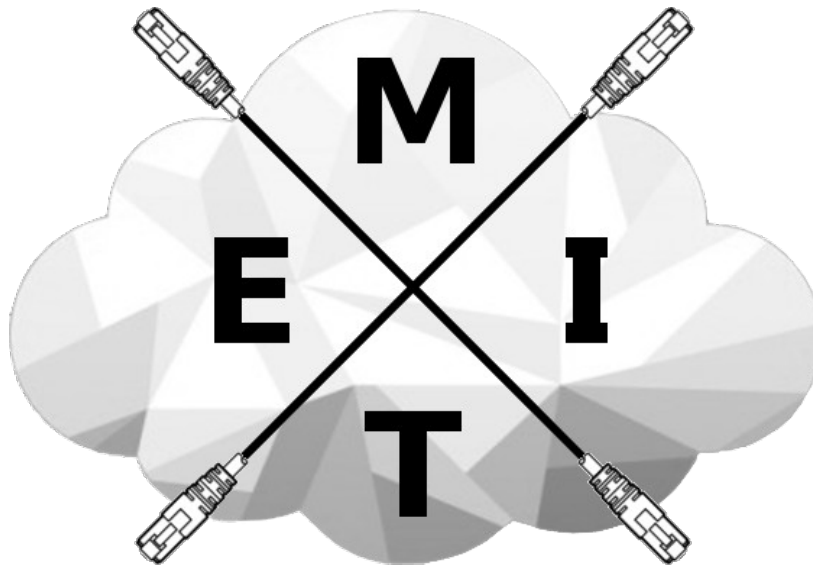


Dpto. INFORMÁTICA - I.E.S. PUERTO DE LA CRUZ

MÓDULO PROYECTO

C.F.G.S. Administración de Sistemas Informáticos en Red

MEIT



Autores:

Antonio Hernández Domínguez
Miriam Rodríguez Méndez

Miércoles 15 de junio 2016

Tutor:

José Antonio Acevedo Mesa

Índice

1. Introducción.....	1
1.1. Origen y contextualización.....	1
1.2. Objetivo del proyecto.....	2
2. Resumen global.....	3
2.1. Equipo Servidor.....	3
2.1.1. Hardware.....	3
2.1.2. Software.....	6
3. Instalación y configuración del servidor.....	7
3.1. Sistema Operativo.....	7
3.1.1. Instalación (Tarjeta micro SD).....	7
3.1.2. Primer acceso y configuración.....	8
3.2. Servidor Nginx.....	11
3.3. Herramienta php-fpm.....	12
3.4. Servidor FTP.....	14
3.5. Motor MySQL.....	15
3.5.1. Base de datos.....	15
3.6. Repositorio y control de versiones GIT.....	18
4. Aplicación Web.....	20
4.1. Recursos utilizados.....	20
4.2. Configuración de la aplicación web.....	20
4.2.1. La capa del Modelo.....	21
4.2.2. La capa de la Vista.....	21
4.2.3. La capa del Controlador.....	25
4.3. Código PHP.....	25
5. Crear host en NO-IP.....	28
5.1. Abrir puertos (Router).....	31
5.2. Comprobación acceso externo.....	32
6. Conclusiones.....	34

1. Introducción

1.1. Origen y contextualización

El nombre del proyecto o más bien de nuestra idea de empresa -sobre la que vamos a desarrollar nuestro proyecto final- es "MEIT" ("Meet Enterprise Information Technologies"). Hemos elegido este nombre con la intención de reflejar el tipo de actividad en la que se va centrar nuestra idea de negocio; la cual será la de hacer de 'intermediaria' entre todas aquéllas empresas, negocios, freelancers, autónomos... y un largo etcétera, que quieran adentrarse dentro del mundo online y deseen disponer de un asesoramiento informático total, en el que se proceda de forma rápida y con precios/productos adaptados a cada situación y/o cliente.

Empezando por "Meet Enterprise", no cabe duda que la propia traducción ya nos dice que nuestra empresa será, en cierto modo, el punto en el que converjan nuestros clientes (con sus necesidades) y nosotros (junto con nuestros proveedores); y las soluciones que les proporcionaremos a éstos.

Y qué decir de "IT" o "Information Technologies"...simplemente enmarcamos ese punto de convergencia del que hablamos antes en un sector específico, el de la informática y las ciencias de la información.

Visto el nombre y la idea de nuestro proyecto, deseamos que a partir de aquí el desarrollo del mismo sea, como poco, un proyecto interesante en el que podamos englobar todo lo que, con mucha ilusión y esmero, hemos aprendido durante estos 2 cursos.

1.2. Objetivo del proyecto

El objetivo de este proyecto es recopilar todo lo aprendido durante el ciclo y plasmar todo ese conocimiento en un único trabajo. Por ello, partimos de la creación de una empresa, en la que se hará uso de un servidor, en el que pasaremos a montar, cablear y configurar; de manera que queden cubiertas todas las necesidades que se puedan dar y todo aquello que sea indispensable para su correcto funcionamiento.

Éste servidor alojará:

- Una aplicación web con la que poder destacar en internet; con la que los clientes interactuarán y donde la empresa podrá publicitarse y vender sus productos.

◆ Con base en esta idea, se nos presentan las siguientes necesidades:

- Disponer de un equipo (hardware) ha emplear como servidor.
- Disponer de una conexión a internet y un router asociado a la misma (proveedor ISP).
- Disponer del cableado necesario.

◆ Y desde la parte de software necesitamos:

- Un sistema operativo ligero y adecuado al hardware del que dispongamos y al software a implementar.
- Un servidor HTTP con los módulos correspondientes (php, mysql...).
- Un servicio SSH para poder administrar desde remoto el equipo.
- Un servicio FTP para poder subir/bajar archivos de forma remota y sencilla.
- Un controlador de versiones y su respectivo repositorio con el cual poder desarrollar dicha aplicación web de manera rápida, óptima y segura.
- Un motor de base de datos.

2. Resumen global

Teniendo en cuenta las necesidades explicadas en el apartado anterior, el proyecto se va a dividir en diferentes fases o partes, relacionadas entre sí, cada una de estas partes será esencial para que el conjunto funcione como uno solo. A continuación resumiremos las principales partes en las que trabajaremos:

2.1. Equipo Servidor

2.1.1. Hardware

El hardware del que nos vamos a servir será:

- Una Raspberry Pi 2 Model B, cuyas especificaciones son las siguientes:

SoC	Broadcom BCM2836
CPU	ARM11 ARMv7 ARM Cortex-A7 4 núcleos @ 900 MHz
Overclocking	Sí, hasta arm_freq=1000 sdram_freq=500 core_freq=500 over_voltage=2 de forma segura
GPU	Broadcom VideoCore IV 250 MHz. OpenGL ES 2.0
RAM	1 GB LPDDR2 SDRAM 450 MHz
USB 2.0	4
Salidas de vídeo	HDMI 1.4 @ 1920x1200 píxeles
Almacenamiento	microSD
Ethernet	Sí, 10/100 Mbps
Tamaño	85,60x56,5 mm
Peso	45 g
Consumo	5v, 900mA, aunque depende de la carga de trabajo de los 4 cores
Precio	35 dólares

- Un cable ethernet o par trenzado UTP de categoría 5e (Gigabit Ethernet).
- Una tarjeta micro SD con 16 gigabytes de almacenamiento.
- Un transformador (fuente de alimentación) de 2.1 amperios salida USB.
- Un adaptador USB/802.11n para conexiones inalámbricas.
- Una "powerbank" con 13.000 amperios de capacidad y una salida de 2.1 amperios.
- Una caja para la Raspberry con ventilador/disipador de calor acoplado, que irá conectado a los pines GPIO 04 (DC Power 5v) y 06 (Ground).
- Un router con conexión a internet (Proveedor ISP).



La elección de este hardware se debe a su bajo coste, su rápida implementación y su gran versatilidad a la hora de ponerlo en marcha. Hemos decidido disponer, además del cableado para llevar a cabo una instalación fija, de un adaptador USB para realizar conexiones inalámbricas que cumpliera el estándar 802.11n (wifi) y una fuente de alimentación externa (no cableada de manera fija al suministro eléctrico) para así explotar un poco más el tamaño del que será nuestro servidor y su posicionamiento dentro de un espacio específico.

Otro motivo por el cual hemos decidido conectarle una batería externa como fuente de alimentación es para que nuestro servidor no se apague en caso de que se produzca un apagón; con lo que ésta tendría unas 13.000mAh/900mAh = 14.44 horas de uso hasta que se agotase la batería; dependiendo también del uso que hagamos de la misma y el consumo mayor o menor de sus núcleos.



2.1.2. Software

Visto el hardware que vamos a usar, vamos ahora a resumir brevemente el software que implementaremos en nuestro servidor para cumplir con los objetivos que nos hemos propuesto:

- El sistema operativo ha instalar como base de nuestro servidor será "**Raspbian GNU/Linux 8.0 (jessie)**" sin entorno gráfico, puesto que queremos que el sistema sea lo más óptimo posible.
- Implementaremos un servidor HTTP "**Nginx**". La decisión de decantarnos por éste servidor frente a otros como por ejemplo "Apache", es por su bajo consumo de recursos.
- Para poder interpretar código php, instalaremos otra herramienta de la que se servirá Nginx para tal fin; "**php5-fpm**".
- Un motor de bases de datos "**MySQL**".
- Para las conexiones remotas haremos uso del protocolo SSH, instalando así el paquete "**Openssh-server**". Que en este caso viene por defecto en el sistema.
- Como servidor FTP instalaremos el paquete "**vsftpd**".
- Para nuestro repositorio local, instalaremos el software controlador de versiones "**GIT**" (git—core).
- Un host en NO-IP para poder acceder fácilmente desde el exterior al servidor.

La instalación y configuración de éste software se encuentra detallada en los siguientes apartados.

3. Instalación y configuración del servidor

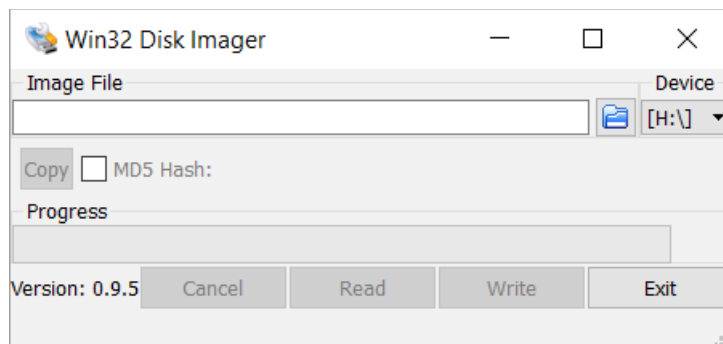
3.1. Sistema Operativo

3.1.1. Instalación (Tarjeta micro SD)

Desde la página oficial de [Raspberry](http://www.raspberrypi.org) descargamos la imagen del sistema operativo "**Raspbian Jessie Lite**".

Una vez descargado, insertamos nuestra tarjeta micro SD en el equipo desde el cual la vamos a formatear para luego grabar la imagen del sistema en la misma.

Para ello, y puesto que el equipo que vamos a emplear para grabar la imagen es un Windows 10, vamos a hacer uso del software "**Win32DiskImager**". Iniciamos la aplicación como administrador, seleccionamos la imagen que nos hemos descargado y la letra asignada a nuestra tarjeta y hacemos click en "*write*".



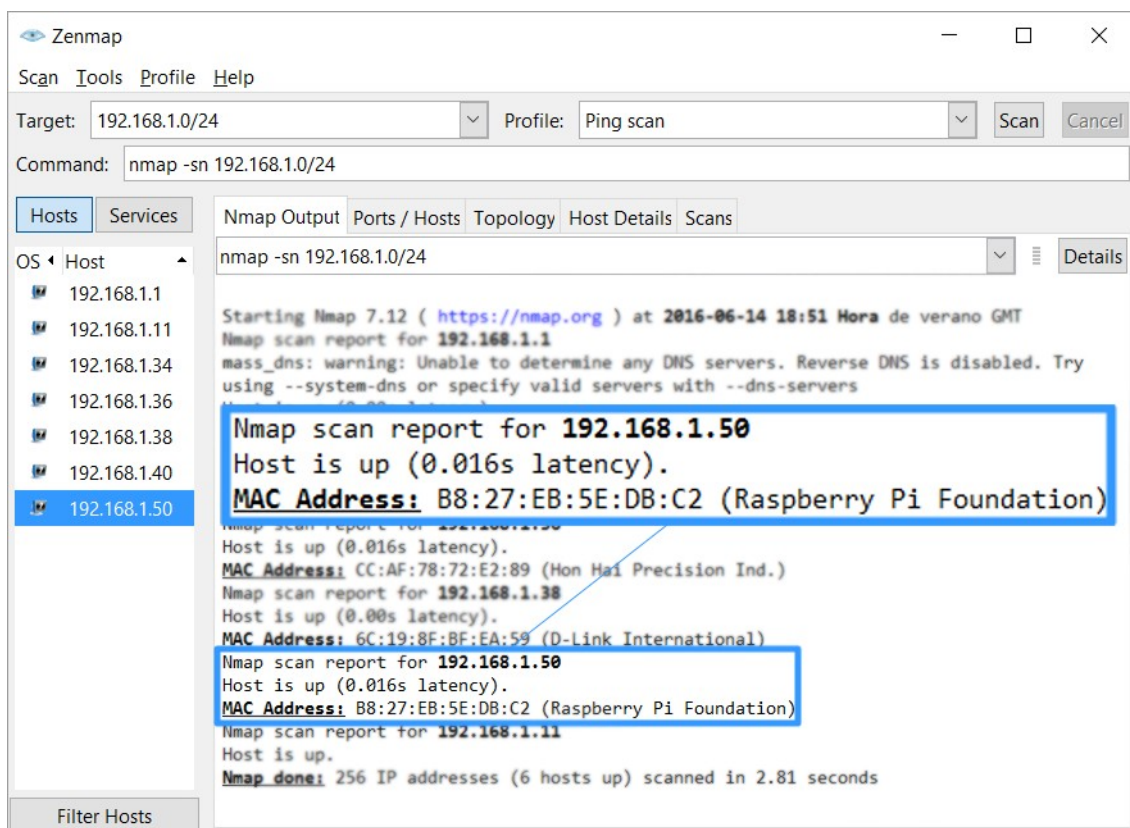
Extraemos la tarjeta en modo seguro y la insertamos en la ranura correspondiente de nuestra Raspberry para arrancar, a posteriori, nuestro servidor.

3.1.2. Primer acceso y configuración

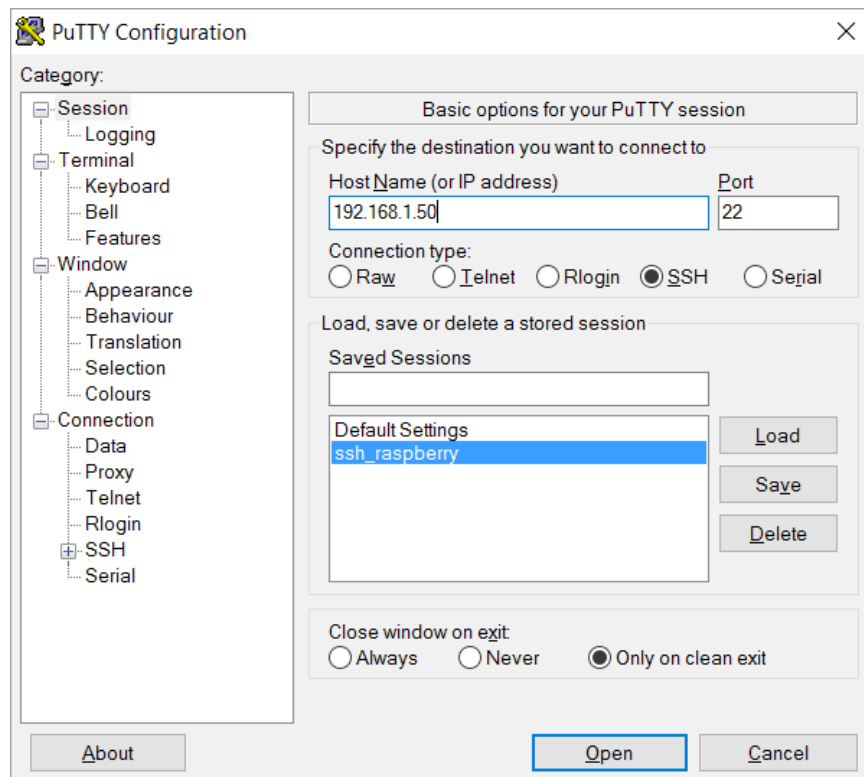
Dado que no disponemos de entorno gráfico, para poder conectarnos a la Raspberry y proceder a su configuración tendremos que hacerlo mediante acceso remoto. Para ésto, vamos a hacer uso del protocolo SSH. Raspbian ya viene con el servidor SSH instalado por lo que sólo tendremos que conocer la dirección IP de nuestro servidor para poder conectarnos.

Para descubrir la IP que le ha asignado el router de forma dinámica a nuestro servidor emplearemos la herramienta “**NMAP**” haciendo peticiones de ping a todos los hosts de la red en busca de dicha IP.

El comando sería: `nmap -sn 192.168.1.0/24`:



Haciendo uso de la herramienta para conexiones remotas **"PuTTY"**, y conociendo la IP asignada a nuestro servidor, realizamos la conexión:



Se intercambiarán las claves pública/privada y posteriormente se nos abrirá la terminal con el inicio de sesión en el que especificaremos el usuario por defecto **"PI"**, cuya contraseña es **"raspberry"**:

```
pi@raspberrypi: ~  
login as: pi  
pi@192.168.1.50's password:  
  
The programs included with the Debian GNU/Linux system are free software;  
the exact distribution terms for each program are described in the  
individual files in /usr/share/doc/*/copyright.  
  
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent  
permitted by applicable law.  
Last login: Tue Jun 14 19:04:11 2016 from 192.168.1.1  
pi@raspberrypi:~$
```

Lo siguiente es ejecutar el comando **"raspi-config"** con permisos de **sudo**, este comando despliega un menú en consola con distintas opciones, ahora simplemente seguimos las indicaciones del asistente **"raspi-config"** y con esto ya tenemos nuestra distribución Raspbian.

Para terminar, vamos a configurar la interfaz de red nuestro servidor (Raspberry Pi) para que tome una misma dirección IP -o dirección estática- y así poder conectarnos, establecer reglas, abrir puertos, etc.... sin tener que modificar dichas configuraciones cada vez que el router le asigne una ip diferente.

Para ello, modificamos el fichero `"/etc/network/interfaces"` y añadimos las siguientes líneas:

```
auto eth0

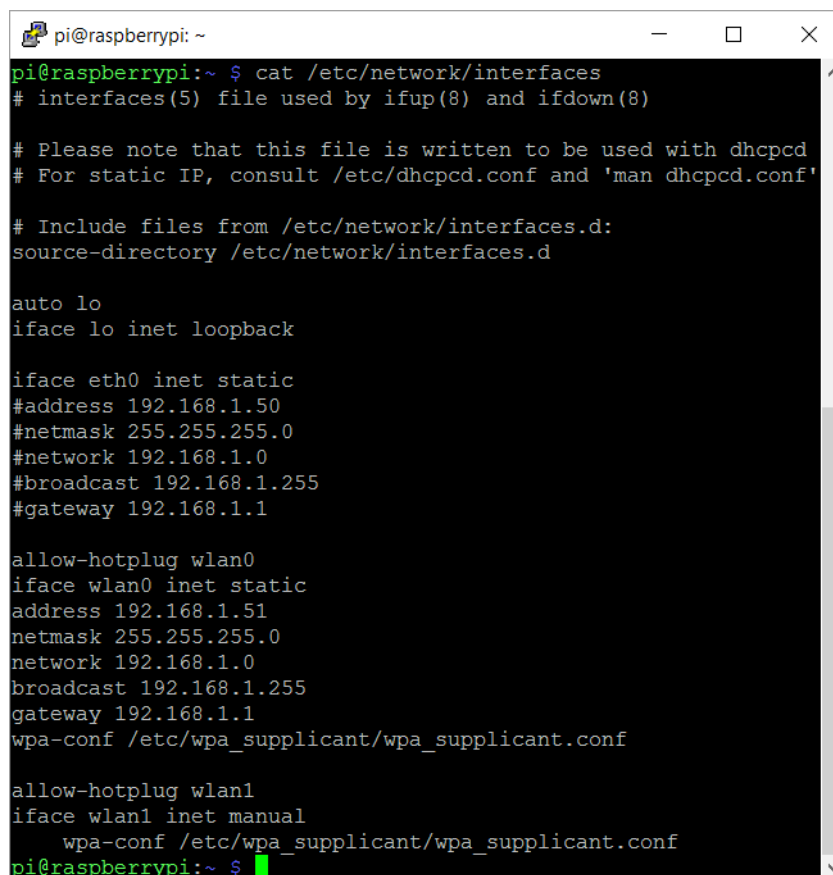
iface lo inet loopback

iface eth0 inet static

address 192.168.1.50

netmask 255.255.255.0

gateway 192.168.1.1
```

A screenshot of a terminal window titled 'pi@raspberrypi: ~'. The terminal shows the command 'cat /etc/network/interfaces' being executed. The output displays the configuration for network interfaces. It includes comments about the file's purpose and usage of dhcpcd. The configuration defines the loopback interface 'lo' and the ethernet interface 'eth0' with a static IP address of 192.168.1.50, a netmask of 255.255.255.0, a network of 192.168.1.0, a broadcast of 192.168.1.255, and a gateway of 192.168.1.1. It also configures wireless interfaces 'wlan0' and 'wlan1' with static IP addresses and includes the wpa-supPLICANT configuration file. The terminal prompt 'pi@raspberrypi:~ \$' is visible at the bottom.

```
pi@raspberrypi:~ $ cat /etc/network/interfaces
# interfaces(5) file used by ifup(8) and ifdown(8)

# Please note that this file is written to be used with dhcpcd
# For static IP, consult /etc/dhcpcd.conf and 'man dhcpcd.conf'

# Include files from /etc/network/interfaces.d:
source-directory /etc/network/interfaces.d

auto lo
iface lo inet loopback

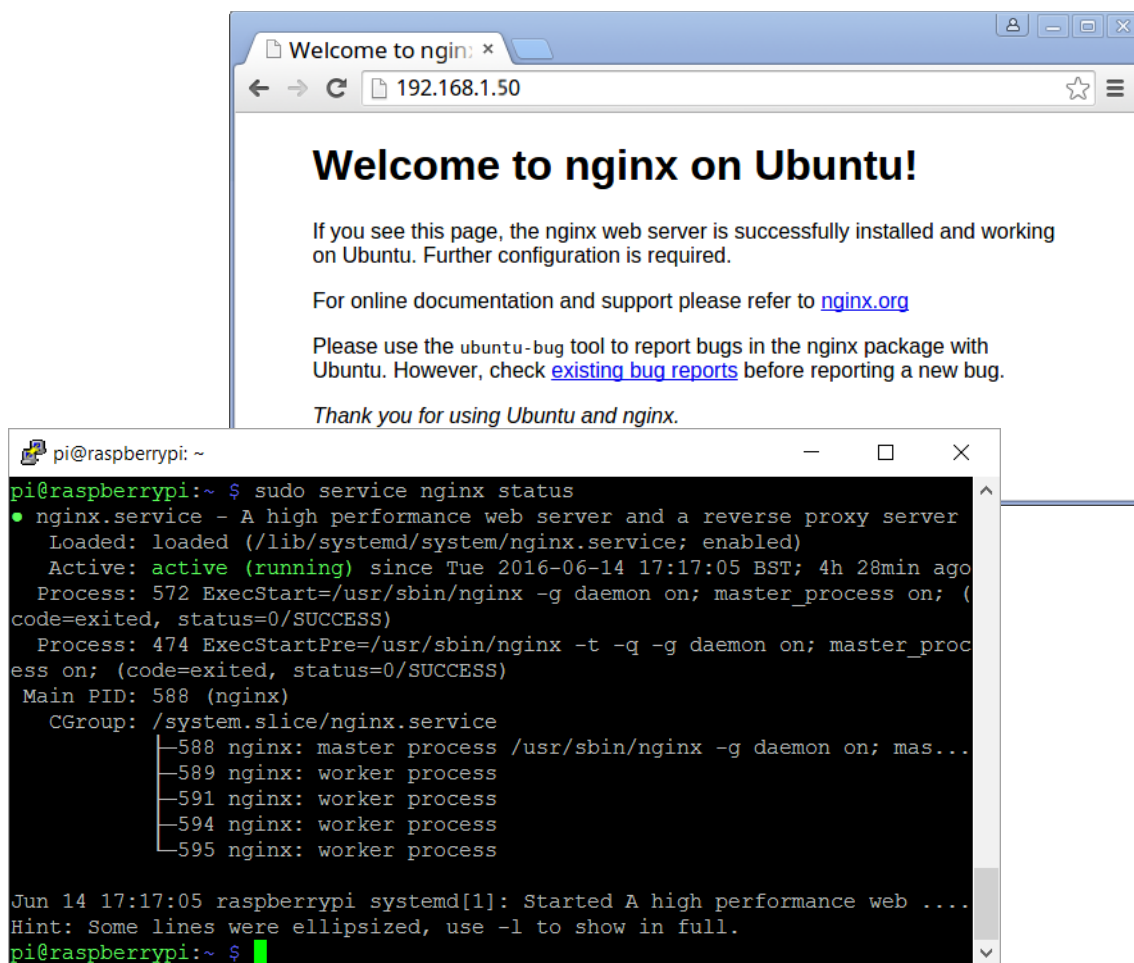
iface eth0 inet static
#address 192.168.1.50
#netmask 255.255.255.0
#network 192.168.1.0
#broadcast 192.168.1.255
#gateway 192.168.1.1

allow-hotplug wlan0
iface wlan0 inet static
address 192.168.1.51
netmask 255.255.255.0
network 192.168.1.0
broadcast 192.168.1.255
gateway 192.168.1.1
wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf

allow-hotplug wlan1
iface wlan1 inet manual
    wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf
pi@raspberrypi:~ $
```

3.2. Servidor Nginx

Para instalar el servidor http "**Nginx**", abrimos una consola y lanzamos el comando "*sudo apt-get install nginx*". Una vez que termine, comprobamos que está en funcionamiento accediendo a él a través de nuestro navegador o haciendo uso del comando "*service nginx status*":



3.3. Herramienta php-fpm

En lugar de instalar php5, instalaremos php5-fpm (FastCGI Process Manager), una implementación alternativa con algunas características adicionales. Si no podemos instalarlo lanzando el comando "*sudo apt-get install php5-fpm*" desde repositorios, podemos agregarlo a mano al *sources.list*:

```
deb http://packages.dotdeb.org stable all  
deb-src http://packages.dotdeb.org stable all
```

Luego actualizamos el repositorio y lanzamos el comando que instale "php5-fpm" junto con los módulos necesarios para hacer uso del motor mysql, entre otros:

```
sudo apt-get update  
sudo apt-get install php5-cli php5-suhosin php5-fpm php5-cgi php5-mysql
```

Y finalmente iniciamos el servicio con:

```
sudo service php5-fpm start
```

Ahora probaremos que php funciona bajo nginx, para ello es necesario modificar ligeramente el archivo nginx.conf. Concretamente, en el bloque http hay que añadir index.php a la directiva index, para que quede *index index.php index.html index.htm*.

A su vez, necesitamos crear la comunicación entre nginx y php mediante un socket, para ello añadimos lo siguiente en el bloque http:

```
upstream php {  
    server unix://var/run/php-fpm.socket;  
}
```

Luego, dentro del bloque `server`, añadimos una regla que permita manejar los archivos `php`:

```
location ~ \.php$ {  
    include fastcgi_params;  
    fastcgi_index index.php;  
    fastcgi_param SCRIPT_FILENAME  
    $document_root$fastcgi_script_name;  
    fastcgi_pass php;  
}
```

Una última modificación al archivo `/etc/php5/fpm/pool.d/www.conf` y agregamos la línea `listen = /var/run/php-fpm.socket`.

3.4. Servidor FTP

Para implementar un servidor FTP vamos a descargar e instalar el paquete **"vsftpd"** con el comando **"sudo apt-get install vsftpd"**. Una vez instalado, editamos el siguiente archivo de configuración:

```
sudo nano /etc/vsftpd.conf
```

Descomentamos las siguientes líneas para permitir la escritura de archivos a los usuarios de la Raspberry Pi.

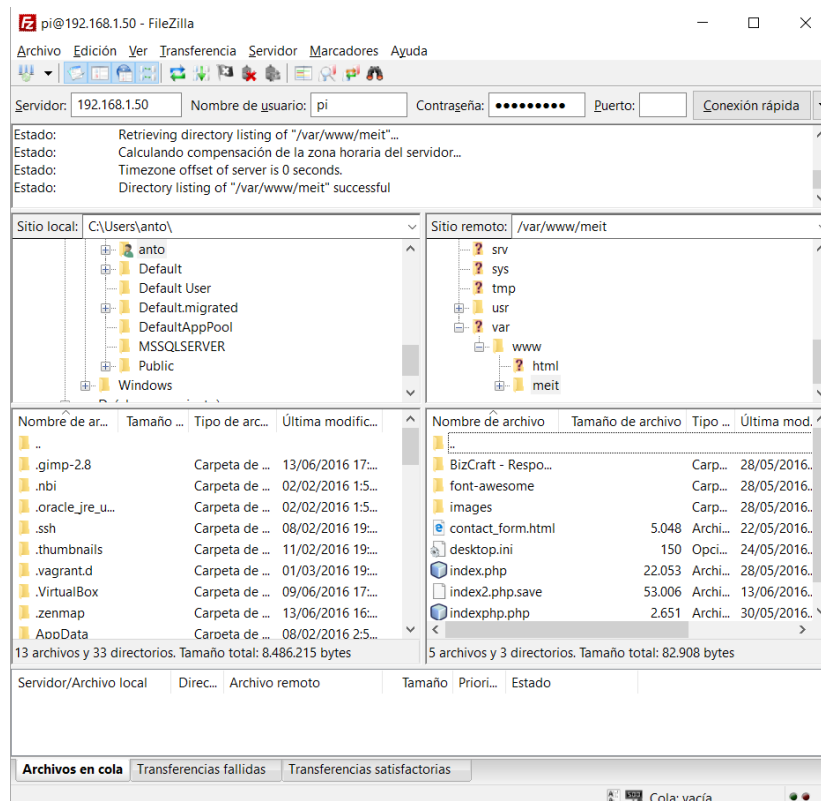
```
local_enable=YES
```

```
write_enable=YES
```

Y por último reiniciamos el servicio:

```
sudo service vsftpd restart
```

Desde un cliente FTP como **"Filezilla"** vemos que podemos acceder al servidor FTP:



3.5. Motor MySQL

Para el almacenamiento, gestión y administración de la información que va a recopilar nuestro servidor de cara a la aplicación web y a los usuarios que hagan uso de la misma; vamos a implementar un motor de base de datos MySQL.

Desde una terminal lanzamos el comando "*sudo apt-get install mysql-server*", cuando nos pida crear un usuario y clave lo rellenamos y terminamos la instalación.

3.5.1. Base de datos

Crearemos una base de datos que almacenará información relacionada con usuarios, servicios y proveedores. Haremos uso de ella a parte de para guardar datos, para gestionar la página web. Contará con siete tablas en principio, cinco de estas tablas estarán relacionadas entre sí, y almacenarán datos de clientes, usuarios, proveedores, pagos y servicios.

Necesitamos crear otras dos tablas individuales, simplemente para llevar cuenta de datos relevantes para el servidor, tales como bajas de usuarios y de servicios.

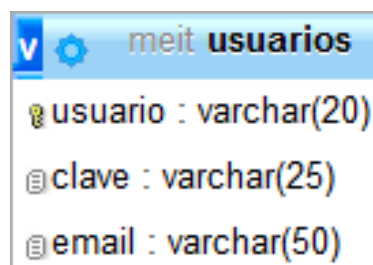
La base de datos que vamos a utilizar se llama 'meit', consta de 7 tablas de las cuales 5 están relacionadas entre sí, como ya se explicó anteriormente, hemos creado 2 tablas para llevar cuenta de los servicios y usuarios que se contraten o revoquen desde el principio, dichas tablas no tienen relación con ninguna otra. A continuación pasamos a explicar cada tabla individualmente.

- Tabla **Cientes**: la tabla clientes cuenta con once campos, de los cuales 'código' hará la función de clave primaria, y tanto 'nombreusuario' como 'codigoproveedor' serán claves foráneas.

Hemos incorporado también otros campos básicos a la hora de definir un cliente, como son 'nombre', 'apellido1', 'apellido2', 'telefono', 'provincia', 'localidad', 'cod_postal' y 'dirección'. El campo 'nombreusuario' lo utilizamos para unir las tablas clientes y usuarios, es necesario a la hora de registrar nuevos usuarios en la aplicación web, para que una vez registrado y logueado, sea el propio usuario el que rellene los campos de cliente en su pestaña 'Perfil'. El campo 'codigoproveedor' lo utilizamos para unir las tablas clientes y proveedores, con esto conseguimos que un cliente pueda tener varios proveedores.



- Tabla **Usuarios**: la tabla usuarios cuenta con tres campos, 'usuario' será la clave primaria, y los campos restantes 'clave' y 'email' son necesarios para el registro o login de los usuarios en la aplicación web.



- Tabla **Servicios**: la tabla servicios cuenta con cinco campos, de los cuales 'codigo' es la clave primaria, y tanto 'usuario' como 'codigoproveedor' son claves foráneas, con esto conseguimos que un mismo servicio tenga diferentes proveedores y que varios usuarios lo puedan contratar.

- Los dos campos restantes son para describir mejor los servicios, 'nombre' y 'precio' hacen la función de definir el nombre del servicio y el precio del mismo.

meit servicios	
🔑	codigo : varchar(15)
📄	nombre : varchar(50)
#	precio : decimal(15,2)
📄	codigoproveedor : varchar(20)
📄	usuario : varchar(20)

- Tabla **Proveedores**: la tabla proveedores consta de cinco campos, 'codigo' es la clave primaria, y el resto de los campos describen mejor el proveedor, como son 'nombre', 'direccion', 'telefono' y 'pais'.

meit proveedores	
📄	nombre : varchar(50)
🔑	codigo : varchar(20)
📄	direccion : varchar(50)
📄	telefono : varchar(20)
📄	pais : varchar(50)

- Tabla **Servicios_baja** y **Usuarios_baja**: estas dos tablas son esencialmente informativas, para que los administradores lleven cuenta de los usuarios que se han registrado y dado de baja en la aplicación y los servicios que se han contratado o revocado.

meit usuarios_baja	
📄	usuario : varchar(20)
📄	email : varchar(20)

meit servicios_baja	
📄	codigo : varchar(15)
📄	usuario : varchar(20)

3.6. Repositorio y control de versiones GIT

La idea es que en desarrollo (on-line) nunca se debe editar nada, se debe hacer en local y sólo subir los cambios cuando realmente esté todo listo. Esto lo podemos hacer muy cómodamente creando un repositorio dentro de nuestra Raspberry Pi.

Antes de nada, si no lo tenemos ya, instalamos Git:

```
sudo apt-get install git-core
```

Creamos un directorio dentro del home y generamos un repositorio vacío:

```
cd /home/git/  
mkdir miweb.git  
cd miweb.git  
git init --bare
```

Ya tenemos nuestro repositorio creado, ahora vamos a hacer que se publiquen directamente los cambios a la raíz del servidor (que estará en "/var/www/meit").

```
cd /hooks/  
sudo nano post-receive
```

Y añadimos lo siguiente:

```
#!/bin/sh  
GIT_WORK_TREE=/var/www git checkout -f
```

Nos aseguramos que tenga los permisos correspondientes:

```
chmod +x post-receive
```

Ahora solamente tendremos que añadir el repositorio a nuestro gestor Git local:

```
git remote add production git@urldelserver.com:miweb.git  
git push production +master:refs/heads/master
```

De aquí en adelante los cambios los haremos en nuestro ordenador, y publicaremos nuestra web haciendo push de los cambios.

En caso de querer partir de datos ya alojados en un directorio en concreto, (como por ejemplo código de una web, etc..) y cuyos ficheros queremos convertirlos en un repositorio; simplemente nos situamos en dicha carpeta y lanzamos el comando "*git init*" para que se creen los ficheros de nuestro repositorio.

Para que podamos trabajar con el directorio en el que hemos creado nuestro repositorio tenemos que editar el fichero ".*git/config*" y poner las siguientes líneas atendiendo a nuestra dirección ip y a la ruta de la carpeta que queremos emplear como "árbol de trabajo":

```
[core]  
    repositoryformatversion = 0  
    filemode = false  
    bare = false  
    logallrefupdates = true  
    symlinks = false  
    ignorecase = true  
    hideDotFiles = dotGitOnly  
[remote "origin"]  
    url = pi@192.168.1.50:/var/www/meit  
    fetch = +refs/heads/*:refs/remotes/origin/*  
[branch "master"]  
    remote = origin  
    merge = refs/heads/master
```

4. Aplicación Web

4.1. Recursos utilizados

Para la realización de la aplicación, hemos utilizado un framework llamado Bootstrap, un framework "es un entorno o ambiente de trabajo para desarrollo; dependiendo del lenguaje normalmente integra componentes que facilitan el desarrollo de aplicaciones como el soporte de programa, bibliotecas, plantillas y más". Como ya mencionamos con anterioridad, Bootstrap nos permite crear interfaces web con HTML, CSS y JavaScript, cuya particularidad es la capacidad que tiene para adaptar la interfaz del sitio web al tamaño del dispositivo en que se visualice.

Escogimos éste en concreto porque lo hemos utilizado en las empresas, y podremos sacar partido de las tareas que ya hayamos hecho. Nos hemos apoyado en una plantilla que hemos descargado y configurado a nuestro gusto.

Trabajamos con php para conectarnos y realizar consultas a la base de datos.

4.2. Configuración de la aplicación web

Para la elaboración de la aplicación web hemos utilizado un framework llamado Bootstrap, el cual hemos explicado anteriormente, y lenguaje php para la interacción con la base de datos.

Para explicar mejor la configuración de dicha aplicación nos basaremos en el Modelo - Vista - Controlador, el cual separa la aplicación en tres partes principales. A continuación se explica una petición MVC típica.

4.2.1. La capa del Modelo

El modelo representa la parte de la aplicación que implementa la lógica de negocio, esto significa que es responsable de la recuperación de datos, así como de su procesamiento, validación, asociación y cualquier otra tarea relativa a la manipulación de dichos datos.

Los objetos del modelo pueden ser considerados como la primera capa de la interacción con cualquier base de datos que podría estar usando la aplicación.

En nuestro caso se podría decir que el código php actuaría como modelo, ya que gracias a su interacción con la base de datos podemos acceder a datos o visualizarlos desde la aplicación.

4.2.2. La capa de la Vista

La vista hace una presentación de los datos del modelo estando separada de los objetos del modelo. Es responsable del uso de la información de la cual dispone para producir cualquier interfaz de presentación de cualquier petición que se presente.

Por ejemplo, como la capa de modelo devuelve un conjunto de datos, la vista los usaría para hacer una página HTML que los contenga.

Si nos centramos en nuestra aplicación, se podría decir que consta de cinco vistas diferentes, dependiendo de lo que el usuario desee visualizar. A continuación las describiremos una a una.

Vista principal o 'index.php': esta vista muestra la portada o inicio de nuestra aplicación web, presenta nuestra empresa, nuestro logo y nuestros servicios.



En la parte superior derecha se puede observar un barra de navegación creada para recorrer esta primera vista más fácilmente.

Si seleccionamos cualquier elemento de dicha barra nos llevará a la parte de la página que tenga asociada.



Vista de los usuarios: esta vista nos muestra un entorno preparado para los usuarios registrados en nuestra página, con la posibilidad de completar su registro rellenando datos personales y pudiendo volver a la vista anterior por medio un un botón.

MEITUSUARIO

Perfil

Mis productos y servicios

Mis proveedores

Contratar

Salir

Información personal

Nombre:

Primer Apellido:

Segundo Apellido:

Teléfono:

Provincia:

Localidad:

Código postal:

Dirección:

Guardar Cambios

Vista productos y servicios: aquí nos encontramos con la visualización en pantalla de las compras realizadas por los clientes.

MEITUSUARIO

Mis Servicios y Pagos

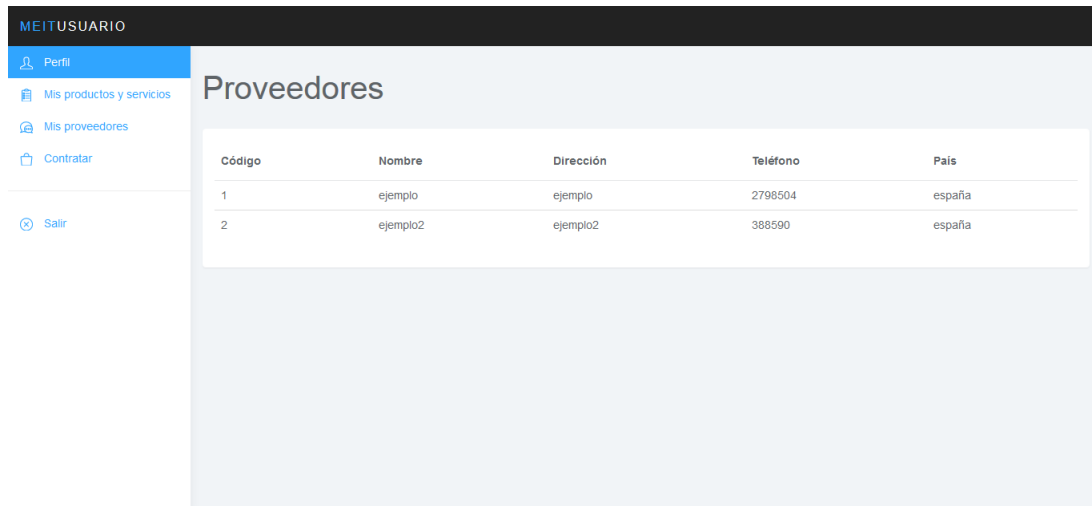
Servicios

Revocar	Código	Nombre	Precio	Código Proveedor
Revocar Seleccionados				

Pagos

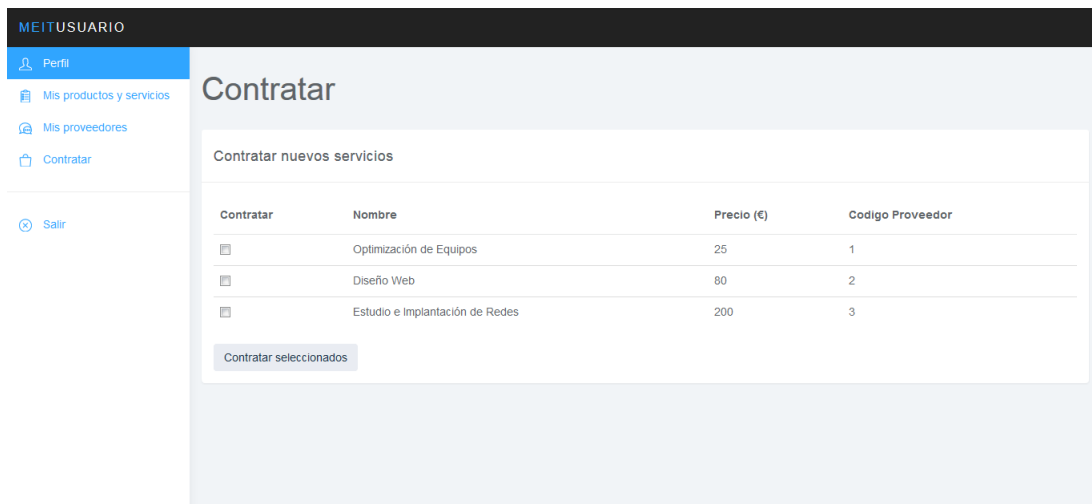
Número de Pago	Cantidad	Fecha Contrato	Fecha Vencimiento
----------------	----------	----------------	-------------------

Vista proveedores: en esta vista veremos los proveedores de los servicios o productos que nuestros clientes tienen contratados, así el usuario puede llevar cuenta de todo.



Código	Nombre	Dirección	Teléfono	País
1	ejemplo	ejemplo	2798504	españa
2	ejemplo2	ejemplo2	388590	españa

Vista Contratar: esta vista como bien describe su nombre, facilita a los usuarios la contratación de nuevos servicios.



Contratar	Nombre	Precio (€)	Código Proveedor
<input type="checkbox"/>	Optimización de Equipos	25	1
<input type="checkbox"/>	Diseño Web	80	2
<input type="checkbox"/>	Estudio e Implantación de Redes	200	3

Contratar seleccionados

4.2.3. La capa del Controlador

La capa del controlador gestiona las peticiones de los usuarios. Es responsable de responder la información solicitada con la ayuda tanto del modelo como de la vista.

En resumen, los controladores son como administradores cuidando que todos los recursos necesarios para completar una tarea se deleguen a los trabajadores más adecuados.

4.3. Código PHP

Hemos decidido trabajar con PHP para conectarnos y sacar información de la base de datos, ya que ha sido el lenguaje que más conocemos y el único con el que hemos trabajado en clase. Contamos con un index principal que contiene todo el código referente a login de usuarios ya registrados y registro de nuevos usuarios.

Conectarse por medio de PDO implica una sintaxis un tanto diferente pero muy sencilla:

```
$host = "localhost";  
$user = "root";  
$pwd = "root";  
$dbName = "meit";  
  
$connect = new PDO("mysql:host=".$host.";dbname=".$dbName.";",$user,$pwd);//connection to the dat
```

Para realizar consultas a la base de datos utilizamos la siguiente sintaxis, primero la preparamos para a continuación ejecutarla:

```
$query = $connect -> prepare("SELECT * FROM usuarios");//preparamos la consulta para sacar la info de la ba  
$query -> execute(array());//ejecutamos dicha consulta..
```

5. Crear host en NO-IP

Debido a que no disponemos de una dirección IP externa estática, dado que nuestro proveedor ISP nos proporciona una dinámica; necesitamos alguna manera de conocer la dirección IP de nuestro modem de banda ancha, para tener una dirección de acceso a nuestra red.

Existe el servicio de www.no-ip.com, que nos permite crear un dominio virtual (o si tenemos un nombre de dominio libre, lo pueden configurar también ahí), y este dominio virtual, asignarlo a una dirección IP donde esté conectado nuestro servidor. Entonces, el primer paso obligado, es ingresar en www.no-ip.com y crear una cuenta nueva, para escoger un subdominio de los varios que se tienen disponibles en dicho servicio.

The screenshot displays the 'My No-IP :: Dynamic DNS' web interface. The browser's address bar shows the URL <https://my.noip.com/#/dynamic-dns>. The page features a dark sidebar on the left with navigation options: Dashboard, Dynamic DNS (Free), Hostnames, Groups, Dynamic Update Client, My Services, Account, and Support Center. The main content area is titled 'Dynamic DNS' and includes a search bar and a table for managing hostnames. The table has columns for Hostname, IP / Target, Type, and Status. A single entry is visible: 'melt.ddns.net' with IP '81.37.69.67' and status 'Active'. To the right of the table is an 'Add Hostname' button. Further right, there's a 'Service Level' section indicating 'Free' and a warning that free hostnames expire every 30 days. Below this is a 'Hostname Count' gauge showing '1 / 3' and a 'Buy More Hostnames' link. At the bottom right, there is a 'Feedback' button.

Una vez que hemos creado y configurado la cuenta, vamos a trabajar en nuestro Raspberry Pi. La idea es que periódicamente detecte la dirección IP pública a la que está conectado el modem, y la envíe al servicio no-ip para actualizar el dominio virtual. Para esto, debemos instalar el paquete no-ip en nuestro Raspberry, usando los siguientes comandos:

```
mkdir no-ip  
cd no-ip  
wget http://www.no-ip.com/client/linux/noip-duc-linux.tar.gz  
tar -zxvf noip-duc-linux.tar.gz  
cd noip-2.1.9-1/  
make  
sudo make install
```

Al empezar la instalación, nos solicitará nuestros datos de la cuenta de no-ip, verificará el dominio virtual creado y nos preguntará el tiempo de actualización de la dirección ip externa. Podemos dejar todos los valores por defecto tal cual, y continuar hasta finalizar la instalación.

Podemos iniciar el servicio ejecutando "*sudo /usr/local/bin/noip2*" pero lo ideal sería que inicie automáticamente en el arranque del PI. Para esto debemos crear el archivo */etc/init.d/noip2* con el comando:

```
sudo nano /etc/init.d/noip2
```

y colocamos esto dentro del contenido del mismo:

```
#!/bin/bash

### BEGIN INIT INFO

# Provides: Servicio No-IP

# Required-Start: $syslog

# Required-Stop: $syslog

# Default-Start: 2 3 4 5

# Default-Stop: 0 1 6

# Short-Description: arranque automatico para no-ip

# Description:

#

### END INIT INFO

sudo /usr/local/bin/noip2
```

Guardamos el archivo, y finalmente damos permisos de ejecución, y lo colocamos en la cola de arranque:

```
sudo chmod +x /etc/init.d/noip2
```

```
sudo update-rc.d noip2 defaults
```

Así garantizamos que el dominio virtual creado en no-ip tenga siempre asignada nuestra dirección IP externa de nuestro modem.

5.1. Abrir puertos (Router)

Una vez que tengamos nuestro dominio virtual creado tenemos que abrir los puertos del router correspondientes para habilitar el acceso a los distintos servicios de la Raspberry. Para ello, estableceremos una regla para cada servicio, en total serán 3, que tome la IP estática que le habíamos dado a nuestro servidor y el puerto local por el que se da cada servicio:

- Puerto externo 5001 se asignará al puerto interno 80, dedicado al servidor HTTP.
- Puerto externo 5002 se reservará al puerto interno 22, para las conexiones SSH.
- Puerto externo 5003 se destinará al puerto interno 21, para el servidor FTP.

The screenshot shows the 'Configurador Router Fibra Óptica' interface. The 'Puertos' section is active, displaying a 'Configuración Puertos' form and a 'Tabla actual de mapeo de puertos'.

Configuración Puertos

Rellena los siguientes campos y pulsa el botón **Añadir**. Ten en cuenta que para abrir un rango de puertos debes usar el siguiente formato : 5001:5010

Nombre regla de puertos:

Dirección IP:

Protocolo:

Abrir Puerto/Rango Externo (WAN): (ej: 5001:5010)

Abrir Puerto/Rango Interno (LAN): (ej: 5001)

Añadir

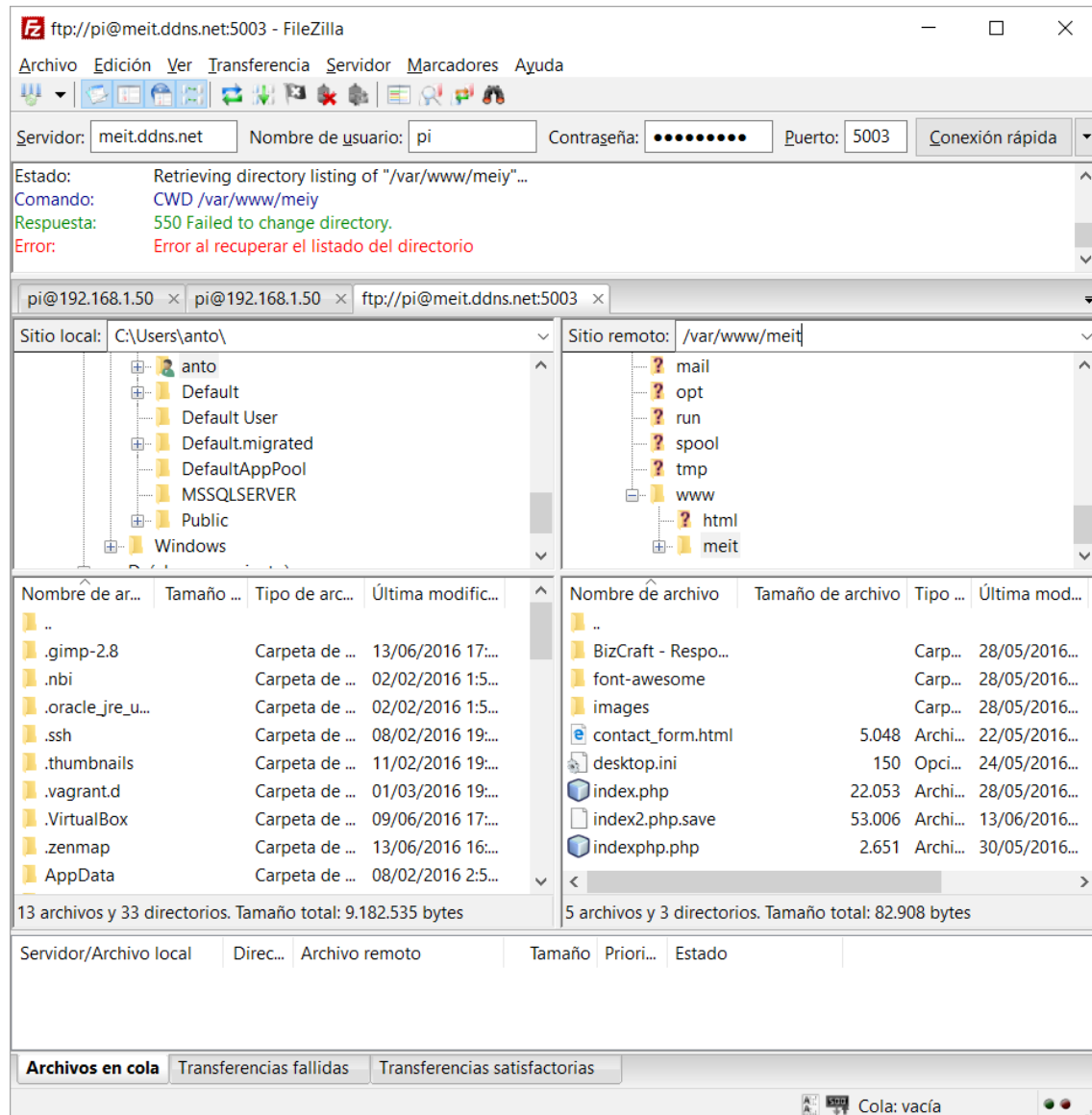
Tabla actual de mapeo de puertos

	Nombre	Protocolo	Puerto/Rango Externo	Puerto/Rango Interno	Dirección IP	Activar
✗	http_raspberry	TCP	5001	80	192.168.1.50	ON <input checked="" type="checkbox"/>
✗	raspberrry_admin	TCP	5002	22	192.168.1.50	ON <input checked="" type="checkbox"/>
✗	raspberrry_ftp	TCP	5003	21	192.168.1.50	ON <input checked="" type="checkbox"/>

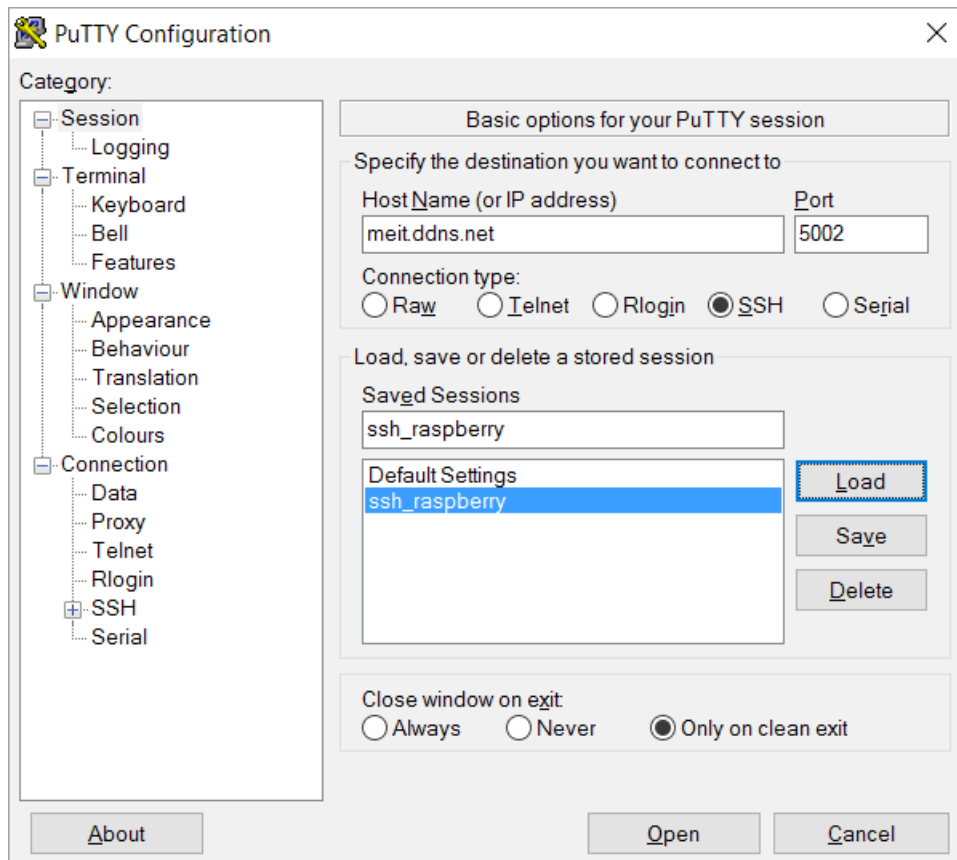
© Telefónica de España S.A.U. Todos los derechos reservados v1.1

5.2. Comprobación acceso externo

Probamos ahora acceder con el nombre de nuestro dominio virtual a los distintos servicios:



Acceso remoto externo SSH:



Clonación de repositorio con nombre de dominio virtual (externo):

```

MINGW64:/c/Users/anto/Desktop
anto@NAVE MINGW64 ~/Desktop
$ git clone ssh://pi@meit.ddns.net:5002/var/www/meit
Cloning into 'meit'...
The authenticity of host '[meit.ddns.net]:5002 ([81.37.69.67]:5002)' can't be es
tablished.
ECDSA key fingerprint is SHA256:A0riL6mdjpSITg+RMaOPuA42UjMLzj0qq6RC1IRPaZM.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '[meit.ddns.net]:5002,[81.37.69.67]:5002' (ECDSA) to
the list of known hosts.
Cloning into 'meit'...
pi@meit.ddns.net's password:
remote: Counting objects: 106, done.
remote: Compressing objects: 100% (106/106), done.
remote: Total 106 (deltaRec eiving 12), reuobsed 0je (dctes: 92% (98/106), 1.59
lt MiB | 1.56 Ma Oi)
Receiving objects: 100% (106/106), 2.37 MiB | 1.56 MiB/s, done.
Resolving deltas: 100% (12/12), done.
Checking connectivity... done.

anto@NAVE MINGW64 ~/Desktop
$ ls meit/
BizCraft - Responsive Html5 Template_files/  font-awesome/  index2.php.save
contact_form.html                           images/        indexphp.php
desktop.ini                                  index.php

anto@NAVE MINGW64 ~/Desktop
$

```

6. Conclusiones

Para terminar con éste documento queremos destacar las buenas sensaciones que hemos tenido durante el desarrollo de éste proyecto y la habilidad y buen hacer que hemos sabido demostrar con los problemas que nos íbamos encontrando, y con el resultado final del mismo que, para nosotros, ha logrado alcanzar los objetivos propuestos desde un principio; cumpliendo con la premisa de englobar todos los aspectos vistos durante estos 2 cursos del ciclo en Administración de Sistemas Informáticos en Red.

Por todo ello, consideramos que éste proyecto es un digno ejemplo de lo que hemos logrado aprender o, más bien, del nivel que hemos alcanzado como estudiantes de éste ciclo y como administradores de sistemas; y de las capacidades o habilidades que hemos adquirido como tal. Yendo incluso un poco más allá de lo visto en las clases, promovidos siempre por la curiosidad y por el querer saber más, más y más...