Proyecto Vihostdom

Autor: Paulo Maté de Nicolás

Indice

Introducción al Proyecto

El objetivo de este proyecto es poner en marcha una aplicación web capaz de gestionar la interacción de los clientes con los productos de una empresa de registro de dominios, alojamiento y desarrollo web (Emulando en la medidad de lo posible a 1&1, godaddy o namecheap). Además de ser capaz de instalar y configurar toda la parte de red y de servidor correspondiente.

La empresa del proyecto se llamará Vihostdom (**Vi**rtual **host**ing and **dom**ain).

En un principio la aplicación web se realizó escribiendo código php desde cero sin ningún tipo de herramienta.

Más tarde se descubrió la tremenda mejora en la calidad y el tiempo de desarrollo que un framework podía ofrecer y se portó el proyecto utilizando Codeigniter (2.1.4), un framework php open source.

La aplicación web desarrollada servirá para contratar, modificar o revocar los servicios de la empresa.

Para la interacción del cliente con su servicio contratado se recurre al panel de control Froxlor, un panel open source que se encargará de automatizar la interacción entre cliente y servidor y que llevará a cabo cualquier cambio producido en la configuración del servidor utilizando el cron de linux.

Tanto la aplicación web como el panel y alguno datos de muestra estarán online en las siguientes direcciones:

http://vihostdom.servehttp.com/
http://asir09.servehttp.com/
http://informatico.servehttp.com/

Contratadas sin ningún coste a través del servicio de dns dinámico no-ip.

Estas tres direcciones apuntarán al servidor que se encuentra en mi domicilio. Dicho servidor es una Raspberry Pi modelo B totalmente configurada y con las medidas de seguridad oportunas.

Creación de la aplicación web

Para la realización de la aplicación web se han utilizado los siguientes recursos:

Codeigniter (2.1.4)

Jquery (1.11.1)

Twitter Bootstrap (3.1.1)

Normalize CSS (3.0.1)

Fatly Theme (Bootstrap Theme)

Jquery FullPage Plugin

Jquery MixItUp Plugin

Jquery Qtip Plugin

Fotografías encontradas en bancos de imágenes gratuitas.

Codeigniter es un framework php open source basado en el patrón modelo-vista-controlador y propiedad de Ellislab. La elección de este framework se debe a su baja curva de aprendizaje, la comunidad y documentación detrás del proyecto, su gran velocidad y su ligereza.

jQuery es una biblioteca de JavaScript que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web.

Twitter Bootstrap es un framework open source para diseñar sitios y aplicaciones web. Contiene plantillas de diseño con tipografía, formularios, botones, cuadros, menús de navegación y otros elementos de diseño basado en HTML y CSS, así como, extensiones de JavaScript opcionales adicionales.

Normalize es un reseteador de código CSS multiplataforma, moderno y compatible con HTML5.

Flatly Theme es un tema de Bootstrap que simplemente se compone de una hoja css la cual cambia el conjunto de colores y animaciones CSS de la aplicación web.

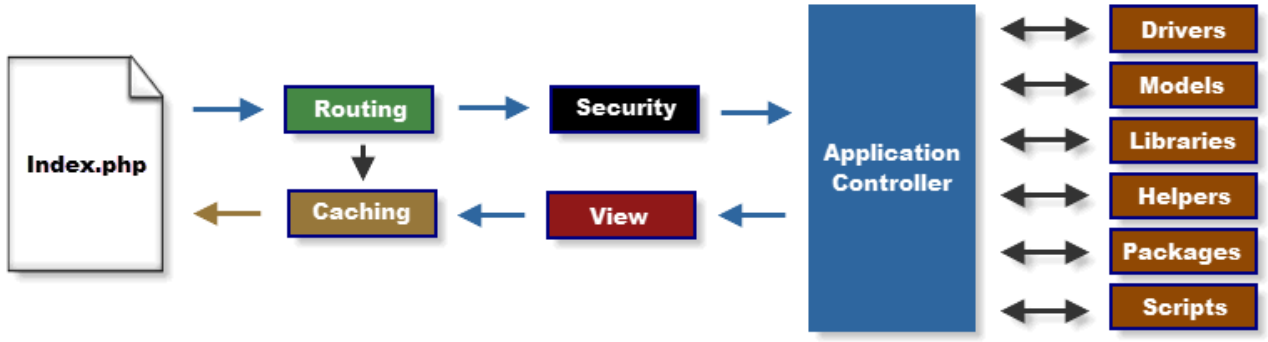
FullPage Plugin se encarga de las animaciones de la portada haciendo posible navegar por ella de una manera distinta.

MixItUp Plugin se encarga de ordenar o filtrar elementos HTML usando animaciones y transiciones.

Qtip Plugin se encarga de proporcionar pequeños mensajes de ayuda en ciertos formularios. Usado en el formulario de registro.

La aplicación web está basada en el patrón modelo-vista-controlador, esto se consigue gracias a la facilidad que da Codeigniter para implementar este patrón. Este patrón es ampliamente usado y se basa en separar las vistas (la parte visible por el usuario, es decir, la interfaz web), los controladores (la parte lógica de la aplicación y la que se encarga de realizar las operaciones necesarias) y los modelos (que se encargan del tratamiento de datos y de las conexiones con la base de datos).

Para entenderlo mejor el diagrama de flujo de Codeigniter es el siguiente:



Una URL de Codeigniter de ejemplo sería la siguiente:

http://ejemplo.com/index.php/controlador/funcion/

La parte de index.php se ha eliminado de la aplicación usando un módulo de apache como se explicará más adelante. También deberemos especificar en el archivo \config\config.php la siguiente línea:

```
$config['index_page'] = '';
```

El archivo \config\router.php se ha configurado de la siguiente forma:

```
$route['datos'] = 'usuarios/datos';
```

Para eliminar también el nombre del controlador usado, de esta forma quedan unas urls más limpias.

En el archivo *\config\autoload.php* cargaremos las siguientes librerías y helpers:

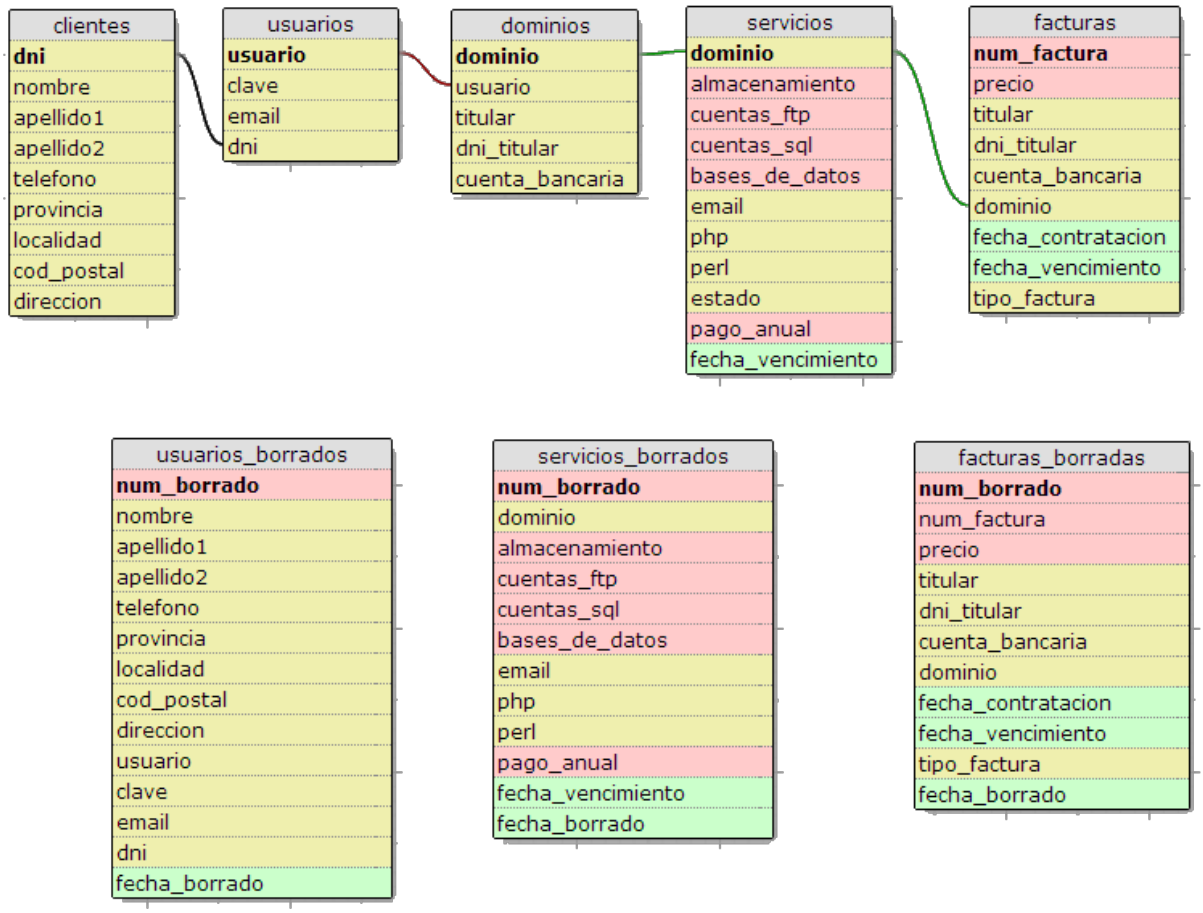
```
$autoload['libraries'] = array('form_validation', 'database',
'session');

$autoload['helper'] = array('form', 'url');
```

Form_validation nos servirá para validar formularios con ciertas reglas predefinidas o algunas que nosotros mismos añadamos, Database para realizar la conexión con la base de datos y realizar las operaciones necesarias y Session para utilizar sesiones.

Form lo utilizaremos para cargar formularios de una forma más fácil y que se envíen al controlador correspondiente, Url lo usaremos para poder trabajar con funciones de tratamiento de urls.

El diseño de la base de datos que utilizaremos es el siguiente:



La tabla Clientes se encargará de almacenar la información de los clientes que se registren, algunos campos son opcionales.

La tabla Usuarios contendrá la información necesaria para iniciar sesión en el sitio web, asociarle a un cliente y contactarle a través del email.

La tabla Dominios contendrá los dominios registrados, cada dominio asociado a un usuario y la información de quien contrató el dominio y del pago realizado.

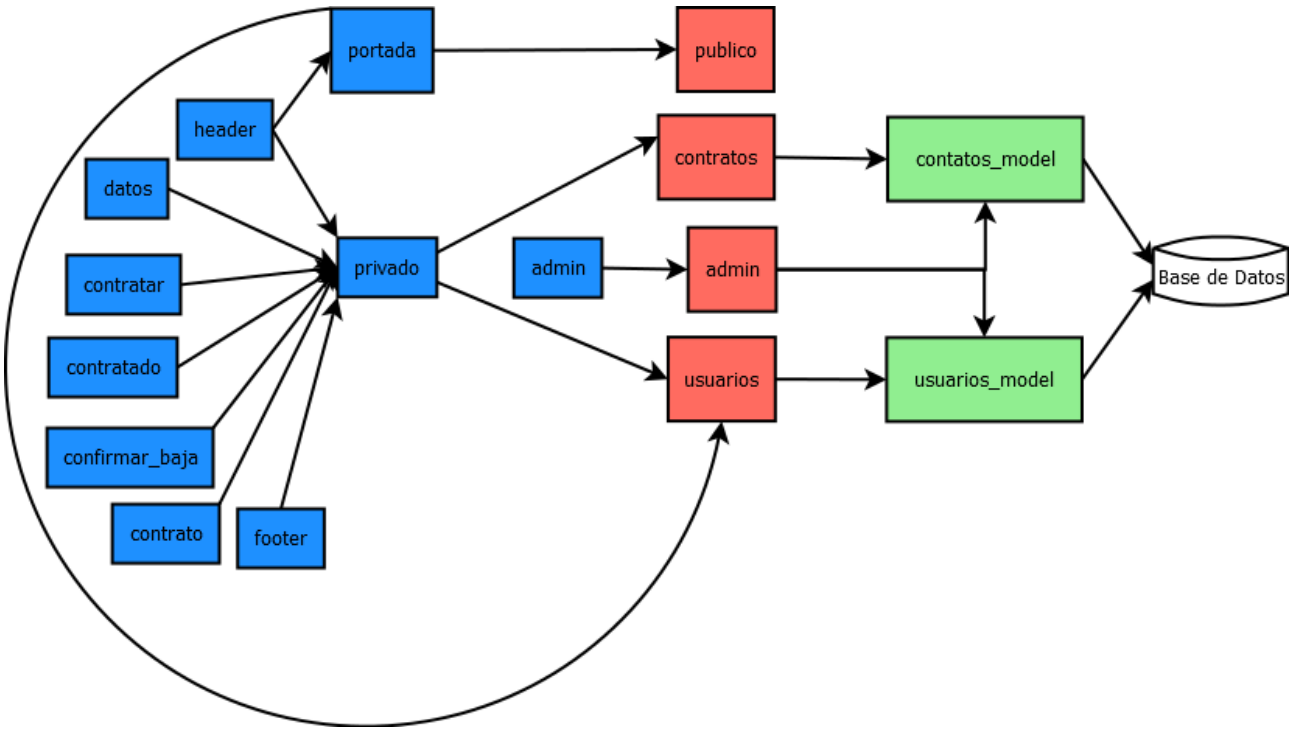
La tabla Servicios contendrá la información de los servicios asociados a cada dominio junto con el pago anual y la fecha de vencimiento.

La tabla Facturas contendrá las facturas asociadas a los dominios, cada factura estará numerada.

Las tablas Usuarios_Borrados, Servicios_borrados y Facturas_Borradas contendrán la información de las anteriores tablas en el momento en el que fueron borradas, ya sea un dominio o una cuenta entera.

El script de creación de esta base de datos se detalla más adelante.

Este es el diagrama de la aplicación web:



La aplicación se compone de dos modelos, cuatro controladores y diez vistas.

Las Vistas:

Hay dos vistas principales que serán las que cargaremos realmente, Portada y Privado.

Portada es la vista que cargaremos para mostrar nuestra web y sus contenidos a todos los usuarios que accedan. En ella se cargará el contenido de la vista Header el cual contiene el menú principal y los formularios de inicio de sesión y registro, despúes cargará el contenido que la propia vista portada contiene.

Privado será la vista que se cargará una vez el usuario haya iniciado sesión y se introduzca en la zona privada de la web (datos, contratar, contratado, etc), cargaremos el contenido de la vista Header que al detectar la variable de sesión del usuario mostrará su panel de usuario, después cargará el contenido de la vista que se le pida dependiendo de la función a la que haya accedido (datos, contratar, contratado, etc) y por último se cargará el contenido de la vista Footer.

La vista datos contiene los formularios necesarios para el cambio de información de la cuenta de usuario o el borrado de la misma.

La vista contratar contiene el formulario de contratación de un servicio.

La vista contratado, la cual es la que se cargará por defecto al iniciar sesión, contiene un botón para contratar un servicio, y, en el caso de que ya exista algún servicio contratado, también muestra un botón de borrado de servicio y una lista de los servicios contratados con un menú que permite ordenarlos o filtrarlos a gusto del usuario y acceder a su modificación si así se desea. En el caso de que el usuario pinche en el botón de contratar servicio será redirigido a la vista contratar, y si pincha en el botón de borrado será redirigido a la vista Confirmar_Baja para confirmar el cese del servicio en cuestión.

La vista contrato carga un formulario cuya función es modificar las condiciones del contrato añadiendo o quitando características al servicio contratado.

La vista Footer contiene un menú parecido al que se encuentra en el panel del usuario por simple comodidad, el aviso de Copyright y las redes sociales de la empresa.

La vista admin, cuando se carga (haya o no un usuario registrado activo), muestra un formulario de acceso, una vez iniciada la sesión muestra una interfaz de administración que permite activar o eliminar los contratos de los clientes y contactar con ellos por email.

Los Controladores:

El controlador Público es el que se encarga de cargar la vista portada a través de su funcion por defecto index(). Contiene además dos funciones privadas (_validation() y _repopulate()) que se encargan de mostrar los errores de los formularios de inicio de sesión o registro y de rellenar los campos con los valores enviados para ahorrar al usuario la tarea de volver a escribirlos. Este controlador además carga dos variables a la vista: $title que contiene el título que se mostrará en la vista Portada y $header que cargará el contenido de la vista Header.

El controlador Usuarios contiene las siguientes funciones:

index(), es la función por defecto del controlador, significa que si se accede sin especificar el nombre del controlador en la url se cargará esta función, como usamos el archivo routes.php para no mostrar el nombre del controlador esta función devuelve un error 404 que carga el contenido de una página de codeigniter la cual se puede modificar para mostrar un mensaje propio. Se ha hecho de esta forma por si, aunque sea improbable, alguien accede directamente usando la dirección: http://ejemplo.com/usuarios/

registro(), es la función que se encarga de validar los campos enviados por el formulario de registro, en el caso de acceder a esta función sin haber enviado variables POST muesta un error 404, tanto si la validación de los datos es válida como errónea, esta función redirige a la vista Portada enviando los mensajes oportunos de creación de cuenta correcta o errónea.

login(), al igual que la anterior comprueba que ha recibido POST, valida los datos y redirige a Contratado si el inicio de sesión es correcto o a Portada con el mensaje oportuno en caso de error.

logout(), destruye las sesiones actuales y redirige a Portada.

datos(), comprueba que el usuario ha iniciado sesión, si es así carga la vista Datos, en caso contrario, carga el error 404, el formulario de modificación de datos se envía a esta misma función por lo que comprueba si se ha recibido POST y si es así valida los datos y muestra los mensajes oportunos. El formulario de modificación carga los valores del usuario actual en sus campos, sin embargo sólo a algunos de estos campos se les está permitida la modificación. También permite el cambio de contraseña.

borrar(), comprueba que el usuario ha iniciado sesión y se encarga de borrar la cuenta actual del usuario junto con toda su información, sus contratos y sus facturas.

El controlador Contratos contiene las siguientes funciones:

index(), se comporta de igual forma que la del controlador Usuarios.

contratado(), comprueba que el usuario ha iniciado sesión, esta función carga la vista Contratado que muestra los servicios contratados por el usuario actual. También se encarga de recibir POST del formulario de la vista Contratar y validar los datos, en el caso de una validación correcta añade el nuevo contrato a la base de datos y lo muestra al cargar la vista, en caso contrario redirige a la vista Contratar y muestra los mensajes oportunos.

contratar(), comprueba que el usuario ha iniciado sesión, esta función carga la vista Contratar que carga un formulario con los campos necesarios para contratar un nuevo servicio, lo envía a contratado() que se encarga de la validación, en caso de error se vuelve a cargar esta página con los mensajes de error oportunos.

contrato(), comprueba que el usuario ha iniciado sesión, también comprueba que el dominio pasado por url en forma de variable GET es del usuario actual, si no es así redirige a la vista Contratado. Esta función carga la vista Contrato que contiene un formulario con las características actuales del dominio especificado, este formulario da la opción de modificar el contrato actual y calcula el precio de la modificación. También sirve para modificar la información del pago anual del dominio.

confirmar_baja(), comprueba que el usuario ha iniciado sesión y que el dominio especificado es del usuario actual, carga la vista Confirmar_Baja que simplemente se compone de un panel en el que hay un mensaje de precaución y un botón para confirmar la baja del dominio actual. Una vez pulsado el botón se envía a esa misma página variables POST con el dominio y la confirmación de la baja, se comprueba que el dominio pasado por POST y por GET siguen coincidiendo y se procede a dar de baja el dominio junto con sus facturas.

dominio(), comprueba que esta función recibe una llamda ajax y que recibe dos valores POST (dominio y terminación), si no es así muestra el error 404. Valida los valores POST recibidos con una expresión regular haciendo que cumpla las reglas de un dominio y devuelve el mensaje oportuno. Esta función es llamada desde la vista Portada sección Dominios usando ajax para comprobar que un dominio está disponible con la función de PHP dns_check_record(), de esta forma se puede comprobar sin tener que recargar la página web para cada comprobación de dominio.

El controlador Admincontiene las siguientes funciones:

index(), se comporta de igual forma que la del controlador Contratos.

paulo(), esta función no tiene reroute por lo que sólo podrá ser accedida a través de la dirección http://ejemplo.com/admin/paulo lo que añade una medida de seguridad. Carga la vista Admin, la cual primero carga primero un formulario de acceso, si el acceso es correcto se vuelve a cargar mostrando esta vez la interfaz de administración. La variable de sesión es diferente a la de un usuario normal y en este caso sólo el usuario paulo tiene acceso a la interfaz de administración, en esta interfaz se encuentran dos botones que activan o eliminan dominios.

activar(), comprueba que un usuario admin ha iniciado sesión, recibe un dominio por GET, lo activa y redirige a la vista Admin.

borrar(), comprueba que un usuario admin ha iniciado sesión, recibe un dominio por GET, lo eliminay redirige a la vista Admin.
Los Modelos:

El modelo Usuarios_Model:

registro(), obtiene los valores necesarios por POST recibidos en la funcion registro() del controlador usuarios y los inserta en la base de datos.

login(), se le pasa un array con los datos de conexion y devuelve un resultado.

datos(), hace un select a la base de datos que obtiene los datos del usuario actual y los devuelve para rellenar el formulario de la vista Datos.

update_datos($data), se le pasa un array con los datos del usuario actual modificados y los actualiza en la base de datos.

change_pass($data), se le pasa un array con la nueva contraseña y la actualiza en la base de datos. La comprobación de la contraseña actual para realizar el cambio se hace llamando a la función login() .

check_usuario($usuario), se le pasa un array con el usuario y devuelve un valor booleano en caso de que exista o no en la base de datos, se utiliza para registrar nuevos usuarios.

check_dni(), se le pasa un array con el dni y devuelve un valor booleano en caso de que exista o no en la base de datos, se utiliza para registrar nuevos clientes asociados a un usuario.

borrar(), borra la cuenta completa de un usuario pasando toda la información a las tablas en las que se almacenan las cuentas, servicios y facturas borradas.

El modelo Contratos_Model:

cotratos(), devuelve los contratos del usuario actual.

Contratar($dominios, $servicios, $facturas), se le pasan tres arrays que contienen todos los datos necesarios al contratar un servicio y los inserta en la base de datos.

check_domain($dominio), recibe un dominio y devuelve un booleano dependiendo de si ese dominio existe o no en la base de datos.

check_owner(), devuelve un booleano dependiendo de si el dominio pasado por GET pertenece al usuario actual o no.

servicios(), devuelve un array que contiene la información del dominio pasado por GET.

update_servicios($dominios, $servicios, $facturas), recibe tres arrays conteniendo la información de las modificaciones hechas al dominio pasado por GET y actualiza la base de datos.

fecha_vencimiento(), obtiene el dominio pasado por GET y devuelve su fecha de vencimiento.

domiciliacion(), obtiene el dominio pasado por GET y devuelve los datos de la domiciliación del pago.

facturas(), obtiene el dominio pasado por GET y devuelve un array de arrays que contienen la información de cada factura.

borrar_dominio(), obtiene el dominio pasado por GET y borra el dominio, los servicios asociados y sus facturas pasando esa información a las tablas de servicios y facturas borradas.

conseguir_dominios(), devuelve un array de arrays conteniendo la información necesaria de cada dominio para administrarlos desde la interfaz de administración.

activar_dominio(), obtiene el dominio pasado por GET y lo activa cambiando el valor del campo “estado” en la base de datos.

actualizar_dominios(), selecciona todos los dominios con la fecha de vencimiento ya pasada a la actual y actualiza su estado a “renovando”.

Configuración del servidor

Para la completa puesta en marcha del servidor hemos utilizado los siguientes recursos:

-Raspberry pi Modelo B

-Sistema Operativo Raspbian

-Apache

-MySQL

-PHP

-PhpMyAdmin

-Fail2ban

-Dos2unix

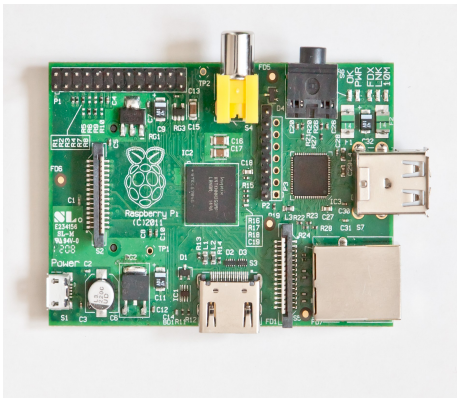
-Froxlor

-Bind9

-ProFTPd

-Postfix

-Courier



Instalamos el sistema operativo Raspbian utilizando el programa win32diskimager, seleccionamos el archivo .img que contiene el sistema operativo, elegimos la tarjeta SD de destino y seleccionamos “write”. Una vez termine introducimos la tarjeta en la raspberry pi y la conectamos a la corriente.

El sistema iniciará automáticamente y tendrá ssh activado por defecto, podemos acceder utilizando el usuario pi y la contraseña raspbery. Lo primero que hacemos es ejecutar el script raspi-config que viene por defecto en el sistema y ejecutamos las siguientes opciones:

-Expandimos el sistema de archivos
-Cambiamos la contraseña para el usuario pi
-Deshabilitamos el escritorio
-Cambiamos la zona horaria a Madrid
-Colocamos el hostname que deseemos a nuestro sistema (“raspiserver” en este caso)
-Cambiamos el tamaño de la memoria asignada a la GPU al mínimo (16 MB)
-Actualizamos la raspberry pi

Cerramos el script raspi-config y empezamos con la configuración de red, editamos el archivo */etc/network/interfaces* y colocamos la siguiente configuración:

```
iface eth0 inet static
     address 192.168.1.100
     netmask 255.255.255.0
     gateway 192.168.1.1
```

Denegamos el acceso del usuario root por ssh editando el archivo */etc/ssh/sshd_config* modificando la línea "PermitRootLogin no".

Por último creamos un nuevo usuario, le añadimos a los grupos en los que se encuentra pi, nos cambiamos al nuevo usuario y deshabilitamos al usuario pi con la orden:

```
usermod -L -e 1 pi
```

Instalamos los servicios de la siguiente forma:

```
apt-get install apache2 php5 mysql-client mysql-server libapache2-
mod-php5 php5-mysql fail2ban dos2unix
```

Con esto tendremos instalado:

Apache: es un servidor web HTTP de código abierto.

MySQL: es un sistema de gestión de bases de datos relacional, multihilo y multiusuario desarrollado como software libre.

PHP: es un lenguaje de programación de uso general de código del lado del servidor originalmente diseñado para el desarrollo web de contenido dinámico.

Fail2ban: es una aplicación escrita en Python para la prevención de intrusos en un sistema, que actúa penalizando o bloqueando las conexiones remotas que intentan accesos por fuerza bruta.

Dos2unix: es un convertidor de texto en formato DOS de windows a formato unix/linux.

Configuraremos apache de la siguiente forma:

Editamos el archivo */etc/apache2/apache2.conf* y añadimos la línea "ServerName raspiserver"

Habilitamos el módulo "rewrite" ejecutando:

```
a2enmod rewrite
```

Reiniciamos apache con el comando:

```
service apache2 restart
```

Para configurar Fail2ban copiamos el archivo jail.conf a jail.local que será el que realmente se use para la configuración actual:

```
cp /etc/fail2ban/jail.conf /etc/fail2ban/jail.local
```

Y editamos las siguientes líneas:

```
ignoreip = 127.0.0.1/8
bantime  = -1
maxretry = 3
```

Ignore ip sirve para excluir direcciones de red de un baneo accidental.
Bantime es el tiempo en segundos durante el cual una ip está baneada, para valores negativos no se quitará el baneo nunca.
Maxretry es el número de intentos fallidos antes de que se banee una dirección.

Dos2unix, MySQL y PHP no necesitan ninguna configuración extra.

Una vez hecho esto instalaremos el panel froxlor. Se trata de un panel de administración del servidor de código libre. Su función es automatizar la configuración del servidor a la hora de la contratación o modificación de los servicios por parte de los clientes, para ello actualiza la configuración del servidor y cualquier cambio llevado a cabo cada cierto tiempo haciendo uso de la herramienta cron de linux. Es capaz de administrar usuarios y por cada usuario dominios, bases de datos, acceso ftp, límite de almacenamiento online y servicios email, php y perl. Además es capaz de limitar el tráfico y lleva un registro de éste el cual es accesible para cada usuario.

Para instalar el panel primero añadimos el repositorio y la clave ejecutando lo siguiente:

```
echo 'deb http://debian.froxlor.org wheezy main' >
/etc/apt/sources.list.d/froxlor.list
apt-key adv --keyserver pool.sks-keyservers.net –recv-key
FD88018B6F2D5390D051343FF6B4A8704F9E9BBC
```

Actualizamos el sistema

```
apt-get update
apt-get upgrade
```

Instalamos los paquetes necesarios:

```
apt-get install froxlor php5-curl
```

Una vez hecho esto el propio paquete froxlor instalará las dependencias necesarias, entre las cuales se inciuyen el servidor dns Bind9, el servidor ftp proFTPd, y los servicios de email courier y postfix entre otros.

Nos pedirá algunos datos como el dominio de mail en el que se encuentra nuestro servidor (vihostdom.servehttp.com) o si deseamos ejecutar algunos servicios de forma independiente, dejaremos las opciones por defecto.

Una vez tenemos froxlor instalado es hora de configurarlo, para hacerlo accedemos al servidor apache con la ip privada y añadiendo /froxlor, nos aparecerá una bienvenida, comprobamos que satisfacemos todas las dependencias, rellenamos los datos necesarios como el nombre de la base de datos que usará, la cuenta de administrador, etc. Iniciamos sesión y comprobamos que todo funciona correctamente, ahora subimos los archivos de configuración que nos proporciona el panel al servidor a través de sFTP con cualquier cliente disponible (filezilla en este caso) y ejecutaremos el siguiente script bash que yo mismo he hecho para colocar los archivos de configuración en su sitio:

```
# Script de configuracion automatica de froxlor

# Ver comandos ejecutados
set -x

# Webserver (HTTP) Apache

mkdir -p /var/customers/webs/
mkdir -p /var/customers/logs/
mkdir -p /var/customers/tmp
chmod 1777 /var/customers/tmp
a2dismod userdir

service apache2 restart

read -p "Servidor WEB Configurado, continuar..." any

# Nameserver (DNS) Bind9

apt-get install bind9
echo "include \"/etc/bind/froxlor_bind.conf\";" >>
/etc/bind/named.conf.local
touch /etc/bind/froxlor_bind.conf
chown root:bind /etc/bind/froxlor_bind.conf
chmod 0644 /etc/bind/froxlor_bind.conf

/etc/init.d/bind9 restart

read -p "Servidor DNS Configurado, continuar..." any

# Mailserver (SMTP) Postfix/Courier

groupadd -g 2000 vmail
useradd -u 2000 -g vmail vmail
mkdir -p /var/customers/mail/
chown -R vmail:vmail /var/customers/mail/
apt-get install postfix postfix-mysql libsasl2-2 libsasl2-modules
libsasl2-modules-sql
mkdir -p /var/spool/postfix/etc/pam.d
mkdir -p /var/spool/postfix/var/run/mysqld

cp ./main.cf /etc/postfix/main.cf
cp ./mysql-virtual_alias_maps.cf /etc/postfix/mysql-
```

```
virtual_alias_maps.cf
cp ./mysql-virtual_mailbox_domains.cf /etc/postfix/mysql-
virtual_mailbox_domains.cf
cp ./mysql-virtual_mailbox_maps.cf /etc/postfix/mysql-
virtual_mailbox_maps.cf
cp ./mysql-virtual_sender_permissions.cf /etc/postfix/mysql-
virtual_sender_permissions.cf
cp ./smtpd.conf /etc/postfix/sasl/smtpd.conf
cp ./aliases /etc/aliases

chown root:root /etc/postfix/main.cf
chown root:postfix /etc/postfix/mysql-virtual_alias_maps.cf
chown root:postfix /etc/postfix/mysql-virtual_mailbox_domains.cf
chown root:postfix /etc/postfix/mysql-virtual_mailbox_maps.cf
chown root:postfix /etc/postfix/mysql-
virtual_sender_permissions.cf
chown root:root /etc/postfix/sasl/smtpd.conf
chmod 0644 /etc/postfix/main.cf
chmod 0640 /etc/postfix/mysql-virtual_alias_maps.cf
chmod 0640 /etc/postfix/mysql-virtual_mailbox_domains.cf
chmod 0640 /etc/postfix/mysql-virtual_mailbox_maps.cf
chmod 0640 /etc/postfix/mysql-virtual_sender_permissions.cf
chmod 0600 /etc/postfix/sasl/smtpd.conf

newaliases
/etc/init.d/postfix restart

read -p "Servidor SMTP Configurado, continuar..." any

# Mailserver (IMAP/POP3) Courier

apt-get install courier-pop courier-imap courier-authlib-mysql

cp ./authdaemonrc /etc/courier/authdaemonrc
cp ./authmysqlrc /etc/courier/authmysqlrc

/etc/init.d/courier-authdaemon restart
/etc/init.d/courier-pop restart

read -p "Servidor IMAP/POP3 Configurado, continuar..." any

# FTPserver (FTP) ProFTPd

apt-get install proftpd-basic proftpd-mod-mysql

cp ./sql.conf /etc/proftpd/sql.conf
cp ./modules.conf /etc/proftpd/modules.conf
cp ./proftpd.conf /etc/proftpd/proftpd.conf

/etc/init.d/proftpd restart

read -p "Servidor FTP Configurado, continuar..." any
```

```
# Otras utiidades:

# Crond

cp ./froxlor /etc/cron.d/froxlor

/etc/init.d/cron restart

read -p "Cron Configurado, continuar..." any

# Awstats

apt-get install awstats
cp /usr/share/awstats/tools/awstats_buildstaticpages.pl /usr/bin/
mv /etc/awstats/awstats.conf /etc/awstats/awstats.model.conf
sed -i.bak 's/^DirData/# DirData/' /etc/awstats/awstats.model.conf
# Please make sure you deactivate awstats own cronjob as Froxlor
handles that itself
rm /etc/cron.d/awstats

read -p "Awstats Configurado, continuar..." any

# Libnss

apt-get install libnss-mysql-bg nscd

cp ./libnss-mysql.cfg /etc/libnss-mysql.cfg
cp ./libnss-mysql-root.cfg /etc/libnss-mysql-root.cfg
cp ./nsswitch.conf /etc/nsswitch.conf

chmod 600 /etc/libnss-mysql.cfg /etc/libnss-mysql-root.cfg

/etc/init.d/nscd restart

read -p "Libnss Configurado, continuar..." any

# Logrotate

apt-get install logrotate

cp ./froxlor2 /etc/logrotate.d/froxlor

chmod 644 /etc/logrotate.d/froxlor

# apt automatically adds a daily cronjob for logrotate
# you do not have to do anything else :)

read -p "Logrotate Configurado, continuar..." any


# Configurar PHP usando FCGID o FPM

# Configuracion FCGID
```

```
apt-get install apache2-suexec libapache2-mod-fcgid php5-cgi
a2enmod suexec fcgid

service apache2 restart

read -p "FCGID Configurado, continuar..." any
```

Una vez hecho esto el panel froxlor estará completamente funcional.

En este momento crearemos un directorio para nuestra web dentro de */var/www* que es donde apache sirve nuestras webs:

```
mkdir /var/www/vihostdom
```

A este directorio subiremos todos los archivos de nuestra página web completamente finalizada a través de sFTP.

Ahora utilizaremos la interfaz web phpMyAdmin para agilizar las operaciones realizadas a la base de datos. Lo primero será crearla con el siguiente código SQL:

```
create database vihostdom;

use vihostdom;

create table clientes (
nombre varchar(50) not null,
apellido1 varchar(50) not null,
apellido2 varchar(50) not null,
dni varchar(9) not null,
telefono varchar(9) not null,
provincia varchar(50),
localidad varchar(50),
cod_postal varchar(5),
direccion varchar(50),
primary key (dni)
) charset=utf8 ENGINE=InnoDB;
create table usuarios (
usuario varchar(20) not null,
clave char(32) not null,
email varchar(50) not null,
dni varchar(9) not null,
primary key (usuario)
) charset=utf8 ENGINE=InnoDB;
```

```
create table dominios (
usuario varchar(20) not null,
dominio varchar(253) not null,
titular varchar(50) not null,
dni_titular varchar(9) not null,
cuenta_bancaria varchar(20) not null,
primary key (dominio)
) charset=utf8 ENGINE=InnoDB;

create table servicios (
dominio varchar(253) not null,
almacenamiento int not null,
cuentas_ftp int not null,
cuentas_sql int not null,
bases_de_datos int not null,
email char(2) not null,
php char(2) not null,
perl char(2) not null,
estado varchar(15) not null,
pago_anual float not null,
fecha_vencimiento date not null,
primary key (dominio)
) charset=utf8 ENGINE=InnoDB;

create table facturas (
num_factura int not null auto_increment,
precio float not null,
titular varchar(50) not null,
dni_titular varchar(9) not null,
cuenta_bancaria varchar(20) not null,
dominio varchar(253) not null,
fecha_contratacion date not null,
fecha_vencimiento date not null,
tipo_factura varchar(15) not null,
primary key (num_factura)
) charset=utf8 ENGINE=InnoDB;

create table servicios_borrados (
num_borrado int auto_increment,
dominio varchar(253) not null,
almacenamiento int not null,
cuentas_ftp int not null,
cuentas_sql int not null,
bases_de_datos int not null,
email char(2) not null,
php char(2) not null,
perl char(2) not null,
pago_anual float not null,
fecha_vencimiento date not null,
fecha_borrado date not null,
primary key (num_borrado)
) charset=utf8 ENGINE=InnoDB;
```

```
create table usuarios_borrados (
num_borrado int auto_increment,
nombre varchar(50) not null,
apellido1 varchar(50) not null,
apellido2 varchar(50) not null,
dni varchar(9) not null,
telefono varchar(9) not null,
provincia varchar(50),
localidad varchar(50),
cod_postal varchar(5),
direccion varchar(50),
usuario varchar(20) not null,
clave char(32) not null,
email varchar(50) not null,
fecha_borrado date not null,
primary key (num_borrado)
) charset=utf8 ENGINE=InnoDB;

create table facturas_borradas (
num_borrado int auto_increment,
num_factura int not null,
precio float not null,
titular varchar(50) not null,
dni_titular varchar(9) not null,
cuenta_bancaria varchar(20) not null,
dominio varchar(253) not null,
fecha_contratacion date not null,
fecha_vencimiento date not null,
tipo_factura varchar(15) not null,
fecha_borrado date not null,
primary key (num_borrado)
) charset=utf8 ENGINE=InnoDB;
```

Una vez hecho esto deberemos editar el fichero del servidor de apache por defecto que sirve nuestra web. Para ello ejecutamos los siguientes comandos:

```
cp /etc/apache2/sites-available/default /etc/apache2/sites-
available/vihostdom.servehttp.com
a2dissite default
```

El primer comando hace una copia del fichero por defecto y lo renombra a nuestro dominio, el segundo deshabilita el archivo por defecto que usa apache para servir las webs.

Editamos el contenido de */etc/apache2/sites-available/vihostdom.servehttp.com* cambiando:

```
DocumentRoot /var/www/vihostdom
```

De esta forma cada vez que alguien acceda a nuestro servidor, la web por defecto que saldrá será esa. Se podría haber utilizado la directiva

```
ServerName vihostdom.servehttp.com
```

Para que nuestro servidor sólo respondiese a los que acceden a través de esa url pero en este caso se ha preferido que se muestre siempre la web por defecto dando igual la dirección que se escriba.

Ahora para poder acceder a nuestro panel de control en el directorio */var/www/froxlor* desde nuestra web que se encuentra en */var/www/vihostdom* deberemos habilitar el módulo alias:

```
a2enmod alias
```

Y crear el alias correspondiente en el fichero de host virtual (/etc/apache/sites-available/vihostdom.servehttp.com) añadiendo:

```
Alias /froxlor /var/www/froxlor
```

Reiniciamos apache para que los cambios surtan efecto:

```
service apache2 restart
```

Puesta en marcha online

Creamos una cuenta gratuita en http://www.noip.com/ que ofrece tres hosts gratuitos y la redirección de cada uno de ellos a una dirección ip pública dinámica que le especifiquemos utilizando su página web o instalando el cliente.

Creamos los hosts gratuitos e instalamos el cliente en nuestro servidor linux especificando nuestro usuario, contraseña, los hosts a actualizar y la frecuencia de envío de la ip.

Una vez realizado esto, nuestro servidor cada cierto tiempo enviará la ip pública de la red en la que se encuentre, de esta forma en caso de que cambie se actualizará automáticamente en los servidores dns de no-ip y el tiempo en que se pierda la conexión será mínimo.

En el router doméstico abriremos los siguientes puertos:

Puerto	Servicio
20-21 TCP	FTP
22 TCP	SSH
25 TCP	SMTP
80 TCP	HTTP
110 TCP	POP3
143 TCP	IMAP4

Y los redirigimos a la ip privada de nuestro servidor (192.168.1.100 en este caso).

Una vez abiertos nuestro servidor y sus servicios serán completamente accesibles desde internet.