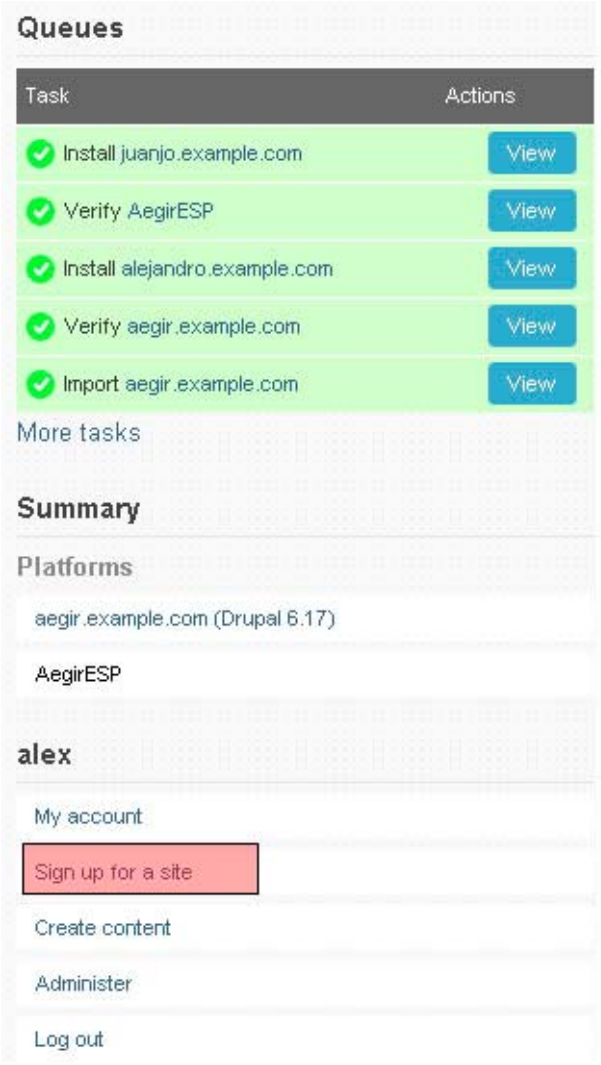


Ilustración 17 - Menú lateral Aegir

2) Accederemos a un menú dónde elegimos el nombre del dominio, el lenguaje (solo los disponibles en tu plataforma por defecto pues no podemos elegir otra), el mail del administrador (tendremos que confirmarlo), el nombre del cliente y su organización.

3) Hacemos clic en Sign up (ilus. 18)

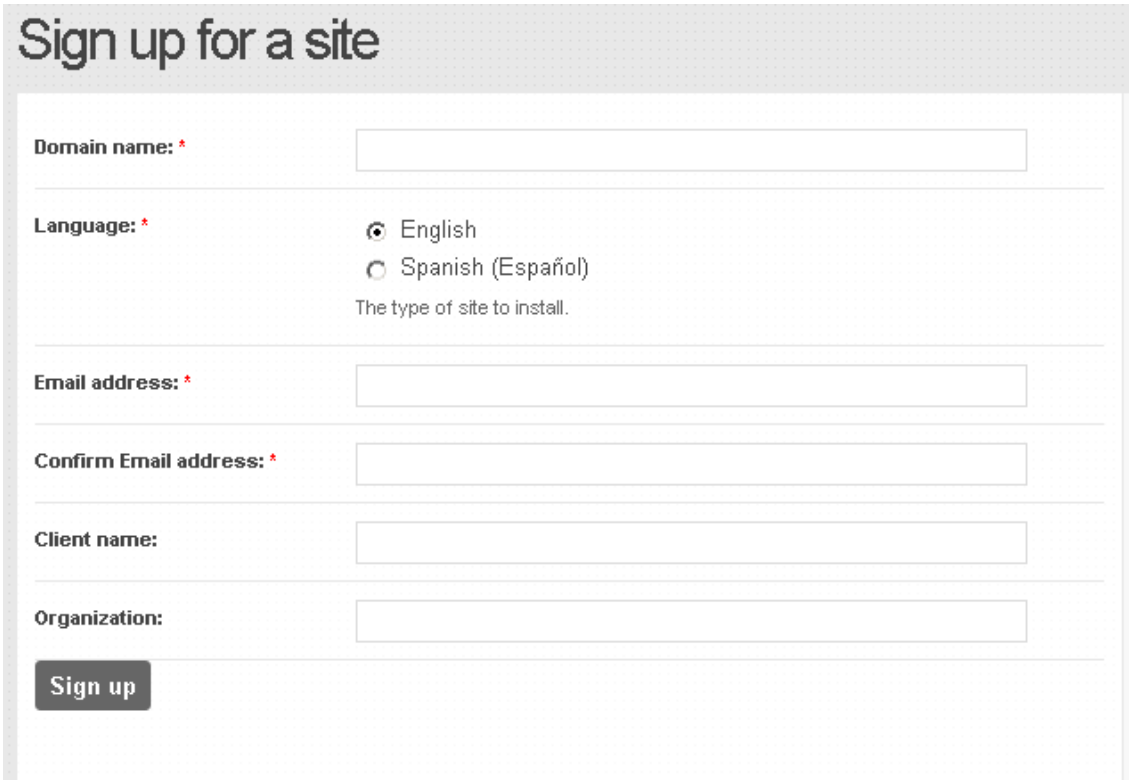


Ilustración 18 - Formulario nuevo sitio

Forma Personalizada

1) En el menú de administración, vamos a hacer clic en Content Management >> Createcontent >> CreateSite. (ilus. 19)

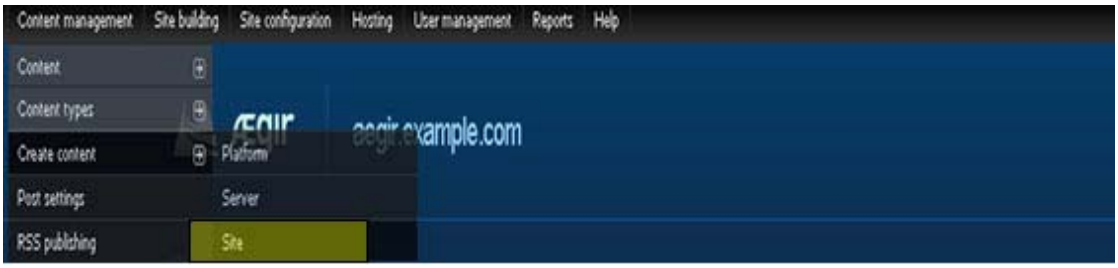


Ilustración 19 - Menú superior Aegir (2)

2) Primero tendremos que poner el dominio de nuestro nuevo sitio, previamente deberemos haber configurado el DNS del dominio, para que apunte a nuestro servidor.

3) Seguidamente, tenemos que elegir nuestro cliente. Este debemos haberlo creado antes.

4) Después elegimos tanto los diferentes alias que puede tener nuestro sitio, la plataforma en la que lo instalaremos, el idioma y el perfil de instalación.

5) Hacemos clic en Aceptar. (ilus. 20)

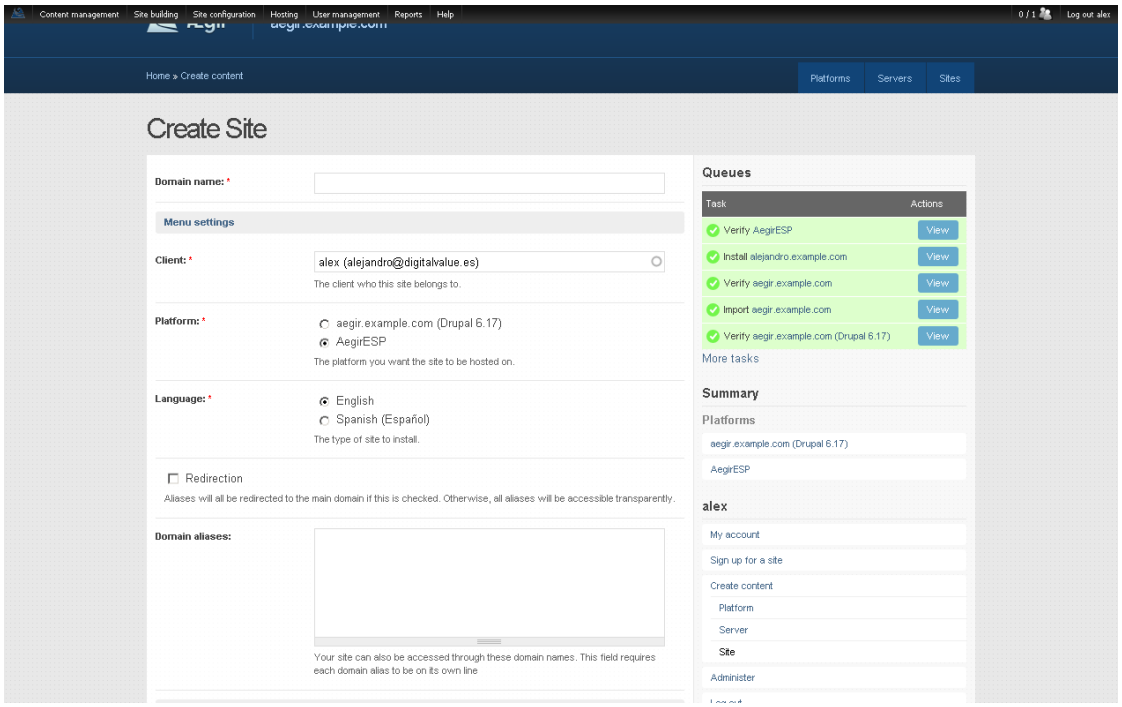


Ilustración 20 - Pantallazo Aegir (Crear Sitio)

6) Nuestro cliente recibirá un mail en su correo que le permitirá entrar en su nuevo sitio para poder empezar de inmediato a administrarlo.

Migración de un sitio

Migrar un sitio nos va a permitir mover nuestro sitio de una plataforma a otra. Con el fin normalmente de mantenerlo actualizado, por ejemplo para mejorar la seguridad de nuestros sitios. Por ejemplo, pongamos que tenemos 100 sitios web basados en Drupal alojados en nuestro servidor en la plataforma 6.15 y aparece la nueva versión de Drupal 6.17 que resuelve algunos problemas de seguridad, mediante la migración podremos con un solo clic migrar nuestras 100 webs a la nueva plataforma. Pero no solo sirve para

actualizar el código base de nuestras webs, también puede servir para dar acceso a nuevos módulos o temas. Por ejemplo en el caso de una empresa que tiene un tipo de cliente al que ofrece el servicio básico que solo da derecho al uso de los temas y los módulos básicos de Drupal y otro al que ofrece un servicio más caro llamado avanzado que tiene muchos más temas y módulos para elegir. Si el cliente decide mejorar su contrato podremos cambiarle de plataforma para que tenga más posibilidades con un solo clic.

Hacer un Backup

1) Antes de hacer una migración se debería siempre hacer un backup del sitio para poder recuperar la información de este en el caso de que la migración falle o nuestro sitio no funcione como esperamos en la nueva plataforma.

Los sitios los podemos migrar individualmente o conjuntamente, de forma muy similar. Esto veremos cómo hacerlo en esta misma guía de uso.

Verificar que la plataforma destino sea compatible

2) Vamos a la página principal de Aegir, seleccionamos el sitio que queremos migrar, o en el caso de que queramos migrar varios sitios de una plataforma, seleccionamos esta desde el menú plataformas situado en la barra lateral derecha. (ilus. 21)

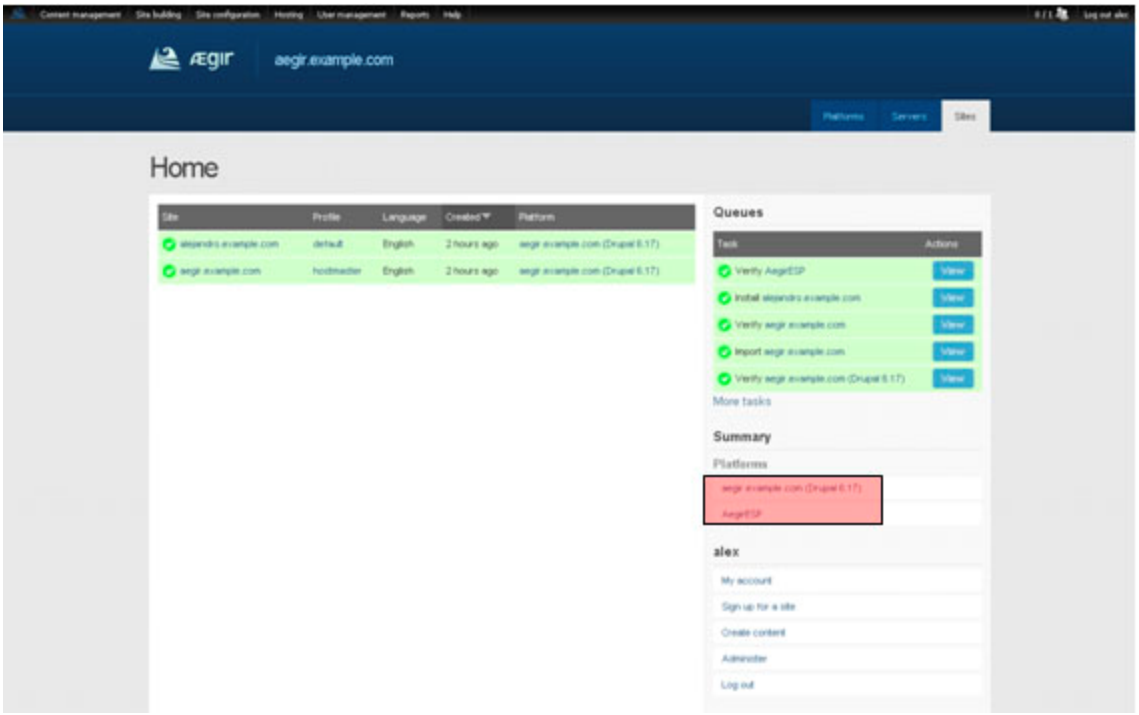


Ilustración 21 - Página principal Aegir

3) En la pantalla de administración del sitio o de la plataforma, en el apartado "Migrate" hacemos clic sobre "run". (ilus. 22)

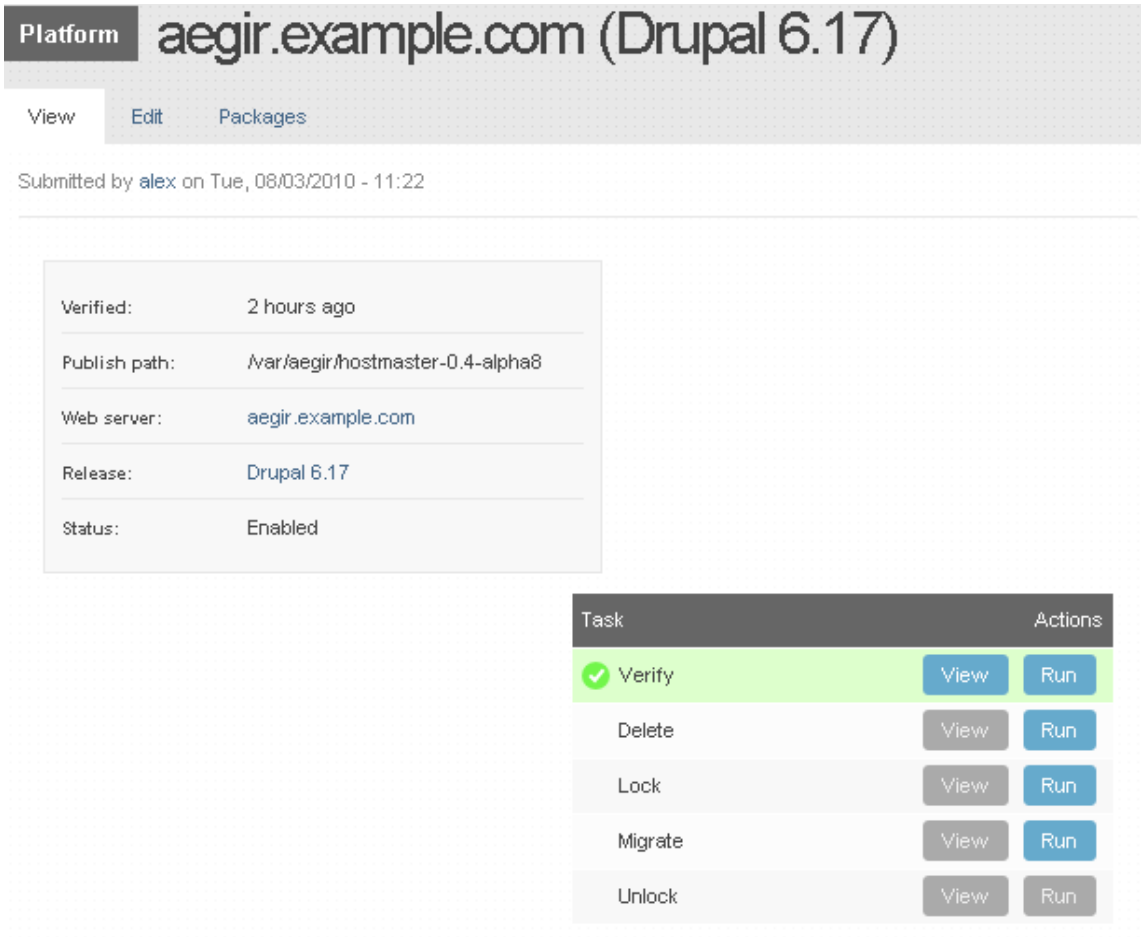


Ilustración 22 - Pantalla administración de un sitio en Aegir

4) Se nos mostrará un nuevo panel con las plataformas que disponemos para migrar nuestros sitios. Seleccionamos la plataforma destino y hacemos clic en el botón "Submit". (ilus. 23)

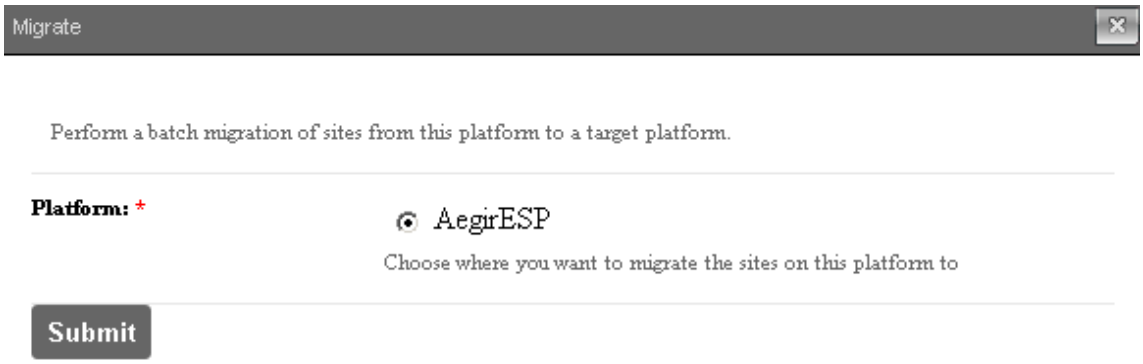


Ilustración 23 - Menú migrar

5) Se nos mostrará una pantalla que nos dirá que sitios se pueden migrar y cuáles no. Al lado de cada uno de ellos tendremos el link “compare” que nos permitirá saber que módulos están presentes, perdidos o anticuados en la plataforma destino con respecto al origen. (ilus. 24)

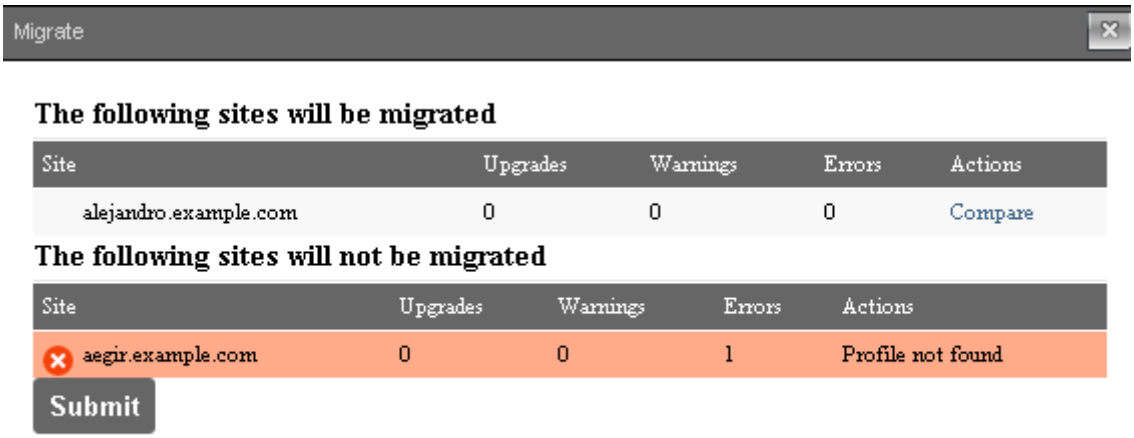


Ilustración 24 - Pantalla de confirmación

Si tenemos módulos en la carpeta modules de la carpeta de nuestro site es decir específicos de ese sitio en concreto (no presentes en /sites/all/modules) Aegir no los encontrará y les pondrá la etiqueta de perdidos (Missing) Cuando migremos estos paquetes serán copiados por lo que no tenemos porque instalarlos vía drush.

Tenemos que estar seguros de que todos los paquetes importantes estén presentes y actualizados antes de la migración para un correcto funcionamiento de nuestro sitio en la nueva plataforma.

Si falta algún paquete tendremos que actualizarlo desde una sesión shell. Estando en la carpeta de la plataforma a la que queremos migrar podremos actualizar o descargar los módulos que nos falten mediante drush:

```
/var/aegir/drush/drush.php dl module1 module2 themename 1
```

Podemos poner todos los módulos que deseemos en la misma línea. Esto descargará la última versión compatible de cada uno de los paquetes desde drupal.org.

6) Después de haber descargado estos módulos que nos faltaban tendremos que ir a la web de administración de nuestra plataforma destino y hacer clic en tabverify, esto actualizará la información de aegir sobre los modulos que hay disponibles en esta plataforma.

Migrar

7) Ahora ya sí que podemos hacer clic sobre el boton “Submit”. Aegir añadirá esta tarea a la cola de tareas y se procesará la próxima vez que se ejecute el cron.

Esto moverá nuestro directorio site/example.com al código base de nuestra plataforma destino y ejecutará el update.php de este sitio por nosotros.

Posibles problemas

Si hay algún problema lo primero sería borrar la caché de nuestro sitio. Desde nuestro sitio, entrando como administrador vamos a la sección mantenimiento y hacemos clic sobre el botón borrar caché.

Si tenemos problemas con el módulo imagecache, lo mejor sería entrar en el panel de configuración de image cache y seleccionar “rebuild all the images”

En el caso de que sigamos teniendo problemas con la migración de nuestro sitio siempre podemos restaurarlo, siempre que previamente hiciésemos el backup.

Hacer backups de los sitios

En Aegir 0.4 alpha8, solo podemos hacer el backups de nuestros sitios individualmente. Es decir si tenemos que migrar de una plataforma a otra todas los sitios o simplemente queremos hacer copias de seguridad de nuestros sitios, tendremos que hacer un backup sitio por sitio.

1) Entramos en la página inicial de nuestro portal Aegir, hacemos clic sobre el sitio del cual queremos hacer una copia de seguridad. Esto hará que accedamos a todas las opciones que tenemos sobre un sitio.

2) En el apartado Backup, hacemos clic sobre run. (ilus. 25)

3) Nos sale una ventana que nos pide confirmación y hacemos clic sobre el botón backup. La tarea se va a quedar en la cola de tareas de Aegir.

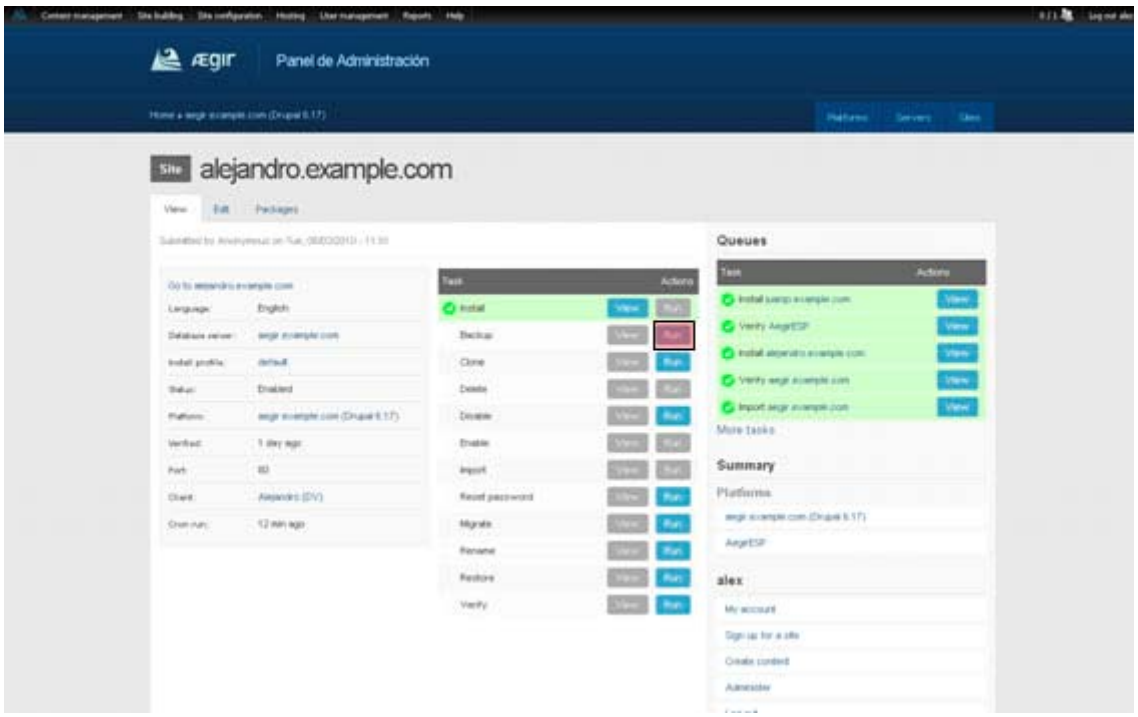


Ilustración 25 - Cola de tareas Aegir

Tendremos que esperar a que la tarea se ponga de color verde antes de hacer alguna migración o alguna otra tarea que ponga en compromiso nuestro sitio. Aegir tiene una cola de tareas y cada cierto tiempo ejecuta sus tareas pendientes.

Validar un sitio

Esta función comprueba que nuestro sitio funcione correctamente y no tenga ningún problema. También sirve para que en el caso de que hayamos añadido un módulo nuevo a nuestro sitio web o que lo hayamos activado Aegir tenga constancia de ello y nos lo muestre en las características del sitio.

1) Entramos en la página inicial de nuestro portal Aegir, hacemos clic sobre el sitio del cual queremos hacer una copia de seguridad. Esto hará que accedamos a todas las opciones que tenemos sobre un sitio.

2) En el apartado verify hacemos clic sobre run, nos aparece una ventana donde hacemos clic sobre verify. (ilus. 26)

Ilustración 26 - Menú tareas

3) Esperamos a que se ejecute el cron (que la tarea se ponga en color verde o en caso de error en rojo).

4) Hacemos clic sobre la tarea llamada “verify nuestro sitio” nos aparecerá una ventana similar que recopila datos del estado de nuestro sitio web. (ilus. 27)

Ilustración 27 - Resultado verificación de un sitio

Clonar un sitio

Esta función de Aegir nos permite clonar nuestro sitio en la plataforma que queramos y asignándole el dominio que queramos.

1) Entramos en la página de administración de nuestro sitio como en los procesos anteriores. Y hacemos clic sobre Run en la pestaña “Clone”. (ilus. 28).

Ilustración 28 - Menú tareas

2) Nos aparecerá una pantalla dónde podemos decidir en qué plataforma queremos clonar nuestro sitio. Tenemos que tener en cuenta que antes habría que comparar las plataformas. Cómo hemos visto anteriormente en la sección “Migración de un sitio”. Para este fin, al lado de cada plataforma nos aparece la opción “Compare Platforms”.

Podemos también cambiar el nombre del dominio de nuestro sitio por el que deseemos y añadir los alias de dominio que necesitemos. (ilus. 29)

Ilustración 29 - Formulario para clonar un sitio

III. VALIDACIÓN DE LA PLATAFORMA. RENDIMIENTO

En este apartado, principalmente vamos a buscar la configuración hardware y software óptima para nuestra plataforma. Para ello empezaremos describiendo los objetivos que buscamos, describiendo el montaje de prueba que hemos realizado, para luego tomar una cantidad de resultados y analizarlos.

III.1. OBJETIVOS

Tenemos que tener en cuenta que la plataforma "Portales Municipales" va a albergar un número creciente de páginas web. En un principio serán "sólo" 200 páginas pero en un futuro la necesidad puede aumentar. En esta parte del proyecto se persigue validar la configuración óptima de nuestra plataforma.

Buscamos que esta sea escalable pues no se sabe la cantidad de sitios que podrían haber en un futuro por lo que buscamos saber cómo evolucionará el rendimiento de esta ante cambios en el hardware y el software de la misma.

III.2. DESCRIPCIÓN DEL MONTAJE DE PRUEBA

III.2.1 HARDWARE

Para comprobar el correcto funcionamiento de nuestra plataforma de Portales Municipales hemos utilizado 2 servidores.

Uno, con el sistema operativo Linux CentOS 5 de 64 bits nos ha servido para realizar las pruebas de rendimiento. Mediante el programa ApacheBench nos ha permitido testear nuestro servidor web a diferentes cargas.

En el otro hemos instalado un sistema de virtualización ya que en la práctica contendrá diversas máquinas virtuales. Una de ellas es nuestro servidor web de pruebas. El sistema de virtualización empleado es VMwarevSphere 4.1 por lo tanto nuestro servidor físico tiene instalado ESXivSphere 4.1.

El hecho de utilizar una plataforma de virtualización nos ha permitido probar el rendimiento de nuestro servidor con diferentes configuraciones de hardware fácil y económicamente. Además en el entorno final de producción se usa este mismo sistema de virtualización.

	Máquina con Apache Benchmark **(Simulador de clientes)**	**Máquina con VMWarevSphere** **(Host del servidor web)**
Equipo	**Fabricante:** HP **Placa Base:** HP Proliant DL160 G6 **CPU Cores:** 4 CPU cores a 2.0 GHz **Tipo de procesador:** Intel® Xeon® E5504, 4MB de caché, 2000 MHz, QPI de 4.8 GT/s **Memoria RAM:** 4092 MB DDR3-1330 **Disco Duro:** 2 * 250 GB SATA2 en RAID 0+1	**Fabricante:** BULL **Modelo:** R410 **CPU Cores:** 4 CPU cores a 2.4 GHz **Tipo de procesador:** Intel® Xeon® CPU X3220 a 2.40 GHz, 2*2MB de cache, FSB 1066 **Memoria** RAM: 4092 MB DDR2-667 MHz **Disco Duro:** 250 GB SATA2 7200RPM
Sistema Operativo	Linux CentOS 5.3 - 64 bits	vSphereESXi 4.1 (host de vmWarevCenter)
Herramienta de pruebas	ApacheBenchVersion 2.0.40-dev	

Tabla 1 - Tabla de Hardware

Ahora pasamos a detallar el sistema operativo, software y configuración hardware por defecto de nuestro servidor virtual. Cabe mencionar que ya que hemos probado el servidor virtual con diferentes configuraciones. Se ha probado el rendimiento para 1, 2 y 4 cores.

	Máquina Virtual (Servidor Web)
Características	**CPU:** 2 CPU cores a 2.4 Ghz **Memoria RAM:** 2048 MB **Disco duro:** 50 GB
Sistema Operativo	Linux CentOS 5.3 – 64 bits
Software	**Servidor Web:** Apache 2.0 **Versión PHP:** 5.2.10 **Acelerador PHP:** eAccelerator v0.9.5.2

Tabla 2 - Configuración Máquina Virtual del Servidor

Durante las pruebas tuvimos que utilizar un tercer servidor (ver figura) para realizar pruebas, al que instalamos Windows 7 Ultimate de 64 bits y en el que instalamos la aplicación webstress server 7.2. Finalmente viendo que a grandes cantidades de tráfico el cliente era el factor limitante decidimos utilizar el otro servidor con webstress.

	Máquina de pruebas con WebStress Server
Características	**Fabricante:** Dell **Modelo:** Dell Poweredge 1950 **CPU:** 2 CPU's Intel Xeon 5160 con cores cada uno **Tipo Procesador:** Intel Xeon 5160, doble core a 3GHz, 2*2MB de memoria cache, FSB 1333MHz **Memoria RAM:** 12 GB DDR2-667
Sistema Operativo	Windows 7 Ultimate 64 bits
Software	Webstress Server 7.2

Tabla 3 - Configuración Máquina de Pruebas

Adjuntamos los resultados de benchmarking de los procesadores que hemos utilizado según la web cpubenchmark.net a 4 de octubre de 2010 comparados con los 10 mejores procesadores del momento. (ilus. 30, 31 y 32)

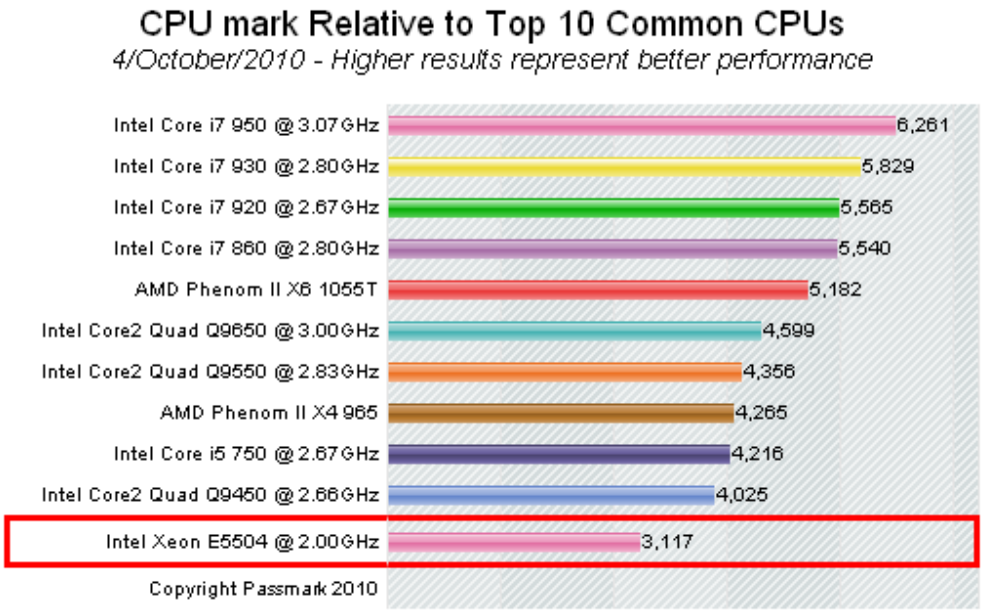


Ilustración 30 - Benchmarking Intel Xeon E5504

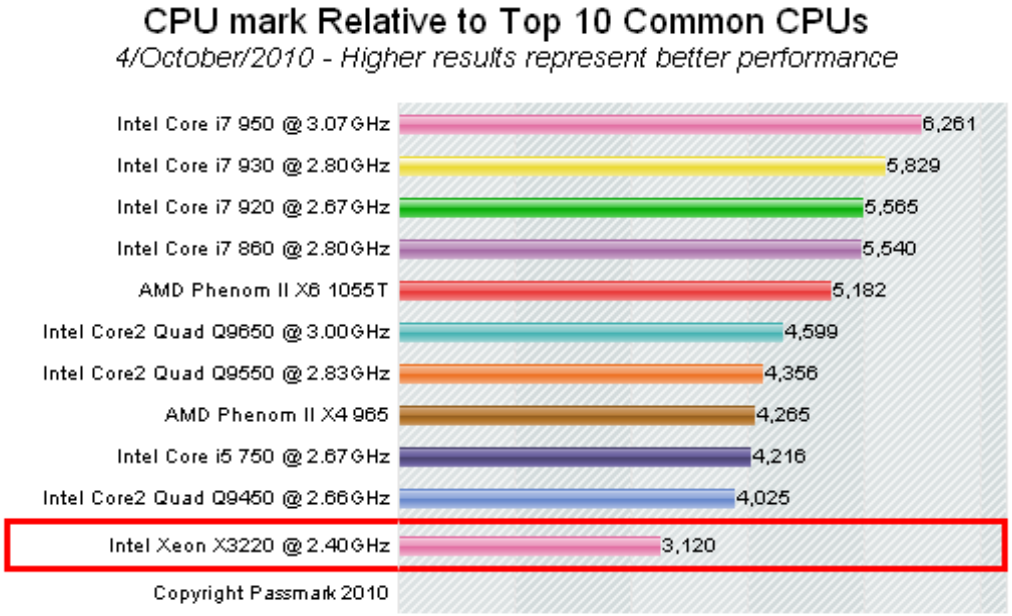


Ilustración 31 - Benchmarking Intel Xeon X3220

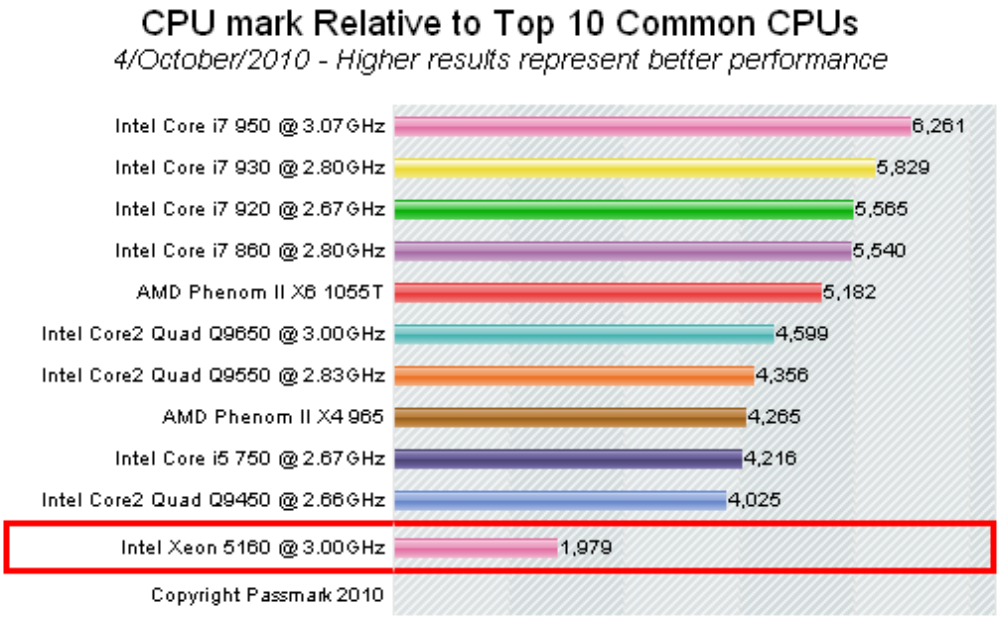


Ilustración 32 - Benchmarking Intel Xeon 5160

III.2.2 LA RED DE PRUEBAS

En un principio utilizamos la red de la oficina (red local ethernet 10 Gbps) para interconectar el servidor y las máquinas de pruebas, pero después de realizar diversas pruebas nos dimos cuenta que como cabía esperar la red era un factor limitante. Así que decidimos crear una pequeña red para interconectarlos. Primero cuando habían solo 2 máquinas utilizamos una conexión directa entre las 2 y después viendo que íbamos a añadir una tercera máquina añadimos a nuestra red un switch de 1 Gbps. Finalmente para las pruebas utilizamos la siguiente red ethernet de 1 Gbps. (ilus. 33)

VMware crea para cada interfaz de red un switch virtual para interconectar todas las máquinas virtuales.

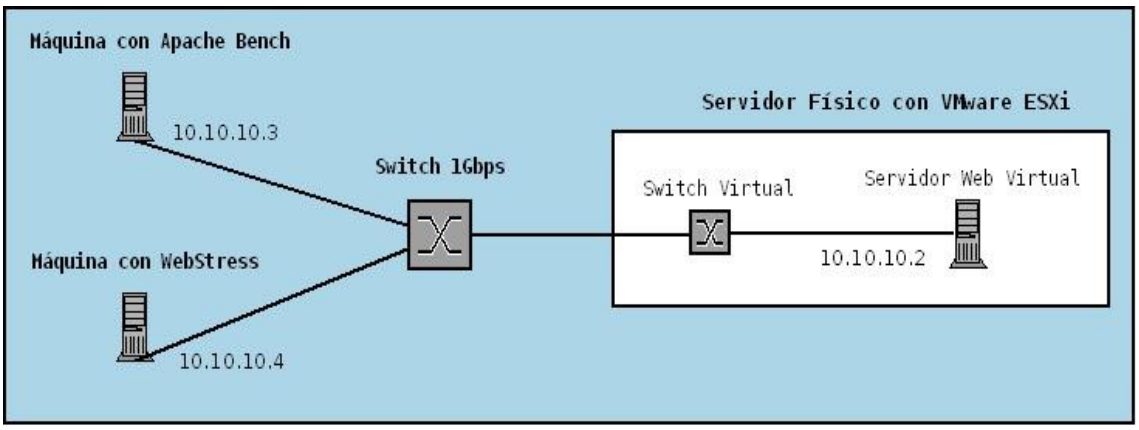


Ilustración 33 - Diagrama de la Red de pruebas definitiva

Las 3 máquinas contaban de dos interfaces de red, lo que nos ha permitido estar también a la vez conectadas a la red de la oficina con lo que las hemos podido administrar remotamente.

Viendo que para obtener resultados reales del rendimiento de nuestro servidor web necesitamos que la red tuviese un alto rendimiento ya que si no se transforma en el factor limitante de la prueba era necesario comprobar que la red llegaba a 1 Gbps de conexión. Para comprobarlo hemos montado un servidor ftp en nuestro servidor virtual y descargado un fichero pesado (unos 4 GB) desde las otras máquinas, y hemos visto que la velocidad de descarga era de 98 MB/s (784 Mbps), lo cual es más o menos lo esperado.

III.2.3 PLATAFORMA DE VIRTUALIZACIÓN – VMWARE VSPHERE

Como hemos comentado se ha utilizado la plataforma de virtualización Vmware vSphere 4.1, que nos permite la creación de múltiples máquinas virtuales en múltiples hosts. Esto es debido a que el entorno de producción final de nuestra plataforma "Portales Municipales" será el de los servidores de la diputación de Valencia. Allí utilizan esta plataforma con el fin de optimizar el rendimiento y la utilización de sus servidores.

Además como ya hemos comentado anteriormente el uso de una máquina virtual nos permite poder intercambiar el número de cores utilizados por esta sin necesidad de las complicaciones que nos daría el tener que hacerlo físicamente. Además de lo que supondría en coste de material.

Como en el momento de las pruebas no contábamos con la licencia completa, de la que si dispone la Diputación, hemos utilizado la versión de prueba que nos permite utilizar todas las funcionalidades existentes para este sistema durante un periodo limitado a 60 días. Esto incluye Tolerancia a Fallos, migración de máquinas en caliente, hosts y máquinas virtuales ilimitados...

Vmware vCenter 4.1 es la última versión de esta plataforma y nos permite administrar de forma centralizada gran cantidad de máquinas virtuales. Esta plataforma está compuesta de distintas aplicaciones, de las cuales hemos hecho uso principalmente de tres.

- Por un lado tenemos ESXi, que es el hypervisor autoinstalable, sistema operativo que se instala sobre los hosts y sobre los que corren las máquinas virtuales. Crea una capa de abstracción sobre el hardware de nuestras máquinas.

- Después contamos con Vmware vSphere vCenter que nos permite centralizar la administración de la plataforma, la creación de clusters virtuales, y la configuración en general de todos los hosts. Esta aplicación solo puede ser instalada en su versión 4.1 sobre un sistema operativo de 64 bits. Se recomienda además que este sobre una máquina virtual de uno de los hosts de la plataforma aunque también puede perfectamente estar en un equipo externo al sistema de virtualización.

- Finalmente, contamos con el cliente vSphere que nos aporta una interfaz gráfica para administrar el sistema, o bien conectándonos a cada uno de los hosts individualmente o bien directamente al servidor vCenter y así poder administrar de forma centralizada.

Se adjunta un ejemplo de cómo podría estar montada una plataforma Vmware vSphere con vCenter. (ilus. 34)

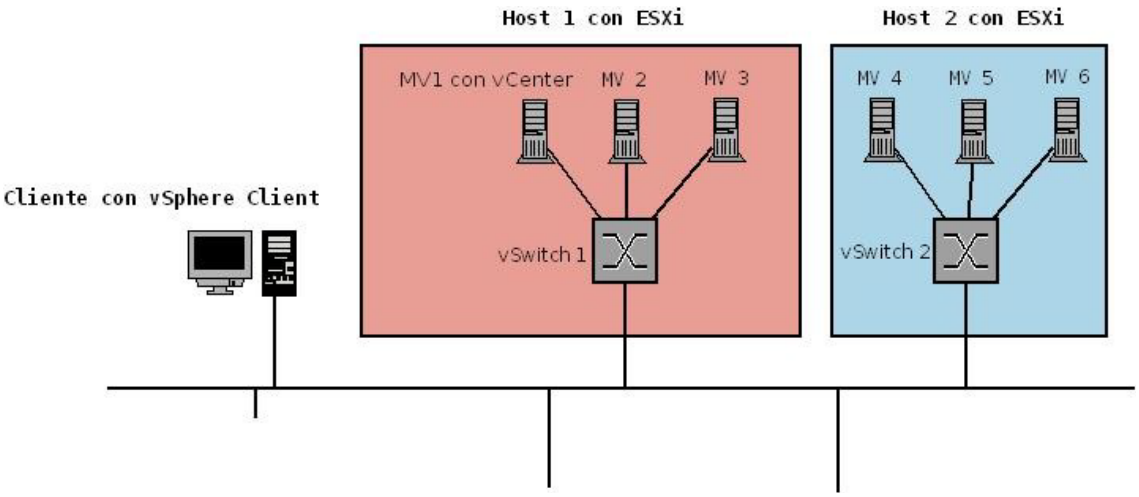


Ilustración 34 - Ejemplo de Plataforma VMWare vSphere vCenter 4.1

III.2.4 SOFTWARE SERVIDOR

En esta parte, hemos descrito todos el software empleado en nuestro servidor, sistema operativo, servidor web, de base de datos…

Sistema Operativo - Linux CentOS 5.3

En el servidor, hemos utilizado como sistema operativo Linux CentOS 5.3 (Community Enterprise Operating System) en su versión para procesadores de 64 bits. Esta distribución de Linux está basada en Red Hat Enterprise Linux y esta compilada por voluntarios a partir del código fuente liberado por Red Hat. A diferencia de este que es distribuido en formato CD-ROM o DVD-ROM a suscriptores, es completamente gratuito.

Apache 2.2.3

A su vez hemos instalado también Apache 2.2.3. Apache es un servidor web HTTP de código abierto disponible para la mayoría de plataformas. Es el más usado actualmente, por el hecho de ser una aplicación modular, de código abierto, multi-plataforma, extensible y popular, por lo que es fácil encontrar ayuda en la red.

Apache se puede optimizar facilmente mediante un fichero de configuración. De hecho, durante las pruebas, dependiendo de la configuración hardware y de software que hemos utilizado, hemos debido hacerlo. Para ello se modifica el fichero /etc/httpd/conf/httpd.conf

MySQL 5.0.77

Junto a Apache hemos necesitado también de un servidor de bases de datos. Para ello hemos utilizado MySQL, sistema de bases de datos relacionales, multihilo y multiusuario. MySQL se ofrece bajo licencia GNU GPL , a diferencia de Apache, donde el software es desarrollado por una comunidad pública y el copyright del código esta en posesión del autor individual, MySQL es patrocinado por una empresa privada (MySQL AB), que posee el copyright de la mayor parte del código. Esto hace que las empresas que quieran utilizarlo para un uso privativo deban comprar a la empresa una licencia especifica que les permita este uso. Existen APIs que permiten a aplicaciones escritas en gran cantidad de lenguajes comunicarse con MySQL. En nuestro caso utilizaremos la API de PHP. Este gestor de bases de datos es el usado por DRUPAL.

MySQL al igual que Apache consta de un fichero de configuración para modificar el comportamiento de este gestor de bases de datos según la configuración del equipo y/o las necesidades del sistema que vamos a alojar. Por defecto MySQL trae varias configuraciones por defecto intercambiables dependiendo del tamaño del servidor de bases de datos que vamos a utilizar. En el caso de nuestro servidor, hemos optado por usar para todas las pruebas la configuración para servidores muy grandes o enormes. (huge servers) Para ello re-nombramos myhuge.cnf y lo copiamos en la carpeta /etc sustituyendo así el que hay por defecto.

Añadimos al final de este documento, en la sección “Anexos” el fichero de configuración de mysql tal y como lo hemos utilizado para todas las pruebas.

PHP 5.2.10

Finalmente hemos instalado en el servidor virtual PHP 5.2.9. Aunque a día de hoy ya está disponible la versión 5.3 hemos instalado la 5.2 debido a problemas de

compatibilidad con DRUPAL el gestor de contenido que utilizan nuestras webs. PHP es un lenguaje de programación interpretado, diseñado para la creación de webs dinámicas. Aunque actualmente puede ser utilizado desde una interfaz de línea de comandos o en la creación de aplicaciones gráficas usando bibliotecas como Qt o GTK+, principalmente se utiliza como lenguaje interpretado del lado del servidor (server-side scripting)

PHP o PHP Hypertext Pre-procesor es un lenguaje multiplataforma orientado a la creación de websdinámicas con uso de bases de datos como en nuestro caso MySQL. Cuando el cliente hace una petición al servidor para que le envíe una página web, el servidor ejecuta el intérprete de PHP, este procesa el script solicitado que generará el contenido de la web de manera dinámica (por ejemplo obteniendo información de una base de datos o de un archivo de texto) y el resultado es enviado por el intérprete al servidor (en nuestro caso Apache), quien a su vez se lo envía al cliente.

eAccelerator 2.2.0

Acelerador de PHP usado en algunas de las configuraciones de pruebas. Este programa se explicará más adelante.

Módulos Boost6.x-1.18 y Authcaché

Dos módulos de Drupal que mediante sistemas de caché permiten mejorar su rendimiento. Están explicados más detalladamente más adelante.

III.2.5 WEB DE PRUEBAS

El fin de estas pruebas de rendimiento como ya hemos explicado es comprobar el rendimiento real de la plataforma “Portales Municipales” y comprobar la carga máxima aceptada por el mismo en según que situaciones y configuraciones. Para así elegir las más convenientes según el tráfico potencial que tenga en su entorno de producción.

Hay que tener en cuenta que el proyecto en sí, pretende producir y alojar en servidor un total que rondará los 200 sitios webs, pero en el momento de este estudio estos no estaban disponibles. Así que hemos optado por utilizar una web modelo, que

consideramos cumple el rendimiento medio que tendrán las webs finales, es la web del Ayuntamiento de Cheste. (ilus. 35)

Los resultados obtenidos provienen de hacer peticiones al home del sitio, pues es la página web del sitio que más recursos requiere (imágenes, consultas a bases de datos, applets)

Ilustración 35 - Web de Pruebas del Ayuntamiento de Cheste (www.cheste.es)

III.2.6 SOFTWARE CLIENTE:

Hablamos de los clientes para hablar de las 2 máquinas que nos sirven para testear nuestro servidor web virtual.

Durante las pruebas, hemos utilizado principalmente 2 aplicaciones dedicadas a medir el rendimiento de nuestro servidor.

Primero, optamos por probar una aplicación gráfica bajo el sistema operativo Microsoft Windows, WebStress Server Tools 7.2 debido a la simplicidad del mismo y el hecho de que nos sacaba gráficas de manera muy cómoda. Para las pruebas en las que no se usaba ningún tipo de caché o solo la caché por defecto que aporta Drupal, el rendimiento del programa era óptimo. Sin embargo, cuando añadimos a nuestro servidor web un acelerador de PHP y el módulo Boost de Drupal, el cliente se tornó en el factor limitante de las pruebas por lo que nos era imposible sacar el rendimiento máximo del servidor para estas configuraciones. Aunque intentamos, repetir las pruebas utilizando una máquina física más potente, los resultados fueron similares, lo cuál hizo que tuviésemos que cambiar nuestra forma de proceder.

Decidimos que debíamos cambiar la herramienta de test. Y optamos por la herramienta que nos proporciona Apache para este fin, Apache Benchmark. Esta aplicación a diferencia de la primera esta solo disponible en modo gráfico. Para las prueabs de rendimiento con ab hemos instalado en la máquina cliente el sistema operativo Linux CentOS. 5.3

III.3 DESCRIPCIÓN DE LA PRUEBA

La prueba ha consistido en generar a través de programas de testeo una carga controlada sobre el servidor web de portales municipales sobre el que se había configurado un portal modelo.

La carga se realiza sobre una página única siendo esta la portada de la web.

La carga se ha ido aumentando progresivamente hasta alcanzar la capacidad máxima del servidor.

Hemos identificado esta situación por el aumento de los tiempos medios de respuesta más allá de 1 segundo para más del 1% de las consultas.

Se ha verificado en todos los casos que no se aplicaba limitación alguna ni en la memoria ram utilizada, ni en el ancho de banda (salvo en el último caso como se verá). Se han configurado los servidores con memoria ram suficiente para que no fuera un factor limitante de las pruebas.

Se han realizado las pruebas para 3 configuraciones del servidor virtual diferentes, con 1 core, 2 cores y 4 cores.

Se han tomado 3 medidas consecutivas sin periodo de descanso y se ha tomado la última obtenida, considerando correcta la prueba si la desviación entre medidas calculada fuera inferior al 10% de la media.

Al ser las pruebas consecutivas, a efectos del servidor ha sido una prueba 'triple'.

En todos los casos también se ha realizado una prueba de carga superior para confirmar que la carga era máxima y el rendimiento se deterioraba a partir de las medidas tomadas.

Inicialmente se utilizó el software Webstress del que no se pudieron obtener datos fiables, al ser en muchos casos la carga del cliente muy superior a la del servidor. No se probó la configuración para varios cliente 'atacando' el servidor al no disponer de equipamiento suficiente.

Las pruebas se han realizado con el software ab utilizando el comando.

```
ab -n numero_de_peticiones -c
numero_de_peticiones_concurrentesurl
```

Por ejemplo, para uno de nuestros casos:

```
ab -n 10000 -c 400 http://www.pruebascheste.com/index.php
```

Se le indican dos parámetros, el número de peticiones y el número de clientes (peticiones concurrentes). El número de clientes es parámetro de esfuerzo debiendo ser el número de peticiones lo suficiente grande para generar un estado de estabilidad en la carga por parte de todo el sistema. (ilus. 36)

Concurrency Level
Time taken for tests seconds
Complete requests
Failed requests
Write errors
Total transferred bytes
HTML transferred bytes
Requests per second [#/sec] (mean)
Time per request [ms] (mean)
Time per request [ms] (mean, across all concurrent requests)
Transfer rate [Mbytes/sec] received
Connection Times (ms)
Connect
Processing
Waiting
Total
Percentage of the requests served within a certain time (ms)

Ilustración 36 - Datos de los resultados de los informes

La prueba tiene el objeto de determinar la relevancia de los parámetros de cpu y de distintas mecanismos de aceleración sobre el rendimiento de la plataforma de portales municipales.

El parámetro que hemos intentado maximizar es el de peticiones atendidas por segundo, para así poder extrapolar el número máximo de peticiones aceptadas por hora de servicio del servidor.

III.3.1 MECANISMOS DE ACELERACIÓN DEL GESTOR DE CONTENIDOS

Para poder describir los mecanismos de aceleración del gestor de contenidos (Drupal 6) que hemos utilizado, es necesario definir el concepto de caché.

Una caché es un conjunto de datos duplicados de otros más difíciles de obtener. Cuando se accede por primera vez a un dato que requiere de un proceso costoso para obtenerlo, por ejemplo múltiples consultas a una base de datos, se almacena. En los siguientes accesos al contenido se devuelve la copia ya guardada anteriormente, comprobando antes que no se haya modificado. De aquí, se deduce que será más eficiente el uso de la caché en un sitio que se actualice pocas veces, por ejemplo una web meramente informativa que en un sitio que este en constante cambio, como por ejemplo una red social o un foro.

De forma genérica podríamos definir un sistema de caché con el siguiente diagrama. (ilus. 37)

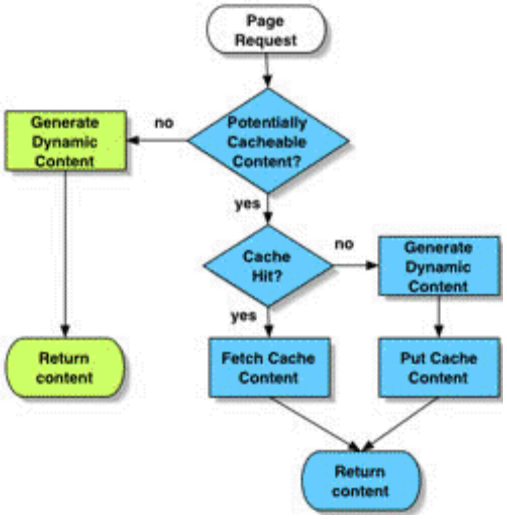


Ilustración 37 - Mecanismo de funcionamiento de un sistema de Caché

Según el sistema de caché que utilicemos podemos almacenar estas copias de contenido en distintos sitios del sistema, en memoria primaria (memoria RAM), en memoria secundaria (disco) o en la propia base de datos (aunque pueda parecer que no haya una mejora de rendimiento por el hecho de hacer una consulta a la base de datos, hay que tener en cuenta que en este caso sacamos todo el contenido de la página en una sola consulta a esta mientras que sin sistema de caché tendremos que hacer múltiples consultas)

En el caso de Drupal, el gestor de contenido que nos interesa, podemos guardar bloques, variables, filtros, formularios, menús o páginas completas. Este factor puede llegar a ser muy importante en el sistema de caché que utilicemos.

Los usuarios anónimos de un sitio suelen limitarse a leer contenidos, normalmente no interactúan pues la mayoría de funcionalidades que proporcionan interacción requieren de una cuenta de usuario. En actividades de interacción podríamos mencionar: comentar, valorar un contenido, buscar, agregar contactos, etc. dependiendo de los módulos que estén instalados en Drupal puede haber cientos de formas de interacción que se tienen que efectuar en tiempo real y que harían que un contenido se modifique muy a menudo, por lo que sería inviable guardar una copia en caché porque en breve quedaría desfasada de su contenido real. Afortunadamente en la mayoría de sitios de

contenidos, la distribución de usuarios será 80% anónimos y 20% autenticados como máximo por lo que la masa crítica verá la página cacheada y no tendrá bloques con contenido personalizado.

Aún así existen métodos para mejorar también el rendimiento en sitios muy dinámicos.

Caché Drupal

Drupal tiene una función en el núcleo que permite activar la cache por página (ilus. 38), esto quiere decir que cada vez que se solicita un contenido por primera vez Drupal hace cientos de consultas, procesa esta información y la renderiza guardando ese HTML final en un motor de caché, esta copia será utilizada para servir el contenido a un coste menor de procesamiento en la siguientes peticiones de los usuarios anónimos.

La caché de página no sirve para usuarios autenticados porque los bloques de contenido personalizados (por ejemplo el saludo de usuario “Hola Juan”) no tendrían sentido para otro usuario. En ese caso hay otras soluciones como la carga de los bloques personalizados de forma dinámica con una petición AJAX que se sobrepone al contenido cacheado.

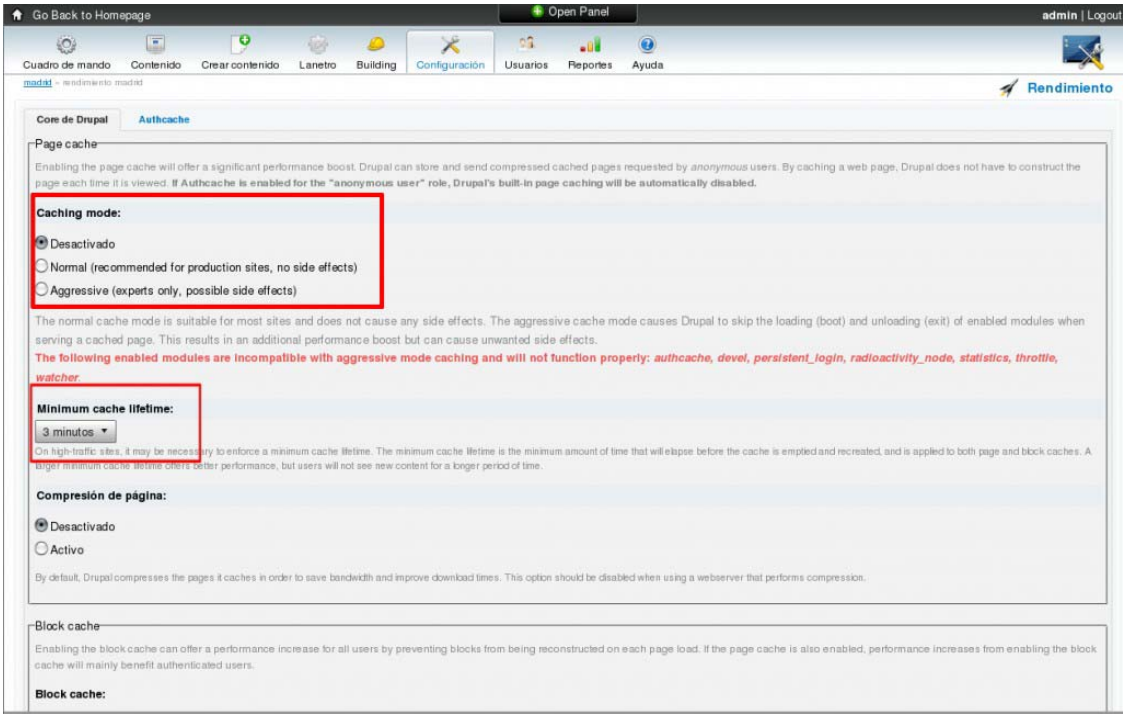


Ilustración 38 - Formulario de configuración de la caché del core de Drupal

Un gran problema de la caché del núcleo de Drupal es que no tiene reglas configurables de caché, si necesitas cachear solo ciertas páginas u omitir otras, o cachear todo pero exceptuando ciertas páginas, olvídalo, es imposible con la función del núcleo. Afortunadamente existen otros módulos que permiten añadir estas funcionalidades a la caché de Drupal.

Acelerador de PHP – Eaccelerator

Para explicar que es un acelerador de PHP, primero debemos tener en cuenta que PHP es un lenguaje de programación interpretado. Es decir, por cada petición de una página PHP, el servidor primero traduce/compila PHP, en un lenguaje intermedio (bytecode) que finalmente es interpretado por un intérprete de PHP o máquina virtual (ZendEngine) que ejecuta el código y nos proporciona el código estático HTML final. (ilus. 39)

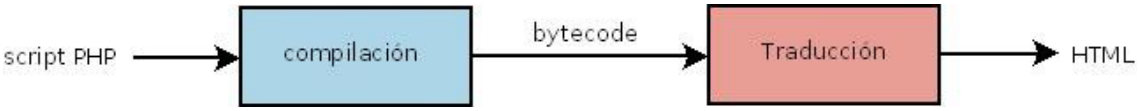


Ilustración 39 - Proceso de compilación de PHP

Un acelerador de PHP (ilus. 18) utiliza un sistema de caché para almacenar en memoria los scripts en código bytecode, ahorrándonos el realizar la fase de traducción/compilación para cada petición de la web en PHP. Cada vez que el servidor recibe una petición de un script de php, (ilus. 40)

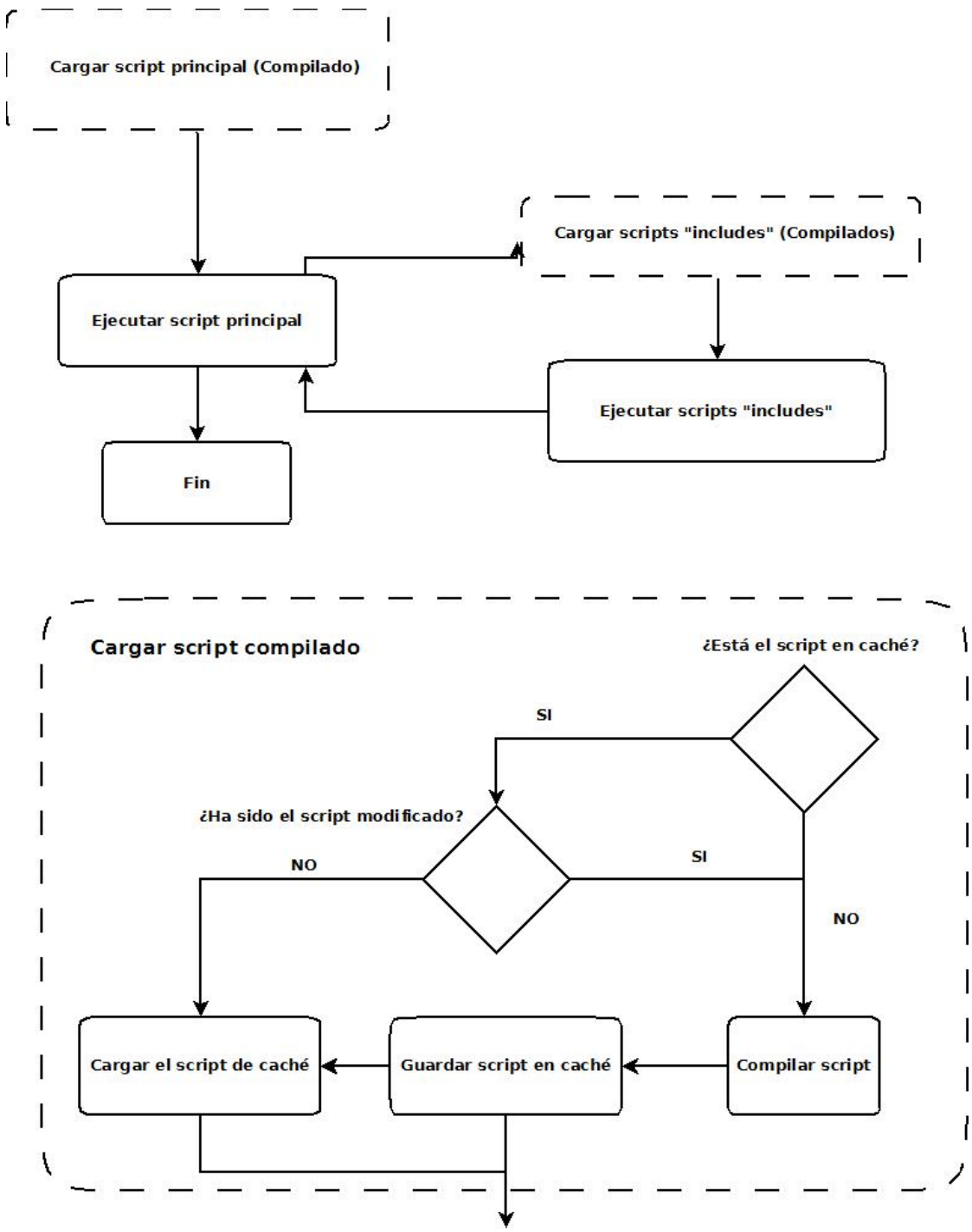


Ilustración 40 - Mecanismo de funcionamiento de un acelerador de PHP

Si el script de php que se nos solicita esta en caché y no ha sido modificado desde que se almacenó se carga de la caché, en caso contrario se compila y se sirve. En el caso de que el script incluya otros scripts se sigue el mismo proceso para todos ellos recursivamente.

Existen muchos aceleradores de PHP o cachers de PHP, entre ellos los más utilizados son Alternative PHP Cache (APC), XCache y Eaccelerator, en nuestro caso hemos optado por el último ya que en estudios de terceros encontrados en la red es el que mejores resultados proporcionaba con el CMS Drupal.

Módulo Boost

Como ya sabemos, Drupal es un CMS modular, algunos de sus módulos nos permiten aumentar el rendimiento de su sistema de caché, es el caso del módulo Boost. (ilus.41)

Boost a diferencia del sistema de caché por defecto que almacena la caché en la base de datos, lo almacena en disco duro en una carpeta creada por el administrador del servidor lo cual hace que el acceso a base de datos sea casi inexistente. Este mecanismo lo proporciona creando un directorio de contenido estático de nuestro sitio y que se regenera con cada ejecución de las tareas programadas por el cron. El tiempo de caducidad de las páginas es facilmente configurable desde el panel de configuración.

Cachea y comprime en formato gzip, tanto contenido html, como ajax, xml, javascript o css.

Este módulo nos permite un aumento muy significativo en rendimiento y escalabilidad para sitios que reciben una gran cantidad de usuarios anónimos ya que hay que tener en cuenta que este módulo no entra en funcionamiento cuando entra un usuario autentificado. Para estos casos habrá que utilizar otros módulos.

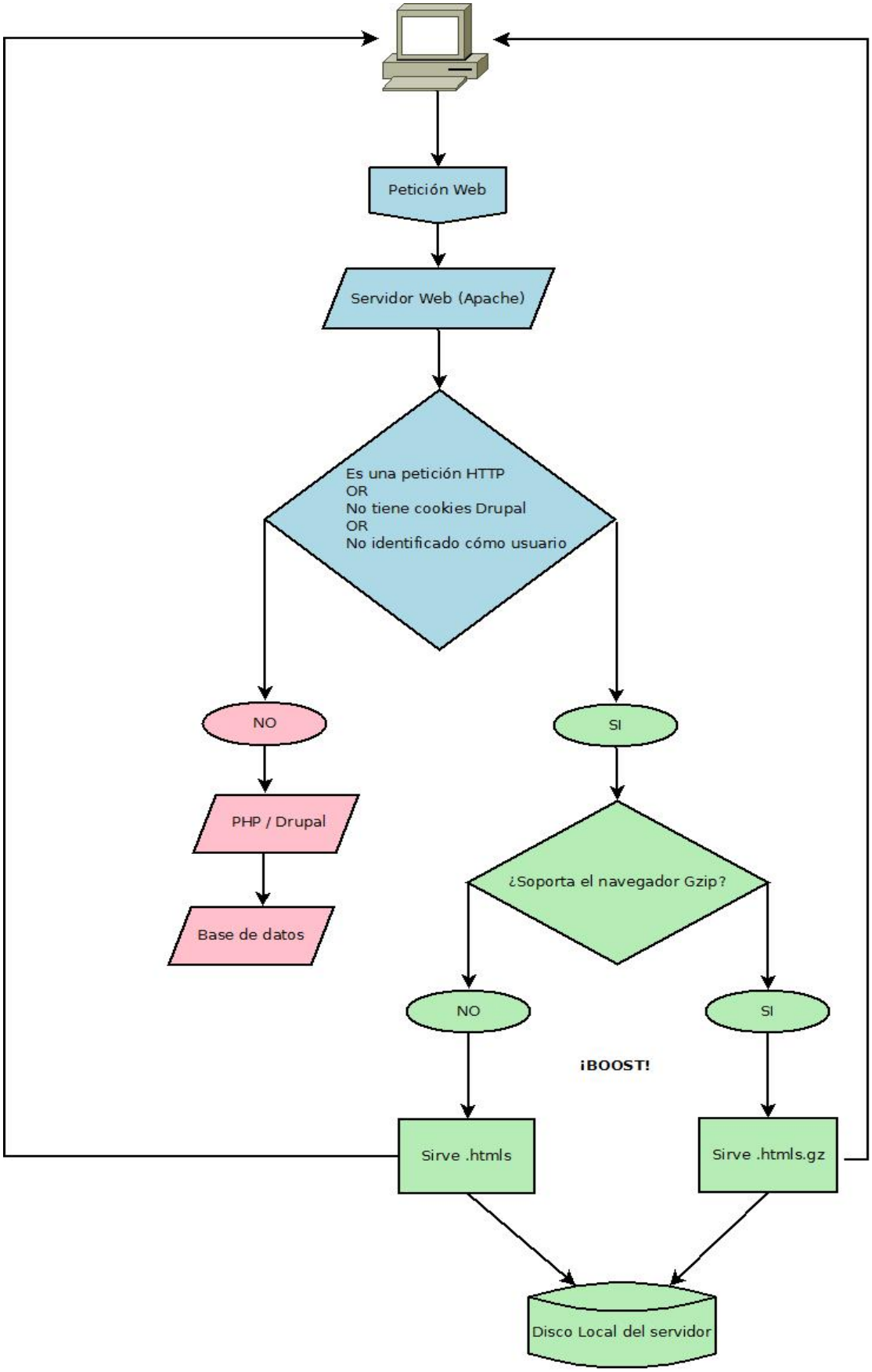


Ilustración 41 - Mecanismo de funcionamiento del módule Boost de Drupal

Módulo Authcaché:

El módulo Authcaché (ilus. 42) es similar a Boost, salvo que este nos permite mejorar el rendimiento también para usuarios autentificados. Esto se consigue porque primero el módulo carga de caché la parte estática de la web común tanto a usuarios anónimos como autentificados y es después que por medio de AJAX se modifican las partes exclusivas a los usuarios autentificados.

Por ejemplo, pongamos una web en la que el contenido para un usuario autentificado llamado “Pepito” solo cambie en un texto en el que pone “Hola Pepito”. Primero se cargará la web tal como la ven los usuarios anónimos es decir sin el texto “Hola Pepito”. Después de esto, por medio de Ajax se pedirá el contenido exclusivo para el usuario “Pepito”, se hará una consulta a la base de datos y se devolverá el texto “Hola Pepito”. Podemos ver un esquema de este proceso en la siguiente figura. En combinación con el módulo Boost este módulo aumenta considerablemente el rendimiento de nuestro servidor web.

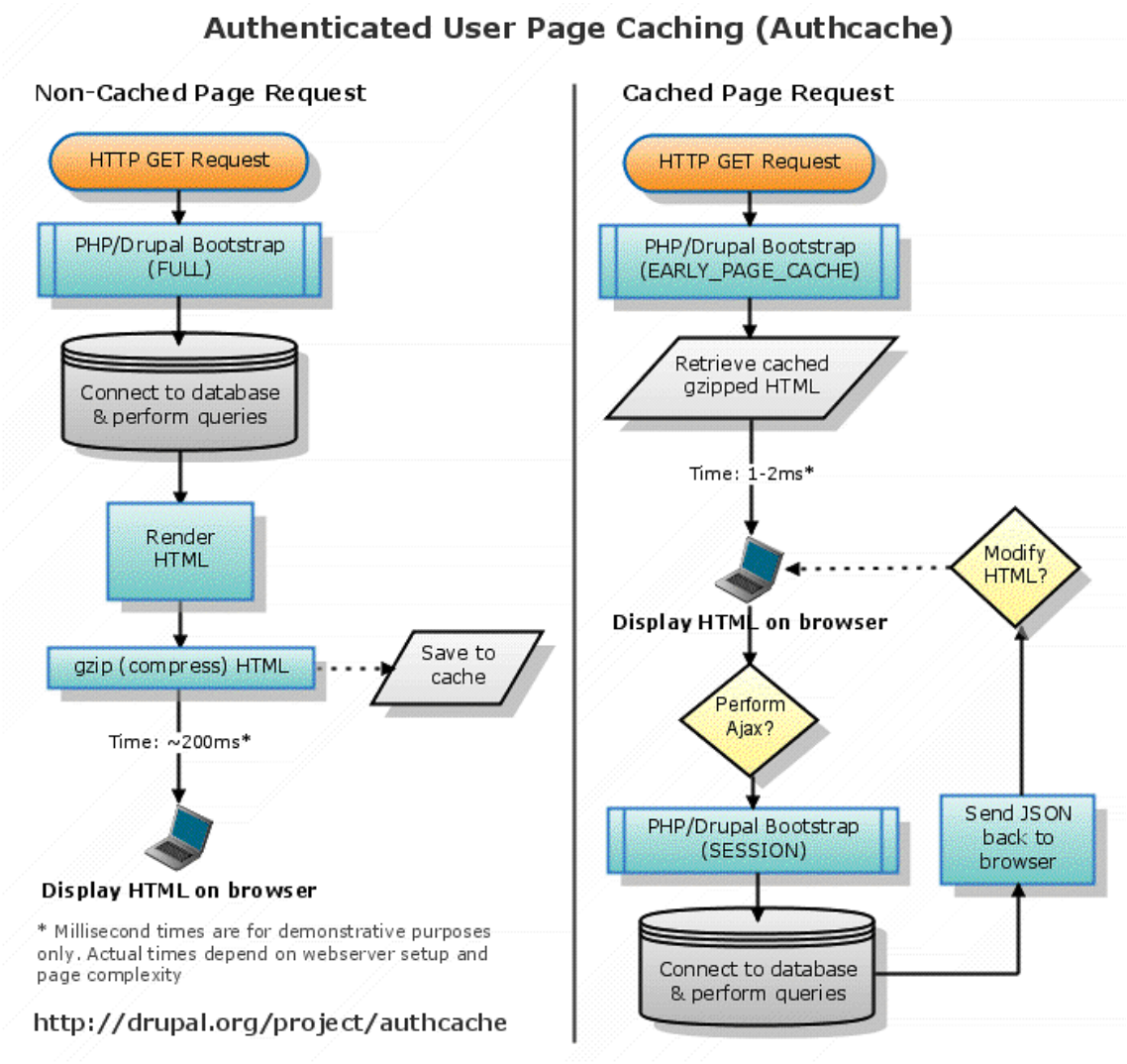


Ilustración 42 - Módulo de Drupal para aceleración de usuarios autenticados

III.4 RESULTADOS DE LA PRUEBA

Vemos los resultados que nos han proporcionado las pruebas realizadas.

Hemos recuperado de todas las pruebas realizadas el número de peticiones por segundo aceptadas por el servidor en función tanto de un factor hardware (número de núcleos de la máquina virtual), como según la configuración software utilizada, uso o no uso de diferentes sistemas de caché. (tabla 4)

Requests per second [/sec] (mean)	1 core	2 cores	4 cores
Sin cache y eAccelerator	0,94	1,78	2,98
Sin cache	1,07	1,96	3,64
Con cache de Drupal	40,45	77,43	143,48
Con cache de Drupal y eAccelerator	220,82	392,19	601,82
Con cache Drupal + eAccelerator + Boost	760,54	1202,49	1203,99

Tabla 4 - Tabla rendimiento según el número de núcleos

El rendimiento de la plataforma sin uso de ningún tipo de caché es muy pobre, al igual que el uso de eAccelerator sin combinar con otro tipo de caché, ya sea la propia o añadiéndole módulos como Boost o Authcache.

El uso de la caché de Drupal aumenta en el orden de 4000 % con respecto a sin ningún tipo de caché.

Si a esto le añadimos un acelerador de php como eAccelerator obtenemos de nuevo un incremento del rendimiento de entre el 400 % al 550 %.

Finalmente añadiéndole a la caché de Drupal + eAccelerator el módulo Boost y haciendo el test únicamente para usuarios autentificados obtenemos un incremento del 300 % al 350%.

Observamos también que seguramente sería posible aumentar el número de peticiones por segundo máximo para la última prueba (caché Drupal + eAccelerator + Boost y 4 cores) con una máquina de testeo más potente pues los resultados quedan estabilizados y no hay aumento de rendimiento). Aunque es posible que el factor limitante sea la red pues ya llegaba a unas velocidades de transmisión de 76 Mbps (aunque teoricamente según la prueba de red que hicimos anteriormente debería llegar hasta mas o menos 95 Mbps).

Concurrency Level	1 core	2 cores	4 cores
Sin cache y eAccelerator	2	2	4
Sin cache	1	2	4
Con cache de Drupal	5	10	15
Con cache de Drupal y eAccelerator	20	30	40
Con cache Drupal + eAccelerator + Boost	75	250	400

Tabla 5 - Máximas peticiones concurrentes

Estos resultados son las peticiones concurrentes que habían configuradas cuando se llegó al valor máximo de peticiones respuestas por segundo pero no son un valor límite para el servidor. Las pruebas completas están en un anexo al final del documento.

III.5. CONCLUSIONES

Las pruebas realizadas nos ofrecen una información muy clara sobre los mecanismos adecuados de aceleración de la plataforma de portales municipales, y de los mecanismos a aplicar en la configuración final de la misma.

El módulo Boost se ha demostrado muy efectivo para mostrar el contenido estático del portal a usuarios anónimos. El funcionamiento de este módulo de almacenar la información de la caché en disco, tiene un rendimiento muy superior al estándar de Drupal que almacena la información en la base de datos.

El rendimiento obtenido para una configuración del servidor de 2 núcleos ha sido de 1202 peticiones por segundo, con una concurrencia de 250, y un tiempo del 99% de peticiones por debajo de 850ms y un tiempo medio de respuesta de 208 ms. El servidor no llegó a saturarse en esta configuración pero la plataforma de prueba ya permitió comprobar cargas mayores al alcanzarse el límite de transferencia TCP/IP del enlace Gigabit en 76,75 Mbytes/segundo de transferencia de información efectiva..

Es posible que una mejor configuración del mysql , o que la utilización de una instancia de mysql que utilizara memoria RAM para almacenar la información, pudiera modificar este resultado. Sin embargo el gran volumen de páginas a cachear hace inviable esta solución para nosotros.

Peticiones por hora	1 core	2 cores	4 cores
Sin cache y eAccelerator	3384	6408	10728
Sin cache	3852	7056	13104
Con cache de Drupal	145620	278748	516528
Con cache de Drupal y eAccelerator	794952	1411884	2166552
Con cache Drupal + eAccelerator + Boost	2737944	4328964	4334364

Tabla 6 - Peticiones atendidas en una hora

Para usuarios identificados, donde no se puede aplicar la de Boost ni la configuración de Cache de Drupal, la configuración de eAccelerator no ha mostrado ninguna mejoría y el rendimiento esperable es el del sistema sin aceleración. En este caso el rendimiento vendrá fijado por la potencia del hardware del servidor. Existen algunas soluciones estudiadas en este informe como el módulo AuthCache el cuál hemos comentado previamente y que nos aumentarían el rendimiento para usuarios autentificados que aunque no hemos podido comprobar debería acelerar el proceso de forma a similar a como lo hace Boost.

IV. CONCLUSIONES

Como conclusión a este proyecto podemos decir que hemos llevado a cabo la elección y validación de un sistema de explotación multientidad para el proyecto Portales Municipales de la Diputación de Valencia con éxito.

Aunque en un principio nos planteábamos el diseño y desarrollo del mismo, durante el estudio de requisitos del proyecto nos dimos cuenta de que existían ya alternativas para llevar a cabo las tareas que necesitábamos. En este caso el programa Aegir nos ha sido de gran ayuda ya que aúna la mayoría de características buscadas para nuestra plataforma de explotación. Con esta aplicación desarrollada por terceros a mano sólo necesitábamos validar la plataforma además de crear los manuales de usuario necesarios para un correcto manejo de la misma.

Al iniciar el proyecto además de validar el rendimiento de la plataforma también pensábamos que sería necesario crear un sistema de recuperación ante fallos pero gracias a las herramientas de virtualización de las que hace uso la Diputación no nos ha sido necesario centrarnos en ese problema pues ya vienen soluciones integradas en estas herramientas. Así que lo único problemático ha sido el buscar una configuración software y hardware óptima para nuestra plataforma.

Además como conclusiones personales podemos sacar que en mi caso ha sido la primera vez que hemos podido participar en un proyecto llevado a cabo en una empresa por lo cual nos ha aportado una experiencia que no teníamos. Resaltamos cómo positivo que este proyecto nos ha permitido por primera vez trabajar en el entorno profesional de la empresa. Además de esto también nos ha permitido profundizar en materias que no se tocan mucho en la facultad de informática. En este sentido, hemos aprendido a preparar la documentación de nuestra plataforma, también hemos cogido bastantes conocimientos en cuanto a administrar la plataforma de virtualización vmware vSphere y finalmente destacaríamos que hemos aprendido a preparar un entorno para realizar pruebas de rendimiento (*profiling*)

Cómo conclusión podemos decir que los objetivos tanto de trabajo, es decir el proyecto que tenía que cumplir la empresa, cómo personales se han cumplido satisfactoriamente.

V. BIBLIOGRAFÍA

- http://szeged2008.drupalcon.org/program/sessions/deploying-and-maintaining-drupal-sites-using-aegir-hosting-system, [consultado el 01/08/2010]
- http://drupal.groups.com, [consultado el 01/08/2010]
- http://szeged2008.drupalcon.org/files/aegir.pdf, [consultado el 03/08/2010]
- http://sf2010.drupal.org/conference/sessions/aegir-hosting-system-one-drupal-rule-them-all, [consultado el 01/08/2010]
- http://www.boe.es/aeboe/consultas/bases_datos/doc.php?id=BOE-A-2007-12352, [consultado el 20/09/2010]
- http://es.wikipedia.org/wiki/Cache, [consultado el 01/08/2010]
- http://www.tratonera.com/?q=node/58, [consultado el 03/09/2010]
- http://cambrico.net/drupal/cache-en-el-desarrollo-de-drupal-6, [consultado el 01/08/2010]
- http://community.aegirproject.org/notebook, [consultado el 18/08/2010]
- http://www.drupblue.com/blog/instalando-aegir, [consultado el 09/08/2010]
- http://groups.drupal.org/node/23712, [consultado el 01/09/2010]
- http://info.vmware.com/content/9270_SP_Gen?src=PaidSearch_Google_PaidSearch_Google_EMEA-South_SpanIB_VI_General_VMware_Search&gclid=CPCplLOQhKsCFUEMfAod3WrA3g, [consultado el 10/08/2010]

VI. ANEXOS

VI.1 RESULTADO DE LAS PRUEBAS DE RENDIMIENTO AL COMPLETO

VI.1.1 CABECERA COMÚN A TODAS LAS PRUEBAS

Server Software	Apache/2.2.3
Server Hostname	www.pruebascheste.com
Server Port	80
Document Path	/
Document Length bytes	66195 bytes

VI.1.2 BOOST + EACCELERATOR

Hardware	BOOST 1CORE 2GB	BOOST 2 CORES 2GB	BOOST 4CORE 2GB
Concurrency Level	75	250	400
Time taken for tests seconds	13,15	42	41,53
Complete requests	10000	50000	50000
Failed requests	0	0	0
Write errors	0	0	0
Total transferred bytes	668841856	3344879704	3345438632
HTML transferred bytes	664921464	3325268336	3325824912
Requests per second [#/sec] (mean)	**760,54**	**1202,49**	**1203,99**
Time per request [ms] (mean)	98,61	208	332,23
Time per request [ms] (mean, across all concurrent requests)	1,32	0,83	0,83
Transfer rate [Mbytes/sec] received	48,51	76,72	76,83

Connection Times (ms)	min	mean[+/-sd]	median	max	min	mean[+/-sd]	median	max	min	mean[+/-sd]	median	max
Connect	0	12 31.0	13	2999	0	40 218.8	27	9027	0	22 91.2	20	3032
Processing	43	85 14.5	83	344	7	166 172.4	118	8590	11	305 2747.0	86	38887
Waiting	1	28 15.6	28	71	5	40 124.4	29	2486	6	224 2724.1	22	37724
Total	43	97 32.6	96	3046	7	206 278.5	145	9279	11	328 2747.5	104	38893

Percentage of the requests served within a certain time (ms)			
	50% 96	50% 145	50% 104
	66% 98	66% 168	66% 106
	75% 99	75% 173	75% 110
	80% 100	80% 201	80% 121
	90% 110	90% 280	90% 140
	95% 123	95% 425	95% 179
	98% 140	98% 611	98% 483
	99% 143	99% 849	99% 813
	100% 3046 (longest request)	100% 9279 (longest request)	100% 38893 (longest request)

VI.1.3 EACCELERATOR + CACHÉ DRUPAL

	eAccelerator 1 Core 2GB	eAccelerator 2 Cores 2GB	eAccelerator 4 Cores 2GB
Concurrency Level	20	30	40
Time taken for tests seconds	226.429462 seconds	25.497656 seconds	16.616276 seconds
Complete requests	50000	10000	10000
Failed requests	0	0	0
Write errors	0	0	0
Total transferred bytes	3333900000	666780000	666780000
HTML transferred bytes	3309750000	661950000	661950000
Requests per second [#/sec] (mean)	**220,82**	**392,19**	**601,82**
Time per request [ms] (mean)	90,57	76,49	66,47
Time per request [ms] (mean, across all co	4,53	2,55	1,66
Transfer rate [Kbytes/sec] received	14378,7	25537.72 [Kbytes/sec] received	39187,6

	Connection Times (ms)	Connection Times (ms)	Connection Times (ms)
	min mean[+/-sd] median max	min mean[+/-sd] median max	min mean[+/-sd] median max
Connect	0 1 1.4 0 9	0 0 0.7 0 5	0 0 0.4 0 6
Processing	7 88 430.4 48 33125	6 75 283.7 46 7303	7 65 65.8 50 773
Waiting	3 74 421.5 35 32445	4 56 234.7 36 7274	4 49 51.3 39 740
Total	7 90 430.4 49 33125	7 75 283.7 46 7303	8 65 65.9 50 773

Percentage of the requests served within a certain	Percentage of the requests served within a	Percentage of the requests served within a ce
50% 49	50% 46	50% 50
66% 85	66% 62	66% 64
75% 111	75% 75	75% 76
80% 133	80% 86	80% 85
90% 187	90% 130	90% 118
95% 238	95% 180	95% 166
98% 310	98% 263	98% 263
99% 389	99% 348	99% 369
100% 33125 (longest request)	100% 7303 (longest request)	100% 773 (longest request)

VI.1.4 CACHÉ DRUPAL

Hardware		Caché 2 Cores 2GB		Caché 4 Cores 2GB	
Concurrency Level		5	10		15
Time taken for tests seconds	123.604721 seconds	64.577020 seconds		69.696272 seconds	
Complete requests		5000	5000		10000
Failed requests		0	0		0
Write errors		0	0		0
Total transferred bytes	333390000 bytes	333390000 bytes		666840816 bytes	
HTML transferred bytes	330975000 bytes	330975000 bytes		662010333 bytes	
Requests per second [#/sec] (mean)		**40,45**	**77,43**		**143,48**
Time per request [ms] (mean)		123,61	129,15		104,54
Time per request [ms] (mean, across all conc	24.721 [ms] (mean, across all concurrent requests)	12.915 [ms] (mean, across all concurrent requests)		6.970 [ms] (mean, across all concurrent requests)	
Transfer rate [Kbytes/sec] received	2634.01 [Kbytes/sec] received	5041.67 [Kbytes/sec] received		9343.56 [Kbytes/sec] received	

	Connection Times (ms)	Connection Times (ms)	Connection Times (ms)
	min mean[+/-sd] median max	min mean[+/-sd] median max	min mean[+/-sd] median max
Connect	0 0 1.2 0 12	0 0 1.8 0 9	0 0 1.7 0 8
Processing	24 122 29.1 121 735	25 127 296.8 117 16635	28 103 64.3 98 3036
Waiting	21 98 26.1 98 727	22 98 204.8 91 11123	23 80 50.5 76 2816
Total	24 122 29.1 121 735	25 128 296.7 118 16635	28 103 64.3 99 3036

Percentage of the requests served within a certain time	Percentage of the requests served within a certain time (ms)	Percentage of the requests served within a certain time (ms)
50% 121	50% 118	50% 99
66% 127	66% 130	66% 111
75% 131	75% 138	75% 119
80% 134	80% 143	80% 125
90% 143	90% 161	90% 142
95% 151	95% 176	95% 159
98% 173	98% 205	98% 185
99% 197	99% 264	99% 219
100% 735 (longest request)	100% 16635 (longest request)	100% 3036 (longest request)

VI.1.5 SIN CACHÉ DRUPAL

Hardware	Sin Caché 1 Core 2 GB	Sin Caché 1 Core 2 GB	Sin Caché 1 Core 2 GB
Concurrency Level	1	2	4
Time taken for tests seconds	93.789976 seconds	51.65208 seconds	27.481037 seconds
Complete requests	100	100	100
Failed requests	0	0	0
Write errors	0	0	0
Total transferred bytes	6669300	6669300	6669300
HTML transferred bytes	6619300	6619300	6619300
Requests per second [#/sec] (mean)	**1,07**	**1,96**	**3,64**
Time per request [ms] (mean)	937.900 [ms] (mean)	1021.304 [ms] (mean)	1099.241 [ms] (mean)
Time per request [ms] (mean, across all cc	937,9	510,65	274,81
Transfer rate [Kbytes/sec] received	69,43	127,52	236,96

	Connection Times (ms)					Connection Times (ms)					Connection Times (ms)				
	min	mean[+/-sd]	median	max		min	mean[+/-sd]	median	max		min	mean[+/-sd]	median	max	
Connect	0	0	0.0	0	0	0	0	0.2	0	1	0	0	0.3	0	1
Processing	907	937	24.4	934	1069	966	1020	32.0	1016	1107	1043	1085	31.7	1079	1221
Waiting	865	894	22.6	892	1011	907	956	31.1	950	1044	968	1009	28.3	1005	1128
Total	907	937	24.4	934	1069	966	1020	32.0	1016	1107	1043	1085	31.7	1079	1221

Percentage of the requests served within a certa	Percentage of the requests served within a certain ti	Percentage of the requests served within a
50% 934	50% 1016	50% 1079
66% 946	66% 1033	66% 1094
75% 949	75% 1042	75% 1104
80% 951	80% 1046	80% 1110
90% 957	90% 1066	90% 1127
95% 968	95% 1081	95% 1152
98% 1053	98% 1105	98% 1159
99% 1069	99% 1107	99% 1221
100% 1069 (longest request)	100% 1107 (longest request)	100% 1221 (longest request)

VI.1.6 SIN CACHÉ DRUPAL + EACCELERATOR

		66193 bytes	66193 bytes
Hardware			
Concurrency Level	2	2	4
Time taken for tests seconds	106.159776 seconds	56.124165 seconds	33.531018 seconds
Complete requests	100	100	100
Failed requests	0	0	0
Write errors	0	0	0
Total transferred bytes	6669300 bytes	6669300 bytes	6669300 bytes
HTML transferred bytes	6619300 bytes	6619300 bytes	6619300 bytes
Requests per second [#/sec] (mean)	0.94 [#/sec] (mean)	1.78 [#/sec] (mean)	2.98 [#/sec] (mean)
Time per request [ms] (mean)	2123.196 [ms] (mean)	1122.483 [ms] (mean)	1341.241 [ms] (mean)
Time per request [ms] (mean, across all co	1061.598 [ms] (mean, across all concurrent reque	561.242 [ms] (mean, across all concurrent	335.310 [ms] (mean, across all concurrent requ
Transfer rate [Kbytes/sec] received	61.34 [Kbytes/sec] received	116.03 [Kbytes/sec] received	194.21 [Kbytes/sec] received

	Connection Times (ms)	Connection Times (ms)	Connection Times (ms)
	min mean[+/-sd] median max	min mean[+/-sd] median max	min mean[+/-sd] median max
Connect	0 0 0.1 0 1	0 0 0.5 0 5	0 0 0.7 0 6
Processing	996 2121 951.5 2025 6470	1007 1121 258.1 1050 2799	1228 1324 46.9 1309 1449
Waiting	962 1888 955.3 1891 6337	974 1084 256.7 1012 2756	1194 1277 44.0 1265 1401
Total	996 2121 951.5 2025 6470	1007 1121 258.1 1050 2799	1228 1324 46.9 1310 1449

Percentage of the requests served within a certain t	Percentage of the requests served within a c	Percentage of the requests served within a certai
50% 2025	50% 1050	50% 1310
66% 2074	66% 1077	66% 1337
75% 2227	75% 1104	75% 1351
80% 2314	80% 1196	80% 1370
90% 2876	90% 1251	90% 1392
95% 4448	95% 1366	95% 1426
98% 5584	98% 2799	98% 1448
99% 6470	99% 2799	99% 1449
100% 6470 (longest request)	100% 2799 (longest request)	100% 1449 (longest request)

VI.2 FICHERO DE CONFIGURACIÓN DE MYSQL (MY.CNF)

```
# Example MySQL config file for very large systems.
#
# This is for a large system with memory of 1G-2G where the system runs mainly
# MySQL.
#
# You can copy this file to
# /etc/my.cnf to set global options,
# mysql-data-dir/my.cnf to set server-specific options (in this
# installation this directory is /var/lib/mysql) or
# ~/.my.cnf to set user-specific options.
#
# In this file, you can use all long options that a program supports.
# If you want to know which options a program supports, run the program
# with the "--help" option.


# The following options will be passed to all MySQL clients
[client]
#password       = your_password
```

```
port            = 3306
socket          = /var/lib/mysql/mysql.sock



# Here follows entries for some specific programs


# The MySQL server
[mysqld]
port            = 3306
socket          = /var/lib/mysql/mysql.sock
skip-locking
key_buffer = 384M
max_allowed_packet = 1M
table_cache = 512
sort_buffer_size = 2M
read_buffer_size = 8M
read_rnd_buffer_size = 8M
myisam_sort_buffer_size = 64M
thread_cache_size = 8
query_cache_size = 32M
# Try number of CPU's*2 for thread_concurrency
thread_concurrency = 8



# Don't listen on a TCP/IP port at all. This can be a security enhancement,
```

```
# if all processes that need to connect to mysqld run on the same host.

# All interaction with mysqld must be made via Unix sockets or named pipes.

# Note that using this option without enabling named pipes on Windows

# (via the "enable-named-pipe" option) will render mysqld useless!

#

#skip-networking



# Disable Federated by default

skip-federated



# Replication Master Server (default)

# binary logging is required for replication

log-bin=mysql-bin



# required unique id between 1 and 2^32 - 1

# defaults to 1 if master-host is not set

# but will not function as a master if omitted

server-id       = 1



# Replication Slave (comment out master section to use this)

#

# To configure this host as a replication slave, you can choose between

# two methods :

#
```

```
# 1) Use the CHANGE MASTER TO command (fully described in our manual) -
#    the syntax is:
#
#    CHANGE MASTER TO MASTER_HOST=<host>, MASTER_PORT=<port>,
#    MASTER_USER=<user>, MASTER_PASSWORD=<password> ;
#
#    where you replace <host>, <user>, <password> by quoted strings and
#    <port> by the master's port number (3306 by default).
#
#    Example:
#
#    CHANGE MASTER TO MASTER_HOST='125.564.12.1', MASTER_PORT=3306,
#    MASTER_USER='joe', MASTER_PASSWORD='secret';
#
# OR
#
# 2) Set the variables below. However, in case you choose this method, then
#    start replication for the first time (even unsuccessfully, for example
#    if you mistyped the password in master-password and the slave fails to
#    connect), the slave will create a master.info file, and any later
#    change in this file to the variables' values below will be ignored and
#    overridden by the content of the master.info file, unless you shutdown
#    the slave server, delete master.info and restart the slaver server.
#    For that reason, you may want to leave the lines below untouched
```

```
#    (commented) and instead use CHANGE MASTER TO (see above)

#

# required unique id between 2 and 2^32 - 1

# (and different from the master)

# defaults to 2 if master-host is set

# but will not function as a slave if omitted

#server-id       = 2

#

# The replication master for this slave - required

#master-host     =   <hostname>

#

# The username the slave will use for authentication when connecting

# to the master - required

#master-user     =   <username>

#

# The password the slave will authenticate with when connecting to

# the master - required

#master-password =   <password>

#

# The port the master is listening on.

# optional - defaults to 3306

#master-port     =  <port>

#

# binary logging - not required for slaves, but recommended
```

```
#log-bin=mysql-bin



# Point the following paths to different dedicated disks

#tmpdir         = /tmp/

#log-update     = /path-to-dedicated-directory/hostname



# Uncomment the following if you are using BDB tables

#bdb_cache_size = 384M

#bdb_max_lock = 100000



# Uncomment the following if you are using InnoDB tables

#innodb_data_home_dir = /var/lib/mysql/

#innodb_data_file_path = ibdata1:2000M;ibdata2:10M:autoextend

#innodb_log_group_home_dir = /var/lib/mysql/

#innodb_log_arch_dir = /var/lib/mysql/

# You can set .._buffer_pool_size up to 50 - 80 %

# of RAM but beware of setting memory usage too high

#innodb_buffer_pool_size = 384M

#innodb_additional_mem_pool_size = 20M

# Set .._log_file_size to 25 % of buffer pool size

#innodb_log_file_size = 100M

#innodb_log_buffer_size = 8M

#innodb_flush_log_at_trx_commit = 1

#innodb_lock_wait_timeout = 50
```

```
[mysqldump]
quick
max_allowed_packet = 16M

[mysql]
no-auto-rehash
# Remove the next comment character if you are not familiar with SQL
#safe-updates

[isamchk]
key_buffer = 256M
sort_buffer_size = 256M
read_buffer = 8M
write_buffer = 2M

[myisamchk]
key_buffer = 256M
sort_buffer_size = 256M
read_buffer = 8M
write_buffer = 2M

[mysqlhotcopy]
```

```
interactive-timeout
```

VI.3 FICHERO DE CONFIGURACIÓN DE APACHE (/ETC/HTTPD/CONF/HTTPD.CONF)

Solo adjuntamos las partes que hemos tenido que ir modificando del fichero según el tipo de prueba que realizábamos, el resto queda por defecto.

```
# prefork MPM

# StartServers: number of server processes to start

# MinSpareServers: minimum number of server processes which are
kept spare

# MaxSpareServers: maximum number of server processes which are
kept spare

# ServerLimit: maximum value for MaxClients for the lifetime of
the server

# MaxClients: maximum number of server processes allowed to
start

# MaxRequestsPerChild: maximum number of requests a server
process serves

<IfModule prefork.c>

StartServers       8

MinSpareServers    20

MaxSpareServers   20

ServerLimit      400

MaxClients       400

MaxRequestsPerChild  10000
```

```
</IfModule>
```

Hemos debido modificar durante las pruebas tanto el número máximo de clientes, debiendo también en algunos casos aumentar ServerLimit pues siempre debe ser superior o igual. Por otro lado hemos modificado según la prueba los parámetros "StartServers", "MinSpareServers", "MaxSpareServers" con tal de optimizar el rendimiento de la prueba.

VII APÉNDICE

VII 1 ILUSTRACIONES

VII 2 TABLAS