

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
GRADO EN INGENIERÍA DEL SOFTWARE

GALERÍA DE ARTE VIRTUAL (G.A.V.)
VIRTUAL ART GALLERY (V.A.G.)

Realizado por
Enrique Ríos Santos
Tutorizado por
Mariemma I. Yagüe Del Valle
Departamento
Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA
MÁLAGA, Diciembre del 2.015

Fecha defensa:
El Secretario del Tribunal

Resumen

Un **sistema de gestión de contenidos** es una plataforma que permite crear una estructura de soporte para la creación y administración de contenidos. Permite a los administradores y a los usuarios centrarse en el contenido de la temática de la página, foro o blog que en los aspectos técnicos de más bajo nivel. Por ejemplo, el uso de la base de datos es totalmente transparente para los administradores.

Este proyecto trata sobre la necesidad ficticia de desarrollar un **portal web artístico**. Contendrá información de artistas, sus obras y exposiciones que ellos mismo creen. Permitirá el registro de usuarios que podrán ver, comentar, valorar, comprar y pujar las obras disponibles. Unos de los objetivos será el de implementar sobre un gestor de contenidos gratuito, así como las demás herramientas utilizadas.

La primera parte del proyecto se centrará en la elección de la plataforma en la que se desarrollará. Seguidamente se conocerá un poco más a fondo el gestor de contenidos seleccionado. Tras diseñar la solución mediante diagramas de casos de usos, se detallará la implementación de los requisitos presentados anteriormente. También se creará un módulo propio que consistirá en un sistema, que permitirá a los usuarios pujar por las obras que se encuentren en subasta.

Palabras Claves

Gestor de contenidos, Drupal, portal web, desarrollo, programación

Abstract

A **content management system** is a platform that lets you create a support structure for the creation and content management. It allows administrators and users to focus on the content of the theme of the page, fórum or blog instead of the technical aspects of the lowest level. For example, the use of the database is completely transparent to administrators.

This project is about the fictional need to develop an **artistic website**. Contain information about artists, works and exhibitions that they themselves organise.. It will allow the user login in order he/she can comment, rate, buy and bid. One of the objectives will be to implement on a free content manager and other tools used.

The first part of the project will focus on the choice of the platform that will be developed. Then it is known a little more about the manager selected content. After designing the solution by use case diagrams, implementation of the requirements set forth above it will be detailed. Own module that will enable users to bid for the artworks under auction will be created too.

Keywords

CMS, Drupal, web portal development, programming

1	INTRODUCCIÓN	1
1.1	MOTIVACIÓN DEL PROYECTO	1
1.2	INTRODUCCIÓN CMS	1
1.3	ELECCIÓN DE GESTOR DE CONTENIDOS.....	3
1.4	DRUPAL.....	8
1.4.1	<i>Historia</i>	9
1.4.2	<i>Funcionamiento</i>	9
1.4.3	<i>Estructura de carpetas</i>	10
1.4.4	<i>Archivos importantes</i>	10
1.5	SOFTWARE UTILIZADO.....	11
2	CAPTACIÓN DE REQUISITOS	13
2.1	ACTIVIDAD DEL SISTEMA	13
2.2	TIPOS DE USUARIOS	13
2.2.1	<i>Usuario No Registrado</i>	13
2.2.2	<i>Usuario Registrado</i>	13
2.2.3	<i>Artista</i>	13
2.2.4	<i>Administrador</i>	14
2.3	REQUISITOS.....	14
3	DISEÑO	19
3.1	DIAGRAMAS DE CASOS DE USOS.....	19
4	INSTALACIÓN DE DRUPAL.....	21
4.1	INSTALACIÓN BÁSICA	21
4.2	INSTALACIÓN DEL IDIOMA ESPAÑOL.....	26
5	DESARROLLO.....	29
5.1	CONFIGURACIÓN DE ROLES.....	29
5.2	CONFIGURACIÓN DEL CONTENIDO	29
5.3	DATOS A ALMACENAR	30
5.4	AÑADIR CONTENIDO EN DRUPAL.....	32
5.4.1	<i>Obra</i>	32
5.4.2	<i>Exposición</i>	35
5.4.3	<i>Currículum Artístico</i>	36
5.5	MÓDULOS UTILIZADOS.....	37
5.6	MÓDULO PROPIO	38
5.6.1	<i>Módulo en Drupal</i>	39
5.6.2	<i>Servicio Web</i>	44
5.6.2.1	Petición	44
5.6.2.2	Respuesta.....	45
5.6.2.3	Código.....	45
5.6.2.4	Clase auxiliar para cifrado	50
5.6.3	<i>Persistencia</i>	50
5.6.4	<i>Excepciones</i>	50
5.7	PRUEBAS UNITARIAS.....	51
5.8	CONFIGURACIÓN DE MENÚS	57
5.8.1	<i>Usuario no registrado</i>	57
5.8.2	<i>Usuario Registrado</i>	58

5.8.3	Artista.....	60
5.9	SOLICITUD DE ARTISTA	61
5.10	BOLETÍN SEMANAL.....	62
5.11	CONFIGURACIÓN DEL ASPECTO	63
6	CONCLUSIONES.....	65
6.1	RESUMEN.....	65
6.2	MEJORAS Y TRABAJOS FUTUROS.....	65
6.3	APRENDIZAJE PERSONAL.....	66
6.4	PROBLEMAS TÉCNICOS ENCONTRADOS.....	66
	REFERENCIAS BIBLIOGRÁFICAS.....	67
	ANEXOS.....	69
1.	INSTALACIÓN DE UN MÓDULO	69
2.	CONFIGURACIÓN SMTP.....	70
3.	CASOS DE USOS	72

1 Introducción

Este primer capítulo se tomará como introducción a los gestores de contenidos. Se conocerá sus posibilidades y limitaciones. Una vez decidido el gestor de contenido que se usará en este proyecto, se tratará de conocer un poco más a fondo así como también su funcionamiento. Por último se comentará las herramientas utilizadas para la realización de este trabajo, siendo como requisito el uso **de software libre**.

1.1 Motivación del Proyecto

Este proyecto surge de la necesidad ficticia de crear una **galería de arte virtual** para gestionar todo lo relacionado con un portal web con dicha temática. El proyecto deberá cubrir las necesidades comunes a un sitio web donde se gestiona usuarios de distintos perfiles, listas de distribuciones de correo y demás funciones, así como las necesidades específicas de una galería de arte como exponer los distintos trabajos de los artistas o incluir un servicio de subastas de las obras artísticas por ejemplo.

La idea principal es la de realizar un portal funcional y completo a partir de un gestor de contenido gratuito. A pesar de ello, no se pretende que sea un proyecto en el cual solo consista en instalar un gestor de contenido y personalizarlo con la temática elegida. Se desarrollará un módulo para personalizar la funcionalidad adecuándose a los requisitos especificados. Consistirá en crear un sistema mediante el cual, los usuarios puedan pujar por las obras de los artistas.

Se seguirá una metodología ágil para el desarrollo del proyecto. En la presente memoria se detallarán las distintas fases captación de requisitos, análisis de la solución, desarrollo, pruebas unitarias, trabajos futuros...

1.2 Introducción CMS

CMS son las siglas en inglés de *Content Management System* que en español significa sistema de gestión de contenidos. Se trata de sistema que se instala en el servidor web, donde se puede administrar mediante su interfaz distintos aspectos para la creación, normalmente, de una página web. Cuando se usa un CMS facilita notablemente la creación de nuestro proyecto web, ya que trámites tan necesarios como un mecanismo para que el usuario se pueda identificar en la página viene ya implementado. Esto sin duda, ahorra tiempo al administrador.

Los CMS nacieron de la necesidad de añadir contenido a Internet de una forma fácil y rápida, ya que, a principios de los años 90, el crecimiento de organizaciones que publicaban gran cantidad de contenido en Internet, necesitaban de continuas actualizaciones y personalizar sus sitios web. Era el caso de revistas en línea, páginas de agencias de noticias, periódicos, publicaciones corporativas... Posteriormente, usuarios individualmente comenzaron a desear escribir su propio contenido y publicar sus propios textos, fotografías o vídeos.

1.2- Introducción CMS

En septiembre de 1.999 fue lanzado el primer CMS, *Xpedio*, que fue desarrollado por la empresa IntraNet Solutions. A partir del año 2.000 van apareciendo progresivamente los CMS más conocidos actualmente, como por ejemplo **Wordpress** en 2.003, **Drupal** en 2.001 o **Joomla!** En 2.005. Hoy en día podemos contar con miles de CMS, de diferente naturaleza, lenguaje o propósito.

El trabajo a realizar, una vez se ha escogido e instalado el CMS, es en menor o mayor medida, la de completar el sitio web con el contenido que se desee. Es decir, el objetivo de los CMS es librar al administrador de tareas de diseño e implementación y focalizar su esfuerzo en la productividad de su proyecto. Esto no quiere decir que el público objetivo sea un perfil técnicamente bajo, ya que las posibilidades que ofrece son muchos mayores, y podemos implementar nuestros propios módulos para darle una personalización que no nos ofrezca el CMS de forma nativa.

Recién instalado lo que ofrece un CMS u otro varía mucho, pero lo que tienen todos en común es la posibilidad de añadir funcionalidades fácilmente gracias a **módulos o plugins**. Los módulos o plugins son pequeños añadidos que extiende el funcionamiento de nuestro CMS. Existe una gran variedad, pero un abuso de estos extras puede provocar inestabilidad y errores en nuestro proyecto. Es recomendable no añadir más de lo estrictamente necesario.

Una de las características más importante es la **seguridad** que por un lado se ve incrementada al usar un CMS, ya que sus versiones estables han sido concienzudamente testeadas de tal forma que el número de errores sea el menor posible. Aunque por otro lado, como en cualquier proyecto software, existen fallos de seguridad que pueden poner en peligro nuestro sistema, pero si el CMS en cuestión que estemos usando dispone de una gran comunidad o los desarrolladores del proyectos están lo suficientemente comprometidos, será subsanado en poco tiempo. También cabe destacar, que si un CMS es muy usado, los “esfuerzos” de los hackers irán centrados a estudiar y vulnerar la seguridad del sistema.

El **diseño visual** es otro de los fuertes, ya que suelen existir mucha variedad de plantillas o temas que cambian por completo el aspecto de un sitio web. Normalmente es fácil instalarlas y aplicarlas. Aunque no siempre es buena idea cambiar el *skin* ya que muchos de ellos no están lo suficientemente probados y pueden llegar a generar errores o incompatibilidades con módulos o plugins ya instalados. Cuando deseemos actualizar nuestro CMS a una nueva versión debemos tener en cuenta la compatibilidad de las plantillas y módulos que tenemos instalados actualmente.

Como en cualquier elección que se haga, también los gestores de contenidos tienen puntos negativos:

- Tienes que adaptarte al CMS que elijas, con sus ventajas y desventajas. En un futuro puede que desees migrar a otro y eso puede ser una ardua tarea.
- Las actualizaciones pueden producir fallos en los módulos o plugins que estén instalados al no ser compatibles con la nueva versión.
- Menor velocidad. Dado que usamos un código que se genera automáticamente, el rendimiento será menor que el de un código implementado a medida.
- Como anteriormente se expuso, la seguridad puede ser una desventaja ya que al

1.3- Elección de Gestor de Contenidos.

ser un código abierto al público puede ser estudiado y explotado. No obstante, si el CMS cuenta con una comunidad lo suficientemente grande, será un punto a favor.

- **Diseño.** A pesar de que la mayoría de los gestores de contenido cuentan con innumerables capas de personalización, incluso muchas de ellas de pago, tal vez ninguna se adecue a nuestro proyecto completamente. Nuevamente, un proyecto implementado a medida será más cuidadoso en el aspecto visual.

1.3 Elección de Gestor de Contenidos.

En esta sección se decidirá el gestor de contenidos que usaremos para el proyecto. Es una decisión importante ya que deberemos elegir el gestor de contenidos adecuado para la dimensión y naturaleza de nuestro proyecto. Debemos tener en cuenta que hay que elegir un CMS según las características de nuestro trabajo, y no elegirlo porque es el que usamos habitualmente o es el más usado por los desarrolladores de sitios web. Cada proyecto requiere de distintos esfuerzos, por eso debemos de ser cuidadosos a la hora de decantarnos por el CMS final.

Primero de todo, descartamos los CMS que sean privativos o de pago, ya que en el objetivo de este proyecto trata de desarrollarlo sobre un **sistema libre y gratuito**. De este modo quedan descartados *Alterain, Cascade Server, Contegro, DotNetNuke, Dot.orange, Ektron, Elcom Technology, EpiServer...*

No todos los CMS están enfocados para el mismo uso. Se pueden clasificar según funcionalidad:

- **Blogs;** pensados para páginas de propósitos personales.
- **Foros;** pensados para compartir opiniones de usuarios visitantes a la página web.
- **Wikis;** pensados para el desarrollo colaborativo.
- **Enseñanza;** plataforma para contenidos de enseñanza on-line.
- **Comercio electrónico;** plataforma de gestión de usuarios, catálogo, compras y pagos.
- **Publicaciones digitales;** para desarrollar periódicos o revistas digitales.
- **Difusión de contenido multimedia;** para distribuir audio e imagen principalmente.
- **Propósito general;** Ninguno de los anteriores, la temática y el contenido lo decide el desarrollador.

Dado la anterior clasificación y la naturaleza de nuestro proyecto debemos centrar la elección en el último grupo, propósito general.

Un buen recurso online para conocer los diferentes CMS y detalles sobre estos, es <http://www.cmsmatrix.org>. Se trata de una página web donde nos listan todos los gestores de contenidos filtrados por los criterios que elijamos. La lista que nos ofrece es incluso demasiado extensa. Estamos hablando de que la lista tiene más de 1.280 CMS diferentes, por

1.3- Elección de Gestor de Contenidos.

lo que intentar comparar todos resulta imposible. Para filtrar el número de CMS, debemos ordenarlos por algún tipo de criterio.

Volviendo a lo que nos interesa podemos buscar los gestores de contenido que en principio nos pueda interesar y realizar una comparación para ayudarnos a tomar decisiones.

Como comentamos anteriormente, necesitamos un CMS de propósito general. En esta categoría nos encontramos con **Wordpress, Drupal y Joomla!**. Son los más populares y con mayor apoyo de la comunidad. Esto último es un punto a favor respecto a otros CMS desconocidos. Una comunidad activa hace mejorar al producto, ya que se detectan posibles errores más rápidamente o se desarrollan módulos o extensiones, que nos pueden resultar útiles en un futuro. En definitiva, un CMS crecerá más eficientemente si cuenta con un gran número de usuarios detrás. Veamos una comparativa de los tres:

	Drupal	Joomla!	WordPress
Última Versión	7.28	3.3	3.9.1
Precio	Gratuito	Gratuito	Gratuito
Base de datos	MySQL	MySQL	MySQL
Licencia	Open Source	Open Source	Open Source
Lenguaje programación	PHP	PHP	PHP
Acceso Root	No	No	No
Seguridad			
Captcha	Mediante extensión	Mediante extensión	Mediante extensión
Verificación Email	Sí	Sí	Sí
Historial de login	Sí	Sí	Mediante Extensión
SSL Compatible	Sí	Sí	Sí
SSL Logins	Mediante Extensión	Sí	Sí
SSL Páginas	Mediante Extensión	Sí	Limitado
Facilidad de uso			
Contenido Arrastrar y Soltar	Mediante Extensión	No	Sí
Discusión por email	Mediante Extensión	Mediante Extensión	Mediante Extensión
URLs Amigables	Sí	Sí	Sí
Redimensionar Imágenes	Mediante	Sí	Sí

1.3- Elección de Gestor de Contenidos.

	Drupal	Joomla!	WordPress
	Extensión		
Suscripciones	Mediante Extensión	Sí	Mediante Extensión
Plantilla de lenguaje	Sí	Sí	No
WYSIWYG Editor	Mediante Extensión	Sí	Sí
Archivos Zip	No	No	Mediante Extensión
Administración			
Gestión de publicidad	Advertising Management	Sí	No
Gestión de activos	Sí	Sí	Sí
Portapapeles	No	No	No
Programación contenidos	Mediante Extensión	Sí	Limitado
Administración Online	Sí	Sí	Sí
Sub-rutas	Sí	Sí	Sí
Temas / Skins	Sí	Sí	Sí
Papelera	No	Sí	Sí
Estadísticas Web	Sí	Sí	Mediante Extensión
Aplicaciones			
Blog	Sí	Sí	Sí
Chat	Mediante Extensión	Mediante Extensión	Mediante Extensión
Clasificados	Mediante Extensión	Mediante Extensión	Mediante Extensión
Foro	Sí	Mediante Extensión	Mediante Extensión
Calendario de eventos	Mediante Extensión	Mediante Extensión	Mediante Extensión
FAQ	Sí	Sí	Mediante Extensión
Gráficas	Mediante Extensión	Mediante Extensión	Mediante Extensión

1.3- Elección de Gestor de Contenidos.

	Drupal	Joomla!	WordPress
Libro de visita	Mediante Extensión	Mediante Extensión	Mediante Extensión
Contacto por formulario	Mediante Extensión	Sí	Mediante Extensión
Periódico	Mediante Extensión	Mediante Extensión	Mediante Extensión
Galería de fotos	Mediante Extensión	Mediante Extensión	Sí
Encuestas	Sí	Sí	Mediante Extensión
Motor de búsqueda	Sí	Sí	Sí
Mapa del sitio	Mediante Extensión	Mediante Extensión	Mediante Extensión
RSS	Sí	Sí	Sí
Tests / Concursos	Mediante Extensión	Mediante Extensión	Mediante Extensión
Wiki	Mediante Extensión	Mediante Extensión	Mediante Extensión
Comercio			
Administración de inventario	Mediante Extensión	Mediante Extensión	Mediante Extensión
Gestión de pagos	Mediante Extensión	Mediante Extensión	Mediante Extensión
Gestión de envío	Mediante Extensión	Mediante Extensión	Mediante Extensión
Gestión de impuestos	Mediante Extensión	Mediante Extensión	Mediante Extensión
Puntos de venta	Mediante Extensión	Mediante Extensión	Mediante Extensión
Carro de compra	Mediante Extensión	Mediante Extensión	Mediante Extensión
Suscripciones	Mediante Extensión	Mediante Extensión	Mediante Extensión
Lista de deseos	Mediante Extensión	Mediante Extensión	Mediante Extensión

Sólo se ha tenido en cuenta las extensiones gratuitas. Las tres plataformas cuentan con

1.3- Elección de Gestor de Contenidos.

extensiones de pago, pero como el objetivo del proyecto es hacerlo sobre un software de código abierto y gratuito no se han tenido en cuenta para la comparativa.

Como se puede observar cualquiera de los tres nos serviría para realizar el proyecto, ya que de forma nativa o mediante alguna extensión prácticamente cubren nuestras necesidades.

Todos ellos están escritos en **PHP**. Están compuestos por un núcleo y módulos básicos de gestor de contenido, así como *plugins* para añadir funcionalidades adicionales. También constan de innumerables plantillas para aplicar un diseño rápido al sitio web (los hay gratuitos y de pagos, nuevamente nosotros nos centraremos en el primer grupo). Los tres CMS cuentan con una interfaz para desarrolladores y en principio no se requieren grandes conocimientos de programación, tal vez *Drupal* sea el más exigente en ese sentido. Eso es un punto a favor para elegir el CMS final, ya que no queremos utilizar un CMS que nos limite, sino que nos permita desarrollar a nosotros mismo las funcionalidades que no se puedan cubrir con módulos o *plugins*, o simplemente estos últimos no sean de nuestro agrado. Lógicamente todo depende de la magnitud del proyecto, si se pretende hacer una web sencilla o un blog personal, no se necesita ningún conocimiento de programación, a medida que el proyecto se vuelva más exigente, necesitaremos estar mejor formados en este ámbito para poder afrontarlo satisfactoriamente.

¿Qué perfiles pueden trabajar con estos CMS? Básicamente vamos a diferenciar en dos perfiles, los administradores *site builders* y los *programadores puros*. Al primer grupo pertenecen los administradores que construyen sus portales web a base de click y ciñéndose exclusivamente a las posibilidades que le brindan los CMS y las extensiones (módulos y *plugins*) que estén disponibles. Dicho perfil no requiere de conocimientos de programación, ya que la construcción del sitio se hará a base de editores gráficos y herramientas diseñadas para tal fin. Uno de los objetivos de los CMS es ofrecer a este perfil, la posibilidad de construir un *website* de forma sencilla, rápida y segura sin tener conocimientos de programación.

El segundo perfil, el de los programadores puros, consta de profesionales que son capaces de leer e interpretar el código de los CMS y programar módulos para implementar una funcionalidad en concreto y de forma personalizada. Es decir, es capaz de ampliar las posibilidades que ofrece los gestores de contenidos.

Este proyecto está pensado para una mezcla de ambos, ya que vamos a aprovechar las ventajas que ofrecen los CMS pero también ampliar y/o crear nuevas funcionalidades. Para ello será necesario conocimiento sobre PHP, que es el lenguaje en el que están escritos los CMS que estamos estudiando utilizar y con el que finalmente trabajaremos.

Tanto Joomla! como Wordpress están enfocados para el perfil de *site builders*, mientras que Drupal lo está para los *programadores puros*. Esto no quiere decir que están enfocados exclusivamente para ese grupo de administradores. Cualquiera de los tres puede ser usado por un administrador de los dos grupos. Una vez más, tiene un peso mayor el tamaño y naturaleza del proyecto que el perfil del administrador.

¿Qué ofrece cada uno recién instalados? Uno de los requisitos de cualquier proyecto software es la estabilidad y más cuando estamos hablando de un portal web, el cual será

1.4- Drupal

interactivo con los usuarios y accesible las 24 horas. Uno de los motivos por lo que un CMS se puede volver inestable es la cantidad de módulos o *plugins* que tenga instalado. El riesgo aumenta si las extensiones son de terceros y no son sometidos a pruebas de testeo suficientemente estrictas para garantizar un funcionamiento aceptable. Otro de los problemas viene a la hora de las actualizaciones, podemos actualizar la versión de nuestro CMS y que ésta no sea compatible con alguna extensión, esto nos creará problemas de estabilidad. A mayor número de módulos instalados, es más fácil que alguno de ellos produzca una inconsistencia y perjudique al buen comportamiento del sistema. Por todo esto, es importante tener en cuenta lo que nos ofrece nuestro CMS recién instalado, es decir, sin instalar ningún módulo o *plugins* adicional.

Drupal en principio ofrece menos módulos básicos tras su instalación que *Joomla!* y *Wordpress*. Éstos últimos ofrecen más funcionalidad tras ser instalado, pero esto no es un punto negativo, ya que es multipropósito, así que podemos crear nuestro sitio más preciso. Por ejemplo, Drupal puede no incluir un módulo que para *Joomla!* o *Wordpress* sea básico, pero para nuestro proyecto es prescindible, por lo tanto nuestro sitio crecerá solo con los módulos que realmente son necesarios para nosotros. Así pues, tenemos más flexibilidad y libertad a la hora de añadir funcionalidades con *Drupal*.

En cuanto al sistema de módulos, hay que resaltar que el de *Drupal* tiene más granularidad, es decir, que cuando se quiere incorporar una nueva funcionalidad tal vez no exista un módulo que lo realice directamente, sino que tengamos que instalar varios, cada uno con una “subrutina” específica, que en su conjunto consigue nuestra funcionalidad deseada. Esto puede ser contraproducente, ya que puede ser considerado como una ventaja, ya que si quieres una funcionalidad, la puedes de alguna forma diseñar a tu gusto añadiendo módulos menores hasta lograrla. Pero también resulta más cómodo añadir solo un módulo que cubra tus necesidades. En definitiva pensamos que es mejor poseer una mayor granularidad ya que así podemos adaptar mejor los módulos a nuestras necesidades, es decir, podemos construir nuestra funcionalidad a nuestro gusto mediante pequeños módulos utilizando la máxima de **divide y vencerás**.

Otra de las ventajas de *Drupal* es la posibilidad de hacer cambios en el tablero de administración, al contrario que en *Joomla!* y *Wordpress* donde es bastante más complicado realizar los cambios que queramos.

En cuanto a la curva de aprendizaje la de *Joomla!* y *Wordpress* son similares, mientras que la de *Drupal* es notoriamente más dura. Las razones son que *Drupal* está más orientado a un perfil programador. Dispone de menos módulos a comparación con los otros dos CMS, por lo que elaborar una funcionalidad puede ser más tedioso.

En conclusión, vamos a optar por *Drupal* ya que nos parece que es la opción que mejor se ajusta a nuestro proyecto. Nos va a ofrecer más posibilidades ya que es más flexible, aunque por otra parte, eso nos añadirá más dificultad.

1.4 Drupal

Este capítulo tiene como objetivo conocer un poco más el gestor de contenidos elegido. Se

1.4- Drupal

comentará cómo y quién lo creó y se detallará su funcionamiento.

1.4.1 Historia

CMS lanzado oficialmente el 1 de enero de 2.001. Es distribuido bajo los términos de la licencia *GNU/GPL*. En 1.999 *Dries Buytaert* inició el desarrollo de *Drupal* en solitario. Su idea inicial era crear un tablón de anuncios y mensajes para utilizarlo de modo personal. Tras un año, observó que mucha gente se empezaba a interesar por su proyecto, viendo así una oportunidad para avanzar en el mismo. Para ello, decidió seguir con el proyecto haciéndolo código abierto, para invitar a otros programadores al desarrollo.

Dries se centró en la investigación de canales RSS, la moderación de contenido y otras tecnologías de Internet. En 2.001, *Drupal.org* pasó a entrar en funcionamiento. Pero no fue hasta 2.005, cuando tomó notoriedad en la red.

Hoy en día se estima que la comunidad de *Drupal* asciende a más de **630.000 usuarios y desarrolladores**, lo cual hace que el soporte, actualización y expansión sea constante.

Para hacer funcionar *Drupal* solo se necesita un servidor web capaz de ejecutar PHP como *Apache*, *Cherokee*, *Lighttpd*... y una base de datos como *MySQL*, *PostgreSQL*, *SQLite*... Los requisitos para la versión 6 de Drupal son PHP 4.4.0 o superior, mientras que para la versión 7 se precisa una versión 5.2.5 o superior de PHP.

Se calcula que se usa en más de **1.5 millones de sitios web**, principalmente es notorio en los dominios .edu, de los cuales el 39% reportan el uso de Drupal. Por ejemplo el mismo gobierno americano confía en este CMS para desarrollar la página web de la casa blanca (www.whitehouse.gov). Prueba de que es un CMS actual y totalmente vivo es que la organización de los juegos olímpicos del año 2.016 de Río de Janeiro también ha optado por desarrollar su sitio web con Drupal (www.rio2016.com)

1.4.2 Funcionamiento

El funcionamiento se basa en la interacción del núcleo con los módulos de *Drupal*. El núcleo provee las funcionalidades básicas de gestión de las distintas entidades y una interfaz de programación que los módulos utilizan para implementar nuevas funcionalidades.

Las funcionalidades de los módulos varían ampliamente en su propósito, actualmente Drupal.org reporta más de 18.000 módulos publicados en sus diferentes versiones.

Gracias a la arquitectura modular de *Drupal*, cualquier desarrollador con conocimientos de PHP puede crear sus propios módulos para implementar características y funcionalidades adicionales a las de la instalación base. Una vez desarrollados deben de ser enviados para una revisión de seguridad y estándares, una vez aprobados son publicados en Drupal.org.

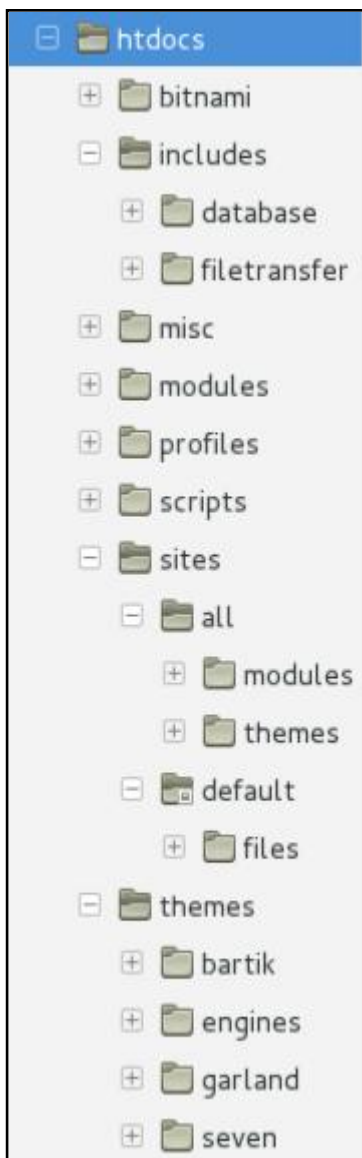
Cada módulo publicado cuenta con repositorio de versiones GIT y sistema de reporte de casos donde los usuarios del módulo reportan errores, realizan solicitudes de nuevas características y proponen cambios al código por medio de parches o comentarios.

1.4- Drupal

1.4.3 Estructura de carpetas

Al instalarse *Drupal* se crean de forma automática muchas carpetas. En este apartado comentaremos las más importantes.

- **includes:** Contiene la infraestructura del núcleo de *Drupal*.
 - **includes/database:** contiene la lógica del sistema de abstracción de bases de datos.
 - **includes/filetransfer:** contiene código para transferencias de archivos con ftp y ssh.
- **misc:** JavaScript, iconos, imágenes...



- **modules:** Contiene todos los módulos instalados por defecto.
- **profiles:** Incluye los perfiles de instalación.
- **scripts:** Para ejecutar línea de comandos.
- **sites:** Contiene los módulos que instalemos y configuremos, además de los archivos de los usuarios para la gestión de su contenido.
 - **sites/default:** Se almacena el archivo settings.php, con la configuración del sitio web.
 - **sites/default/files:** Se almacenan los archivos que se suben al sitio web, como imágenes de usuarios y archivos multimedia.
 - **sites/all/modules:** Contiene los módulos que hemos instalados.
 - **sites/all/themes:** Contiene los temas que hemos instalados.
- **themes:** Contiene todos los temas instalados por defecto.

1.4.4 Archivos importantes

En el directorio raíz podemos encontrar varios archivos importantes para nuestro sitio web. A continuación los vamos a describir brevemente:

- **authorize.php:** script administrativo para ejecutar operaciones autorizadas como instalar un nuevo tema o módulo desde *Drupal*.
- **cron.php:** ejecuta tareas periódicas
- **index.php:** punto de entrada para las peticiones del

1.5- Software utilizado

usuario

- **install.php:** punto de entrada para el instalador de *Drupal*
- **robots.txt:** implementación por defecto del estándar de exclusión
- **update.php:** actualiza el esquema de base de datos después de una actualización de la versión de Drupal.
- **xmlrpc.php:** recibe peticiones XML-RPC y debe ser borrado cuando no se pretenda recibir estas peticiones.

1.5 Software utilizado

Para el diseño de los diagramas de casos de uso, hemos elegido el programa *argoUML*, ya que se trata de un sencillo programa software libre con el que hacer varios tipos de diagramas de fácil manejo.

Para realizar el proyecto de una forma más ordenada, simple y segura, hemos optado por crear una máquina virtual donde instalaremos, configuraremos e implantaremos todo lo necesario para llevar a cabo la creación del portal web. Para este propósito existe una página que nos resulta muy útil, www.bitnami.com. Es una página donde recopila máquinas virtuales con multitud de entornos para desarrollar casi cualquier proyecto que podamos pensar. En nuestro caso encontramos con dos máquinas virtuales que nos podrían resultar adecuadas para este proyecto.

- La primera se trata de una máquina virtual con todo lo necesario para empezar un proyecto con Drupal. El sistema operativo es Ubuntu 14.04, que a fecha de la redacción de esta memoria es la última versión estable de esta popular distribución de Linux. Para el buen funcionamiento de Drupal ya viene preinstalado **MySQL 5.5.39**, **PHP 5.4.32**, **Apache 2.4.10** y **phpMyAdmin 4.2.7.1** entre otras utilidades. La versión de **Drupal** que nos ofrece es la **8.0.0-alpha14**. Por lo que no es una versión estable, ni tan siquiera se trata de una versión Beta. Esto es una cosa que debemos tener en cuenta, ya que no es recomendable instalar la última versión que esté disponible si no es una versión estable. En caso de desarrollar el proyecto en una versión Alpha nos exponemos a fallos de compatibilidad, bugs aún no corregidos y demás problemas típicos de las versiones aún en desarrollo. A cambio, podremos disfrutar de las últimas mejoras, no siendo muy notables sobre la última versión estable. Por todo esto, descartamos la utilización de ésta máquina virtual.
- La segunda consta con la versión 7.6.0 de Debian con la preinstalación de la suite “**LAMP**”. LAMP es un acrónimo usado para describir un sistema de infraestructura de internet que usa las siguientes herramientas:

Linux: Kernel en el que se basa el sistema operativo, en nuestro caso Debian 7.6.0.

Apache: Servidor web con la versión 2.4.10

MySQL: Base de datos con la versión 5.6.20

1.5- Software utilizado

PHP, Perl y Python: Lenguajes de programación con la versión 5.6.0 de PHP que es la que más nos interesa.

Por último cabe mencionar que esta suite se puede encontrar para otras plataformas, como son Windows (**WAMP**) o Mac (**MAMP**).

Por tanto, constamos con un sistema operativo en el cual está instalado lo necesario para empezar a usar Drupal (y prácticamente cualquier proyecto web), por lo que solo nos falta instalar la última versión de *Drupal*. A fecha de la redacción de esta memoria dicha versión es la **7.31**.

Para el desarrollo del módulo propio se utilizará *Netbeans* como entorno de programación. Desplegaremos la aplicación creada en un servidor de aplicaciones gratuito como es *Glassfish*.

2 Captación de requisitos

Este capítulo tratará sobre la documentación de los requisitos del proyecto. Se detallará las necesidades de la plataforma, así como los objetivos a cubrir. Se listará todos los tipos de usuarios que compondrá el sitio web, como también sus roles.

2.1 Actividad del sistema

La actividad normal de nuestro sitio consistirá en el almacenamiento y exposición de las obras de los artistas registrados en la plataforma. Podrán exponerlas ordenadas por exposiciones o individualmente. Los usuarios registrados, podrán, aparte de visualizar todas las obras, pujar o directamente comprar las obras que les interese. Los administradores por su parte tendrán todos los privilegios y podrán gestionar de forma manual cada una de las funciones de los perfiles restantes. También deberán moderar los comentarios que los usuarios realicen en las obras que los artistas publiquen.

2.2 Tipos de Usuarios

En el sistema se podrá encontrar cuatro tipos de usuarios. A continuación se expondrá sus principales características.

2.2.1 Usuario No Registrado

Este perfil de usuario será el más limitado. Cualquier usuario de internet que acceda a la página será un usuario no registrado. Solo podrá realizar dos acciones, registrarse en la plataforma para pasar a ser un usuario registrado o identificarse con su cuenta de usuario, para así poder disfrutar de todos sus privilegios.

2.2.2 Usuario Registrado

El usuario registrado podrá visualizar todas las exposiciones de los artistas. En las obras de las exposiciones podrá realizar comentarios, pujar sobre las obras que les interese comprar si éstas están a la venta con dicha modalidad o comprar directamente si así está estipulado por el artista de dicha obra. También existirá un sistema de votación, mediante el cual podrán valorar las obras.

2.2.3 Artista

Este perfil solo estará disponible de forma limitada. Será un administrador, quien dará de alta a cada artista que previamente lo solicite. Cada artista podrá crear cuantas exposiciones quiera. Tendrá posibilidad de crear, modificar o eliminar obras. Podrá ampliar sus datos de perfil y currículum cuando lo desee.

Además de las características propias del perfil, también posee las de usuario registrado.

2.3- Requisitos

2.2.4 Administrador

Este perfil de usuario será el encargado de la administración del portal web. No cualquiera puede solicitar ser administrador, tan solo habrá un número limitado de ellos.

Una de las tareas principales es la de administración de las cuentas de los usuarios. Podrá dar de alta, modificar o dar de baja cualquier tipo de cuenta existente en el sistema. Es el encargado de moderar los comentarios que los usuarios vayan introduciendo. Administra todo el contenido del sitio, para un correcto funcionamiento, y mantenimiento del sistema. Añade nuevas exposiciones, detalles de cada uno de los cuadros, galerías multimedia donde se exponen los mismos...

2.3 Requisitos

En esta sección se detallarán los requisitos tanto funcionales como no funcionales del sistema. Todos ellos irán identificados por un código para facilitar la trazabilidad en un futuro. El patrón que se seguirá será el siguiente:

[RF | RNF][000 .. 999]

Es decir, empezarán por un **RF** en caso de los requisitos funcionales y por **RNF** en caso de los no funcionales. Seguidamente se le asignará un número de tres cifras, empezando por el 000 y acabando en el 999 si fuese necesario.

Nombre	RF0001 – Registro de usuarios
Descripción	Cualquier visitante a la web tiene la posibilidad de registrarse en la misma
Actores Principales	Usuario no registrado

Nombre	RF0002 – CRUD de Artista
Descripción	Un administrador crea/consulta/modificar/eliminar un perfil de artista. Consultar significa ver todos los datos registrado en el sistema.
Actores Principales	Administrador

Nombre	RF0003 – CRUD exposición
Descripción	Un administrador puede crear, modificar y eliminar una exposición de cualquier artista. Un artista puede crear, modificar y eliminar una exposición que sea suya

2.3- Requisitos

Actores Principales	Administrador, Artista
----------------------------	------------------------

Nombre	RF0004 – CRUD obra
Descripción	Un administrador puede crear, modificar y eliminar una obra de cualquier exposición. Un artista puede crear, modificar y eliminar una obra que sea suya
Actores Principales	Administrador, Artista

Nombre	RF0005 – Solicitar nueva contraseña
Descripción	Cuando un usuario no recuerda de su contraseña, puede solicitar una nueva mediante un formulario.
Actores Principales	Todos

Nombre	RF0006 – Iniciar Sesión
Descripción	Antes de acceder al contenido es necesario iniciar sesión para identificarse en el sistema
Actores Principales	Todos

Nombre	RF0007 – CRUD de Usuarios
Descripción	Un administrador crea/consulta/modificar/eliminar un perfil de usuario. Consultar significa ver todos los datos registrado en el sistema.
Actores Principales	Administrador

Nombre	RF0008 – Bloquear cuenta de usuario
Descripción	En determinadas ocasiones un usuario con los suficientes privilegios puede decidir bloquear la cuenta de otro de forma temporal
Actores Principales	Administrador

2.3- Requisitos

Nombre	RF0009 – Comentar una obra
Descripción	El actor puede escribir un comentario en cualquier obra
Actores Principales	Usuario Registrado, Artista, y Administrador

Nombre	RF0010 – Pujar por una obra
Descripción	El actor puede pujar por una obra que se esté subastando
Actores Principales	Usuario Registrado, Artista, y Administrador

Nombre	RF0011 – Comprar Obra
Descripción	El actor puede comprar una obra que esté a la venta.
Actores Principales	Usuario Registrado, Artista, Administrador y Administrador de Drupal.

Nombre	RF0012 – Gestionar Currículum Vitae artístico
Descripción	El artista puede gestionar su propio currículum vitae artístico donde podrá indicar sus méritos e hitos destacados.
Actores Principales	Artista

Nombre	RF0013 – Crear Subasta
Descripción	El artista puede seleccionar que obras que quiera poner a la venta en forma de subasta.
Actores Principales	Artista

Nombre	RF0014 – Valorar Obra
Descripción	El actor puede valorar una obra mediante una calificación
Actores Principales	Usuario Registrado, Artista y Administrador

Nombre	RF0015 – Añadir Obra a Exposición
Descripción	Un artista podrá añadir una obra suya a una exposición también suya. Los administradores podrán añadir una obra cualquiera a una

2.3- Requisitos

	exposición que sea del mismo artista que la obra.
Actores Principales	Artista, Administrador

Nombre	RF0016 – Eliminar Obra de exposición
Descripción	Un artista podrá eliminar una obra suya de una exposición también suya. Los administradores podrán eliminar una obra cualquiera a una exposición que sea del mismo artista que la obra.
Actores Principales	Artista, Administrador

Nombre	RF0017 – Moderar Comentario
Descripción	El actor principal podrá moderar un comentario de cualquier usuario realizado en cualquier obra. Podrá modificarlo o eliminarlo.
Actores Principales	Administrador

Nombre	RF0018 – Formulario
Descripción	El actor principal podrá rellenar un formulario para hacérselo llegar a los administradores.
Actores Principales	Todos

Nombre	RF0019 – Solicitud de Artista
Descripción	El actor principal podrá Solicitar mediante un formulario el deseo de ser considerado como artista, y así poder disfrutar de los privilegios de este rol.
Actores Principales	Usuario Registrado

Nombre	RF0020 – Conversión de Usuario a Artista
Descripción	El actor principal podrá Solicitar mediante un formulario el deseo de ser considerado como artista, y así poder disfrutar de los privilegios de este rol.
Actores Principales	Administrador

2.3- Requisitos

3 Diseño

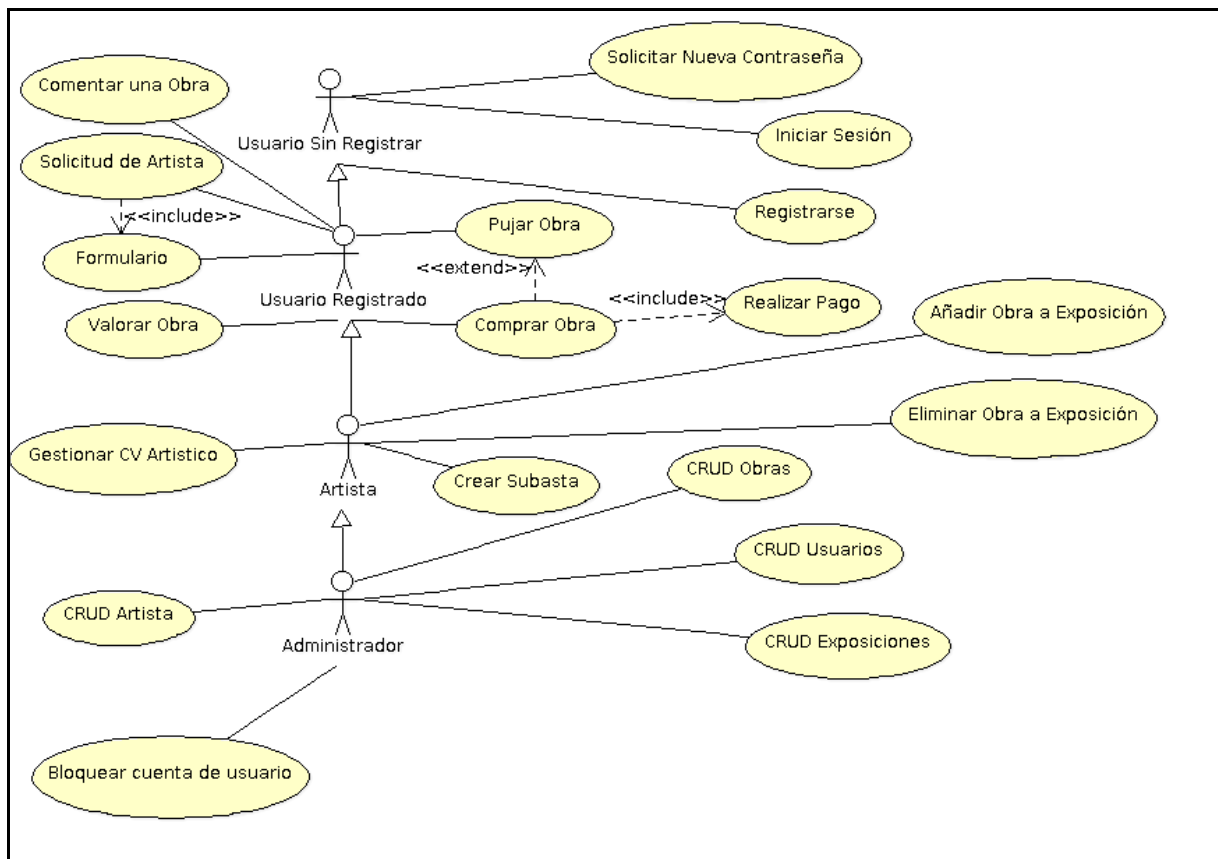
Este capítulo trata sobre el diseño de la solución, un planteamiento lógico de cómo se va a abordar el problema para satisfacer los requisitos previamente identificados.

3.1 Diagramas de Casos de Usos

En esta sección incluiremos los diagramas de casos de usos implicados en el desarrollo del sistema. Se identifican con un número de tres cifras precedidas de CU (Caso de Uso). En el anexo se detalla cada caso de uso mediante un cuadro.

Los usuarios están generalizados, es decir, el usuario que se encuentra debajo de otro, posee los privilegios del primero. Por lo tanto el perfil que se encuentra arriba, es el usuario con menos privilegios, y éste es, el usuario no registrado. Por el contrario el usuario con más privilegios y por lo tanto se encuentra en la parte baja, es el administrador.

Administrador *Drupal* no está incluido en el diagrama de casos de uso, ya que no pertenece a la lógica de la galería de arte virtual.



3.1- Diagramas de Casos de Usos

4 Instalación de Drupal

4.1 Instalación Básica

Como se comentó en la sección donde se comenta el software utilizado usaremos una máquina virtual con *Debian* en la cual ya viene instalado gran parte de lo que necesitamos. *Apache*, *MySql* y *PHP* entre otras herramientas.

Arrancamos los servidores con el comando `ctlscript.sh start`, obteniendo la salida:

```
usuario@debian:~/Lampstack-5.4.31-0$ ./ctlscript.sh start
140916 18:24:35 mysqld_safe Logging to '/home/usuario/Lampstack-5.4.31-0/mysql/data/mysqld.Log'.
140916 18:24:35 mysqld_safe Starting mysqld.bin daemon with databases from /home/usuario/Lampstack-5.4.31-0/mysql/data
/home/usuario/Lampstack-5.4.31-0/mysql/scripts/ctl.sh : mysql started at port 3306
Syntax OK
/home/usuario/Lampstack-5.4.31-0/apache2/scripts/ctl.sh : httpd started at port 8080
```

Tal y como indica en la última línea, el servidor web Apache se encuentra escuchando en el puerto 8080, además podemos apreciar como el servidor *MySQL* también ha sido arrancado. Comprobamos que lo primero es así:

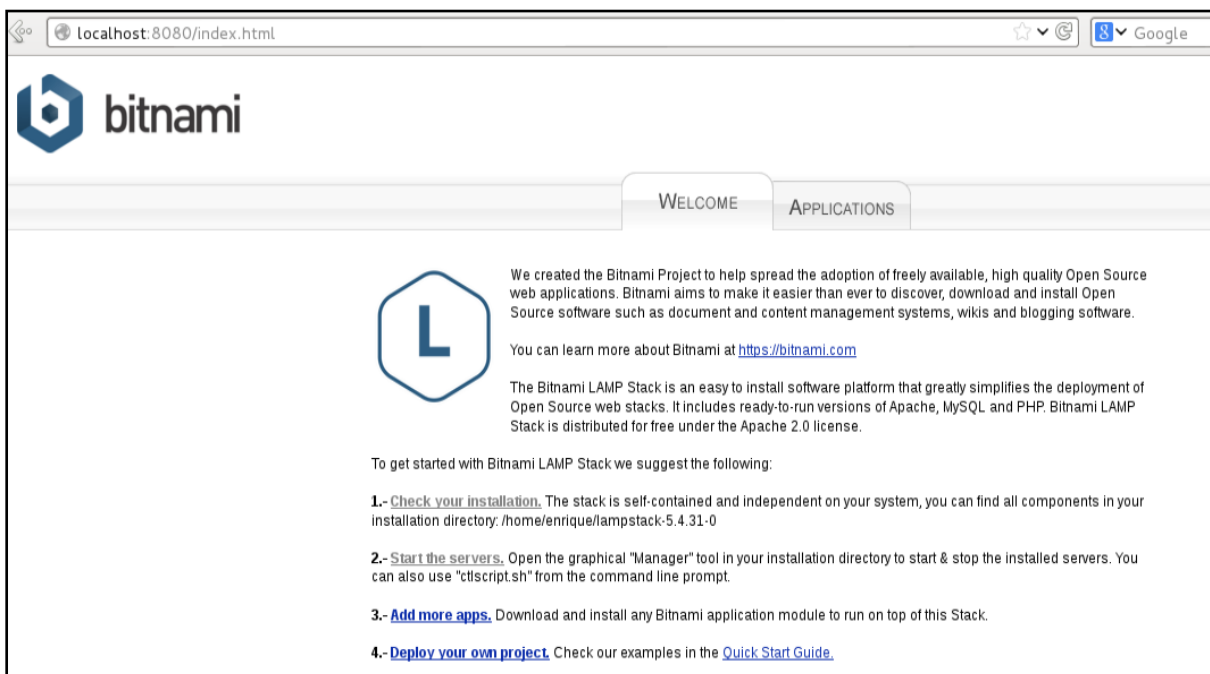


Figura 4.1- Pantalla de bienvenida del servidor Apache

Accedemos con el navegador a nuestro propio equipo por el puerto 8080 (<http://localhost:8080>) y podemos observar una pantalla de bienvenida que por defecto han

4.1- Instalación Básica

colocado los creadores de la herramienta Binami.

Para mayor comodidad vamos a configurar nuestro archivo `/etc/hosts` para que cuando accedamos desde nuestro navegador a la dirección <http://gav.es:8080> automáticamente nos redirija hacia la IP de la máquina virtual donde está el servidor web Apache. Es decir, desde la máquina anfitriona, podemos acceder a nuestro sitio web desde el dominio `gav.es` a través del puerto 8080.

Ahora tenemos todo preparado para proceder con la instalación de Drupal. Primero accedemos a la página oficial <http://www.drupal.org> y comprobamos la última versión disponible de forma estable. Se trata de la 7.31. Como comentamos anteriormente, existen versiones posteriores, pero aún en fase de desarrollo. Optamos por la última estable ya que así evitaremos fallos de estabilidad y bugs no corregidos aún.

Se nos descarga un comprimido donde viene una carpeta con el nombre de Drupal y la versión. Tras descomprimirla en un lugar de fácil acceso, como puede ser la carpeta personal de nuestro usuario, abrimos un archivo de texto con el nombre de `INSTALL`. En él podemos leer como debemos instalar nuestro CMS. La instalación es sencilla, tan solo tendremos que copiar el contenido de la carpeta anteriormente descomprimida al directorio público de nuestro servidor web. En este caso, dicho directorio se encuentra en

```
/home/usuario/lampstack-5.4.31-0/apache2/htdocs
```

Podemos comprobar que se encuentran los archivos generados por Bitnami. Los borramos y copiamos los archivos de Drupal, estando en la carpeta personal (*/home/usuario*)

```
rm -r Lampstack-5.4.31-0/apache2/htdocs
```

```
cp -r drupal-7.31/* drupal-7.31/.htaccess Lampstack-5.4.31-0/apache2/htdocs
```

Comprobamos que todo ha ido bien dirigiéndonos al navegador accediendo por el puerto 8080

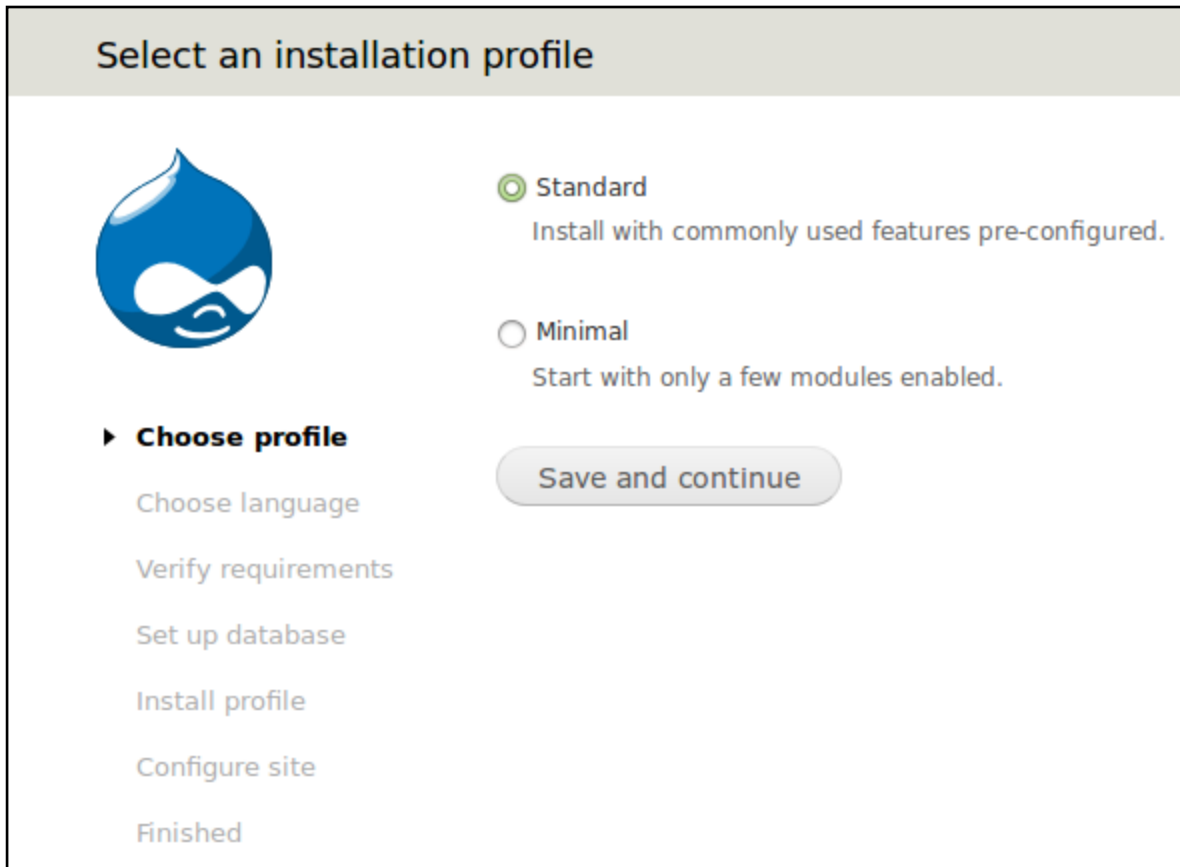


Figura 4.2- Primera pantalla de la instalación de Drupal

Efectivamente podemos ver la pantalla de instalación de Drupal.

4.1- Instalación Básica

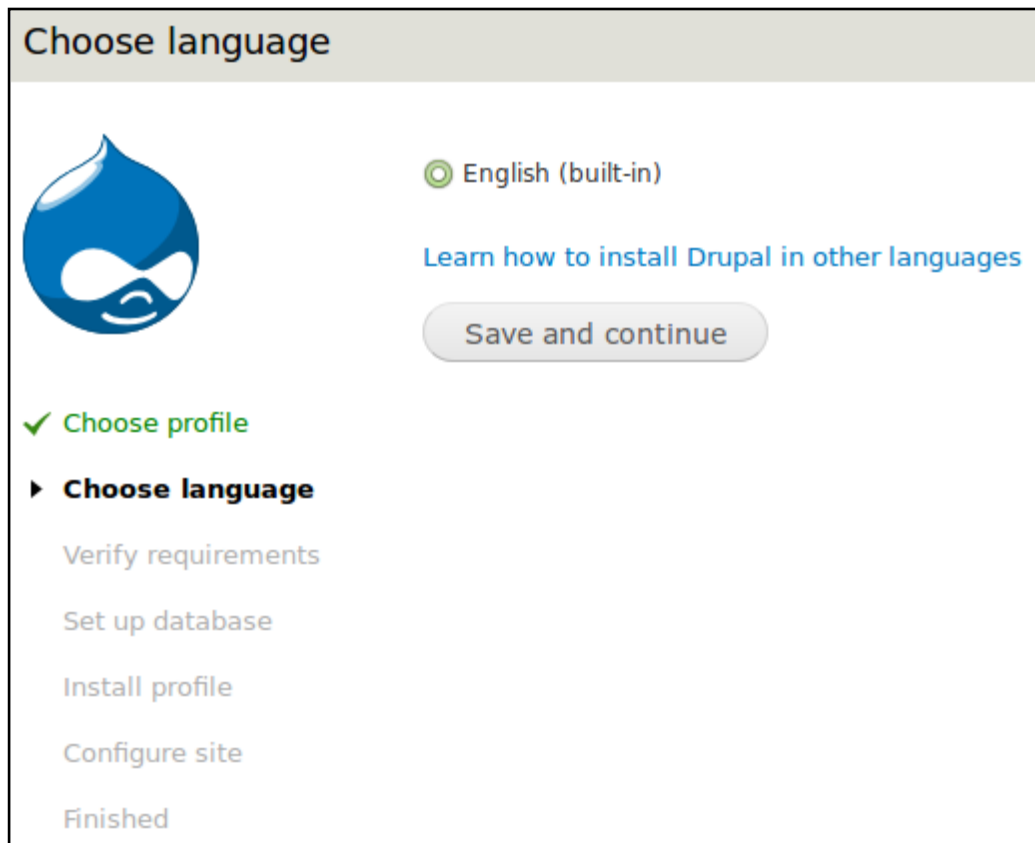
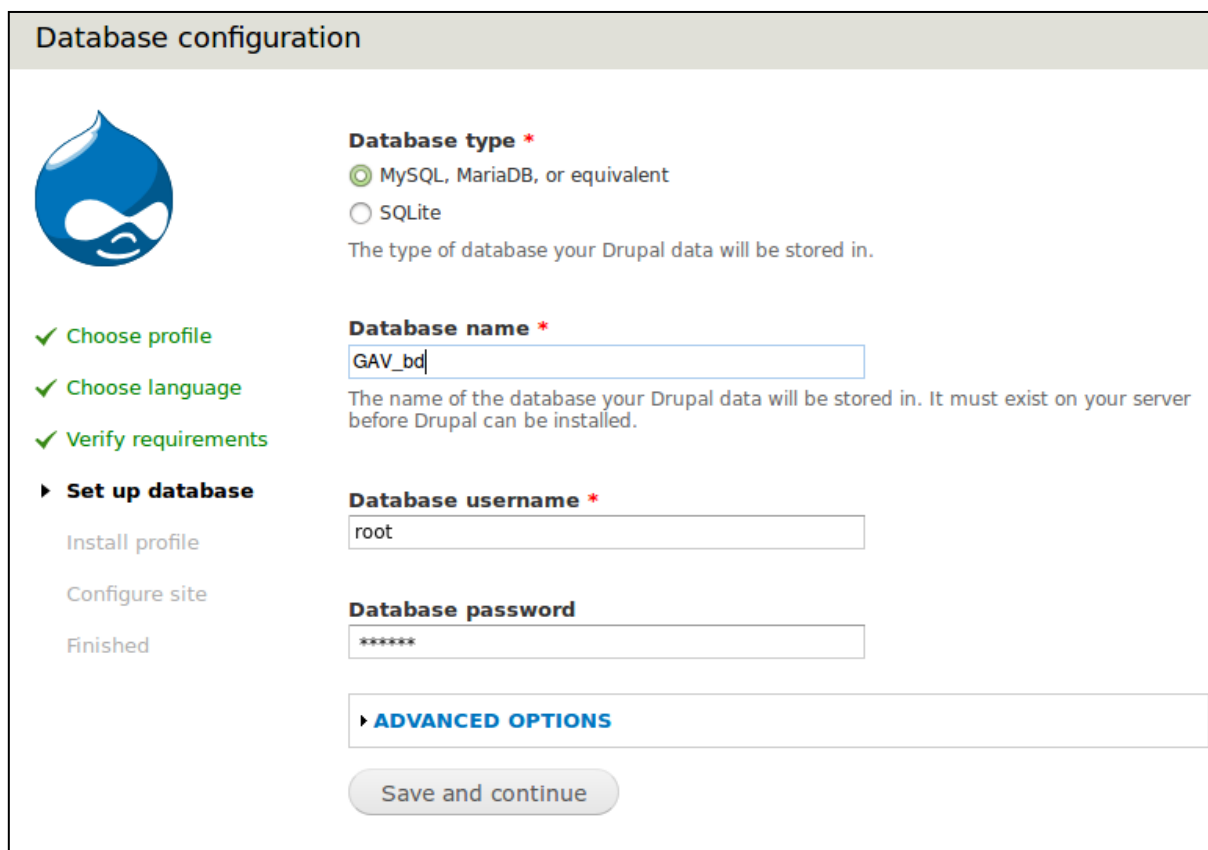


Figura 4.3- Selección de idioma

En la siguiente pantalla no podemos elegir otro idioma que no sea el inglés, aunque podemos instalar un idioma posteriormente.

4.1- Instalación Básica



The screenshot shows the 'Database configuration' screen in Drupal. On the left, there is a sidebar with a Drupal logo and a list of steps: 'Choose profile', 'Choose language', 'Verify requirements', and 'Set up database'. The 'Set up database' section is expanded, showing 'Install profile', 'Configure site', and 'Finished'. The main content area contains the following fields and options:

- Database type ***: Radio buttons for 'MySQL, MariaDB, or equivalent' (selected) and 'SQLite'. Below it, a note says 'The type of database your Drupal data will be stored in.'
- Database name ***: A text input field containing 'GAV_bd'. Below it, a note says 'The name of the database your Drupal data will be stored in. It must exist on your server before Drupal can be installed.'
- Database username ***: A text input field containing 'root'.
- Database password**: A text input field containing '*****'.
- ADVANCED OPTIONS**: A blue button with a right-pointing arrow.
- Save and continue**: A grey button at the bottom.

Figura 4.4- Pantalla de configuración de la conexión de la base de datos

Antes de continuar debemos crear la base de datos para conectarla con *Drupal*. Para crear la base de datos, nos dirigimos a la aplicación *phpMyAdmin*, que ya está instalada ya que pertenece al paquete que instalamos de Bitnami.



The screenshot shows the phpMyAdmin login screen. At the top, there is the phpMyAdmin logo and the text 'Bienvenido a phpMyAdmin'. Below this, there is a language selection dropdown menu labeled 'Idioma - Language' with 'Español - Spanish' selected. Underneath is a login section labeled 'Iniciar sesión' with a blue lock icon. It contains two input fields: 'Usuario:' and 'Contraseña:'. At the bottom right of the login section is a 'Continuar' button.

Tras acceder como usuario *root*, procedemos a crear la base de datos que guardará los datos de nuestro proyecto.

La creamos con el nombre *gav_bd*. Más adelante, diseñaremos la estructura que contendrá. Una vez creada podemos seguir con la instalación de *Drupal*.

Figura 4.5- Pantalla de identificación de *phpMyAdmin*

Ya solo nos falta indicar el nombre de nuestro proyecto y elegir el del nuestro usuario de *Drupal* para ver por primera vez nuestro portal web. En nuestro caso hemos elegido un dominio ficticio (*GAV.es*).

4.2- Instalación del idioma español

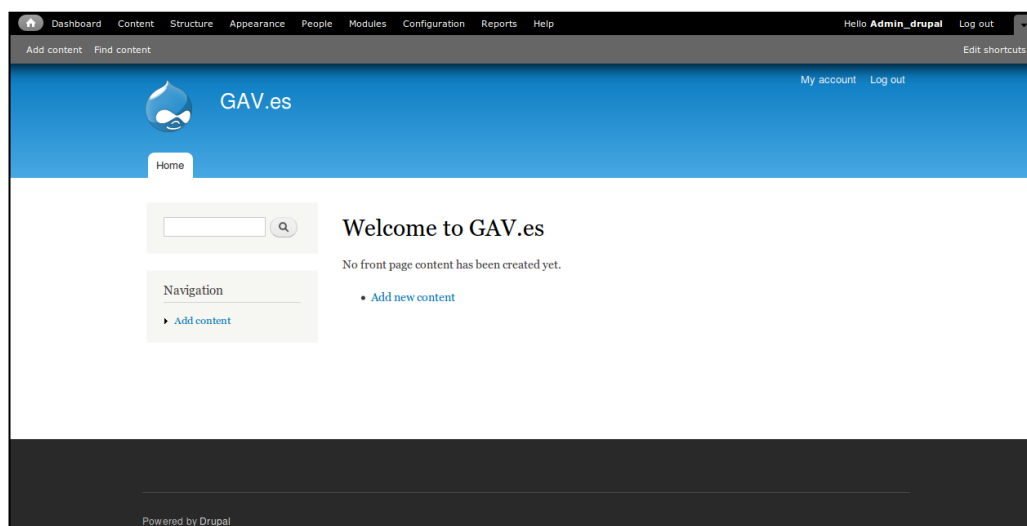
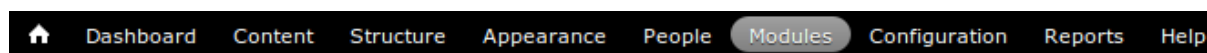


Figura 4.6- Pantalla de bienvenida de Drupal

4.2 Instalación del idioma español

Para finalizar la instalación de Drupal, vamos a traducirlo al español. Accedemos a la dirección <http://localize.drupal.org/translate/downloads> y descargamos el paquete de español correspondiente a nuestra versión de Drupal. Se trata de un fichero “.po”, el cual debemos colocar en la ruta `profiles/standard/translations/`.

Posteriormente debemos asegurarnos que el módulo *locale* está activo. Para comprobarlo,



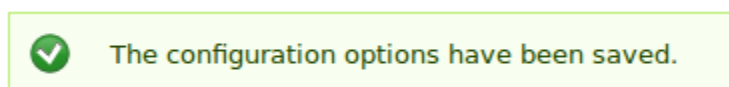
vamos a la sección de módulos, en la barra superior.


Figura 4.7- Barra de administración

<input type="checkbox"/>	Locale	7.31	Adds language handling functionality and enables the translation of the user interface to languages other than English. Required by: Content translation (disabled)
--------------------------	---------------	------	--

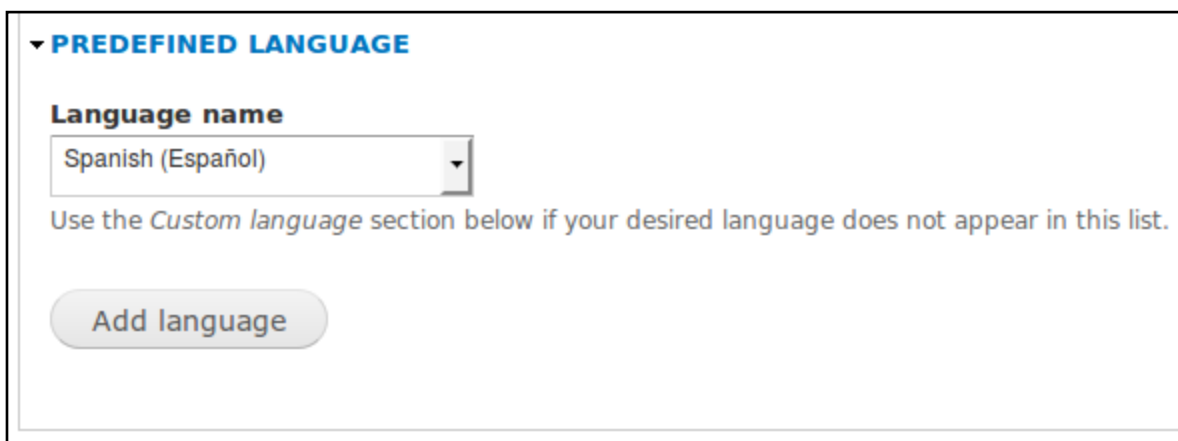
Figura 4.8- Módulo “Locale” aún deshabilitado

En nuestro caso está deshabilitado. Para activarlo, lo seleccionamos y confirmamos pulsando: Si todo ha ido correctamente veremos el siguiente mensaje:



Para configurar el idioma español, pulsamos sobre el botón:  **Configure**

4.2- Instalación del idioma español



▼ **PREDEFINED LANGUAGE**

Language name

Spanish (Español) ▼

Use the *Custom language* section below if your desired language does not appear in this list.

Add language

Figura 4.9- Menú de selección del idioma

Elegimos el idioma el español. Ahora solo nos falta elegirlo como por defecto y ya tendremos Drupal totalmente traducido a nuestro idioma.

4.2- Instalación del idioma español

5 Desarrollo

Es el capítulo más extenso, donde se lleva a cabo la resolución del problema planteado en anteriores capítulos. Se detallará la implementación de un módulo propio, además de los instalados procedentes de la página oficial de *Drupal*.

5.1 Configuración de roles

Lo primero que vamos a añadir al sistema son los distintos roles que va a existir en nuestra galería de arte virtual.

Rol es un grupo de usuarios con unos permisos concretos. Por defecto Drupal crea tres roles, Administrador, Usuario Registrado y Usuario No Registrado. Suelen ser tres roles muy comunes en los portales web, de hecho nosotros lo habíamos contemplado con anterioridad.

Para administrar Drupal necesitamos un usuario, es el que se crea al instalarlo. En nuestro caso le hemos llamado *Admin_drupal*, para diferenciarlo de los usuarios que realmente van a componer el portal web. Este usuario corresponde a la persona o personas que desarrollan el proyecto, no formando parte del personal encargado de administrar el sitio o añadir contenidos. Por lo que tiene el rol de *administrador Drupal*.

Para crear los dos roles que nos falta, nos dirigimos a la pantalla de roles que se encuentra en la ruta *Administración/Usuarios/Permisos/Roles*. A cada rol va asociado un “peso”, un número que ordena de mayor a menor importancia los roles. Por defecto los pesos son, por orden ascendente, Usuario no registrado, Usuario registrado, Artista, Administrador y Administrador Drupal. El rol de artista tiene más permisos que el usuario registrado pero menos que el administrador, por el contrario, el administrador de Drupal ha de poseer todos los permisos, por lo que tras añadir estos dos roles la tabla de pesos de los roles queda de la siguiente forma:

ROL	PESO
Usuario no registrado	0
Usuario registrado	1
Artista	2
Administrador	3
Administrador Drupal	4

5.2 Configuración del contenido

Dado que la principal actividad de nuestro sitio es la de albergar y exponer obras de arte, debemos de contar con un módulo que nos permita subir dicho contenido.

5.3- Datos a almacenar

Antes de la versión 7 de *Drupal*, un módulo que cumplía con lo expuesto anteriormente era el *Content Construction Kit (CCK)*, a partir de la séptima versión la funcionalidad de dicho módulo fue integrada en el núcleo de *Drupal*. Así pues, ya no es necesario instalarlo para hacer uso del mismo.

5.3 Datos a almacenar

Tenemos que decidir la información que necesitamos almacenar en la base de datos para el correcto funcionamiento de nuestra plataforma. *Drupal*, aparte de la información que nosotros decidamos almacenar, conserva muchísima más información necesaria para interactuar su núcleo con el contenido que aportemos y los módulos presentes. Aquí listaremos los datos que necesitamos y que no están contemplados por defecto.

Más detalladamente:

- ❖ Artista:
 - Nombre
 - Apellidos
 - Nombre de usuario
 - Zona horaria
 - Ciudad
 - Sexo
 - Currículum Artístico
- ❖ Obra:
 - Título de la obra
 - Descripción
 - Técnica:
 - Óleo
 - Cera
 - Témpera
 - Tinta
 - Fresco
 - Graffiti
 - Acrílico
 - Collage
 - Acuarela
 - Otro
 - Material
 - Papel
 - Cartón
 - Vidrio
 - Lienzo
 - Cobre
 - Otro
 - Dimensiones

5.3- Datos a almacenar

- En venta
 - Checkbox que indica si está o no a la venta
- Precio
 - En caso de estar en subasta o en venta
- ID máximo pujador
 - ID del usuario que es máximo pujador (0 por defecto)
 - No es visible para los usuarios
 - Se utiliza en el módulo *Pujador*
- Máximo pujador
 - Nombre del usuario que ostenta la puja mayor
- Estado
 - Visible para todos los roles, solo editable por los administradores o artistas (siempre y cuando sea contenido propio)
 - Vendido
 - No vendido
 - En subasta
- Imagen
 - Ilustración de la obra representada
- Valoración
 - Los usuarios podrán valorar la obra con una nota entre 0 y 5
- Fecha límite
 - En caso de que la obra esté en subasta, la fecha hasta la cual se admiten pujas
- ❖ Exposición:
 - Título
 - Descripción
 - Fecha (De inicio y fin)
 - Localización
 - Un mapa indicando el lugar donde se celebra
 - Obras expuestas
- ❖ Currículum Artístico
 - Título
 - Resumen
 - Formación
 - Nombre de formación
 - Centro
 - Fecha
 - Premios
 - Publicaciones
 - Título
 - Fecha

5.4 Añadir contenido en Drupal

Una vez decidido los datos que deben ser persistentes en la base de datos, debemos implementarlo en Drupal. Vamos a detallar como sería añadir los tipos de contenidos que necesitamos.

5.4.1 Obra

Primero debemos entrar en el menú de *estructura* de la barra del administrador.

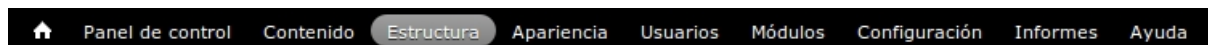


Figura 5.1- Barra de administración

Seguidamente seleccionamos *Añadir tipo de contenido*



Figura 5.2- Añadir tipo de contenido

La pantalla para editar es muy básica. Consta de un nombre identificativo al tipo de contenido, una descripción, la cual es solo visible para los administradores, un título y un texto que se mostrará en la cabecera del formulario, válido para indicar al usuario las directrices oportunas. Además, consta también de varias opciones como permitir al usuario la posibilidad de previsualizar el contenido antes de enviarlo.

5.4- Añadir contenido en Drupal

The screenshot shows the 'Editar' (Edit) page for a content type named 'Obra'. The breadcrumb trail is 'Inicio » Administración » Estructura » Tipos de contenido'. The page has several tabs: 'EDITAR', 'GESTIONAR CAMPOS', 'GESTIONAR PRESENTACIÓN', 'CAMPOS DE COMENTARIOS', and 'PRESENTACIÓN DE COMENTARIOS'. The main content area includes a 'Nombre *' field with the value 'Obra' and a description field. Below these are sections for 'Opciones del formulario de envío', 'Opciones de publicación', 'Opciones de presentación', 'Opciones de comentarios', and 'Opciones del menú'. The 'Etiqueta del campo de título *' is set to 'Obra'. The 'Previsualizar antes de enviar' section has 'Opcional' selected. The 'Explicación o directrices para envíos' section contains a text area with instructions.

Figura 5.3- Editar una obra

Drupal por defecto solo incluye un título y un cuerpo. Para añadir los datos que nos falta debemos hacerlo desde la opción *administrar campos* en el menú de *Tipos de contenidos*.

The screenshot shows the 'Gestionar Campos' (Manage Fields) page for a content type named 'Obra'. The breadcrumb trail is 'Inicio » Administración » Estructura » Tipos de contenido » Obra'. The page has several tabs: 'EDITAR', 'GESTIONAR CAMPOS', 'GESTIONAR PRESENTACIÓN', 'CAMPOS DE COMENTARIOS', and 'PRESENTACIÓN DE COMENTARIOS'. The main content area is a table with columns: 'ETIQUETA', 'PESO', 'PADRE', 'NOMBRE DE SISTEMA', 'TIPO DE CAMPO', 'CONTROL', and 'OPERACIONES'. Below the table are sections for 'Agregar nuevo campo' and 'Añadir un campo existente'.

ETIQUETA	PESO	PADRE	NOMBRE DE SISTEMA	TIPO DE CAMPO	CONTROL	OPERACIONES
Obra	-5	- Ninguno -	title	Elemento del módulo Node		
Body	-4	- Ninguno -	body	Texto largo y resumen	Área de texto con un resumen	editar eliminar

Agregar nuevo campo

Etiqueta: PESO: -3 PADRE: - Ninguno - TIPO DE CAMPO: - Seleccione un tipo de campo - CONTROL: - Seleccione un control -

Añadir un campo existente

Etiqueta: PESO: -2 PADRE: - Ninguno - TIPO DE CAMPO: - Seleccione un campo existente - CONTROL: - Seleccione un control -

Figura 5.4- Campos por defecto de un nodo

Para el campo *Fecha Límite*, debemos utilizar el módulo *Date*, el cual nos permite añadir un popup flotante para facilitar la selección de una fecha.

El otro campo que no utiliza un tipo de datos estándar es el widget encargado de la votación por parte de los usuarios.

5.4- Añadir contenido en Drupal

Una vez añadidos los atributos detallados en la sección anterior, la configuración de campos de nuestro tipo de contenido *Obra*, lucirá de la siguiente forma:

Pensamientos volátiles

Vista

Enviado por Artista1 el Jue, 10/08/2015 - 18:31

Foto:



Valoración:
☆☆☆☆☆
Sin votos aún
Puedes valorar esta obra indicando el número de estrellas

Descripción:
Representa lo efímeros que pueden ser los pensamientos

Técnica:
Óleo

Material:
Lienzo

Dimensiones [Alto x Ancho]:
120cm
120cm

En venta:
Sí

Precio:
3 001 458.00€

Estado:
En Subasta

Máximo pujador:
admin_drupal

Figura 5.5- Vista de una obra

Añadimos los permisos a cada rol. Al usuario anónimo y usuario registrado no le corresponde ningún permiso, mientras que al rol de artista, si le indicamos que pueda crear, editar y eliminar una obra que sea suya. Lógicamente, para los roles de administración los poderes son absolutos y podrán crear, editar y eliminar cualquier obra del sistema.

5.4- Añadir contenido en Drupal

PERMISO	USUARIO ANÓNIMO	USUARIO AUTENTICADO	ARTISTA	ADMINISTRADOR	ADMINISTRADOR DRUPAL
Ver revisiones del contenido	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Restablecer revisiones del contenido	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Eliminar revisiones del contenido	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Article: Crear contenido nuevo	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Article: Editar contenido propio	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Article: Editar cualquier contenido	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Article: Borrar contenido propio	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Article: Borrar cualquier contenido	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Obra: Crear contenido nuevo	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Obra: Editar contenido propio	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Obra: Editar cualquier contenido	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Obra: Borrar contenido propio	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Obra: Borrar cualquier contenido	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Figura 5.6- Permisos de los roles existentes

5.4.2 Exposición

De igual forma creamos una exposición. Cuando un artista crea una exposición, decide qué obras va a exponer.

Drupal nativamente no ofrece la posibilidad de referenciar un nodo desde un campo de otro. Una vez más debemos de usar un módulo para satisfacer un requisito de nuestro proyecto. El módulo (o uno de ellos, Drupal al ser tan modularizado, es posible solucionar problemas de distintas formas) encargado para ello es *Entity Reference*. Antes de instalar *Entity Reference* comprobamos que tiene como requisito otro módulo, por lo que procedemos también a instalar *Entity*. Este último módulo tiene como función ampliar la API de Drupal en cuanto creación, modificación y eliminado de entidades.

Obras	field_obra	Entity Reference	Casillas de selección / botones de opciones
-------	------------	------------------	---

Figura 5.7- Campo 'Obras' en el tipo de contenido 'Exposición'

Ahora ya podemos añadir un campo que referencie a otra entidad, en este caso al nodo *Obra*. Debemos de indicar en las opciones del campo, que el Artista puede añadir ilimitadas obras.

Uno de los campos es la localización del evento, por lo que se necesita un módulo que nos ofrezca la posibilidad de situar una dirección en un mapa. *Geolocation* nos ofrece lo que necesitamos, además contiene otro módulo para representar el mapa mediante la API de *Google Maps*. Por lo que va a dar a nuestro sitio una mejor presencia, ya que actualmente los mapas más habituales por la red son los de *Google*.

Para el campo fecha de inicio y fecha fin, utilizamos el mismo módulo utilizado para el campo fecha límite del nodo *Obra*.

En resumen, los campos de la entidad *Exposición* queda de la siguiente forma

5.4- Añadir contenido en Drupal

ETIQUETA	NOMBRE DE SISTEMA	TIPO DE CAMPO	CONTROL
+ Exposición	title	Elemento del módulo Node	
+ Descripción	body	Texto largo y resumen	Área de texto con un resumen
+ Fecha	field_fecha	Fecha	Pop-up calendar
+ Localización	field_localizacion	Geolocation	Google Map
+ Obras	field_obra	Entity Reference	Casillas de selección / botones de opciones

Figura 5.8- Campos de Exposición

PERMISO	USUARIO ANÓNIMO	USUARIO AUTENTICADO	ARTISTA	ADMINISTRADOR	ADMINISTRADOR DRUPAL
Exposición: Crear contenido nuevo	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Exposición: Editar contenido propio	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Exposición: Editar cualquier contenido	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Exposición: Borrar contenido propio	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Exposición: Borrar cualquier contenido	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Figura 5.9- Permisos asignados a Exposición

5.4.3 Currículum Artístico

El último tipo de contenido, o entidad que vamos a añadir a nuestro proyecto se trata de un *Currículum Artístico*. Su función será la de que cada artista pueda rellenar además de su perfil, con los típicos datos personales, también lo relacionado con su profesión/afición.

Nuestro *Currículum Artístico* consta de un resumen donde el autor puede describir su formación y/o experiencia de una forma libre. Es habitual que los artistas reciban premios, por ello, se agrega un campo donde se puede especificar el galardón otorgado. Para este campo, se ha optado por un texto corto. Como es imposible saber de ante mano los premios de cada artista, nuestra plataforma estipula que se puede especificar un número ilimitado de ellos, o incluso ninguno.

Los dos últimos campos hacen referencia a otro nodo. Por un lado, la formación, que es un nodo que contiene el centro donde se impartieron, la fecha, en la cual se debe indicar el periodo y un título que identifique de forma clara la preparación y conocimientos adquiridos. Al igual que los premios, se puede añadir un número ilimitado (incluyendo ninguno) de formación.

Por el otro, el artista puede indicar las publicaciones editadas a lo largo de su carrera. El nodo creado para ser referenciado por el campo *Publicaciones* consta de un título y del año de

5.5- Módulos utilizados

publicación del mismo. Para satisfacer la selección y representación de la fecha, se utiliza de nuevo un campo tipo *Fecha*, del módulo *Date*.

Tras configurar los campos como se ha descrito previamente, el contenido del currículum queda de la siguiente manera:

ETIQUETA	NOMBRE DE SISTEMA	TIPO DE CAMPO
⊕ Currículum Artístico	title	Elemento del módulo Node
⊕ Resumen	body	Texto largo y resumen
⊕ Formación	field_formacion	Entity Reference
⊕ Premios	field_premios	Texto
⊕ Publicaciones	field_publicaciones	Entity Reference

Figura 5.10- Campos de la entidad Currículum Artístico

5.5 Módulos utilizados

En esta sección comentaremos los módulos que han sido necesarios instalar para configurar el sitio.

- **STMP**
 - Trata para configurar los envíos de correos a los usuarios registrados.
- **Conditional fields**
 - Permite añadir condiciones a la hora de visualizar los nodos. Por ejemplo, el campo *Precio de venta al público*, del nodo obra, solo es visible si el check box *Está en venta*, está activado.
- **Field Permissions**
 - Posibilita tratar independientemente los permisos de los campos. Al igual que se puede configurar los permisos de creación, edición, borrado y visualización de los nodos, gracias a este módulo se puede aplicar a los campos.
- **Views**
 - Mejora la representación de los nodos.
- **Voting API**
 - Permite a los usuarios votar por las obras. Cada obra tendrá un sistema de votación, por la que los usuarios podrán valorarlas fácilmente.
- **Fivestar**
 - Representación gráfica del módulo anterior. Posibilidad de representar los votos recibidos mediante diferentes imágenes.

5.6- Módulo propio

- **Linked Field**
 - Añade la posibilidad de añadir links en los nodos. Por ejemplo, en las obras que se encuentran en subasta, se añade un link que lleva al usuario al formulario de puja.
- **Date**
 - Agrega la posibilidad para añadir un calendario para seleccionar una fecha. Por ejemplo, a la hora de poner una obra en subasta, el artista debe seleccionar la fecha límite.
- **Entity Reference**
 - Posibilita crear un campo que referencie a un nodo. Por ejemplo, en el tipo de contenido *Exposición*, creamos un campo que sea *Obras*, ya que *Exposición* consta de una o varias *Obras*.
- **Inline Entity Form**
 - Es un módulo que incrementa las funcionalidades del anterior. Una vez hemos referenciado una entidad en un tipo de contenido, con *Inline Entity Form* seremos capaces de añadir un nuevo nodo referenciado en el mismo formulario del nodo que lo referencia. Por ejemplo, en la entidad *Currículum Artístico*, podemos crear, directamente en el formulario de creación, un nuevo nodo del tipo *Formación*. Gracias a ello, nos facilita las tareas de creaciones de contenido que hacen referencia a otros nodos.
- **Geolocation**
 - Añade un mapa como campo. El tipo de contenido *Exposición*, contiene un campo llamado *Localización*. En dicho campo añadimos en un texto el lugar donde se celebrará la exposición. Al renderizar la vista, el campo *Localización* se representa con un mapa del tipo *Google Maps*.
- **Simplenews**
 - Crea un sistema de boletines, con los cuales podremos informar de nuestro contenido a los usuarios que así lo deseen.

5.6 Módulo propio

En muchas ocasiones algunas de las funcionalidades de nuestro sitio no se satisfacen con los módulos existentes. Podemos resolver dicha situación creando un módulo desde cero, o ampliando las funcionalidades de algún módulo existente a base de parches (“patches”).

Para satisfacer el requisito de que un usuario pueda pujar por una obra que esté en venta, vamos a implementar un módulo desde cero. El módulo recibirá el nombre de **Pujador**.

Las obras que estén en el estado de *en subasta*, se habilitará automáticamente un link que de acceso a un formulario donde el usuario pueda realizar la puja. Dicho formulario será implementado mediante un módulo en Drupal. El tratamiento de datos vendrá a cargo del propio módulo, que los enviará a un servicio web donde se realizará las comprobaciones oportunas, así como la lógica de la funcionalidad.

Pujador consta de tres proyectos. El primero es un módulo propio de Drupal, el cual será el

5.6- Módulo propio

encargado de la vista. Al ser un módulo de Drupal el lenguaje utilizado será **PHP**.

Para manejar la lógica, es decir, la parte del controlador será un servicio web desplegado en un servidor de aplicaciones (en este caso Glassfish).

Por último, el módulo **EJB**, que realiza un mapeo objeto-relacional con la base de datos que crea Drupal.

El lenguaje utilizado para los dos últimos proyectos es Java. Para la comunicación entre la vista y el controlador se ha elegido un formato de la información mediante **JSON** para una comunicación ágil y eficiente. Toda la información entre cliente y servidor se realiza cifrada con el algoritmo **DES** y el modo **ECB**.

5.6.1 Módulo en Drupal

Para la parte de la vista, realizaremos un módulo en *Drupal* que implemente un formulario donde se muestren los datos de la obra por la que se va a realizar la puja.

Añadir un módulo realizado por nosotros es muy fácil. Primero debemos de crear un directorio (con un nombre identificativo de nuestro módulo) en la ruta `/sites/all/modules/`. Como mínimo dentro del directorio recién creado, ha de constar con dos ficheros, `pujador.info` y `pujador.module`. Opcionalmente puede incluir `pujador.install`, donde contendría parámetros de configuración, esquemas de la base de datos, información para la desinstalación...

pujador.info: Contendrá la información acerca del módulo.

```
name = Pujador

description = "Módulo propio que crea un sistema para pujar por obras"

package = Módulos propios

version = 1.0

core = 7.x
```

Figura 5.11- Contenido de pujador.info

- **name:** Es el nombre del módulo, el que se mostrará en la página de configuración de los módulos, dentro del panel del administrador.
- **description:** Pequeña descripción que se mostrará en la página de módulos.
- **package:** Es posible agrupar nuestros módulos en paquetes, para que así se muestren ordenados en el listado de módulos.
- **version:** Indica la versión del módulo
- **core:** Indica la compatibilidad con Drupal 7.x

Además existe posibilidad de ampliar la información con:

- **dependencies:** Señala si depende de otro(s) módulo(s)
- **php:** Versión de PHP que usa.

5.6- Módulo propio

- **configure:** Es la ruta URL del formulario de configuración del nuevo módulo.

MÓDULOS PROPIOS				
ACTIVADO	NOMBRE	VERSIÓN	DESCRIPCIÓN	OPERACIONES
<input type="checkbox"/>	Pujador	1.0	Módulo propio que crea un sistema para pujar por obras	

Figura 5.12- Vista del módulo en la página de configuración

El sistema de módulos de Drupal se basa en el concepto de hooks. Para que nuestro módulo se comuniquen con el núcleo de Drupal, tan solo debemos sobrescribir el comportamiento de los hooks del sistema. Por ejemplo el hook `hook_help()` se usa para proporcionar información al usuario. En nuestro caso para definir el comportamiento de nuestro módulo, debemos sobrescribirlo de la siguiente forma: `pujador_help()`. Como se aprecia, basta con renombrar la función para que empiece por el nombre de nuestro módulo, respetando el nombre del hook (a partir del guión bajo ‘_’).

- **pujador.module:** Archivo donde está la lógica del módulo. En este caso consta de la creación de un formulario de la puja de una obra.

También se implementa la recogida de los datos del formulario, tratamiento y envío de los mismos al servicio web.

Por seguridad se cifra la información que se manda al servicio web. Se utiliza el algoritmo DES con el modo ECB.

Drupal posee una característica, por medio de la cual, es posible programar periódicamente la ejecución de tareas. A través del hook correspondiente a dicha característica, `hook_cron`, es posible indicar a Drupal que ejecute periódicamente una tarea necesaria en nuestro módulo.

Pujador

Pensamientos volátiles



Fecha límite para pujar:
2016-10-08 16:30:00

Precio Actual
65000.25€

Máximo pujador
usuario1

Precio de la puja *
 €

Enviar

Figura 5.13- Vista del módulo Pujador

En el módulo `pujador`, se implementa una tarea que sobrescribe el hook de cron, que comprueba el estado de las obras y la fecha límite de subasta. Concretamente recorre todas las obras que tengan el estado seleccionado como “En subasta” y comprueba si la fecha límite para pujar es posterior a la fecha y hora en la que se está ejecutando el `pujador_cron`. En caso de que la obra no haya recibido ninguna puja, pasará al estado “No vendido”, en caso contrario, pasará al estado “Vendido”.

A continuación se explicará las partes más relevantes de dicho método.

5.6- Módulo propio

```
$consulta = new EntityFieldQuery();  
  
$nodos = $consulta->entityCondition('entity_type', 'node')  
  
>propertyCondition('type', 'obra')  
  
>fieldCondition('field_estado', 'value', 'subasta', '=')  
  
>execute();
```

Figura 5.14- Consulta para obtener las obras en subasta

En la variable *\$nodos* contendrá un array de nodos (del tipo obra), los cuales cumplan la condición que el estado esté fijado como subasta. Es decir, devuelve todas las obras que actualmente está en subasta.

```
if (!empty($nodos['node'])) {  
  
    foreach ($nodos['node'] as $valor) {  
  
        $obra = node_load($valor->nid);  
  
        //Determina si la puja ha excedido su tiempo límite  
  
        $field_fecha_limite = field_get_items('node', $obra,  
'field_fecha_limite')[0]['value'];  
  
        $tiempo_limite = new DateTime($field_fecha_limite);  
  
        $ahora = new Datetime('now');  
  
        $maximo_pujador_id = field_get_items('node', $obra,  
'field_maximo_pujador_id')[0]['value'];  
  
        if ($ahora > $tiempo_limite && $maximo_pujador_id != 0) {  
  
            //Comportamiento si la obra ha sido vendida  
  
            $texto_log = "El tiempo de puja para la obra " . $obra->title  
            . " ha excedido, la obra pasa a vendida";  
  
            actualizar_campo($obra, 'vendido', 'estado');  
  
        }else if ($ahora > $tiempo_limite && $maximo_pujador_id === '0') {  
  
            //Comportamiento si la hora no ha recibido ninguna puja  
  
            $tiempo_limite = null;  
  
            $texto_log = "El tiempo de puja para la obra " . $obra->title  
            . " ha excedido, la obra pasa a no vendida";  
  
            actualizar_campo($obra, 'noVendido', 'estado');
```

5.6- Módulo propio

```
    }  
  
    if (isset($texto_log)) {  
  
        watchdog('Pujador', $texto_log);  
  
        actualizar_campo($obra, 0, 'en_venta');  
  
    }  
  
}  
  
}
```

Figura 5.15- Método que sobrescribe el comportamiento del hook cron

En caso de que la variable *\$nodos* esté vacía, no seguiría la ejecución del cron. En caso contrario, se trataría cada obra devuelta. Para ello, se almacena en una variable local (*\$obra*) la información de la obra en subasta gracias a la función de Drupal destinado para ello, indicándole el id.

Para comprobar si la fecha límite ha excedido simplemente se compara con la hora del servidor. También se recupera el id del usuario que actualmente es el máximo pujador. Con éstos últimos dos valores se puede comprobar si una obra ha de pasar al estado 'Vendido', 'No vendido' o no variar.

En caso de que el estado de una obra se modifique se registrará un mensaje. Estos mensajes son visibles en el apartado de "Informes/Mensajes recientes del registro", donde podremos cerciorarnos del buen funcionamiento del cron.

TIPO	FECHA	MENSAJE
cron	11/01/2015 - 12:51	Ejecución de cron terminada.
Pujador	11/01/2015 - 12:51	El tiempo de puja para la obra Obra prueba Cron...
Pujador	11/01/2015 - 12:51	El tiempo de puja para la obra Obra prueba Cron No...

Figura 5.16- Registro de mensajes

Podemos ver con más detalle pulsando en los mensajes.

5.6- Módulo propio

TIPO	Pujador
FECHA	Domingo, Noviembre 1, 2015 - 12:51
USUARIO	Anónimo (no verificado)
UBICACIÓN	http://localhost/admin/config/system/cron?render=overlay&render=overlay
REFERENTE	http://localhost/admin/config/system/cron?render=overlay
MENSAJE	El tiempo de puja para la obra Obra prueba Cron No Vendida ha excedido, la obra pasa a no vendida
GRAVEDAD	aviso
NOMBRE DEL SERVIDOR	127.0.0.1
OPERACIONES	

Figura 5.17- Mensaje detallado del registro. Obra no vendida.

TIPO	Pujador
FECHA	Domingo, Noviembre 1, 2015 - 12:51
USUARIO	Anónimo (no verificado)
UBICACIÓN	http://localhost/admin/config/system/cron?render=overlay&render=overlay
REFERENTE	http://localhost/admin/config/system/cron?render=overlay
MENSAJE	El tiempo de puja para la obra Obra prueba Cron Vendida ha excedido, la obra pasa a vendida
GRAVEDAD	aviso
NOMBRE DEL SERVIDOR	127.0.0.1
OPERACIONES	

Figura 5.18- Mensaje detallado del registro. Obra vendida

Se puede apreciar el resultado tras la ejecución del cron de nuestro módulo con dos obras creadas intencionadamente para tal fin. La primera imagen trata de una obra creada con una fecha límite anterior a la actual y sin puja. Por lo que el estado cambia a 'No vendida'. Mientras que en la segunda imagen, se creó la obra con la misma intención que la anterior salvo que en esta ocasión si ha recibido una puja.

La periodicidad de los hooks de cron es configurable desde el panel de control de Drupal.

5.6- Módulo propio

- `configuracion.php`: Archivo donde se encuentran las configuraciones constantes del módulo.

```
<?php

class Configuracion{

    const URL_BASE = 'http://[VPS_IP]:8080/gavWS/webresources/pujador/';

    const URL_PUJAR = 'pujar';

    const CLAVE_CIFRADO = [CLAVE PRIVADA]

}
```

Figura 5.19- Archivo configuracion.php

La constante `URL_BASE` ha de ser modificada una vez instalado el módulo con la dirección IP del servidor donde esté alojado el servicio web encargado de la lógica del módulo.

La segunda constante es referente al contexto de la operación del servicio web.

Por último, se ha de configurar la clave que se utilizará para el cifrado de datos. Obviamente dicha clave, debe de coincidir con la clave utilizada en el servicio web para un correcto procesamiento de los datos.

5.6.2 Servicio Web

REST es el estándar utilizado para implementar el servicio web. Mediante una petición POST se envía la información del formulario necesaria para llevar a cabo la puja.

Se utiliza la librería *Gson* para la serialización y la deserialización en formato JSON.

5.6.2.1 Petición

El formato del envío de datos ha de ser un JSON cifrado (mediante el algoritmo DES, utilizando el modo ECB) con los siguientes campos:

Usuario: Es el id del usuario que realiza la puja.

Contraseña: El resultado de la función MD5 de la contraseña del usuario que realiza la puja. Su función consiste en autenticar la identidad del emisor de la petición.

Obra: Id de la obra por la que se realiza la puja.

Cantidad: Puja en euros que ofrece el usuario por la obra.

Ejemplo (antes de cifrado):

```
{"usuario":1,"contrasena":"$$Drg49DXp8YugM9yZ8AiDZVev9Pnjke\DXEE8o7.xFGYcvEvEqJb","obra":15,"cantidad":1500}
```

5.6- Módulo propio

5.6.2.2 Respuesta

La respuesta del servicio web será, al igual que la petición, un JSON cifrado de la misma forma.

Contendrá dos campos, resultado y mensaje. En el primero se indicará mediante OK (todo ha ido bien) o KO (se ha producido algún error) el estado de la petición. En el segundo campo, se especificará con más detalle el resultado de la operación.

Ejemplo (antes de cifrado):

```
{"Resultado":OK,"Mensaje":"Puja realizada correctamente"}
```

5.6.2.3 Código

En esta sección se comentará el código realizado para la implementación del servicio web.

5.6.2.3.1 Método pujar

Este método se define con las siguientes cabeceras:

`@POST`

`@Path("pujar")`

`@Consumes("application/json")`

`@Produces("application/json")`

Método Post con el contexto pujar. El tipo de datos que tanto consume como produce, es JSON.

```
Gson gson = new Gson();  
  
String respuesta = new String();  
  
Map<String, String> mapaRespuesta = new HashMap<>();  
  
mapaRespuesta.put("Resultado", "KO");  
  
Cifrador cifrador = Cifrador.getInstance();
```

Figura 5.20- Definición de variables en el método 'pujar' del servicio web

Lo primero que se hace es definir las variables que serán utilizadas en el método. La variable mapaRespuesta es un mapa de *String*, la cual a posteriori, será la serialización de la respuesta. Se inicializa con el resultado a "KO", y se modificará a "OK" si todo ha ido bien.

```
String peticionPujadescifrado = cifrador.Descifrar(peticionPujadescifrado);
```

Figura 5.21- Definición de la petición descifrada en el método 'pujar' del servicio web

Descifra la petición recibida mediante la clase auxiliar que veremos con detalle en la sección 5.6.2.4.

5.6- Módulo propio

```
Type tipoMapa = new TypeToken<Map<String, Object>>() {}.getType();

//Recoger datos de la petición

JsonReader jsonReader = new JsonReader(new
StringReader(peticionPujaDescifrado));

jsonReader.setLenient(true);

LinkedTreeMap<String, Object> mapaPeticion = gson.fromJson(jsonReader,
tipoMapa);

Integer idUsuario = ((Double) mapaPeticion.get("usuario")).intValue();

String pass = mapaPeticion.get("contrasena").toString();

Integer idObra = ((Double) mapaPeticion.get("obra")).intValue();

Double cantidadPuja = (Double) mapaPeticion.get("cantidad");

respuesta = pujarObra(idUsuario, pass, idObra, cantidadPuja);
```

Figura 5.22- Parte del código del método 'pujar' del servicio web

Para manipular los datos recibidos con comodidad se guardan en memoria mediante un tipo mapa. Gracias a la librería *Gson* deserializamos la petición en el objeto tipo mapa antes definido. Una vez obtenidos los parámetros necesarios se realiza la llamada al método donde se efectúa la puja que se detallará en la sección siguiente.

```
mapaRespuesta.put("Resultado", "OK");

} catch (NullPointerException | JsonParseException jpe) {

    Logger.getLogger(ServicioWeb.class.getName()).log(Level.SEVERE,
null, jpe);

    respuesta = "Error al recibir los datos: " +
jpe.getMessage();

} catch (ClassCastException cce) {

    Logger.getLogger(ServicioWeb.class.getName()).log(Level.SEVERE,
null, cce);

    respuesta = "Formato de datos incorrecto: " +
cce.getMessage();

} catch (PujadorExcepcion ex) {

    Logger.getLogger(ServicioWeb.class.getName()).log(Level.SEVERE,
null, ex);

    respuesta = ex.getMessage();
```

5.6- Módulo propio

```
} catch (Exception e) {  
  
    respuesta = "Error en el servidor";  
  
} finally {  
  
    mapaRespuesta.put("Mensaje", respuesta);  
  
}  
  
String respuestaFinal = gson.toJson(mapaRespuesta);  
  
return cifrador.Cifrar(respuestaFinal);
```

Figura 5.23- Extracto de código del método 'pujar'

Se asigna *OK* a la futura respuesta si no se ha recogido ninguna excepción del método *pujarObra*. Seguidamente, se adjunta el mensaje devuelto del método que realiza la puja y tras cifrarlo se devuelve. En caso de captar alguna excepción, se registra y se adjunta el mensaje de error en la respuesta para posteriormente serializarla y cifrarla antes de ser devuelta.

5.6.2.3.2 Método auxiliar pugarObra

```
private String pugarObra(Integer idUsuario, String pass, Integer idObra,  
Double cantidadPuja) throws PujadorExcepcion
```

Figura 5.24- Cabecera del método auxiliar pugarObra

Método privado que realiza la puja tras hacer las comprobaciones oportunas. Como parámetros recibe el id del usuario que realiza la petición, así como la cantidad pujada y el id de la obra. En caso de algún error, se elevará una excepción del tipo *PujadorExcepcion*.

```
Node nodo = nodoC.find(idObra);  
  
Users usuario = usuarioC.find(idUsuario);  
  
if (nodo == null) {  
  
    throw new PujadorExcepcion("Obra no encontrada");  
  
}  
  
if (usuario == null) {  
  
    throw new PujadorExcepcion("Usuario no encontrado");  
  
} else {  
  
    if (!pass.equals(usuario.getPass())) {  
  
        throw new PujadorExcepcion("Contraseña incorrecta");  
  
    }  
  
}
```

5.6- Módulo propio

```
}  
  
}
```

Figura 5.25- Método auxiliar del servicio web, 'pujarObra'

Mediante los controladores implementados en la parte de la persistencia, recogemos los datos de la obra seleccionada para la puja, así como también la del usuario que realiza la acción.

Tras disponer de toda la información necesaria, es hora de realizar las comprobaciones oportunas para verificar si la petición de puja es correcta. En primer lugar, comprobamos que la obra exista. Para realizarlo simplemente debemos cerciorarnos que el objeto obra, que hemos recibido del controlador correspondiente es distinto de *null*. De igual forma, realizamos la misma comprobación sobre el objeto *usuario*. Además, debemos también autenticar al usuario mediante el cotejo de la contraseña que recibimos en la petición y la almacenada en la base de datos. Realmente, y como hemos comentado anteriormente, comparamos el resultado de la función hash *SHA512* que realiza Drupal sobre la contraseña del usuario.

En caso de no cumplir con algún requisito, se elevará una excepción del tipo *PujadorExcepcion* con un texto identificativo.

```
FieldDataFieldPvpPK pvpPK = new FieldDataFieldPvpPK("node",  
Short.parseShort("0"), idObra, "und", 0);  
  
FieldDataFieldPvp pvp = pvpC.find(pvpPK);  
  
if (cantidadPuja < pvp.getFieldPvpValue().doubleValue()) {  
  
    throw new PujadorExcepcion("Puja inferior al precio actual");  
  
}
```

Figura 5.26- Comprobaciones en el método 'pujarObra'

Comprobamos que la cantidad pujada es superior a la cantidad almacenada en la base de datos, elevando una excepción en caso contrario.

```
FieldDataFieldEstadoPK estadoPK = new FieldDataFieldEstadoPK("node",  
Short.parseShort("0"), idObra, "und", 0);  
  
FieldDataFieldEstado estado = estadoC.find(estadoPK);  
  
if (estado == null || !estado.getFieldEstadoValue().equals("subasta")) {  
  
    throw new PujadorExcepcion("La obra no tiene el estado de subasta");  
  
}
```

Figura 5.27- Comprobaciones en el método 'pujarObra'

5.6- Módulo propio

El método continúa en caso de que la obra tenga el estado *subasta*.

```
FieldDataFieldMaximoPujadorIdPK maximoPujadorIdPK = new
FieldDataFieldMaximoPujadorIdPK("node", Short.parseShort("0"), idObra,
"und", 0);

FieldDataFieldMaximoPujadorId maximoPujadorId =
maximoPujadorIdC.find(maximoPujadorIdPK);

if (maximoPujadorId == null) {

    throw new PujadorExcepcion("Error al recuperar al máximo pujador");

}

if (maximoPujadorId.getFieldMaximoPujadorIdValue().equals(idUsuario)) {

    throw new PujadorExcepcion("El usuario ya es el máximo pujador");

}
```

Figura 5.28- Comprobaciones en el método 'pujarObra'

Por último, comprobamos que el usuario que solicita pujar la obra, no es ya el máximo pujador.

```
//Actualiza ID del máximo pujador

maximoPujadorId.setFieldMaximoPujadorIdValue(idUsuario);

//Actualiza el nombre del máximo pujador

FieldDataFieldMaximoPujadorPK maximoPujadorPK =

new FieldDataFieldMaximoPujadorPK("node", Short.parseShort("0"), idObra,
"und", 0);

FieldDataFieldMaximoPujador maximoPujador =
maximoPujadorC.find(maximoPujadorPK);

maximoPujador.setFieldMaximoPujadorValue(usuario.getName());

//Actualiza el valor pujado

pvp.setFieldPvpValue(BigDecimal.valueOf(cantidadPuja));

//Reflejamos los cambios en la base de datos

maximoPujadorC.edit(maximoPujador);

maximoPujadorIdC.edit(maximoPujadorId);

pvpC.edit(pvp);
```

5.6- Módulo propio

```
return "Puja realizada correctamente";
```

Figura 5.29- Manipulación de la base de datos para la realización de la puja

Una vez validado todo, se procede a manipular la base de datos para registrar la puja. Para ello debemos modificar el id del máximo pujador con la del usuario que solicita la puja, así como la cantidad pujada en la obra objetivo. Por último devolvemos el mensaje de éxito, ya que si se hubiese producido algún problema se hubiera elevado una excepción con anterioridad.

Tal vez no sea la mejor práctica, modificar la base de datos directamente sin que Drupal tenga constancia, pero para la realización de este proyecto considero interesante poder distribuir la lógica del módulo creando un servicio web independiente de la plataforma Drupal.

5.6.2.4 Clase auxiliar para cifrado

Clase que se rige por el patrón **singleton** para que solo exista una instancia de ella.

Se utiliza el algoritmo *DES* con el modo *ECB*. La realización del cifrado lo lleva a cabo la librería *javax.crypto.Cipher*.

5.6.3 Persistencia

Esta parte del proyecto se ha generado automáticamente mediante las herramientas que nos ofrece un proyecto *JAVA EE*. Simplemente configurando la conexión a la base de datos donde se encuentra nuestro proyecto Drupal tendremos lo necesario para generar las entidades y los controladores. Para ello, es necesario configurar un recurso *JDBC* en el servidor de aplicaciones, en nuestro caso *Glassfish*.

Debido a que no necesitamos nada más que consultar y modificar las entidades, no es preciso modificar prácticamente nada del código que no autogenera java.

5.6.4 Excepciones

```
public class PujadorExcepcion extends Exception {  
  
    public PujadorExcepcion() {  
  
    }  
  
    public PujadorExcepcion(String msg) {  
  
        super(msg);  
  
    }  
  
}
```

Figura 5.30- Clase *PujadorExcepcion* para gestionar las excepciones de forma propia

5.7- Pruebas Unitarias

Creamos una sencilla clase que extiende de *Exception* para gestionar de forma personalizada nuestras excepciones. Simplemente se trata de un método constructor que recoge un mensaje identificativo que lo eleva a la clase padre.

5.7 Pruebas Unitarias

En esta sección utilizaremos la característica que incluye Drupal de forma nativa para crear test y probar los módulos que hemos implementados.

En nuestro caso, y como ejemplo, realizaremos las pruebas necesarias para testear el método `pujador_cron`. Recordamos que dicho método se ejecuta periódicamente comprobando las obras que están en subasta y que hayan excedido la fecha límite cambiando el estado a vendido o no vendido según corresponda.

Lo primero que debemos hacer, es activar el módulo *testing* que por defecto viene desactivado. No hace falta que lo instalemos desde la versión 7 de Drupal ya que desde dicha versión fue integrado al núcleo.

Para escribir test para nuestro módulo es necesario añadir en el fichero *.info* donde se encuentra los ficheros *.test*, que será donde escribamos el código que testea nuestra aplicación. Dentro del fichero *.test* tendremos que sobrescribir algunos métodos que comentaremos a continuación, y además deberemos indicar los métodos que se encargan de realizar las pruebas llamándolos de la siguiente forma: *testNombreDelMétodo*.

```
files[] = test/pujador.test
```

Figura 5.31- Fichero *pujador.info*

Lo anterior indica al módulo *testing* donde se encuentran los archivos que debe ejecutar para realizar nuestro testeo.

```
class TestPujador extends DrupalWebTestCase
```

Figura 5.32- Declaración de la clase *TestPujador*

Nuestro fichero *pujador.test* ha de contener una clase que extienda de *DrupalWebTestCase*, que es la clase que nos proporciona todo lo necesario para crear un entorno seguro donde efectuar las pruebas.

```
public static function getInfo() {  
  
    return array(  
  
        'name' => t('Test de Pujador'),  
  
        'description' => t('Funcionalidad del módulo Pujador'),  
  
        'group' => 'Pujador',  
  
    );  
};
```

5.7- Pruebas Unitarias

```
}
```

Figura 5.33- Método *getInfo* de la clase *TestPujador*

Debemos sobrescribir el método *getInfo()* para proporcionar información a Drupal sobre nuestro archivo de pruebas. En él simplemente debemos devolver un array con los siguientes parámetros:

- **Name:** El nombre que se mostrará en la página donde se agrupan todos los archivos de testeo.
- **Description:** Descripción del archivo, por ejemplo lo que se va a probar.
- **Group:** Grupo en el cual se engloba, por ejemplo es buena idea agruparlos por módulos.

```
function setUp() {  
  
    //Se asegura de que el módulo pujador está habilitado  
  
    parent::setUp(array('pujador'));  
  
    //Realiza el login con un usuario con privilegios de administrador  
  
    $this->privilegedUser = $this->drupalCreateUser(array('create obra  
content', 'extra special edit any obra'));  
  
    $this->drupalLogin($this->privilegedUser);  
  
}
```

Figura 5.34- Método *setUp* de la clase *TestPujador*

El segundo método que debemos sobrescribir es *setUp()*. Se encarga en preparar el “escenario” en el que queremos realizar las operaciones. Siempre se va a lanzar antes de todos los test de la clase. También existe la posibilidad de sobrescribir el método *tearDown()*, que se ejecuta al final de todos los test.

En nuestro caso realizamos dos acciones importantes. La primera es asegurarnos de que el módulo *Pujador* está activado, obviamente sin esto no tendría sentido continuar. Lo segundo, es crear un usuario con los permisos necesarios para crear y editar un nodo de tipo obra. Finalmente, nos logueamos con el usuario recientemente creado.

Cabe comentar que todos estos cambios no se reflejan en la base de datos del proyecto, para ello, el módulo *testing* crea una estructura igual a nuestro sitio en la base de datos. De este modo, se crea un escenario totalmente seguro donde probar nuestro código.

```
private function getFechaAnterior(){  
  
    $fechaAhora = date('Y-m-j G:i:s');  
  
    debug($fechaAhora, 'Fecha Actual', true);  
  
}
```

5.7- Pruebas Unitarias

```
$fechaAnterior = strtotime('-1 day', strtotime($fechaAhora));

$fechaAnterior = date('Y-m-j G:i:s', $fechaAnterior);

return $fechaAnterior;
}

private function getFechaPosterior(){

    $fechaAhora = date('Y-m-j G:i:s');

    debug($fechaAhora, 'Fecha Actual', true);

    $fechaPosterior = strtotime('+1 day', strtotime($fechaAhora));

    $fechaPosterior = date('Y-m-j G:i:s', $fechaPosterior);

    return $fechaPosterior;
}
```

Figura 5.35- Funciones auxiliares de pujador.test

Para probar el método `pujador_cron` es necesario que alguna obra tenga como fecha límite anterior a la fecha a la actual. Por ello, necesitamos dos funciones auxiliares, una que se encarga de devolver una fecha anterior a la actual en un día (`getFechaAnterior()`), y otra de devolver una fecha posterior a la actual en un día (`getFechaPosterior()`).

```
private function crearObra($idMaximoPujador, $fechaLimite){

    $edit = array();

    $edit['title'] = $this->randomName(8);

    $edit["body[und][0][value]"] = $this->randomName(16);

    $this->drupalPost('node/add/obra', $edit, t('Save'));

    $obra_contenido = array(

        'type' => 'obra',

        'title' => 'Obra de prueba',

        'body' => 'Esta obra ha sido creada para la batería de pruebas',

        'field_estado[LANGUAGE_NONE][0][value]' => 'subasta',

        'field_fecha_limite[LANGUAGE_NONE][0][value]' => $fechaLimite,

        'field_maximo_pujador_id[LANGUAGE_NONE][0][value]' =>
        $idMaximoPujador,
    );
}
```

5.7- Pruebas Unitarias

```
);  
  
$obra = $this->drupalCreateNode($obra_contenido);  
  
return $obra;  
  
}
```

Figura 5.36- Método auxiliar para crear una obra

Necesitamos crear una obra que tenga el estado ‘*En subasta*’ para la ejecución del método a probar, *pujador_cron*. El método consta de dos parámetros, uno para indicar el máximo pujador (esto es realmente para indicar si la obra ha recibido o no puja) y fecha límite, que lo usaremos para determinar si la obra va a tener una fecha anterior o posterior según corresponda.

```
function testObraVendida()  
  
    $fechaLimite = getFechaAnterior();  
  
    debug($fechaLimite, 'Fecha Límite', true);  
  
    $obra = crearObra(5, $fechaLimite);  
  
    debug($obra->field_estado[LANGUAGE_NONE][0][value], 'Estado actual',  
true);  
  
    $this->pujador_cron();  
  
    $this->assertEqual($obra->field_estado[LANGUAGE_NONE][0][value], 'vendido', 'La obra ha de tener el  
estado vendido');  
  
    debug($obra->field_estado[LANGUAGE_NONE][0][value], 'Estado actual',  
true);  
  
}
```

Figura 5.37- Test que prueba cuando una obra pasa a ser vendida

En la *Figura 5.37* se encuentra el código del test para una obra en subasta, con la fecha límite excedida y con alguna puja realizada (id del máximo pujador es distinto a 0). Tras la ejecución del método *pujador_cron*, el estado debe de tener como nuevo valor “*vendido*”.

```
function testObraNoVendida() {  
  
    $fechaLimite = getFechaAnterior();  
  
    debug($fechaLimite, 'Fecha Límite', true);  
  
    $obra = crearObra(0, $fechaLimite);  
  
    debug($obra->field_estado[LANGUAGE_NONE][0][value], 'Estado actual',
```

5.7- Pruebas Unitarias

```
true);

    $this->pujador_cron();

    $this->assertEqual($obra->field_estado[LANGUAGE_NONE][0][value], 'noVendido', 'La obra ha de tener el estado no vendido');

    debug($obra->field_estado[LANGUAGE_NONE][0][value], 'Estado actual', true);
}
}
```

Figura 5.38- Test que prueba cuando una obra pasa a no vendida

En la *Figura 5.38* podemos observar el código implementado para el primer test. Se comprueba que una obra que tiene el estado “*en subasta*”, que su fecha límite ha excedido y que no ha recibido ninguna puja (id del máximo pujador es igual a 0), se debe de modificar el estado, tras la ejecución del método *pujador_cron*, al estado “*no vendido*”.

```
function testObraNoExcedida() {

    $fechaLimite = getFechaPosterior();

    debug($fechaLimite, 'Fecha Límite', true);

    $obra = crearObra(0, $fechaLimite);

    debug($obra->field_estado[LANGUAGE_NONE][0][value], 'Estado actual', true);

    $this->pujador_cron();

    $this->assertEqual($obra->field_estado[LANGUAGE_NONE][0][value], 'noVendido', 'La obra ha de tener el estado no vendido');

    debug($obra->field_estado[LANGUAGE_NONE][0][value], 'Estado actual', true);
}
}
```

Figura 5.39- Test de una obra con la fecha límite aún no excedida

En el último test, se crea una obra con la fecha límite posterior a la fecha actual, por lo que su estado ha de permanecer inmutable. En esta ocasión es irrelevante si se pasa algún id de máximo pujador o no, ya que no influye en el comportamiento ya que al no exceder la fecha límite no debe de variar su estado.

Para ejecutar todos estos test debemos irnos a la página del módulo *testing* (si tenemos Drupal en español en el menú se llamará *probando*) que se encuentra en el apartado de *configuración*. Se nos mostrará todo el listado con los test disponibles en el proyecto. Seleccionamos el de nuestro módulo y pulsamos “*Ejecutar pruebas*”

5.7- Pruebas Unitarias

Pujador

Test del módulo Pujador

Funcionalidad del módulo Pujador

Ejecutar pruebas

Si todo ha ido bien, veremos algo como lo siguiente:

TEST DE PUJADOR

Funcionalidad del módulo Pujador
2 pases, 0 fallos, 0 excepciones, y 4 mensajes de depuración

MENSAJE	GRUPO	NOMBRE DE ARCHIVO	LÍNEA	FUNCIÓN	ESTADO
Enabled modules: <i>pujador</i>	Other	pujador.test	18	TestPujador->setUp()	✓
Fecha Actual: 2015-11-1 12:02:11	Debug	pujador.test	26	TestPujador->testObraVendida()	⚠
Fecha Límite: 2015-10-31 12:02:11	Debug	pujador.test	29	TestPujador->testObraVendida()	⚠
Estado actual: subasta	Debug	pujador.test	30	TestPujador->testObraVendida()	⚠
La obra ha de tener el estado vendido	Other	pujador.test	31	TestPujador->testObraVendida()	✓
Estado actual: vendido	Debug	pujador.test	32	TestPujador->testObraVendida()	⚠

Figura 5.40- Resultado de la ejecución del tesObraVendida

TEST DE PUJADOR

Funcionalidad del módulo Pujador
2 pases, 0 fallos, 0 excepciones, y 4 mensajes de depuración

MENSAJE	GRUPO	NOMBRE DE ARCHIVO	LÍNEA	FUNCIÓN	ESTADO
Enabled modules: <i>pujador</i>	Other	pujador.test	18	TestPujador->setUp()	✓
Fecha Actual: 2015-11-1 12:23:29	Debug	pujador.test	37	TestPujador->testObraNoVendida()	⚠
Fecha Límite: 2015-10-31 12:23:29	Debug	pujador.test	40	TestPujador->testObraNoVendida()	⚠
Estado actual: subasta	Debug	pujador.test	41	TestPujador->testObraNoVendida()	⚠
La obra ha de tener el estado vendido	Other	pujador.test	42	TestPujador->testObraNoVendida()	✓
Estado actual: NoVendido	Debug	pujador.test	43	TestPujador->testObraNoVendida()	⚠

Figura 5.41- Resultado de la ejecución del testObraNoVendida

5.8- Configuración de Menús

▼ TEST DE PUJADOR

Funcionalidad del módulo Pujador
2 pases, 0 fallos, 0 excepciones, y 4 mensajes de depuración

MENSAJE	GRUPO	NOMBRE DE ARCHIVO	LÍNEA	FUNCIÓN	ESTADO
Enabled modules: pujador	Other	pujador.test	18	TestPujador->setUp()	✓
Fecha Actual: 2015-11-1 12:24:57	Debug	pujador.test	48	TestPujador->testObraNoExcedida()	⚠
Fecha Limite: 2015-11-2 12:24:57	Debug	pujador.test	51	TestPujador->testObraNoExcedida()	⚠
Estado actual: subasta	Debug	pujador.test	52	TestPujador->testObraNoExcedida()	⚠
La obra ha de mantener el estado	Other	pujador.test	53	TestPujador->testObraNoExcedida()	✓
Estado actual: subasta	Debug	pujador.test	54	TestPujador->testObraNoExcedida()	⚠

Figura 5.42- Resultado de la ejecución del testObraNoExcedida

5.8 Configuración de Menús

En esta sección, y ya tras disponer de toda la información en nuestro proyecto, se trata de modificar el la interfaz con la que el usuario va a interactuar. Para ello vamos a estructurar los diferentes menús según el rol del usuario que esté haciendo uso de la plataforma.

5.8.1 Usuario no registrado

Para el usuario no registrado, no debemos realizar ningún cambio, ya que sus dos únicas funcionalidades son la de identificarse y acceder con el rol que le corresponde a su usuario o registrarse en la web.

GAV.es

Home

Inicio de sesión

Nombre de usuario *

Contraseña *

- [Crear nueva cuenta](#)
- [Solicitar una nueva contraseña](#)

Iniciar sesión

Acceso denegado

Usted no está autorizado para visitar esta página.

Figura 5.43- Pantalla del usuario no registrado

5.8.2 Usuario Registrado

Para el usuario registrado, diseñamos un menú de navegación sencillo. En la parte izquierda de la pantalla creamos un menú con los enlaces más importantes.



Figura 5.44- Menú de navegación para el Usuario Registrado

Para crear los enlaces primeros debemos crear una vista. Las vistas en *Drupal* es la manera de representar la información almacenada en la base de datos. Es como si nuestro sitio web fuese una panadería, las vistas serían los escaparates donde colocamos los productos para que puedan ser visualizados por los clientes.

El módulo de vistas nos ofrece gran cantidad de opciones para personalizarlas. Desde filtros, añadir o quitar campos de los nodos, incluso relaciones con otros nodos. A continuación detallaremos la creación de una de las vistas usadas a modo de ejemplo. Las demás, las hemos realizado de forma semejante.

Para crear una vista nos debemos de ir al menú *admin/structure/views*, al cual se accede mediante el menú de administrador, en la sección estructura y por último *views*. Obviamente solo podemos acceder mientras estemos autenticados con un usuario con los permisos suficientes.

5.8- Configuración de Menús

La vista que vamos a crear, trata de listar las obras que actualmente están en el sistema. Los principales ajustes que vamos a introducir son el formato con el que van a ser representadas.

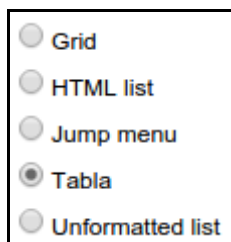


Figura 5.45- Formatos disponibles



Figura 5.46- Campos de la vista obras

Disponemos de los cinco estilos que se muestran en la *Figura 5.45*. Se pueden añadir otros mediante módulos que aporten los suyos propios. Para este ejemplo hemos seleccionado una tabla. En la *Figura 5.46* seleccionamos los campos que se van a mostrar. Para el caso de los dos primeros campos, el primer término que nos encontramos hace referencia al contexto al que pertenece. El segundo es el nombre del campo en la base de datos, mientras que en el último, es la etiqueta que se va a mostrar en la cabecera de la tabla. Se puede observar que el último campo contiene la palabra *autor* al principio, esto es porque indica que es fruto de una relación. Es decir, hemos relacionado el nodo **obra** con el nodo **usuario**, mediante la relación que los une, **un usuario es autor de una obra**. Gracias a esa relación podemos obtener de igual forma que los datos de la obra, la información del autor.

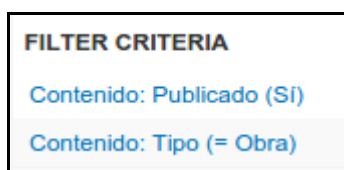


Figura 5.47- Filtros de la vista obras



la vista obras

Figura 5.48- Configuración de la página de

Debemos indicar cuáles son las obras que queremos listar. Para la vista de *obras*, es un filtro sencillo, como se muestra en la *Figura 5.47*. Simplemente exigimos que las obras estén publicadas (un contenido puede estar en la base de datos pero sin publicar, lo que hará que no sea visible) y que el nodo sea del tipo *Obra*. Para la vista *Obras en venta*, además de los dos filtros usados para la vista *obras*, se añade también que el estado sea igual a *en subasta*.

Por último, configuramos la ruta por la que accedemos al listado. Tal y como se muestra en la *Figura 5.48*, para acceder a nuestro listado de obras, tan solo debemos dirigirnos a <http://nuestrodominio/obras>. Para hacerlo más accesible añadimos un menú normal, esto creará un hiperenlace a la dirección anterior en el menú de navegación del

5.8- Configuración de Menús

usuario.

De igual forma, creamos las vistas para las obras que están en subasta, las obras que están en venta, artistas y usuarios.

Para el menú de *Buscar*, primero debemos asignar los permisos necesarios para que los usuarios registrados puedan acceder tanto a la búsqueda simple como a la avanzada. En la página de configuración de búsqueda avanzada (*configuración/Búsqueda y metadatos/Opciones de búsqueda*) podemos comprobar el tanto por ciento de indexación de la página, pudiendo ordenar volver a indexar manualmente. En cada ejecución del cron se indexa el contenido.



The screenshot shows a search interface titled "Buscar". At the top, there are two tabs: "Contenido" (selected) and "Usuarios". Below the tabs, there is a search bar with the prompt "Escriba las palabras clave." and a search icon. Underneath, a section titled "Búsqueda avanzada" is expanded, showing three columns of options:

- Que contenga cualquiera de las palabras:** A text input field.
- Que contenga la frase:** A text input field.
- Que no contenga ninguna de las palabras:** A text input field.

To the right of these fields is a list of content types with checkboxes:

- Sólo de los tipos:**
 - Article
 - Basic page
 - Currículum Artístico
 - Exposición
 - Formación
 - Obra
 - Publicación
- Idiomas:**
 - Spanish
 - English

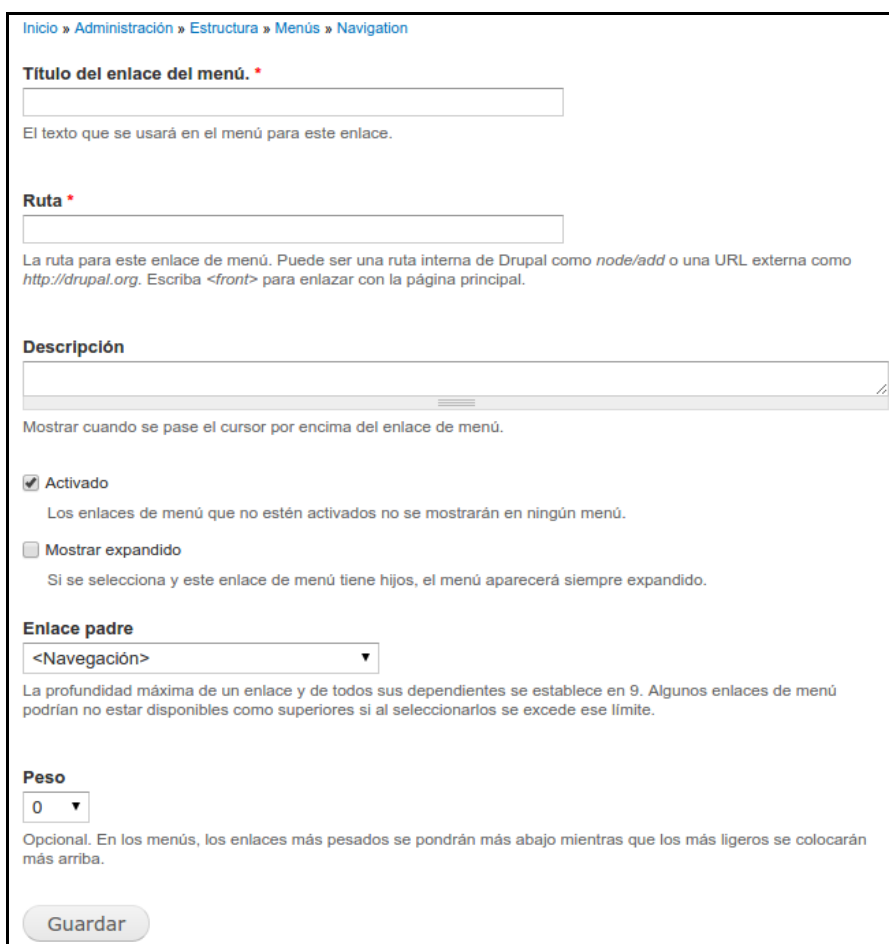
At the bottom left of the advanced search section, there is another search icon.

Figura 5.49- Búsqueda avanzada de contenido

5.8.3 Artista

Los usuarios con el rol de artista, se les aplica todo lo comentado en la sección anterior. Además añadimos un nuevo menú que contendrá los enlaces necesarios para crear un nuevo contenido. Crearemos el menú desde las herramientas de administración *Estructura/Menús*. Dentro del menú *Navegación*, seleccionamos añadir enlace.

5.9- Solicitud de Artista



Inicio » Administración » Estructura » Menús » Navigation

Título del enlace del menú. *

El texto que se usará en el menú para este enlace.

Ruta *

La ruta para este enlace de menú. Puede ser una ruta interna de Drupal como `node/add` o una URL externa como `http://drupal.org`. Escriba `<front>` para enlazar con la página principal.

Descripción

Mostrar cuando se pase el cursor por encima del enlace de menú.

Activado

Los enlaces de menú que no estén activados no se mostrarán en ningún menú.

Mostrar expandido

Si se selecciona y este enlace de menú tiene hijos, el menú aparecerá siempre expandido.

Enlace padre

La profundidad máxima de un enlace y de todos sus dependientes se establece en 9. Algunos enlaces de menú podrían no estar disponibles como superiores si al seleccionarlos se excede ese límite.

Peso

Opcional. En los menús, los enlaces más pesados se pondrán más abajo mientras que los más ligeros se colocarán más arriba.

Guardar

Figura 5.50- Formulario de creación de un enlace

En el título, indicamos el nombre que va a ver el usuario, por lo que se recomienda que sea lo más descriptivo posible. En *Ruta*, especificamos la ruta a la que se dirigirá el usuario tras pulsar en el enlace. Podemos incluir una descripción que se mostrará en un recuadro flotante cuando el usuario pose el cursor sobre el enlace. Para que un enlace sea visible ha de estar activado. Si marcamos la opción de *Mostrar expandido*, se visualizará desplegado por defecto. Debemos añadir el enlace padre que es del que cuelga. Por último, el peso sirve para ordenarlos mediante su importancia (cuanto mayor sea el número, más bajo será su posición).

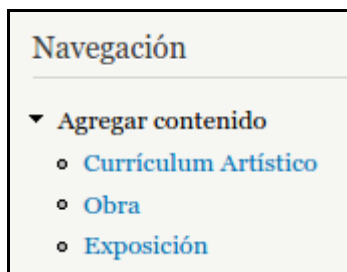


Figura 5.51- Menú para los artistas

5.9 Solicitud de Artista

Un usuario registrado puede solicitar a un administrador el cambio de rol a artista.

5.10- Boletín Semanal

Dispone de un formulario de contacto que podrá encontrar en el menú superior del sitio



Figura 5.52- Menú superior de usuario

Tras rellenar la petición con los datos que abajo se solicitan, el administrador decidirá si atiende positivamente la petición.

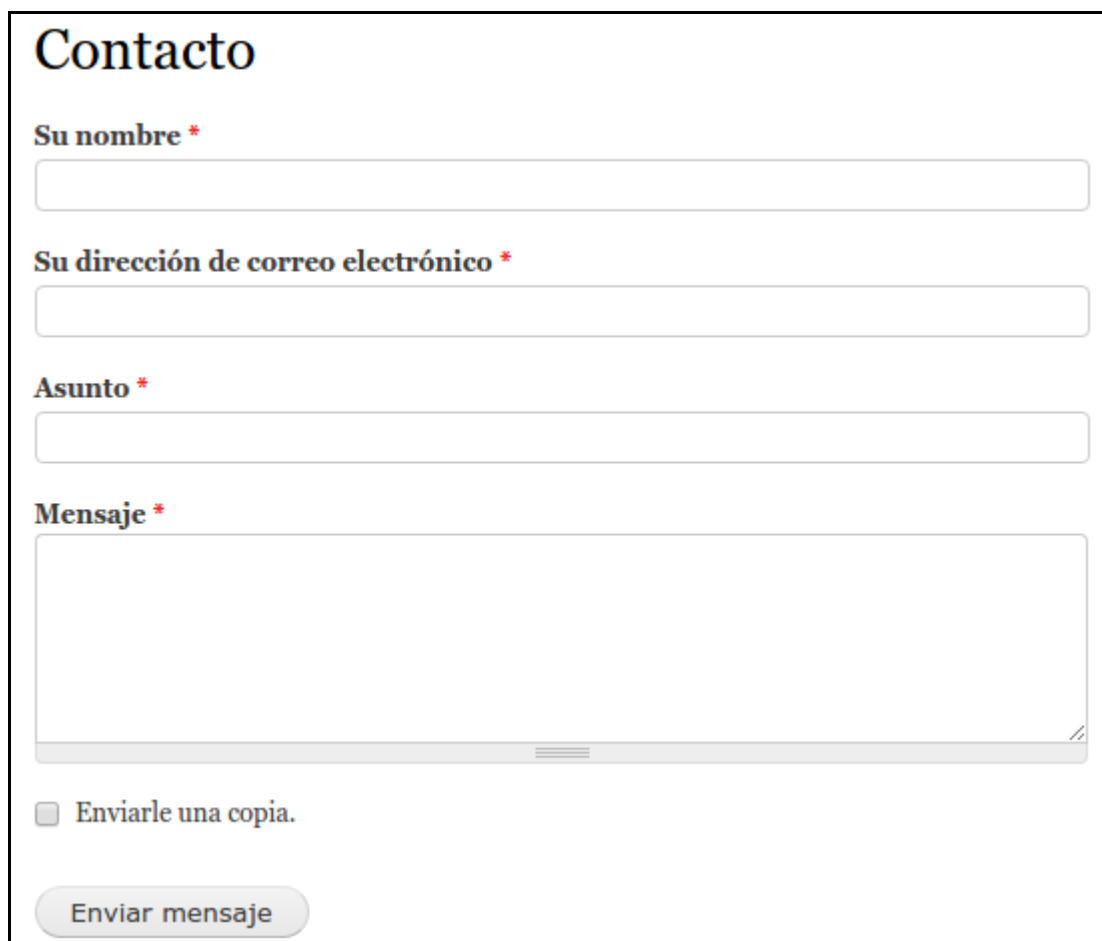
Un formulario de contacto con el título 'Contacto'. Incluye los siguientes campos: 'Su nombre *' (campo de texto), 'Su dirección de correo electrónico *' (campo de texto), 'Asunto *' (campo de texto) y 'Mensaje *' (campo de texto grande). Debajo de los campos hay un checkbox con el texto 'Enviarle una copia.' y un botón 'Enviar mensaje'.

Figura 5.53- Formulario de contacto con la administración de la plataforma

5.10 Boletín Semanal

Debemos mantener a los usuarios informados de las últimas novedades de nuestro sitio. Una solución clásica a este problema es la de implementar un boletín informativo que se envía de forma automática a los usuarios que previamente estén suscritos. Para nuestro sitio vamos a configurar un boletín con las nuevas exposiciones creadas en la última semana. El envío de los boletines se efectúa al mismo tiempo que se ejecuta cron.

En *Drupal 7* no es posible crear boletines de forma nativa, por lo que tenemos que instalar un módulo que satisfaga nuestro requisito. Disponemos de varios para elegir, nosotros hemos optado por *Simplenews*. Una vez instalado, debemos indicarle que contenido se va a incluir en los boletines. Nosotros optamos por las exposiciones, también se decide que todos

5.11- Configuración del Aspecto

los usuarios están suscritos por defecto, dejando a elección del usuario cancelar la suscripción cuando lo desee.

Al editar el tipo de contenido de las exposiciones notamos que tenemos una nueva opción:

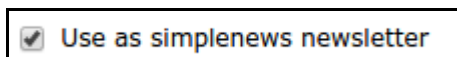


Figura 5.54- Opción para incluir en el boletín

Una vez configurado, todos los suscriptores recibirán el boletín periódicamente con las nuevas exposiciones que se creen.

5.11 Configuración del Aspecto

Siempre es bueno tener una imagen única para nuestro sitio web. Con *Drupal* podemos crear nuestro propio tema para personalizar la interfaz. Obviamente crear un tema desde cero es una ardua tarea. Podemos evitar eso felizmente gracias a la gran colección de temas que nos ofrece *Drupal*.

En el menú del administrador podemos encontrar la opción de *Apariencia*, donde visualizaremos los temas que tenemos actualmente activados. Por defecto, *Drupal 7*, nos ofrece dos temas. En la página oficial de *Drupal* tenemos a nuestra disposición más de 2.000 temas a día de la redacción de este proyecto.

A modo de ejemplo, elegiremos un tema que se adecue a nuestro proyecto para posteriormente instalarlo. El tema *Garland* cumple con los requisitos anteriores. Tras descargarla y descomprimirla en `%rutaDelProyecto%/sites/all/themes`, tan solo tenemos que dirigirnos al apartado encargado de gestionar los temas y activarlo.

Una vez activado y reconfigurado la situación de todos los menús ya tendremos disponible la nueva imagen de nuestro sitio. De forma adicional, creamos un logo fácilmente desde un recurso gratuito, online y sencillo como la página <http://www.designmantic.com/>.



Figura 5.55- Logo de la Galería de Arte Virtual

Esta es la nueva imagen de la plataforma. De una forma sencilla se puede cambiar el aspecto radicalmente.

5.11- Configuración del Aspecto



Figura 5.56- Nuevo aspecto de la plataforma

6 Conclusiones

En este último capítulo trataremos de exponer las conclusiones y resumen del proyecto. También se narrará lo que a nuestro criterio, deberían de ser trabajos futuros para la mejora de la plataforma, que no se han llevado a cabo en este trabajo, ya sea por falta de tiempo o porque excedía del objetivo del mismo.

6.1 Resumen

En este proyecto se ha llevado a cabo una plataforma virtual de arte, así como todo lo referente a un sitio web con requisitos de registro y gestión de usuarios. Como requisito previo, debe de ser implementado sobre un **gestor de contenidos** y todo lo utilizado ha de ser **software libre y gratuito**.

Carezco de experiencia en el desarrollo web con gestores de contenido, por lo cual, considero que la sección de elección del gestor de contenido con el que se ha realizado el proyecto no ha sido condicionado por trabajos pasados. *Drupal* tiene una curva de aprendizaje más dura que otros *CMS*. Considero que es un sistema que, aunque muy modularizado, nos presenta un sitio web con unas funcionalidades básicas recién instalado. Si queremos añadir más funcionalidades, debemos de buscar en el amplio catálogo de módulos que nos ofrece la página oficial de *Drupal*. Como comentamos anteriormente, es muy modularizado, por lo que en varias ocasiones para satisfacer un requisito, nos hemos visto obligados a instalar más de un módulo. Esto tiene como parte buena, que podemos diseñar el funcionamiento de una forma más precisa, pero también tiene su parte mala, es más complejo que instalar un módulo con el único trabajo de configurarlo.

A medida que el proyecto avanzaba, el concepto de *Drupal* va cambiando, pasando de considerarlo no solo un gestor de contenido, sino un **framework** muy completo. En este proyecto hemos ejemplarizado como se utiliza como marco de trabajo, creando un módulo usando las funciones *hook* que interaccionan con el núcleo. Por esto último, tenemos infinitas posibilidades, implementando módulos de cero, modificando los ya creados o simplemente adaptando el comportamiento del núcleo a nuestro gusto.

Uno de los puntos más negativos, ha sido comprobar que la traducción al español está lejos de ser completa. Sobre todo cuando se acude a la instalación de módulos de terceros. Son muchas opciones que no están traducidas o interfaces como botones que carecen de texto en español. Es una pequeña pega que tiene fácil solución pero que puede afean un proyecto perdiendo imagen profesional.

6.2 Mejoras y Trabajos Futuros

Vamos a exponer una serie de tareas que bajo nuestro criterio mejorará sustancialmente la calidad del proyecto.

Se debe de mejorar el **aspecto visual** para hacerlo más atractivo al usuario. Aunque actualmente el diseño es un diseño limpio y ordenado, tal vez una mejora supondría una mejor aceptación.

Conclusiones

Respecto al módulo que hemos implementado, requiere una mejora en la **seguridad**, ya que el cifrado usado es obsoleto y no muy seguro. No es un trabajo muy prioritario, ya que los datos transmitidos no son de especialmente críticos.

Añadir una **pasarela de pago**, para así realizar en la propia plataforma la compra y venta de obras. Actualmente no es posible, simplemente se acuerda un precio y desde fuera del sitio web se debe de efectuar la operación monetaria.

Realizar tareas de **SEO**. En un proyecto real es un área muy importante darse a conocer en la web. Concretamente se debe de centrar en las técnicas de SEO que hagan subir posiciones en el buscador de *Google*, ya que actualmente, y con gran diferencia, es el buscador más utilizado.

6.3 Aprendizaje Personal

Dado que carecía de experiencia en el desarrollo web en general y con gestores de contenido en particular, el aprendizaje ha sido muy alto. He conocido una tecnología como *Drupal*, la cual me ha sorprendido muy gratamente, a pesar de tener una curva de aprendizaje dura. No descarto realizar futuros proyectos con lo aprendido en la realización de este proyecto. He adquirido también nociones sobre **PHP**, que hasta la fecha solo disponía de un conocimiento muy básico. Así mismo, he aumentado el conocimiento de herramientas, como por ejemplo *Git*, que ha sido utilizado mediante *Github* donde está alojado todo el código del módulo *pujador*.

6.4 Problemas Técnicos Encontrados

Los principales problemas encontrados se centraron en el desarrollo del módulo propio. Mi escasa experiencia en PHP causó un retraso evidente en el proyecto. Gracias a los numerosos recursos que se pueden encontrar en la web y los libros referenciados en la sección *Referencias Bibliográficas* pude solventar el problema satisfactoriamente, además de adquirir nuevos conocimientos.

El otro gran escollo encontrado, fue la dificultad a la hora de *debugar* el código, ya que el servidor *Glassfish* se comportaba de una manera un tanto inestable. Finalmente tras indagaciones por la web y reconfigurando el servidor pude terminar el desarrollo de una manera normal.

Referencias Bibliográficas

- Recursos literarios

[Drupal API] Interfaz de programación de aplicaciones de Drupal (Accedido en Mayo del 2015)

<https://api.drupal.org>

[Drupal Módulos] Directorio oficial de módulos de Drupal (Accedido en Mayo del 2015)

https://www.drupal.org/project/project_module

[Drupal API] Directorio oficial de temas de Drupal (Accedido en Mayo del 2015)

https://www.drupal.org/project/project_theme

[CMSMatrix, 2015] Listado de CMS disponibles (Accedido en Mayo del 2015)

<http://www.cmsmatrix.org>

[Heurtel, 2014] Olivier Heurtel. *PHP 5.5 Desarrollar un sitio Web dinámico e interactivo*

[Charte Ojeda, 2004] Francisco Charte Ojeda. *PHP 5*

[Tomlinson, 2015] Todd Tomlinson. *Beginning Drupal 8*

[Hodgdon, 2015] Jennifer Hodgdon. *Programmer's guide to Drupal, 2nd Edition*

[Wikipedia, 2015] Definición de un Sistema de Gestión de Contenidos según Wikipedia (Accedido en mayo del 2015)

http://en.wikipedia.org/wiki/List_of_content_management_systems

[Pilone y Miles, 2008] Dan Pilone y Russ Miles. *Head First Software Development*

[IngSoftware, 2014] Ingeniería del Software de Gestión. Apuntes U.M.A.

[MetdDesSoft] Metodología para el Desarrollo de Software. Apuntes U.M.A.

- Software y Herramientas

[Glassfish] Servidor de aplicaciones (Accedido en Septiembre del 2015)

<https://glassfish.java.net>

[PHP] Lenguaje de código abierto (Accedido en Julio del 2015)

Referencias Bibliográficas

<https://php.net>

[MySQL] Sistema de gestión de base de datos (Accedido en Mayo del 2015)

<https://www.mysql.com>

[Github] Plataforma de desarrollo colaborativo. (Accedido en Mayo del 2015)

<https://github.com/>

[ArgoUML] Programa de modelado para UML. (Accedido en Julio del 2015)

<http://argouml.tigris.org>

[Bitnami] Página que reúne máquinas virtuales de multitud de entornos de desarrollo (Accedido en Mayo del 2015)

<https://bitnami.com>

[Netbeans] Entorno de desarrollo integrado libre (Accedido en Abril del 2015)

<https://netbeans.org>

[Designmantic] Página para crear logos online y gratuitos (Accedido en Octubre del 2015)

<https://www.designmantic.com/es>

Anexos

1. Instalación de un módulo

En este apartado se explicará como añadir un nuevo módulo a nuestro proyecto. Cada vez que queramos añadir una funcionalidad, tendremos que añadir uno o varios módulos. Existen varios tipos:

Core (núcleo):

Son los que se añaden por defecto cuando se instala *Drupal*. Son escasos ya que componen el núcleo, pero en cambio, son imprescindibles para satisfacer las funcionalidades básicas.

Contributed (contribuciones):

Son los desarrollados por la comunidad que da soporte a *Drupal*. Para mayor seguridad, se recomienda no descargarlos desde otra página que no sea la oficial.

Custom (personalizados):

Son módulos que expanden su funcionalidad mediante (generalmente) PHP.

Para instalar el módulo que necesitemos lo primero que debemos de hacer es descargarlo. Por seguridad se recomienda solo descargar desde la página oficial de *Drupal*.

Una vez localizado el módulo en cuestión debemos descargar la versión correspondiente con el núcleo de *Drupal* que estemos usando:

Version	Download	Date
7.x-2.12	tar.gz (30.86 KB) zip (38.65 KB)	2014-Aug-14
6.x-2.3	tar.gz (35.89 KB) zip (47.64 KB)	2009-Aug-15
Development releases		
Version	Download	Date
8.x-3.x-dev	tar.gz (36.66 KB) zip (59.15 KB)	2015-Oct-19
7.x-2.x-dev	tar.gz (30.86 KB) zip (38.66 KB)	2014-Aug-14
6.x-2.x-dev	tar.gz (30.96 KB) zip (38.58 KB)	2013-Oct-19

En nuestro caso, debemos descargar la versión *7.x2.12* ya que es la última versión estable. Debemos evitar las versiones en desarrollo ya que nos pueden acarrear problemas.

Una vez descargado debemos de descomprimir el fichero (ya sea en formato *tar.gz* o *zip*) y copiar el contenido en `sites/all/modules`. En este momento podemos considerar que el módulo está instalado, pero para hacerlo funcionar es necesario activarlo.

Para ello, debemos de irnos como administrador al menú *Módulos*, buscar el módulo que acabamos de instalar y marcar la casilla junto a su nombre. Por último solo debemos guardar los cambios pulsando sobre el botón del final de la página *Guardar configuración*.

2. Configuración SMTP

Cualquier sitio web medianamente grande necesita comunicarse con los usuarios constantemente. La mejor forma y más rápida es mediante correos electrónicos. Para que Drupal pueda gestionar el envío de correos, es necesario configurar un servidor **SMTP**.

En la mayoría de paquetes de servicios de almacenaje web que contratemos para albergar nuestro proyecto constará del servidor **SMTP** ya configurado. Por lo tanto lo único que debemos hacer es configurar Drupal como veremos más adelante.

Si nuestro proyecto lo hospedamos en un servidor propio, debemos de preocuparnos de tener instalado un servidor de correos, como puede ser *Dovecot* <http://www.dovecot.org>.

Otra opción es utilizar una cuenta de correo electrónico de un proveedor como puede ser **Gmail**. Ésta es la solución que hemos optado en este proyecto. Por lo cual solo debemos indicar los datos de los servidores de **Gmail**.

Sea cual sea la solución que optemos, debemos instalar y configurar el módulo **SMTP Authentication Support** que podemos encontrar en la página oficial de Drupal. Una vez instalado (consultar anexo nº 1) accedemos a configurar el módulo en la sección *Módulos* de la barra de menú superior estando autenticado como administrador de Drupal.

MAIL				
ACTIVADO	NOMBRE	VERSIÓN	DESCRIPCIÓN	OPERACIONES
	SMTP Authentication Support	7.x-1.0	Allow for site emails to be sent through an SMTP server of your choice.	Ayuda Permisos Configurar

En la página de configuración del módulo, debemos indicar el servidor SMTP que vamos a utilizar, en nuestro caso el que proporciona Gmail. Con el puerto correspondiente:

<p>SMTP server</p> <input type="text" value="smtp.gmail.com"/> <p>The address of your outgoing SMTP server.</p>	<p>SMTP port</p> <input type="text" value="465"/> <p>The default SMTP port is 25.</p> <p>Use encrypted protocol</p> <input type="text" value="Use SSL"/>
--	--

En el caso de tener un servidor **SMTP** propio deberíamos indicar la dirección donde se encuentra, así como el puerto asignado (por ejemplo, <http://localhost> y puerto 25).

<p>SMTP AUTHENTICATION</p> <p>Leave blank if your SMTP server does not require authentication.</p> <p>Nombre de usuario</p> <input type="text" value="galeria.arte.virtual.es"/> <p>SMTP Username.</p> <p>Contraseña</p> <input type="password"/>
--

Seguidamente debemos facilitar los datos con los que nos identificamos en **Gmail**. En nuestro caso nuestro nombre de usuario es *galeria.arte.virtual.es*.

A partir de ahora ya podremos enviar correos electrónicos con el remitente *galeria.arte.virtual.es@gmail.com*. Obviamente si configuramos un servidor **SMTP** propio o contratamos un dominio propio, el remitente lucirá de una forma más profesional tipo *administracion@gav.es*

En la sección *Módulos/user/Configurar/Opciones de la cuenta/Correos electrónicos* podemos configurar el asunto y cuerpo de los diferentes correos electrónicos disponibles. Por ejemplo, aquí vemos la configuración para el email de bienvenida:

<p>Asunto</p> <p><i>Un administrador ha creado para usted una cuenta en [site:name]</i></p> <p>Cuerpo</p> <p><i>[user:name],</i></p> <p><i>Un administrador del Sitio [site:name] ha creado una cuenta para usted. Ahora puede iniciar una sesión haciendo clic en este enlace o copiándolo y pegándolo en su navegador:</i></p> <p><i>[user:one-time-login-url]</i></p> <p><i>Este enlace es para un único inicio de sesión y le llevará a una página donde podrá establecer su contraseña.</i></p> <p><i>Tras establecer su contraseña, en el futuro podrá iniciar sesión identificándose en [site:login-url] con los datos:</i></p> <p><i>Usuario: [user:name]</i></p> <p><i>Contraseña: Su contraseña</i></p> <p>• <i>El equipo de [site:name]</i></p>
--

Se puede observar el uso de comando entre corchetes para referenciar objetos claves, como puede ser el nombre del usuario al que va dirigido el email [user:name] o el nombre del sitio [site:name].

3. Casos de Usos

Nombre	CU0001 – Registro de usuarios
Descripción	Cualquier visitante a la web tiene la posibilidad de registrarse en la misma
Actores Principales	Usuario no registrado
Satisface	RF001
Flujo Principal	<ol style="list-style-type: none"> 1. Actor principal accede a la página de entrada del sitio web 2. Accede al formulario de registro mediante link “<i>Crear nueva cuenta</i>” 3. Rellena su nombre de usuario y dirección de correo electrónico 4. El sistema valida que el nombre de usuario y la dirección de correo electrónico no estén en uso 5. El sistema envía un correo a la dirección indicada para verificarlo 6. Tras acceder a la dirección indicada en el email, accede al formulario donde establece su contraseña
Caso Alternativo	<p>Nombre de usuario ya usado</p> <p>4ª. El sistema detecta que el nombre de usuario ya está en uso</p> <p>5ª. Se muestra un mensaje anunciando que debe de elegir otro nombre</p> <p>Dirección de correo en uso</p> <p>4b. El sistema detecta que el correo electrónico ya está en uso</p> <p>5b. Se muestra un mensaje anunciando que debe de elegir otro email y da la posibilidad de acceder al formulario de contraseña perdida</p>
Pre-condición	Usuario no debe de estar registrado previamente
Post-condición	Un nuevo usuario se crea en el sistema

Nombre	CU0002 – Crear Artista
Descripción	Un administrador puede crea un perfil de artista.
Actores Principales	Administrador, Administrador de Drupal
Satisface	RF002
Flujo Principal	<ol style="list-style-type: none"> 1. El actor principal accede al menú Usuarios/<i>Añadir usuario</i> 2. Rellena los datos obligatorios en el formulario 3. En el apartado de roles seleccionar Artista

Anexos

Caso Alternativo	
Pre-condición	Artista no registrado previamente
Post-condición	Artista creado en el sistema

Nombre	CU0003 – Consultar Artista
Descripción	Un administrador puede consultar un perfil de artista. Consultar significa ver todos los datos registrado en el sistema, no solo los que muestra el perfil público.
Actores Principales	Administrador, Administrador de Drupal
Satisface	RF002
Flujo Principal	<ol style="list-style-type: none"> 1. El actor principal accede al menú <i>Usuarios</i> 2. Accede al perfil del artista seleccionando su nombre en la lista
Caso Alternativo	
Pre-condición	Artista existente en el sistema
Post-condición	

Nombre	CU0004 – Modificar Artista
Descripción	Un administrador puede modificar un perfil de artista.
Actores Principales	Administrador, Administrador de Drupal
Satisface	RF002
Flujo Principal	<ol style="list-style-type: none"> 1. El actor principal accede al menú <i>Usuarios</i> 2. Accede al formulario de edición mediante el botón <i>editar</i> que se encuentra en la lista de usuarios 3. Modifica los datos 4. Confirma los cambios
Caso Alternativo	
Pre-condición	Artista existente en el sistema
Post-condición	Artista existente en el sistema con datos actualizados

Nombre	CU0005 – Eliminar Artista
Descripción	Un administrador puede eliminar un perfil de artista. El administrador debe decidir si desea conservar los datos del artista o eliminarlos por completo de la base de datos.

Anexos

Actores Principales	Administrador, Administrador de Drupal
Satisface	RF002
Flujo Principal	<ol style="list-style-type: none"> 1. El actor principal accede al menú <i>Usuarios</i> 2. Accede al formulario de edición mediante el botón <i>editar</i> que se encuentra en la lista de usuarios 3. Pulsar en <i>Cancelar cuenta</i> <ol style="list-style-type: none"> 4a. Desactivar la cuenta y mantener su contenido 4b. Desactivar la cuenta y retirar de la publicación su contenido 4c. Eliminar la cuenta y atribuir todo su contenido al usuario <i>Anónimo</i> 4d. Eliminar la cuenta y su contenido
Caso Alternativo	
Pre-condición	Artista existente en el sistema
Post-condición	Artista no existe en el sistema

Nombre	CU0006 – Crear Exposición
Descripción	<p>Un administrador puede crear, modificar y eliminar una exposición de cualquier artista.</p> <p>Un artista puede crear, modificar y eliminar una exposición que sea suya</p>
Actores Principales	Administrador, Administrador de Drupal, Artista
Satisface	RF003
Flujo Principal	<ol style="list-style-type: none"> 1. El actor principal clic en “<i>Exposición</i>”, en el menú “<i>Navegación</i>”, bajo “<i>Agregar contenido</i>” 2. Rellena los datos obligatorios y opcionales que desee 3. Selecciona las obras que desee exponer 4. Pulsa sobre “<i>Guardar</i>”
Caso Alternativo	
Pre-condición	
Post-condición	Nueva exposición creada

Nombre	CU0007 – Consultar Exposición
Descripción	<p>Un administrador puede crear, modificar y eliminar una exposición de cualquier artista.</p> <p>Un artista puede crear, modificar y eliminar una exposición que sea suya</p>

Anexos

Actores Principales	Administrador, Administrador de Drupal, Artista
Satisface	RF003
Flujo Principal	1. El actor principal pulsa sobre el enlace de la exposición
Caso Alternativo	
Pre-condición	Exposición ha de existir en el sistema
Post-condición	

Nombre	CU0008 – Modificar Exposición
Descripción	Un administrador puede crear, modificar y eliminar una exposición de cualquier artista. Un artista puede crear, modificar y eliminar una exposición que sea suya
Actores Principales	Administrador, Artista
Satisface	RF003
Flujo Principal	1. Actor principal accede a exposición 2. Pulsa sobre la pestaña " <i>Editar</i> " 3. Realiza los cambios que desee 4. Pulsa sobre " <i>Guardar</i> "
Caso Alternativo	
Pre-condición	Exposición ha de existir en el sistema
Post-condición	Exposición se modifica con los nuevos cambios

Nombre	CU0009 – Eliminar Exposición
Descripción	Un administrador puede crear, modificar y eliminar una exposición de cualquier artista. Un artista puede crear, modificar y eliminar una exposición que sea suya
Actores Principales	Administrador, Administrador de Drupal, Artista
Satisface	RF003
Flujo Principal	1. Actor principal accede a exposición 2. Pulsa sobre la pestaña " <i>Editar</i> " 3. Pulsa sobre " <i>Eliminar</i> " 4. Volver a pulsar " <i>Eliminar</i> " para confirmar
Caso Alternativo	
Pre-condición	Exposición ha de existir en el sistema
Post-condición	Exposición deja de existir en el sistema

Anexos

Nombre	CU0010 – Crear Obra
Descripción	Un administrador puede crear, modificar y eliminar una obra de cualquier exposición. Un artista puede crear, modificar y eliminar una obra que sea suya
Actores Principales	Administrador, Administrador de Drupal, Artista
Satisface	RF004
Flujo Principal	<ol style="list-style-type: none"> 1. El actor principal clicca en “<i>Obra</i>”, en el menú “<i>Navegación</i>”, bajo “<i>Agregar contenido</i>” 2. Rellena los datos obligatorios y opcionales que desee 3. Selecciona las obras que desee exponer 4. Pulsar sobre “<i>Guardar</i>”
Caso Alternativo	
Pre-condición	
Post-condición	Nueva obra creada

Nombre	CU0011– Consultar Obra
Descripción	Un administrador puede crear, modificar y eliminar una obra de cualquier exposición. Un artista puede crear, modificar y eliminar una obra que sea suya
Actores Principales	Administrador, Administrador de Drupal, Artista
Satisface	RF004
Flujo Principal	<ol style="list-style-type: none"> 1. El actor principal pulsa sobre el enlace de la obra
Caso Alternativo	
Pre-condición	La obra ha de existir en el sistema
Post-condición	

Nombre	CU0012– Eliminar Obra
Descripción	Un administrador puede crear, modificar y eliminar una obra de cualquier exposición. Un artista puede crear, modificar y eliminar una obra que sea suya
Actores Principales	Administrador, Administrador de Drupal, Artista
Satisface	RF004
Flujo Principal	<ol style="list-style-type: none"> 1. Actor principal accede a obra 2. Pulsa sobre la pestaña “<i>Editar</i>” 3. Pulsa sobre “<i>Eliminar</i>” 4. Volver a pulsar “<i>Eliminar</i>” para confirmar

Anexos

Caso Alternativo	
Pre-condición	Obra ha de existir en el sistema
Post-condición	Obra deja de existir en el sistema

Nombre	CU0013 – Modificar Obra
Descripción	Un administrador puede crear, modificar y eliminar una obra de cualquier exposición. Un artista puede crear, modificar y eliminar una obra que sea suya
Actores Principales	Administrador, Administrador de Drupal, Artista
Satisface	RF004
Flujo Principal	<ol style="list-style-type: none"> 1. Actor principal accede a obra 2. Pulsa sobre la pestaña "<i>Editar</i>" 3. Realiza los cambios que desee 4. Pulsa sobre "<i>Guardar</i>"
Caso Alternativo	
Pre-condición	Obra ha de existir en el sistema
Post-condición	Obra se modifica con los nuevos cambios

Nombre	CU0014 – Solicitar nueva contraseña
Descripción	Cuando un usuario no recuerda de su contraseña, puede solicitar una nueva mediante un formulario.
Actores Principales	Todos
Satisface	RF005
Flujo Principal	<ol style="list-style-type: none"> 1. Actor principal accede a la portada del sistema 2. Presiona sobre el enlace <i>Solicitar una nueva contraseña</i> 3. Introducir el nombre de usuario o correo electrónico en el formulario 4. Acceder al enlace enviado al correo electrónico 5. Presionar en <i>Iniciar sesión</i> antes de que se cumpla el periodo de caducidad indicado en la pantalla 6. Introducir una nueva contraseña
Caso Alternativo	<p>Nombre o correo electrónico erróneo</p> <p>3a. Se introduce un nombre de usuario o contraseña no registrada en el sistema</p> <p>4a. Se notifica del error mediante un mensaje en pantalla</p>
Pre-condición	No estar autenticado en el sistema

Post-condición	El actor principal habrá seleccionado una nueva contraseña
-----------------------	--

Nombre	CU0015 – Iniciar Sesión
Descripción	Antes de acceder al contenido es necesario iniciar sesión para identificarse en el sistema
Actores Principales	Todos
Satisface	RF006
Flujo Principal	<ol style="list-style-type: none"> 1. Actor principal accede a la portada del sistema 2. Introduce nombre de usuario y contraseña 3. Accede al sistema autenticado
Caso Alternativo	Nombre o contraseña 3a. Introduce el nombre de usuario o contraseña erróneamente 4a. Se notifica del error mediante un mensaje en pantalla
Pre-condición	Actor principal no autenticado en el sistema
Post-condición	Actor principal autenticado en el sistema

Nombre	CU00016 – Crear Usuario
Descripción	Un administrador puede crear un perfil de usuario.
Actores Principales	Administrador, Administrador de Drupal
Satisface	RF007
Flujo Principal	<ol style="list-style-type: none"> 1. El actor principal accede al menú Usuarios/<i>Añadir usuario</i> 2. Rellena los datos obligatorios en el formulario 3. En el apartado de roles ya viene seleccionado por defecto usuario registrado, no seleccionar ningún otro
Caso Alternativo	
Pre-condición	Usuario no registrado previamente
Post-condición	Usuario creado en el sistema

Nombre	CU00017 – Consultar Usuario
Descripción	Un administrador puede consultar un perfil de usuario. Consultar significa ver todos los datos del usuario que se tiene almacenado en la base de datos.
Actores Principales	Administrador, Administrador de Drupal

Satisface	RF007
Flujo Principal	<ol style="list-style-type: none"> 1. El actor principal accede al menú <i>Usuarios</i> 2. Accede al perfil del usuario seleccionando su nombre en la lista
Caso Alternativo	
Pre-condición	Usuario existente en el sistema
Post-condición	

Nombre	CU00018 – Eliminar Usuario
Descripción	Un administrador puede eliminar un perfil de usuario.
Actores Principales	Administrador, Administrador de Drupal
Satisface	RF007
Flujo Principal	<ol style="list-style-type: none"> 1. El actor principal accede al menú <i>Usuarios</i> 2. Accede al formulario de edición mediante el botón <i>editar</i> que se encuentra en la lista de usuarios 3. Pulsar en <i>Cancelar cuenta</i> 4a. Desactivar la cuenta y mantener su contenido 4b. Desactivar la cuenta y retirar de la publicación su contenido 4c. Eliminar la cuenta y atribuir todo su contenido al usuario <i>Anónimo</i> 4d. Eliminar la cuenta y su contenido
Caso Alternativo	
Pre-condición	Usuario existente en el sistema
Post-condición	Usuario no existe en el sistema

Nombre	CU00019 – Modificar Usuario
Descripción	Un administrador puede modificar los datos de un perfil de usuario.
Actores Principales	Administrador, Administrador de Drupal
Satisface	RF007
Flujo Principal	<ol style="list-style-type: none"> 1. El actor principal accede al menú <i>Usuarios</i> 2. Accede al formulario de edición mediante el botón <i>editar</i> que se encuentra en la lista de usuarios 3. Modifica los datos 4. Confirma los cambios
Caso Alternativo	

Anexos

Pre-condición	Usuario existente en el sistema
Post-condición	Usuario no existe en el sistema

Nombre	CU0020 – Bloquear cuenta de usuario
Descripción	En determinadas ocasiones un usuario con los suficientes privilegios puede decidir bloquear la cuenta de otro de forma temporal
Actores Principales	Administrador, Administrador de Drupal
Satisface	RF008
Flujo Principal	<ol style="list-style-type: none"> 1. Actor principal accede al menú <i>Usuarios</i> 2. Accede al formulario de edición mediante el botón <i>editar</i> que se encuentra en la lista de usuarios 3. En el apartado <i>Estado</i> seleccionar <i>Bloqueado</i> 4. Confirmar cambios
Caso Alternativo	
Pre-condición	Cuenta activa de un usuario
Post-condición	Cuenta bloqueada de un usuario

Nombre	CU0021 – Pujar una obra
Descripción	Un usuario registrado o un artista realiza una puja sobre una obra que se encuentra en subasta
Actores Principales	Usuario registrado, Artista, Administrador, Administrador de Drupal
Satisface	RF010
Flujo Principal	<ol style="list-style-type: none"> 1. Actor principal accede a obra 2. Pulsa sobre “Pujar Obra” 3. Introduce importe a pujar 4. Recibe mensaje de puja realizada
Caso Alternativo	<p>Importe insuficiente</p> <p>4a. Recibe mensaje de error indicando que la cantidad no es suficiente</p> <p>Actor principal ya es máximo pujador</p> <p>4b. Recibe mensaje de error indicando que el actor principal ya es el máximo pujador.</p>
Pre-condición	Obra puesta en subasta
Post-condición	Obra cambia puja máxima y máximo pujador