Array Zone

Kernel Web & CMS

Rubén Arroyo Ceruelo
Kevin Cala Sánchez

Contenido

Plan de Empresa 4
Proyecto: KernelWeb & CMS 4
Proyecto Innovador 4
Necesidad 4
Análisis DAFO del proyecto 4
Público Objetivo 5
Disponibilidad del Proyecto 5
Proyecto: ArrayHosting 5
Disponibilidad del Proyecto 5
Análisis DAFO del proyecto 6
La empresa: ArrayZone 6
Análisis del Entorno 6
Forma Jurídica 7
Organigrama 7
Proceso de Creación de Empresa 8
Calculo de Coste para el primer año 8
Clientes 9
Primera facturación 9
Soporte 9
Instalación de los Servidores y los Servicios Iniciales 10
Instalación del Servidor y panel de Gestión 10
Panel de Control: i-MSCP 10
Implementación de un sistema de Monetización 11
Servicio de SVN 12
Configurando el entorno de desarrollo 13
Servicio NX para la administración 13
ArrayHosting 14
Planes de Alojamiento Web Ofrecidos 14
Plan de Copias de Seguridad 16
Organización de Copias de Seguridad 16
Organización Semanal 17
Planes de Copia para Clientes 17
Framework. ¿Qué son y para qué sirven? 19
Ventajas de utilizar un framework 19
¿Qué tipo de Framework utilizo? 19
¿Por qué KernelWeb y no otro sistema? 20
Kernel Web 21
Desarrollo del Núcleo 21
Contenido del Núcleo 21
Que NO contiene el Núcleo 22
Estructura de KernelWeb 22
Estructura de los proyectos diseñados con KernelWeb 23
Reescritura de la URL 25
Preparando KernelWeb para su uso 26

Instalación de KernelWeb..... 26
Configuración de KernelWeb..... 26
Configuración del Directorio System..... 26
Desarrollo de Plugins 27
¿Qué extensiones hay desarrolladas?..... 27
Quality Assurance 29
Nuestro primer proyecto de KernelWeb 30
Creación del Proyecto..... 30
Configurando el proyecto..... 30
Configurando la aplicación (app.conf.php):..... 30
Sobre las bases de datos (db.conf.php):..... 30
Generando nuestro contenido 30
Hola Mundo 31
Añadiendo el menú..... 31
Añadiendo el controlador..... 32
Vistas:..... 32
Traducciones:..... 33
Finalmente el menú (opcional):..... 33
Para terminar:..... 33
Optimización..... 34
Bibliografía y Enlaces..... 35

Plan de Empresa

Proyecto: KernelWeb & CMS

KernelWeb & CMS es el nombre de proyecto de un framework de código abierto programado en PHP pensado para crear aplicaciones web de una forma rápida y sencilla. KernelWeb está orientado a programadores y KernelCMS a gente con pocos o ningún conocimientos.

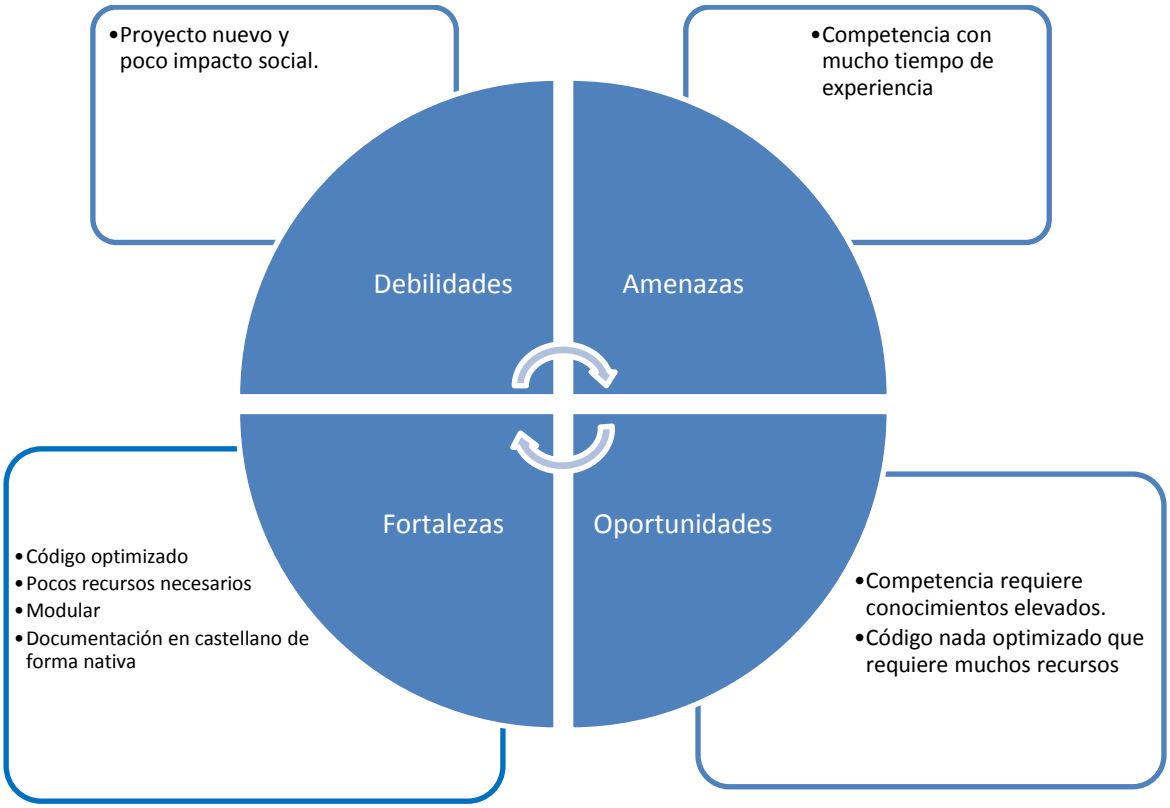
Proyecto Innovador

ArrayZone innova en la optimización y maleabilidad del código. Esto es así ya que KernelWeb, la base del núcleo del framework, viene con las funciones esenciales que toda aplicación web necesita. El framework es extensible gracias a la gran cantidad de plugins que serán ofrecidos. Los plugins son código añadido aparte de la base que solamente depende de funciones internas del Kernel y de sí mismo, con lo que se permite modular toda la programación.

Necesidad

Con alojamientos web cada vez más restrictivos en cuanto a consumo de recursos y código web devorador de recursos, cada vez es más importante la optimización de los recursos de una aplicación.
Al trabajar mayormente con plugins, la optimización del código que se ejecuta es mucho mayor y más corta que la de la competencia, ya que aplicaciones como WordPress o Joomla ejecutan muchas funciones que no son necesarias en gran parte de los casos, reduciendo la eficiencia del equipo.

Análisis DAFO del proyecto



Público Objetivo

KernelWeb está orientado a programadores con conocimientos que necesiten una base con funciones esenciales para utilizar en sus proyectos, el tiempo de implementación de una aplicación basada en KernelWeb es alto ya que requiere, pese a traer bastante trabajo hecho, crear las vistas de la aplicación y realizar todo el tratamiento de datos que sea necesario.
Si bien KernelWeb trae una base sólida para crear una aplicación, KernelWeb & CMS está orientado a usuarios con pocos o ningún conocimiento de programación o con poco tiempo disponible para programar que desean tener su página disponible de forma rápida ya que permite instalar una aplicación web en pocos minutos y sin ninguna complicación y editar su contenido de forma fácil y sencilla con un panel de administración.
Ambos proyectos incluyen un sistema de actualizaciones constantes por parte de ArrayZone.

Disponibilidad del Proyecto

KernelWeb & CMS es un proyecto que está orientado al uso profesional y no profesional y tiene incorporado un sistema multilenguaje puede ser utilizado globalmente independientemente de la localización del usuario.

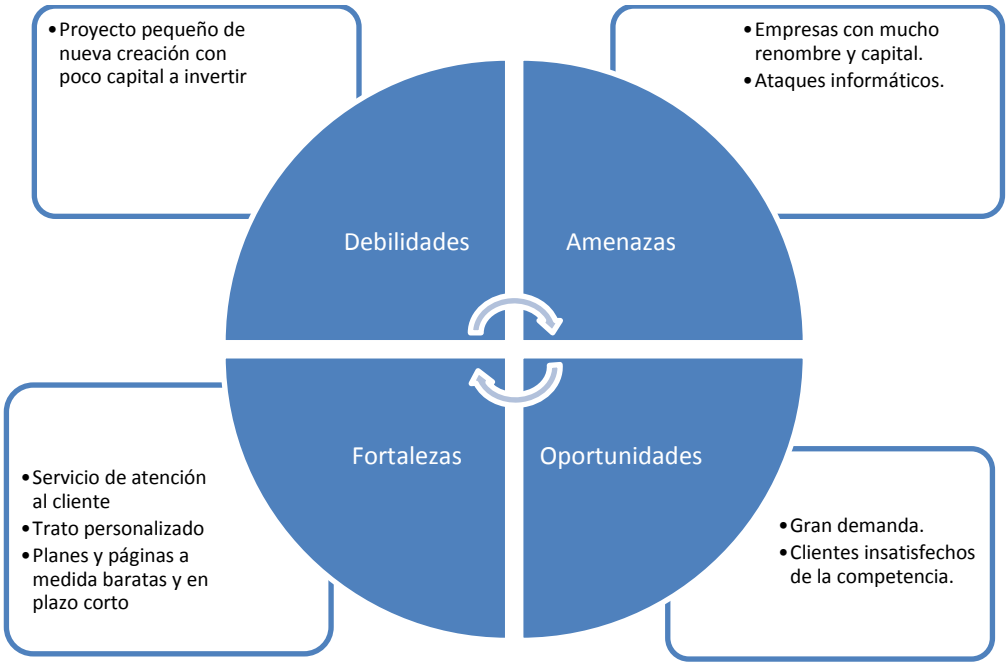
Proyecto: ArrayHosting

ArrayHosting es un servicio que trata en ofrecer un servicio de hospedamiento web dónde los usuarios pueden alquilar espacio en un servidor online 24 horas encendido para su página, tienda o aplicación al servicio de todos los usuarios. Además, el servicio también ofrecerá alquiler de servidores dedicados, tanto unitarios, es decir, dónde el usuario tendrá acceso completo al servidor, como virtuales, donde varios usuarios conviven en un mismo servidor compartiendo sus recursos. El proyecto de ArrayHosting se diferenciará del resto de empresas del mismo sector sobretodo en el servicio, ofreciendo un servicio preparado para atender a PYMES y grandes empresas que necesiten administración de un servicio con alta disponibilidad, disponible 24 horas al día, 7 días de la semana. Además, para incluir a los pequeños empresarios o aquellos que simplemente quieran tener su propia página web personal o una página comunitaria pueden contratarla a un precio muy bajo sin sacrificar un soporte adecuado.

Disponibilidad del Proyecto

ArrayHosting está orientado a trabajar dentro de España, aunque dado la facilidad de compra de servicios extranjeros utilizando Internet, cualquier usuario del mundo podrá contratar los servicios de este proyecto, ya que solo requiere de acceso a internet para hacer uso de él.

Análisis DAFO del proyecto



La empresa: ArrayZone

ArrayZone es la empresa creadora de KernelWeb & CMS que, además de ofrecer el framework PHP, ofrece alojamiento web y con planes de ofrecer Cloud. La financiación a largo plazo de la empresa es la venta de módulos y plantillas de pago y la construcción a medida utilizando KernelCMS de una página web a empresas y particulares.

Análisis del Entorno

Entorno Económico

El entorno económico no es problema ya que ArrayZone tiene proyectos gratuitos, como KernelWeb & CMS y proyectos asequibles como ArrayHosting.

Entorno Tecnológico

El mundo de la tecnología y las telecomunicaciones es muy cambiante con lo que hemos de tener muy en cuenta que todos los productos físicos, como servidores, mejorarán día a día y es posible conseguir productos similares o incluso mejores a menor precio. Por eso, ArrayZone quiere ofrecer servicios que tengan imagen de empresa muy concienciada con el medio ambiente, con lo que siempre que es posible, se intenta utilizar centros de datos limpios y respetuosos con el medio ambiente.

Entorno político y legal

Entendemos que el contenido que ofrezcan nuestros clientes utilizando ArrayHosting puede implicar posibles problemas por parte de la Administración. Por ello y para salvaguardar la imagen de la empresa, debemos crear unos Términos y Condiciones de un Uso Acep-

table del servicio para evitar el mal uso de nuestros servicios y además, evitar posibles multas.
Para ofrecer KernelWeb & CMS crearemos una licencia llamada Array-Zone Public License o AZPL, que permitirá modificar, redistribuir gratuitamente el código fuente. Para utilizar el código con objetivos comerciales se debe contactar con los desarrolladores para conseguir una licencia especial de uso comercial.

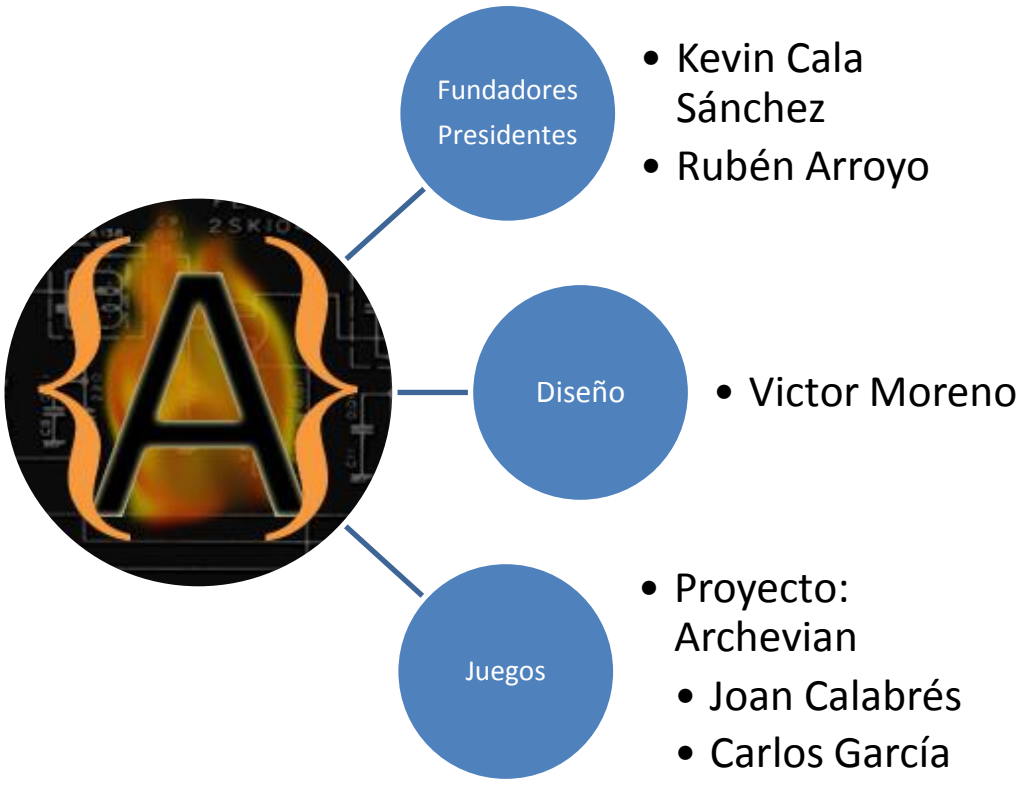
Forma Jurídica

La forma jurídica escogida para este proyecto es la de Sociedad Limitada Nueva empresa, una especialidad de la Sociedad de Responsabilidad Limitada (SRL) que gracias a que su objeto social es genérico, permite una mayor flexibilidad en el desarrollo de actividades empresariales sin necesidad de modificar los estatutos de una sociedad.
La denominación social inicialmente será *Rubén Arroyo Ceruelo ID-CIRCE SLNE* aunque según el portal CIRCE es posible cambiar el nombre de la denominación social a una de fantasía como ArrayZone.

Organigrama

ArrayZone se divide actualmente en tres departamentos:

- Administración y Fundadores: Se encarga de la administración de la empresa.
- Departamento de Diseño: Se encargan de producir los diseños y plantillas para ser utilizados en el proyecto KernelCMS
- Departamento de Juegos: Se encargan del desarrollo de juegos online. Ahora mismo existe un proyecto activo, llamado Archevian.



Proceso de Creación de Empresa

Para crear una empresa hay que realizar las siguientes acciones:

- Registro del Nombre en el Registro Mercantil Central: Coste: 40€.
- Rellenar el Documento Único Electrónico en el PAIT más cercano. Días: 1 mañana
- Obtener el capital mínimo para la fundación de la empresa. Días: lo que tarde tu banco…
- Cita en la notaría para firmar las escrituras: Coste: 60€. Días: 1 mañana.

Tiempo total invertido: 2 mañanas
Coste total de la fundación: 100€

Calculo de Coste para el primer año

Concepto	**Valor**	
Alquiler Servidor	188,61€	Anual
Dominio	8,46€	Anual
Salarios	0,00€	Anual
Coste de los Equipos	814€	Una vez
Coste Instalación del Servidor	12,00€	Una vez

Cada socio debe aportar 1501€ para el capital inicial y así fundar la empresa, aunque hasta que el proyecto no esté finalizado correctamente no se pondrá en marcha la creación de la empresa ya que el tiempo inicial hasta que se empiece a conseguir beneficios puede ser demasiado largo. La fuente de Financiación será con un crédito ICO a pagar en 1 año sin carencia con un 3,337% Nominal y un 3,412% TAE

Clientes

Primera facturación

A riesgo de parecer muy precipitado, ya contamos con cinco clientes, de los cuales tres son compañeros de clase. Los planes contratados por éstos ha sido el de **500 MB** para estudiantes, es decir, 0,72 céntimos / mes * 3 clientes = 2,16 € (a razón de 1 mes e contratación). Uno de los grupos ha contratado dos meses en su lugar (lo que asciende a 2,88€). Aunque parezca una cifra pequeña esto reduce las pérdidas.
Por otro lado, los otros dos clientes tienen contratado el pack de **1GB**, así que pagan anualmente **24€**, lo que asciende a un total de **50,88 €** el primer año. Luego a parte se debe contar instalaciones de CMS en algunos casos.

Soporte

Detrás de toda empresa de hosting hay un soporte que hemos tenido que ofrecer durante el crédito, lo cual nos habrá consumido un total de 10 h aproximadamente, tanto para ayudarles a solventar dudas con sus páginas como para soporte del servidor.

Instalación de los Servidores y los Servicios Iniciales

Instalación del Servidor y panel de Gestión

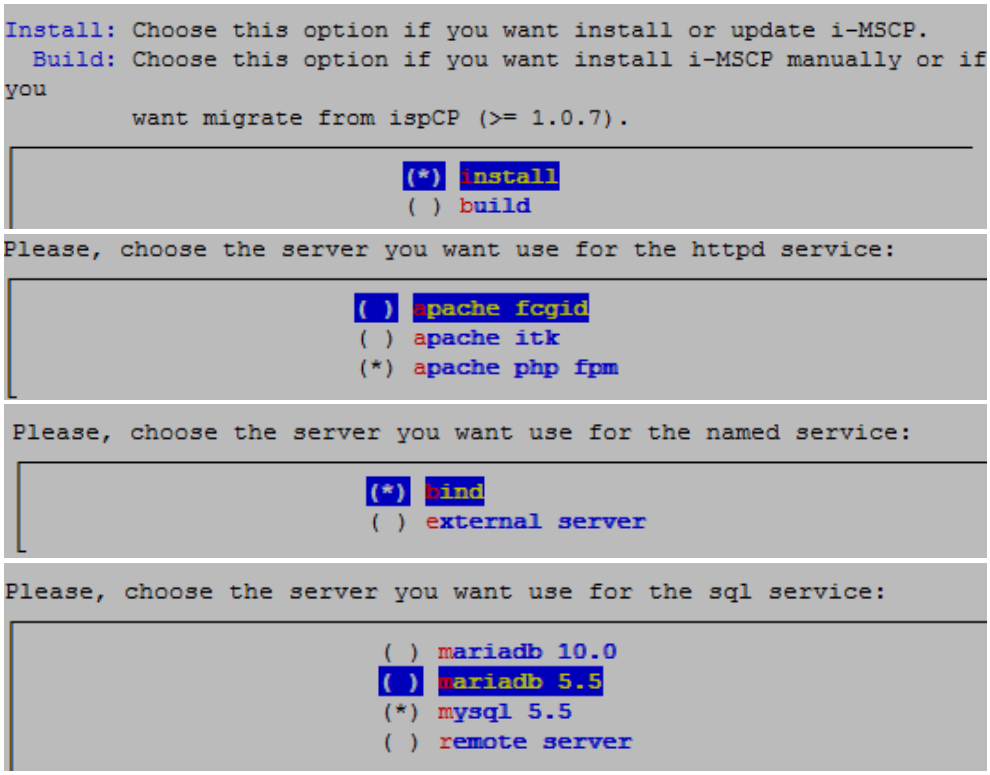
El servidor host de todo el proyecto es un servidor dedicado situado en Gravelines, Francia alquilado al proveedor de alojamiento OVH. Las características del mismo son las siguientes:

- Intel Atom CPU N2800 1,86GHz
- 4GB RAM
- 2x500GB SATA
- RAID 1
- 100Mbps unmetered.

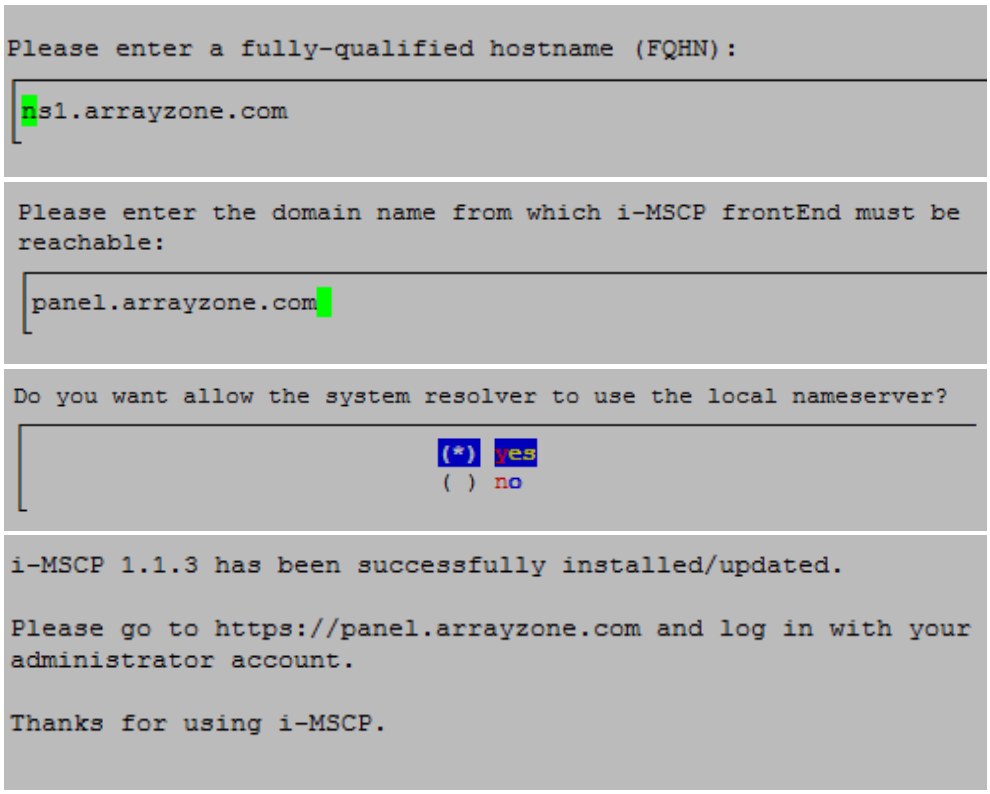
Panel de Control: i-MSCP

En nuestro servidor, aprovechando que lo tenemos, instalaremos un panel de gestión para vender hosting, i-MSCP.

- Ejecutamos "*tasksel install standard*" para tener el servidor con los recursos básicos como Apache2, PHP y MySQL.
- Cambiamos el hostname de la máquina. Al ser nuestro primer servidor dentro de la familia ArrayZone, lo llamaremos array1 *Echo 'array1.arrayzone.com' > /etc/hostname*
- Cambiamos a un directorio temporal *cd /var/tmp*
- Descargamos desde GitHub la última versión de i-MSCP, en este caso la 1.1.4[1] con wget.
- Ejecutamos el instalador: *perl imscp-autoinstall -d*
- Seguimos el instalador:



[1] En el momento de realizar esta parte del proyecto, la versión era la 1.1.4, aunque a día de 26 de Mayo 2014, la versión disponible es la 1.1.9. El servidor está en la versión más actualizada posible para evitar fallos de seguridad.



- Ya tenemos instalado el panel

Implementación de un sistema de Monetización

Para monetizar el servicio, implementaremos una plataforma de pago dentro de ArrayZone. La aplicación escogida es BoxBilling, que ofrece un sistema gratuito sin comisiones, con límite de cinco productos pero que nos va a ir muy bien al principio, hasta que podamos desarrollar nuestra propia pasarela de pago.

Para poder instalarlo, debemos instalar previamente una dependencia llamada ionCube, esta permite desencriptar código que se ha ofuscado con esta aplicación. Para cargar el loader necesitamos saber la versión de php de la que disponemos, para ello podemos hacer un pequeño script php, como por ejemplo:

```
<?php
    // Esta función enseña todos los datos de php.ini
    phpinfo();
?>
```

Nuestro servidor posee la versión 5.4.4, con lo que necesitamos el ioncube 5.4.so

Descargamos los loaders de ionCube de la página principal[2]

Editamos el archivo php.ini del servidor y añadimos la línea *zend_extension = /usr/local/ioncube/ioncube_loader_lin_5.4.so* antes de que aparezca cualquier *zend_extension* en el archivo php.ini

Una vez hecho esto, reiniciamos apache con el comando *service apache2 restart*. Una vez esto, seguimos el instalador de BoxBilling,

[2] http://www.ioncube.com/loaders.php

que te pide información para la base de datos y los datos de conexión de un usuario administrador.

Servicio de SVN

Subversion o SVN es una herramienta de control de versiones open source basada en un repositorio que se asemeja a un sistema de archivos. Entre revisiones se almacena las modificaciones y así se utiliza el menor espacio posible. Subversion permite acceder a los ficheros a través de la red, lo que permite utilizarse alojando el servicio de SVN en un servidor y accediendo a él para crear, modificar o eliminar ficheros.

Las ventajas de utilizar este sistema son:

- Se sigue la historia del archivo a través de copias
- Las modificaciones son atomicas. Es decir, entre reversión y reversión solo hay una versión, no subversiones.
- Solo se envían las diferencias en ambas direcciones, no archivos completos.
- Puede ser utilizado mediante Apache, sobre WebDav. Esto permite que SVN sea utilizado de forma transparente.
- Permite bloquear archivos para que no sean editados por varias personas a la vez.
- Cuando se utiliza con Apache, permite utilizar las opciones de verificación para autenticación.

Pese a estas ventajas, tiene sus problemas:

- El manejo de archivos no es muy completo. Realiza una copia y un borrado.
- Al realizar varias acciones sobre una rama, puede tener errores al aplicar. Esto se soluciona siendo cuidadoso al intentar sincronizar.

El procedimiento de instalación es el siguiente:

```
# apt-get update
# apt-get install subversion
# apt-get install libapache2-svn
```

Ya tenemos instalado el SVN, ahora debemos crear un repositorio para nosotros. Para organizar las cosas, lo asignaremos en /srv con el siguiente árbol:

```
# mkdir -p /srv/svn/repos && mkdir -p /srv/svn/user_access
```

Creamos un repositorio llamado kernelweb en este directorio

```
/srv/svn/repos# svnadmin create kernelweb
```

Creamos los usuarios, para ello nos movemos a /srv/svn/user_access y realizamos el siguiente comando

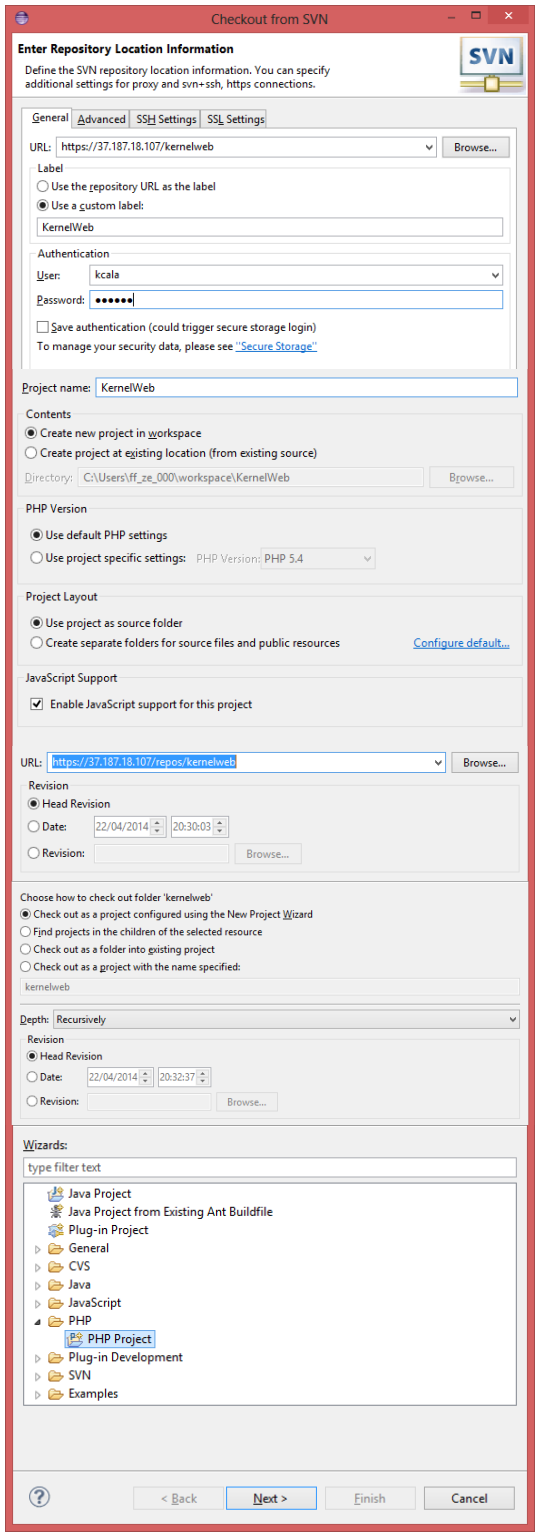
```
/srv/svn/user_access# htpasswd -c kernelweb_passwords kcala
/srv/svn/user_access# htpasswd kernelweb_passwords rarroyo
```

Entramos en /etc/apache2/mods-avaible/dav_svn.conf y añadimos al principio:

```
<Location /repos/kernelweb >
  DAV svn
  SVNPath /srv/svn/repos/kernelweb
  AuthType Basic
  AuthName "ArrayZone KernelWeb Repository"
  AuthUserFile
/srv/svn/user_access/kernelweb_password
  Require valid-user
  SSLRequireSSL
</Location>
```

Para poder utilizar el sistema mediante IDE Eclipse, debemos dar los siguientes permisos.

```
#mkdir /srv/svn/repos/kernelweb/dav
#chown -R www-data:root
/srv/svn/repos/kernelweb/
#chmod -R 644 /srv/svn/user_access/
#chmod -R 777 /srv/svn/repos/kernelweb/
```



Configurando el entorno de desarrollo

Para desarrollar este proyecto utilizaremos un entorno de desarrollo, o IDE, llamado Eclipse RPC. Eclipse es un programa compuesto por herramientas de programación multiplataforma programado en JAVA. Para ello añadiremos varios plugins para poder trabajar tanto como PHP como con SVN.

Necesitamos instalar los siguientes plugins:

- PHP Development Tools (PDT)
- SVNKit 1.8.3 Implementation (Optional)
- Subversion SVN Team Provider

Una vez instalados, realizamos las siguientes acciones para utilizar Eclipse CMS con el servicio de Subversion.

Servicio NX para la administración

Para administrar el servidor mediante ventana y así evitar el problema de la incomodidad de la línea de comandos, vamos a instalar un servidor de NX, en nuestro caso y dada la magnitud de nuestro proyecto, utilizaremos la versión gratuita de NoMachine, ya que ofrece hasta una sesión concurrente por servidor. Para instalar nxserver, debemos hacer lo siguiente:

```
# wget http://goo.gl/ekP13G
# dkpg -i nomachine_4.2.21_1_amd64.deb
```

Para utilizar NX, debemos instalar el cliente en el equipo, al estar utilizando Windows, lo descargamos del enlace indicado en la bibliografía.

ArrayHosting

ArrayHosting es un servicio que trata en ofrecer un servicio de hospedamiento web dónde los usuarios pueden alquilar espacio en un servidor online 24 horas encendido para su página, tienda o aplicación al servicio de todos los usuarios. Además, el servicio también ofrecerá alquiler de servidores dedicados, tanto unitarios, es decir, dónde el usuario tendrá acceso completo al servidor, como virtuales, donde varios usuarios conviven en un mismo servidor compartiendo sus recursos.

Planes de Alojamiento Web Ofrecidos

Como ArrayHosting es un proyecto desarrollado por estudiantes y estamos concienciados en que mucha gente necesita un alojamiento barato y completo. Por ello ponemos a disposición de estudiantes un descuento en la suscripción anual de un plan de hosting.
Los planes disponibles son:

- **Pack 100M**
  - Pack para páginas web que contienen únicamente contenido en HTML y javascript. Incluye:
    - 100 Megabytes de espacio en disco
    - 1.100 Megabytes de transferencia
    - 1 dominio permitido
    - Sin Base de Datos
    - **Sin PHP**
    - Soporte mediante el foro

      Todo por: 5€/año o 0,50€/mes

- **Pack 1G**
  - Páginas pequeñas o proyectos que acaban de empezar.
    - 1.000 Megabytes de espacio en disco
    - 11.000 megabytes de transferencia
    - 2 Dominios permitidos
    - Hasta 5 bases de datos
    - 20 Subdominios
    - **PHP**
    - Soporte mediante foro

      Todo por 20€/año o 2€/mes

- **Pack 2G**
  - Páginas pequeñas con trafico
    - 2.000 megabytes de espacio en disco
    - 22.000 megabytes de transferencia
    - 4 dominios permitidos
    - 50 subdominios

        - PHP
        - Asistencia por correo 24H
          Todo por 30€/año o 3€/mes

- **Pack 5G**
    - Proyectos multidominio
        - 5.000 megabytes de espacio en disco
        - 55.000 megabytes de transferencia
        - 10 dominos permitidos
        - 25 bases de datos
        - 100 subdominios
        - PHP
        - Asistencia por correo 24H
          Todo por: 60€/año o 6€/mes

- **Pack 10G**
    - Proyectos con necesidades de alto rendimiento
        - 10.000 megabytes de espacio en disco
        - 111.000 megabytes de trafico
        - 25 dominio permitidos
        - 50 bases de datos
        - **Subdominios ilimitados**
        - PHP
        - Asistencia por correo 8x5
          Todo por 90€/año o 9€/mes

- **Plan 50G**
    - Empresa de reventa de hosting o proyectos con muchísima necesidad
        - 50.000 megabytes de espacio en disco
        - 555.000 megabytes de transferencia
        - 100 dominios permitidos
        - **Subdominios ilimitados**
        - **Asistencia telefónica 8x5 y correo 24H**
        - **100 bases de datos**
        - Todo por: 15€/mes o 150€/año
        -

Los estudiantes tienen ofertas de hasta un 54% de descuento en los planes anuales. Para beneficiarse de esta promoción, deben enviar escaneado una copia del carné de estudiante del año en curso. Los planes disponibles para estudiantes con hasta 5 meses gratis pagando anualmente son:

- 500MB Estudiantil
    - Estudiantes para realizar pruebas o publicar su curriculum, portifolio…

- 500 megabytes de espacio en disco
- 5.500 megabytes de transferencia
- 1 dominio permitido
- 10 subdominios
- 1 base de datos
- PHP
- Soporte mediante el foro o correo 24H

Todo por: 5€/año o 1€/mes

- Pack 1G Estudiantil
  - Ofrece las mismas características que el Pack 1G.

    Todo por 14€/año (Descuento del 30%)
- Pack 2G Estudiantil
  - Ofrece las mismas características que el Pack 2G.

    Todo por 21€/año (Descuento del 42%)
- Pack 5G Estudiantil
  - Ofrece las mismas características que el Pack 5G.

    Todo por 42€/año (Descuento del 54%)

Plan de Copias de Seguridad

El panel de control iMSCP tiene un sistema automático de copias de seguridad (backups) que permite realizar las mismas de forma automática.

Si entramos en la configuración del panel, situada en */etc/imscp/imscp.conf* disponemos de las siguientes líneas:

```
# Crontab delayed tasks
BACKUP_HOUR = 23
BACKUP_MINUTE = 40
BACKUP_IMSCP = yes
BACKUP_DOMAINS = yes
BACKUP_ROOT_DIR = /var/www/imscp/engine/backup
```

Donde podemos escoger el momento de la copia de seguridad y si se realizan copias de seguridad. Además de este sistema automático, definiremos un sistema de copias gestionado por un cron que realizará copias de seguridad de las partes sensibles del servidor para una rápida recuperación en caso de fallos.

Organización de Copias de Seguridad

Para poder mantener una alta tolerancia a fallos, decidimos realizar varios tipos de copias de seguridad, según lo crítica que sea la información que contenga el volumen.

Por ello, realizaremos las siguientes copias:

- **Para el sistema:** El sistema operativo no va a ser modificado apenas, por ello realizaremos una copia de seguridad del sistema entero. Además, el servidor dispone de dos discos duros de 500GiB organizados en RAID-1, con lo que la posible pérdi-

da de datos es muy reducida, teniendo en cuenta los recursos de la empresa.

- **Para las bases de datos:** Al ser contenido muy sensible realizaremos una copia completa cada día. Una vez los datos almacenados en las bases de datos sean muy altos, fragmentaremos la copia de seguridad para aumentar la tasa de transferencia.
- **Para los archivos de los usuarios:** Los usuarios tienen un límite de espacio en sus dominios. Por ello podemos realizar copias de seguridad semanales completas e incrementales o diferenciales entre semana. Valga decir que el servicio está bajo la redundancia de un RAID-1 en dos discos.

Organización Semanal

Para evitar el uso intensivo de los recursos cuando se está realizando una copia de seguridad, realizaremos por ahora las copias de seguridad a una hora tardía, véase las cuatro de la mañana, para evitar modificaciones mientras se ejecutan las copias.
Una vez la empresa ArrayZone tenga un poco más de éxito, está previsto aumentar el número de servidores, realizando un clúster entre ellos para poder repartir la carga de trabajo y que uno o varios nodos se encarguen de las copias de seguridad mientras el resto da servicio a la red.

	Lunes	Martes	Mierc.	Jueves	Viern.	Sábado	Domin.
Sistema	Una vez						
Base de Datos	4:00 Completa	4:00 Completa	4:00 Completa	4:00 Completa	4:00 Completa	4:00 Completa	4:00 Completa
Archivos de Clientes	4:00 Completa	4:00 Diferen.	4:00 Diferen.	4:00 Diferen.	4:00 Diferen.	4:00 Diferen.	4:00 Diferen.

Planes de Copia para Clientes

Todos los ficheros de los clientes siempre son objeto de copia de seguridad aunque estas no están disponibles para su uso ya que se reservan exclusivamente para posibles fallos de hardware y poder así mantener la alta disponibilidad del servicio.
Pese a esto, los clientes tienen disponible un sistema de copia de seguridad y recuperación de datos automática que pueden contratar mensualmente como una suscripción que se sumaría al coste mensual de los paquetes de hosting que tengan contratado o contratar una recuperación puntual. Los planes de copia incluyen los archivos de los dominios y las bases de datos de los usuarios.

Plan de Copias	Precio Mensual	Recuperaciones al mes	Lunes a Domingo

1S	20€ por Restaurar	Pago por restauración	4:00 Completa (Semanal). Ultima copia de seguridad disponible.
1D	1€/1GB Extra: 0,20€/GB	2	Copia diaria incremental o diferencial con copia completa cada 7 días. Se almacenan hasta 7 días de copias de seguridad.
6D	5€/1GB Extra: 0,20€/GB	5	Copias cada 4 horas incrementales o diferenciales con copias completas cada día. Se almacenan hasta 7 días de copias de seguridad.
24H	20€/1GB: Extra: 0,20€/GB	15	Copias horarias diferenciales con copias completas cada día. Se almacenan hasta 7 días de copias de seguridad.

Este sistema está para aquellos clientes que tengan proyectos especiales que requieran una realización de copias más exhaustiva y ellos no puedan hacerse cargo de las mismas. Además, al realizarse de forma automática, permite olvidarse de las copias y utilizarlas solo cuando hace falta.

Framework. ¿Qué son y para qué sirven?

Un framework es un **esquema para el desarrollo** de una aplicación. Optamos por dar mucha facilidad al usuario pero siempre dentro de unos márgenes de funcionamiento, con lo que se ofrece flexibilidad y seguridad al usuario.
Los Framework están siguiendo el estándar **MVC**, Modelo-Vista-Controlador, fragmentando el código según su función.

- El **modelo** maneja las operaciones lógicas y de manejo de los datos.
- En la **vista** está definida la interfaz visual de la página.
- El **Controlador** define todas las acciones y funciones del tratamiento que se recogen del modelo y se muestran en la vista.

Ventajas de utilizar un framework

Utilizando un framework como KernelWeb se derivan unas ventajas:

- El programador no **necesita plantearse la estructura general** de su aplicación ya que el framework le indica una metodología a seguir para generar un código adecuado al sistema que está utilizando.
- Facilita la **colaboración** entre usuarios ya que al compartir un mismo código fuente todos los programadores es más fácil crear un código que pueda servir a otros usuarios de la red y permite desarrollos colaborativos.
- Facilita el uso de **herramientas o plugins** adaptadas al framework en concreto para facilitar el desarrollo.
- **Mayor seguridad** y nivel de **actualizaciones** del código base, ya que al pertenecer a una comunidad, cuando uno de los miembros reporta un error, suele salir un parche que lo soluciona. Si no utilizáramos un framework, deberíamos localizar primero los posibles fallos e ir viendo con el tiempo los posibles errores.
- La **Curva de Aprendizaje** es exponencial, eso quiere decir que, al principio la evolución será muy lenta ya que aprender el funcionamiento de un framework puede ser igual, en algunos casos, a aprender un lenguaje nuevo, pero una vez se tienen las bases, crear una aplicación desde cero es mucho más rápido que haciéndolo manualmente.

¿Es posible no utilizar un framework? Sí, por supuesto, pero se pierden las ventajas de utilizar un framework.

¿Qué tipo de Framework utilizo?

Al iniciar el análisis del proyecto a realizar se debe tener en cuenta aspectos como el lenguaje, metodología, tipo de aplicación. Al escoger el lenguaje de programación, debemos escoger si programamos desde cero, es decir, creando absolutamente todo el código

de nuestra aplicación nosotros mismos o utilizar una herramienta preparada para proyectos, un framework.

¿Por qué KernelWeb y no otro sistema?

Escoger un framework es igual que escoger un lenguaje de programación, según el proyecto que se quiera desarrollar, es mejor utilizar uno u otro.
Para crear una aplicación nativa en un móvil, la mejor opción, aunque no la única, es *Java for Android*, para programar en aplicaciones multiplataforma, *Java* o *Perl*. Si queremos hacer un proyecto web, mucha gente optaría por PHP en contra de otros lenguajes.
En el entorno web, como hemos visto antes, podemos programar directamente o utilizar un framework como base.
Si escogemos los framework, tenemos complicadísimos como Symphony, Yii, Zend, etc.. Muchos de estos proyectos abogan para ofrecer la mayor cantidad de recursos necesarios, tantos, que es posible que exista alguna función ofrecida que no haya sido utilizada todavía por nadie.
KernelWeb & CMS es un **framework en PHP** que ofrece y apuesta por la reducción de operaciones necesarias para ofrecer una web, aumentando la seguridad en menos código y por lo tanto más optimizado y con menos tiempo de compilación.

Kernel Web

KernelWeb se compone de un Core o Núcleo, que contiene todo lo necesario para poder ejecutar alguna aplicación con KernelWeb, y de los Plugins, que son aplicaciones extras que permiten añadir funcionalidades al Kernel. Mientras que el primero es necesario para que una aplicación funcione bajo KernelWeb la segunda solo es necesario si la aplicación requiere la funcionalidad del plugin.

Desarrollo del Núcleo

Al ser la base funcional de KernelWeb, el Core es la parte más ligera que incluye lo necesario para que, además de ofrecer pequeñas funciones, se puedan utilizar plugins.
KernelWeb está pensado para que esté lo más optimizado posible, de forma que el Kernel solamente traerá funciones esenciales, como paginación, caché, gestión de proyectos e implementación y gestión de plugins. Aunque pueda parecer poca cosa, lo importante del Kernel y donde están las mayores posibilidades de personalización aparece en los plugins.

Contenido del Núcleo

El contenido del Núcleo es:

- **Internacionalización y Localización** (I18N y L10N): El Núcleo viene con un sistema preparado para integrar el lenguaje en los proyectos futuros siguiendo el esquema del estándar I18N.
- **Gestión de las peticiones** (POST, GET,…): Además, incluye un gestor de peticiones, ya sean POST o GET, incluyendo, SESSIONS y REQUESTS para que hacer verificaciones con este sistema sea sencillo. El sistema automáticamente gestiona si existe o no y le aplica un valor indicado por el usuario.
- **Generador de Proyectos:** Permite crear un CMS básico con ejecutar un comando desde el administrador de Kernel (próxima versión).
- **Generador y Controlador de Formularios:** Una función integrada permite generar, controlar y verificar formularios evitando cientos de líneas de verificación.
- **Sistema de Registro** (log): Incluye un sistema de registros ya creado que permite registrar todos los sucesos importantes que sucedan en la aplicación.
- **Gestor de Plugins:** Como el núcleo depende en gran medida de las extensiones que le añadamos, también incluye de serie un gestor para poder incluir y excluir plugins evitando que sean cargados. Este gestor es independiente por cada utilización del Kernel, así pues, cada proyecto tendrá una configuración de Kernel específica.

- **Almacenamiento Caché:** Para optimizar tiempos de carga del contenido, Kernel Web permite activarla a selección y es dependiente del proyecto utilizado.
- **Gestor de plantillas:** Sistema que carga la plantilla indicada por el usuario. Por defecto solo puede haber una plantilla simultánea, para que el usuario pueda cambiarla se tendría que programar de forma manual encima de la propia aplicación (realmente es sencillo hacerlo). Las plantillas que vienen por defecto son todas responsivas (adaptativas), es decir, se ajustan a la resolución del monitor con el fin de que se vea bien la web en cualquier dispositivo (incluidos teléfonos moviles).
- **Carga automática:** Con el fin de optimizar los recursos y las acciones que debe hacer el usuario, las extensiones y otros archivos específicos se cargan de forma automática únicamente cuando son llamados. Esto se hace a través de un "trigger" (disparador) que tiene PHP.

Que NO contiene el Núcleo

KernelWeb está tan optimizado que muchas funciones que se considerarían esenciales no están disponibles. Esto es así ya que estas se encuentran programadas como Plugins, para facilitar que una aplicación que no las requiera no esté obligado a cargarlas aunque no las use. KernelWeb no incluye, por ejemplo:

- Conexión y Gestión de Bases de Datos (en desarrollo)
- Gestión de Usuarios (próximas versiones)

Estructura de KernelWeb

KernelWeb utiliza una estructura de ficheros muy definida. Para aumentar la optimización de la que el proyecto quiere caracterizarse, está diseñado para que el código de KernelWeb esté únicamente en un lugar del servidor y varias webs puedan realizar llamadas al código. Esto permite un ahorro de tiempo, ya que solo hay que actualizar un repositorio[3] en vez de los archivos de todas las páginas y reduciendo costes al tener que ocupar menos espacio para las copias de seguridad de los servicios.
Gracias a que KernelWeb es un framework PHP con tolerancia al Server Side Include[4], con lo que muchas de las respuestas son en código PHP o HTML, no es posible utilizar respuestas javascript o css a no ser que incluyamos dicho contenido en un retorno de php para que se muestre en el navegador o esté en una carpeta accesible desde internet.
El framework está organizado en carpetas, cuya raíz es:

[3] Lugar donde está almacenada la copia pública de KernelWeb
[4] Server Side Include es una funcionalidad de PHP que permite incluir código que no se encuentra directamente disponible desde la raíz de ficheros, para ello se crean ficheros especiales que se asocian

- **Extensions**: son aquellos plugins que complementan la aplicación básica. Se vinculan automáticamente
- **Kernel**: carpeta que contiene las clases principales e integradas de KernelWeb
  - **Config.php**: Configuraciones básicas del kernel (y parte de las aplicaciones).
  - **Kw.php**: Incluye las funciones principales del núcleo, además de ser utilizado como variable global principal.
  - **Kwcontroller.php**: Utilizado en los controladores de los proyectos finales para gestionar algunas acciones.
  - **Kwmodel.php**: En desarrollo. Utilizado para los modelos.
  - **Kwview**.php: Funciones genéricas utilizadas en las vistas.
  - **Rewriter.php**: Función para reescribir (y cambiar el idioma). Dependiendo si hay o no reescritura se invocará de una u otra forma.
- **System**: Gestor de administrador del Kernel
- **Main**.php: Loader de KernelWeb. Se encarga de ejecutar y vincular KernelWeb.

Estructura de los proyectos diseñados con KernelWeb

Al ser un framework que trabaja con el modelo MVC, todos los proyectos en los que se utiliza KernelWeb se debe utilizar una estructura muy definida y que debe ser la siguiente para no tener errores.

- **index.php**: Incluye el kernel (main.php), inicializando el proyecto de forma completamente automática, sin interacción del desarrollador final (siempre y cuando invoque **start()**).
- **Cron**.php: Opcional. Script que debe ejecutarse cada cierto tiempo (a través de un cron por ejemplo) el cual ejecuta tareas de limpieza y optimización (definidas por el usuario).
- **Caché**: Directorio que contiene la caché de la web de forma que no tenemos que recargarlos continuamente desde el servidor, sino que solo se compilan cada cierto tiempo o cuando hagamos modificaciones desde la administración. Cuando mudemos nuestro proyecto, este directorio debe excluirse ya que nuevos archivos de caché serán regenerados. El contenido de este directorio es rotativo, es decir, se eliminan y se crean de forma automática por KernelWeb a través de un cron que hay que iniciar.
- **Private**: Contiene todos los archivos los cuales no deben estar directamente disponibles desde los navegadores cliente. De esta forma evitamos que contenido sensible, como el código PHP pueda llamarse directamente como una url.

- **Config**: Almacena las configuraciones del proyecto en cuestión. Podemos crear tantos archivos de configuración como queramos, aunque KernelWeb utiliza por defecto dos:
  - **app.config.php**: Este archivo se encarga de la configuración principal de la aplicación. Esta puede ser modificada por el usuario externamente o modificar los datos en el código PHP para que carguen otras plantillas para un módulo en específico, etc,…
  - **db.config.php**: Las bases de datos que se cargarán de forma automática. Pese a ser cargadas de forma automática, estas son cargadas única y exclusivamente al utilizarse, eso quiere decir que el usuario se ahorra un tiempo de procesado si no utiliza bases de datos en una página ya que no se intenta contactar con bases de datos.
- **Models**: En este directorio debemos almacenar todos los modelos de configuración para contactar con la base de datos.
- **Modules**: Los módulos de nuestra aplicación se deben almacenar en estar carpeta. Un módulo es un contenedor de proyectos. Por defecto disponemos dos:
  - **Frontend**
  - **Backend**

  Aunque podemos crear tantos como queramos. Un módulo debe contener:
  - **Controllers**: Contiene los controladores principales de la aplicación –aquellos que ejecutan las acciones-.
  - **Views**: Contiene las vistas que se mostrarán, esto no incluye ni las plantillas, es decir, únicamente el contenido en sí.
  - **Data**: Contiene los ficheros de datos, entre ellos podemos encontrar los menús de la página.
  - **Lang**: La carpeta de lenguaje es dependiente del módulo que tengamos cargado. Cada carpeta de lenguaje puede tener tantos lenguajes como localizaciones queramos.
    - **Localización**: Dentro de lang, debemos crear una carpeta por cada idioma que queramos aplicar. El estándar de KernelWeb es i18n, por ejemplo, para poner Español de España, es_ES, inglés de estados unidos, en_US.
      - **Archivo de Lenguaje**: Dentro de la localización, es posible crear todos los archivos que uno quiera para escribir traducciones.

- **Public:** Este directorio contiene todo lo que es visible desde el exterior, véase imágenes, javascript, css, etc… Podemos crear tantos directorios como queramos. Dentro de public encontramos las plantillas que están dentro de *templates*, de forma que es muy fácil añadir, eliminar y escoger plantillas. Cada plantilla tiene su funcionamiento y sus menús, que es cargado a través de las funciones internas de la plantilla y de la caché.

Reescritura de la URL

KernelWeb funciona a través de la reescritura de la barra de direcciones, eso quiere decir que, en lugar de mostrar los GET de la forma habitual (con & y ?), se ocultan automáticamente organizando la URL de forma comprensible para el usuario.
Muchos servicios este detalle no lo permiten (**mod_rewrite** de Apache), por ello existe una función interna del Kernel llamada *rewrite* que se encarga de mostrar la barra de navegaciones según la disponibilidad de utilizar mod_rewrite o no.
Esta función se encuentra dentro del directorio Kernel y en ella podemos ver que disponemos de un condicional que según su resultado carga una función de rewrite u otra. De esta forma evitamos repetir múltiples veces la comprobación del estado de mod_rewrite. Este detalle no hace falta que el programador lo sepa ya que es automático, según la configuración del Kernel.

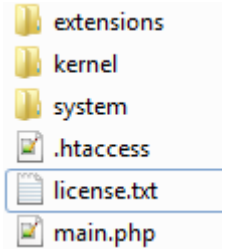
```
if (isset(kw::$config['rewrite']) and kw::$config['rewrite']) {
	// Function when rewrite is on
	function rewriter($module = '', $controller = '', $action = '', $get = '') {}
} else {
	// Rewrite is off? No problem
	function rewriter($module = '', $controller = '', $action = '', $get = '') {}
}
```

Preparando KernelWeb para su uso

Instalación de KernelWeb

Es recomendable realizar comprobaciones de todo lo que se quiera probar en local y una vez se hayan acabado y se quiera subir todo al servidor, solo se deberá mover las cosas a través de FTP o similar.

1. Si no lo tenemos ya, por ejemplo al alquilar un alojamiento web, instalamos un servidor web (si es necesario, también un servicio de base de datos). Un ejemplo de esto puede ser xampp[5].
2. Descargar la última versión de KernelWeb desde la página principal del proyecto.[6]
3. Una vez descargado, descomprimimos el proyecto y lo movemos a la carpeta web de nuestro servicio web. En el caso de utilizar xampp y Windows, dentro de c:\xampp\htdocs). No es necesario que esté en la raíz, puede estar en un subdominio, subdirectorio, alias, etc… ya que detecta automáticamente su localización para evitar problemas.
4. Una vez copiado, tendremos una imagen como la siguiente:



5. ¡Ya tenemos KernelWeb instalado!

Configuración de KernelWeb

Una vez instalado, podemos configurar algunos detalles de ejecución entrando en el directorio Kernel y modificando el archivo config.php

Como KernelWeb viene por defecto en modo desarrollo, no es necesario cambiar ningún dato en un principio.

En el futuro deberíamos cambiar la sección User Configs, que es donde están todas las configuraciones que puede modificar el usuario. La más importante de todas es la variable **debug** ya que es muy importante tenerla desactivada en modo producción.

Configuración del Directorio System

Antes de seguir explicando, debemos tener en cuenta que la carpeta *system* es un **gestor de administración** de KernelWeb. El framework no depende de esta gestor, con lo que podríamos eliminar el directorio system y KernelWeb seguiría funcionando sin problemas.

Para configurar system debemos ir a la ruta {web-dir}/*system/private/config/app.conf.php*.

[5] https://www.apachefriends.org/es/index.html
[6] http://kernel.arrayzone.com

Lo más crítico de este fichero es la configuración de los usuarios. Estos deben tener usuario y contraseña, estando la segunda encriptada en *SHA1*.
El usuario por defecto es **admin/test**.
Desde este mismo archivo tenemos una variable de configuración llamada *enableSystem* la cual permite activar o desactivar el acceso a este módulo de KernelWeb.

Desarrollo de Plugins

Tal y como está explicado en el Desarrollo del Núcleo, los plugins o extensiones añaden funcionalidad a nuestra aplicación. Por ejemplo podemos integrar un sistema automático de Caché HTML o de Paginación.
El framework incluye una carpeta donde todas las extensiones deben ser colocadas para ser llamados desde el sistema. Optimizando el consumo de recursos, las clases y funciones de las extensiones solo serán cargadas cuando estas sean necesarias, por ello hay que seguir un estándar de nombramiento de los archivos, para que estos puedan ser cargados sin problema por el sistema automático de inclusión de extensiones.
Este sistema funciona buscando el nombre de la extensión que se intenta cargar dentro de la carpeta homónima dentro de extensiones.
Por ejemplo, si queremos intentar acceder a la clase *Paginador*, cargamos con kw::init("Paginador"); y el sistema intenta cargar dentro de la carpeta extensiones, el plugin que está situado en /extensiones/Paginador/Paginador.php.

¿Qué extensiones hay desarrolladas?

Hasta la fecha, hay desarrollados 16 extensiones que pueden ser utilizadas por los usuarios si estos quieren, estas son:

- **Base64Encoder:** Clase que puede utilizar el usuario para codificar archivos PHP específicos, útil si pretende hacer una pequeña obfuscación de algún archivo.
- **BasicXML:** Permite utilizar y tratar una archivo XML pasando el contenido a un vector multidimensional que PHP puede utilizar.
- **Cache:** Sistema de gestión de caché para aumentar la velocidad de tratamiento de la información, guardando páginas en caché para evitar compilación del motor PHP.
- **DB (MySQL DataBase Connectors):** Conector para MySQL del sistema de base de datos. Incluye el constructor de consultas para KernelWeb.
- **DirectoryManager (Gestor de Directorios):** Plugin que permite gestionar los archivos, por ejemplo, esta clase es utilizada

para controlar los archivos de caché existentes según su fecha.

- **FileValidation (Validador de Archivos):** Validador de archivos que comprueba el tipo de archivo comprobando tanto extensiones permitidas como código mime del archivo.
- **Filter (Filtro de texto):** Filtra cadenas de texto para evitar que pueda sufrirse ataques con los formularios de la web.
- **FormGenerator (Generador de Formularios):** Generador de formularios. Permite generar y comprobar formularios con muy pocas líneas de texto.
- **FormDBGenerator (Generador de Formularios unidos a la base de datos):** Utilizando el generador de formularios, incluye el contenido en la base de datos utilizando conectores incluidos, como el Conector para MySQL. Esta extensión estará disponible en la próxima versión.
- **Parches para Internet Explorer:** Internet Explorer es un navegador difícil para utilizar algunas características de HTML5 y CSS3, por ello este plugin permite incluir metadatos para que la página web pueda verse correctamente en cualquier versión de Internet Explorer.
- **Sistema de Lenguajes:** Permite utilizar la página web con la tecnología i18n traduciendo en servidor el código que se entrega en HTML al usuario.
- **Mailer:** Extensión que ayuda al usuario a enviar emails, esta extiende a otra librería externa (en nuestro caso PHP Mailer). Para que esta funcione se configura en la aplicación el email con la contraseña. Actualmente solo permite enviar.
- **Lector de extensiones Mime (MimeReader):** Los archivos están identificados con un código en los primeros bytes del fichero. Este plugin permite leer esas extensiones para poder filtrar los ficheros. Al ser un script complejo y muy específico no ha sido desarrollado por nosotros, lo hemos descargado de la página del autor (es de código libre).
- **Paginador:** Esta extensión permite paginar cualquier cosa, ya sean líneas de texto, archivos, resultados de una base de datos, etc…
- **ServerStatus (Estado del Servidor):** Extensión que hace comprobaciones con el estado del servidor.
- **Stringer (Acción sobre cadenas):** Extensión que realiza acciones sobre cadenas, por ejemplo contiene una función llamada *getStringBetween* que recoge la cadena que se encuentra entre dos caracteres o cadenas indicadas.

Quality Assurance

Varios compañeros de clase han utilizado código de KernelWeb para sus proyectos. Nombrando por ejemplo a David García y Albert Ferrer. Han facilitado las pruebas del código reportando errores.
Para hacer pruebas, hemos intentado simular un intento de ataque colocando formularios y algunos mensajes.
El tiempo medio de carga de una página con algunos textos utilizando el MVC del Kernel es de 0,02 segundos y 300 kilobytes de memoria.

Nuestro primer proyecto de KernelWeb[7]

Creación del Proyecto

Una vez tengamos todo configurado, accedemos al gestor de nuestro KernelWeb desde el navegador[8]. Nos pedirá un usuario y contraseña, utilizamos como usuario *admin* y como contraseña *test*.
Cuando entremos tendremos la opción de **crear nuevo proyecto**, esto nos creará donde nosotros queramos del servidor un directorio con el esquema básico para crear nuestra aplicación ya pre-configurada para funcionar con KernelWeb. Además, nos creará por defecto un nuevo módulo llamado **site** (podemos renombrarlo antes de comenzar a hacer cambios). Podemos tener tantos módulos como queramos.
Si no especificamos nada se creará a la misma altura de directorios que el Kernel. Si ya existe un directorio con ese nombre o no hay permisos suficientes no se creará nada.
También se pueden crear esquemas personalizados para que se auto descompriman (como por ejemplo **Kernel CMS**).

Configurando el proyecto

Al margen de las configuraciones que se aplican de forma automática, podemos crear las nuestras propias y configurar la aplicación actual.
Así que nos movemos a **$dirapp/private/config**.
Podemos configurar todos y cada uno de los archivos con nuestras preferencias, si creamos algún archivo más tendremos que incluirlo de forma manual cuando lo queramos usar (o en el index.php).

Configurando la aplicación (app.conf.php):

Este es el archivo encargado de las configuraciones principales de la aplicación, muchas opciones cargan de forma automática, pero entre otras cosas podemos especificar la **plantilla** (template). Por defecto tenemos **offcanvas** (de **getbootstrap**), también podemos optar por **jumbotron** (también de **getbootstrap**).
También desactivaremos la caché (**static $cache = false;**).

Sobre las bases de datos (db.conf.php):

Todas las bases de datos que configuremos en ese archivo serán llamadas **únicamente cuando se necesiten**, si tienes 100 conexiones y solo usas 10, únicamente se iniciarán esas diez con tal de optimizar recursos.

Generando nuestro contenido

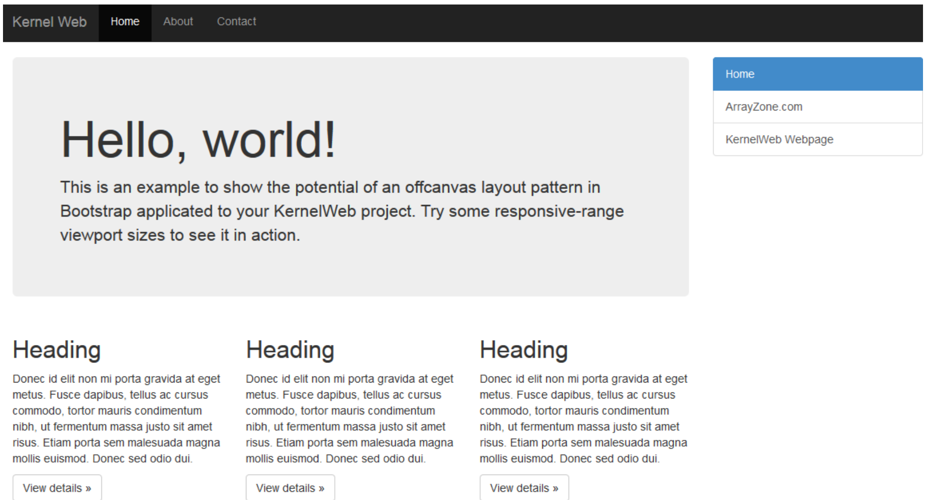
Actualmente, al ser una versión **Alpha** hay muchas cosas que cambiarán en futuras versiones y otras que aún deben hacerse manualmen-

[7] El código de /system aún no se encontraba operativo a día 3 de junio de 2014. Eso quiere decir que todo lo incorporado en el punto *Nuestro Primer Proyecto de KernelWeb* es totalmente teórico.
[8] Localmente suele ser http://localhost/system

te, por tanto, de momento solo nos centraremos en cambiar el contenido. La configuración de los menús actualmente se hace directamente sobre la **plantilla**, recordad que es un **framework de desarrollo**, no es una prioridad (para ello está **Kernel CMS**).

Hola Mundo

Si entramos en http://localhost/**nuestroproyecto** veremos algo parecido a:



El bloque central (desde el **Hello, world!** Hasta el último **heading**) es la página inicial, el resto de cosas se configuran directamente desde la plantilla o desde archivos específicos que veremos poco a poco.
Como ejemplo crearemos un controlador de contacto utilizando **Form-Generator**, una extensión nativa en KernelWeb.

Añadiendo el menú

Lo primero que haremos será ir a **$app-dir/private/modules/$module/menu.php**. En nuestro caso **$module** será **frontend**.
Ahí añadimos al final después de **Home** el siguiente código:

```
    'Contact'=>array(
        'cont'=>'contact',
        'act'=>'index'
    ),
```

```
$menu = array(
    'Home'=>'',
    'Contact'=>array(
        'cont'=>'contact',
        'act'=>'index'
    ),
    'ArrayZone.com'=>array(
        'target'=>'_blank',
        'href'=>'http://arrayzone.com'
    ),
    'KernelWeb Webpage'=>array(
        'target'=>'_blank',
        'href'=>'http://kernel.arrayzone.com'
    ),
```

Con esto le estamos diciendo que, al hacer clic en **Contact** cargue el **controlador** *contact* y su **acción** *index*. Como no existe actualmente mostrará un error, así que vamos a crearlos.
NOTA: si especificamos **module** podemos redirigirlo a otro módulo.

Añadiendo el controlador

Nos vamos a **$appdir/private/modules/$module/controllers/** y creamos el archivo **contact.php**, nótese que tiene el mismo nombre que **cont** del array. Dado que es algo extenso el código únicamente mostraremos la estructura básica (se puede ver el ejemplo real).

```
<?php
class contactController extends KWController {
	function generateForm() {
		// Generamosel formulario (véase el script original)
		$form = new FormGenerator();
		$form->action = rewriter('','','send');
		// ...
		$form->addSubmit(kw::t('Send', 'contact'), 'Send');

		return $form;
	}
	function indexAction() {
		// Cargamos el formulario:
		$form = $this->generateForm();
		// Hacemos que "form" sea accesible desde la vista:
		$this->form = $form;
	}

	function sendAction() {
		// Aquí capturamos la respuesta, volvemos a cargar el formulario
		$form = $this->generateForm();
		// Validamos el formulario
		if ($form->validate()) {
			// Esta validado…
		} else {
			$this->view = 'index';
			// Remostramos el formulario
			$this->form = $form;
		}
	}
}
?>
```

Vistas:

Ahora prepararemos lo que verá el usuario, para ello vamos a **$appdir/private/modules/$module/views/.**

Aquí creamos un directorio para el controlador con el mismo nombre (**contact**) y dentro un archivo **.php** para cada **vista** que estemos cargando, actualmente tendremos "**index**" y "**send**" (las acciones principales), podríamos tener más con otros nombres aunque no existieran las **acciones.**

Index.php:

```
<?php echo '<h1>' , kw::t('Welcome to the contact form', 'contact') ,
'</h1>';
// Mostramos el formulario asignado en $this-> desde el controlador
echo $this->form->showForm(); ?>
```

Send.php:

```
<?php echo kw::t('Your message has been sent', 'contact'); ?>
```

Traducciones:

Otra cosa importante son las traducciones (si tenemos la web en varios idiomas). Vamos a **$appdir/private/modules/$module/lang/.** Después entramos en el directorio del idioma (**es-ES**) y creamos un archivo con el nombre del **controlador** (**contact.php**), dentro ponemos:

```
<?php
return array(
	'Welcome to the contact form'=>'Bienvenido al formulario de contacto',
	'Send'=>'Enviar',
	'Your message has been sent' => 'Tu mensaje ha sido enviado');
?>
```

Finalmente el menú (opcional):

Esto lo podemos hacer de varias formas, no es necesario tener un menú para que un controlador funcione (podemos cargarlo dentro de alguna vista), pero como es un menú que queremos siempre activo iremos a **$appdir/private/modules/$module/data/. CUIDADO:** La plantilla tiene que incluir el menú, si no será inútil (las plantillas por defecto ya incluyen esto con caché).
Dentro de **menu.php** (es el más genérico) añadimos dentro el array:

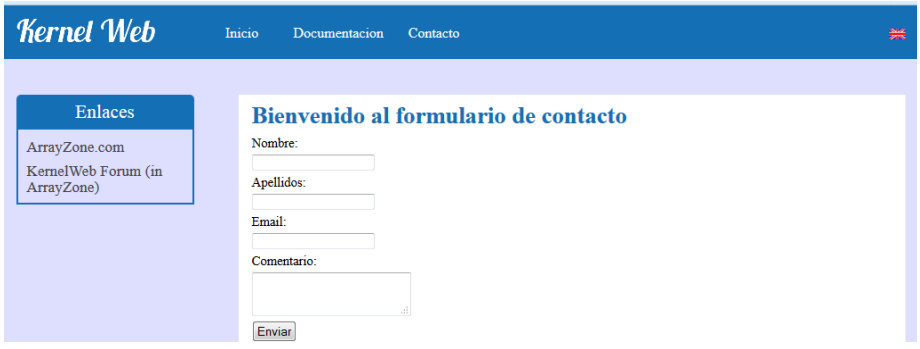
```
	kw::t('Contact', 'generic')=>array(
		'href'=>array(
			'cont'=>'contact',
			'act'=>'index'
		)
	),
```

Para terminar:

Una vez tengamos la aplicación funcionando debemos reactivar la caché para optimizar recursos, para ello vamos a **$appdir/private/config/app.conf.php** y ponemos **$cache** en **true.**

Cuando lo subamos al servidor deberemos dar permisos de escriutra como mínimo a: **$appdir/cache, $appdir/public/img.**

Si cambiamos la plantilla por la **kernelweb** obtendremos una vista como la siguiente:



Optimización

Llamar a una función un millón de veces cuando
3.30s – Función no estática dentro de una clase.
0,98s – Creando un objeto y llamándolo a partir de una clase.
0,95s – Función estática dentro de una clase
0,91s – Función sola

Llamar a una función dos millones de veces cuando
4.80s – Función no estática dentro de una clase.
1,95s – Función dentro de una clase que está extendida a una clase abstracta.
1,85s – Creando un objeto y llamándolo a partir de una clase.
1,60s – Función estática dentro de una clase
1,60s – Función sola

Tiempo de escritura en memoria dos millones de veces al utilizar:
3,60s – Define
0,60s – Variable estática en una clase
0,40s – Variable estándar

Tiempo de lectura en memoria dos millones de veces al utilizar:
0,46s – Variable definida por el sistema (__FILE__)
0,46s – Define
0,34s – Variable estática en una clase
0,32s – Variable estándar

El tiempo…
…en incluir un fichero *utilizando* include() o require() es el mismo.
…que se tarda en utilizar rutas relativas es algo superior que las absolutas.

Llamar a una función…
…con variables definidas por defecto es igual de rápido que si son obligatorias.

La transferencia de variables entre controlador y vista hay dos opciones, utilizando $this->, siendo más sencillo para el usuario de entender el código, aunque utilizar extract() hace algo más rápida la programación, ya que no se debe escribir $this-> en todas las variables que se quiera utilizar.

Bibliografía y Enlaces

ArrayZone
Página Principal de ArrayZone
http://arrayzone.com
Foro de ArrayZone
http://foro.arrayzone.com
Panel de Gestión de Usuarios
https://panel.arrayzone.com
Panel de Cobro
http://billing.arrayzone.com
Página principal de KernelWeb
http://kernel.arrayzone.com

i-MSCP Panel de Gestión de Hosting
https://github.com/i-MSCP/imscp/archive/1.1.5.tar.gz
BoxBilling Panel de Cobro
http://www.boxbilling.com/
ionCube Loaders
http://www.ioncube.com/loaders.php
Subversión:
http://es.wikipedia.org/wiki/Subversion
Como instalar SVN en Debian
http://www.howtoforge.com/debian_subversion_websvn
NoMachine:
Página Principal
https://www.nomachine.com
Enlace de Descarga:
http://goo.gl/ekP13G
IDE Eclipse Kepler
http://www.eclipse.org
Sociedad Limitada Nueva Empresa:
http://goo.gl/8gciS8
Cambio de Denominación Social en la Sociedad Limitada
http://goo.gl/FZKmZa
PcComponentes:
Equipos Informáticos escogidos al iniciar la empresa
http://goo.gl/1y2EhN
Prestamos ICO:
Página Principal de Información sobre los prestamos ICO
http://goo.gl/FAFPcD
XAMPP
https://www.apachefriends.org/es/index.html