

TEMA 6: Despliegue **WAMP-LAMP**

Módulo

Despliegue de aplicaciones web

para los ciclos

Desarrollo de aplicaciones web



Despliegue FP-GS; Tema6:DespliegueWAMP-LAMP

© Gerardo Martín Esquivel, Diciembre de 2020

Algunos derechos reservados.

Este trabajo se distribuye bajo la Licencia "Reconocimiento-No comercial-
Compartir igual 3.0 Unported" de Creative Commons disponible en
<http://creativecommons.org/licenses/by-nc-sa/3.0/>

6.1 Introducción.....	3
6.2 Instalación de WAMP con Wampserver.....	3
6.3 Despliegue en local del WAMP Windows7.....	5
6.3.1 Crear BD y usuario y otorgar permisos.....	5
6.3.2 Ubicar la aplicación.....	6
6.3.3 Asociar la localización con la aplicación.....	6
6.4 Instalación de LAMP con tasksel.....	6
6.4.1 PHPMyAdmin.....	7
6.5 Servicio FTP en Ubuntu (vsftp).....	8
6.5.1 Usuarios y grupos.....	8
6.5.2 Configuración.....	9
ftpd_banner.....	9
max_clients.....	9
max_per_ip.....	9
anonymous_enable.....	9
local_enable.....	9
chroot_local_user.....	10
chroot_list_enable.....	10
local_root.....	10
write_enable.....	10
anon_upload_enable.....	11
anon_mkdir_write_enable.....	11
local_umask.....	11
anon_umask.....	11
anon_max_rate.....	12
local_max_rate.....	12
6.5.3 Registro.....	12
6.5.4 Cliente Filezilla para Windows y para Linux.....	12
6.6 Servicio SSH.....	13
6.6.1 Instalación de SSH en Ubuntu.....	13
6.6.2 Iniciar una sesion SSH desde Ubuntu.....	14
Ejemplo de sesión SSH.....	14
6.6.3 Instalación del cliente SSH en Windows.....	15
6.6.4 scp: copiar ficheros entre equipos.....	15
6.6.5 Aplicaciones gráficas con SSH.....	16
6.7 Despliegue en un servidor remoto.....	17
6.8 Despliegue en un servidor de hosting.....	17

6.1 Introducción

Las siglas **WAMP** hacen referencia al conjunto de software que sirve de base para poner las aplicaciones web a disposición de los usuarios:

- **W Windows**, el sistema operativo.
- **A Apache**, el servidor web.
- **M MySQL** o **MariaDB** (en realidad vale cualquier otro **SGBD**).
- **P PHP** o **Perl** (o cualquier otro lenguaje de script de servidor).

Las siglas **LAMP** hacen referencia al conjunto de software cuando está montado sobre **Linux** (**L**).

Las siglas **XAMP** hacen referencia al conjunto de software con independencia del sistema operativo (**X**) que se use.

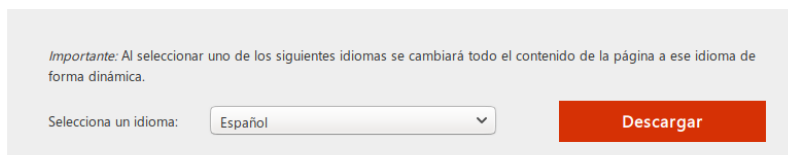
6.2 Instalación de WAMP con Wampserver

Existen muchas versiones y combinaciones para montar **WAMP**. Aquí usaremos un instalador que lo hace todo de una vez: **wampserver**.

Antes de instalar **WAMP** con **wampserver** es necesario descargar e instalar **Visual C++ redistributable para Visual Studio 2012 Update 4**. Esto es necesario para poder ejecutar el instalador de **WAMP** que es una aplicación **C++** creada con **Visual Studio 2012**.

Desde <http://www.microsoft.com/es-ES/download/details.aspx?id=30679>:

Visual C++ Redistributable para Visual Studio 2012 Update 4



Los paquetes redistribuibles de Visual C++ instalan componentes en tiempo de ejecución necesarios para ejecutar aplicaciones de C++ creadas con Visual Studio 2012.

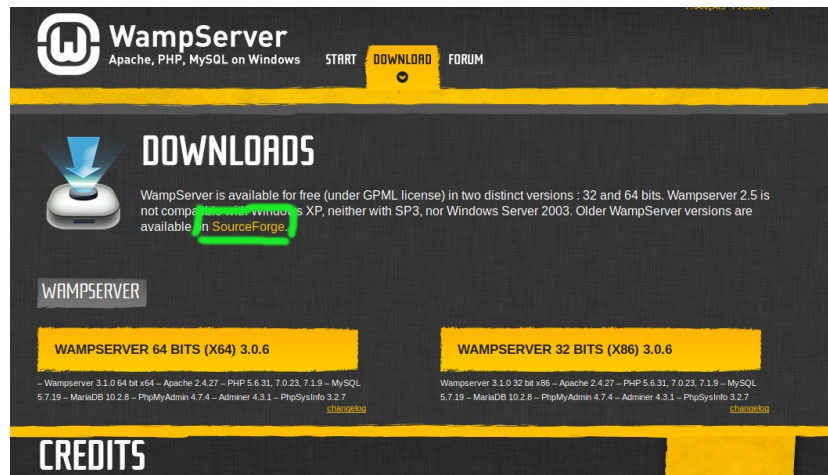
Seleccionamos la versión de 32 bits:

Elige la descarga que quieras

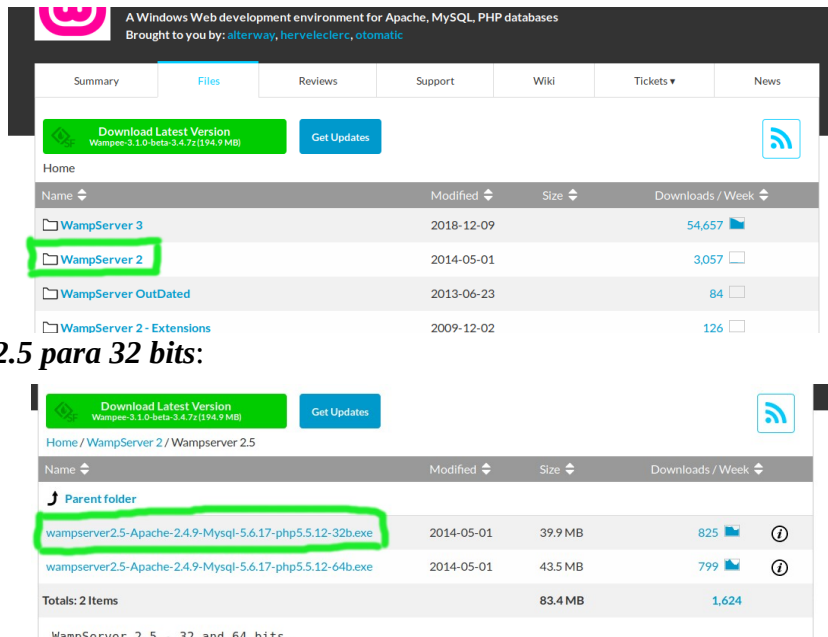
<input type="checkbox"/> Nombre del archivo	Tamaño
<input type="checkbox"/> VSU4\vc_redist_arm.exe	1.4 MB
<input type="checkbox"/> VSU4\vc_redist_x64.exe	6.9 MB
<input checked="" type="checkbox"/> VSU4\vc_redist_x86.exe	6.3 MB

Obtienes un ejecutable que debes instalar previo al **wampserver**. La instalación solo requiere aceptar los términos y esperar pacientemente porque tarda un poco.

Descarga **wampserver** desde <https://www.wampserver.com/#download-wrapper> Ojo! descargamos la **versión 2.5 de 32 bits**, no la última. Para ello selecciona el enlace a **SourceForge** (marca verde en la imagen).




Elegimos **Wampserver 2**:



Y, concretamente, la **versión 2.5 para 32 bits**:

y obtenemos un fichero **.exe**, que debemos ejecutar.

La instalación nos pide aceptar las condiciones y posteriormente pregunta algunos datos como la carpeta donde instalarlo, etc. Presta atención cuando pregunta por el navegador por defecto para elegir el ejecutable que corresponde a tu navegador preferido. También deberías permitir el acceso desde todas las redes.

En la esquina inferior derecha aparecerá el icono de **Wampserver** , en verde si está levantado, en rojo si está caído. Si, al reiniciar, no aparece ese icono, tendrás que iniciarlo con:

INICIO/TODOS LOS PROGRAMAS/WAMPSEVER/START WAMPSEVER

Lo primero que deberías hacer es asignar un editor de textos cómodo al **wampserver**. Por defecto abre el **bloc de notas** y no muestra los ficheros de configuración correctamente por las diferencias en los códigos de fin de línea entre Windows y Linux. Se recomienda instalar gedit para

Windows (el instalador está disponible en la plataforma Moodle del módulo) y establecerlo como editor por defecto, cambiando en `C:\wamp\wampmanager.conf` la línea:

```
editor="notepad.exe"
```

por la línea:

```
editor="ruta\gedit.exe"
```

Naturalmente, habrá que reiniciar el equipo.

6.3 Despliegue en local del WAMP Windows7

El despliegue de una aplicación web en un equipo local es la forma más fácil de despliegue, puesto que tenemos acceso directo al equipo que alojará la aplicación. Hay que hacer tres cosas:

- En el gestor de bases de datos creamos la base de datos y el usuario y otorgamos permisos al usuario sobre esa base de datos.
- Copiamos la aplicación en la carpeta adecuada.
- Establecemos una correspondencia entre una localización y el lugar donde hemos alojado la aplicación para que el servidor pueda localizarla.

6.3.1 Crear BD y usuario y otorgar permisos

Este paso será necesario si nuestra aplicación usa una base de datos, que será casi siempre. La aplicación contará con un nombre de base de datos y un nombre de usuario y contraseña. Lo normal será que la aplicación venga acompañada de un script **SQL** con el que se crean las tablas y quizá se inserten los datos.

Si lo hacemos desde el cliente estándar de **MySQL**, estas son las órdenes necesarias:

```
mysql> CREATE DATABASE nombreDB;  
mysql> CREATE USER nombreUsuario IDENTIFIED BY 'contraseña';  
mysql> GRANT ALL PRIVILEGES ON nombreDB.* TO nombreUsuario;
```

Fíjate en el ejemplo:

```
mysql> CREATE USER peliculas IDENTIFIED BY 'peliculas';  
Query OK, 0 rows affected (0.01 sec)  
  
mysql> CREATE DATABASE peliculas;  
Query OK, 1 row affected (0.00 sec)  
  
mysql> GRANT ALL PRIVILEGES ON peliculas.* to peliculas;  
Query OK, 0 rows affected (0.00 sec)  
  
mysql> 
```

A continuación debemos ejecutar los scripts **SQL** para crear las tablas e insertar los datos.

Nota: Suele ser habitual que el acceso a la base de datos se haga usando **PHPMyAdmin** como cliente, especialmente si el despliegue de la aplicación se hace en un servidor remoto.

6.3.2 Ubicar la aplicación

Las aplicaciones web están compuestas de muchos ficheros y necesitan que esos ficheros estén localizados en sitios concretos para que estén localizables. La distribución de estas aplicaciones se hace en forma de carpeta que contiene subcarpetas, normalmente comprimidas en un solo fichero. Tendremos que ubicar la aplicación en la carpeta adecuada, por ejemplo, si tenemos el servidor que hemos instalado en el apartado anterior con **WampServer**, esa carpeta es:

```
C:/WAMP/APPS
```

6.3.3 Asociar la localización con la aplicación

Sabemos que el usuario final solo puede solicitar **URLs** desde el navegador. Por tanto, necesitamos asociar la aplicación con una **URL** de modo que cuando el usuario solicite la **URL** se ejecute la aplicación.

Eso lo haremos con la directiva **Alias** de **Apache**. Además es necesario configurar las características del acceso a esa localización con la directiva **Directory**. Aquí tenemos un ejemplo:

```
Alias /liga "C:/wamp/apps/liga/"
<Directory "C:/wamp/apps/liga">
    Options Indexes FollowSymLinks
    AllowOverride all
    Require all granted
</Directory>
```

De esta manera cuando el usuario solicite la **URL**:

```
http://nombreServidor.dom/liga
```

se ejecutará la aplicación web.

6.4 Instalación de LAMP con tasksel

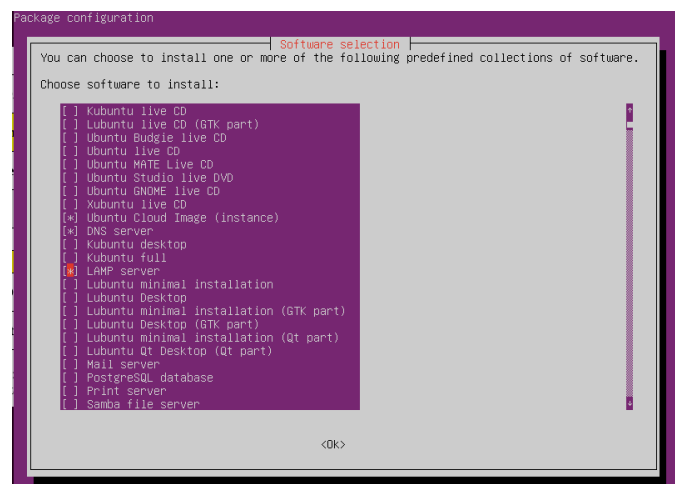
La instalación **LAMP** en **Ubuntu** se puede hacer paquete a paquete o bien, ayudándose del selector de tareas **tasksel** que, mediante un menú nos permite elegir los paquetes a instalar. Probablemente antes que nada tendrás que instalar **tasksel** mediante el paquete del mismo nombre:

```
sudo apt-get update
sudo apt-get install tasksel
sudo tasksel
```

El menú de **tasksel** tiene el siguiente aspecto de la imagen de la derecha.

Aquí, nos movemos por las opciones con la tecla de cursor, marcamos/desmarcamos programas con la barra espaciadora y saltamos al **OK** con la tecla tabulador.

Solamente hay que marcar la opción **Servidor LAMP** sin tocar las otras y dar al **OK**.



En los sistemas **Ubuntu** que ejecutan **MySQL 5.7** (y versiones posteriores), el usuario **root** de **MySQL** está configurado para autenticarse usando el complemento **auth_socket** de forma predeterminada en lugar de con una contraseña. Esto permite mayor seguridad y facilidad de uso en muchos casos, pero también puede complicar las cosas cuando necesita permitir que un programa externo (por ejemplo, **phpMyAdmin**) acceda al usuario.

Para permitir al usuario **root** conectarse a **MySQL** con una contraseña, deberá cambiar su método de autenticación de **auth_socket** a **mysql_native_password**. Para hacer esto, abre el indicador de **MySQL** desde un terminal:

```
sudo mysql
```

y ejecuta, tras el **prompt** de **MySQL**:

```
mysql> ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY 'root';
```

Nota: Observa que proponemos **root** como contraseña para el usuario **root** de **MySQL**. En un sistema en producción deberíamos usar una contraseña segura.

y recargamos la tabla de privilegios para que los cambios tengan efecto:

```
mysql> FLUSH PRIVILEGES;
```

Finalmente, probamos que se puede acceder a **MySQL** con contraseña:

```
mysql -u root -p
```

Tras la instalación de **MySQL**, especialmente en sistemas en producción, podremos configurar los elementos menos seguros con la orden:

```
sudo mysql_secure_installation
```

Nos hará una serie de preguntas a las que respondemos si o no. Especialmente interesante es la primera de ellas que pregunta si queremos instalar el **plugin Validate Password**. Este plugin obliga a usar contraseñas seguras.

Si en algún momento queremos anular la obligatoriedad de contraseñas seguras podemos ejecutar, desde dentro de **MySQL**:

```
mysql> uninstall plugin validate_password;
```

6.4.1 **PHPMyAdmin**

PHPMyAdmin es una aplicación web que hace de cliente **MySQL** proporcionando un entorno gráfico para manejar este gestor de bases de datos.

La instalación en **Ubuntu** solo requiere del paquete del mismo nombre:

```
sudo apt-get update  
sudo apt-get install phpmyadmin
```

Durante la instalación te preguntará por el servidor web que quieres usar (se trata de una aplicación web). Elegimos **apache**.

Como contraseña del usuario **phpmyadmin** usaremos **root** en nuestro entorno de aprendizaje.

6.5 Servicio FTP en Ubuntu (vsftpd)

Para instalar un **servicio FTP** en **Ubuntu** sólo tienes que bajar el paquete **vsftpd** (Very Secure FTP Daemon, demonio FTP muy seguro). Se trata del servidor **FTP** más extendido en distribuciones **Linux**.

```
sudo apt-get update
sudo apt-get install vsftpd
```

Al instalar el servidor se crean el usuario **ftp** y el grupo **ftp**, así como la carpeta en **/srv/ftp** (que será propiedad de **root:ftp**) que es la que queda accesible a los clientes **FTP** que se conecten.

También se genera el fichero de configuración **/etc/vsftpd.conf**.

Para iniciar, parar o comprobar el estado del servicio:

```
/etc/init.d/vsftpd start/stop/status
```

o bien:

```
service vsftpd start/stop/status
```

MUY IMPORTANTE: Si tienes varios terminales abiertos, la parada y reinicio del servicio tienes que hacerla **siempre desde el mismo terminal**. En caso contrario, los cambios de configuración no tendrán efecto.

Puedes consultar la documentación de este servicio en <http://vsftpd.beasts.org>.

6.5.1 Usuarios y grupos

El control de los usuarios de un servicio **FTP** es una tarea muy importante. Existen dos tipos de usuarios **FTP**: **anónimos** y **autenticados**.

- Los **usuarios anónimos** no se identifican por lo que pueden ser un peligro potencial. Para conectarse deben usar el nombre **anonymous** sin contraseña. Si el servidor **FTP** sólo sirve para uso interno de una organización no debería de haber usuarios anónimos (**Ojo! porque por defecto están habilitados**). En el caso de que queramos difundir ficheros de forma general tendremos que permitirlos, pero restringiendo su acceso exclusivamente a esa información pública. En el caso de que se permitan los usuarios anónimos, estos sólo tendrán acceso a la carpeta **/srv/ftp** y a sus subcarpetas.

Nota: Es posible que en versiones posteriores a esta documentación, los usuarios anónimos estén deshabilitados por defecto.

Nota Importante: La carpeta **/srv/ftp** tendrá que ser propiedad de **root** y el grupo **ftp**, pues de lo contrario el acceso de los usuarios anónimos devolverá un error. Para hacer que la carpeta sea propiedad de **root**, desde un terminal teclea el siguiente comando: **chown root:ftp /srv/ftp**

- Los **usuarios autenticados** disponen de un nombre y contraseña, por lo que podemos saber de quién se trata y establecer los permisos que le corresponden. Pueden ser de dos tipos:
 - ➡ Los **usuarios del sistema**, es decir, los usuarios de **Ubuntu** que ya existen en el servidor **FTP** y que acceden con su propio nombre y contraseña. Para estos usuarios podemos decidir si acceden a todas las carpetas del sistema o si, por el contrario, los enjaulamos en una carpeta.

Nota: Desde un terminal puedes crear más usuarios del sistema con el siguiente comando: `adduser --home /home/usuario usuario` (Sustituyendo la parte roja por el nombre del usuario que quieres crear. Se creará el usuario y su carpeta personal).

Nota: Con el comando: `userdel usuario` se borra el usuario (pero no su carpeta, que habrá que borrarla manualmente).

🔹 Los **usuarios virtuales**, que son usuarios exclusivamente del servicio **FTP**.

También podemos crear **grupos de usuarios** con objeto de asignar permisos de forma colectiva. Imagina un servicio **FTP** disponible para todo el alumnado del instituto. Si en algún momento hay que cambiar un permiso para todos ellos, sólo hay que cambiarlo para el grupo que los incluye a todos (por ejemplo, para el grupo profesores o para el grupo alumnado). Sin grupos, esos cambios serían inviables.

6.5.2 Configuración

La configuración general del servidor se establece en el fichero:

```
/etc/vsftpd.conf
```

A continuación se exponen algunas de las directivas que puedes usar en ese fichero:

FTPD_BANNER

Para establecer el mensaje de bienvenida.

```
ftpd_banner="Bienvenido al servidor FTP del IES Murgi."
```

MAX_CLIENTS

Para establecer el número máximo de conexiones admitidas.

```
max_clients=3
```

MAX_PER_IP

Para limitar el número máximo de conexiones admitidas desde una misma **IP**.

```
max_per_ip=2
```

ANONYMOUS_ENABLE

Admite los valores **YES** o **NO**. Para habilitar o negar los accesos anónimos.

```
anonymous_enable=NO
```

Si se habilitan los accesos anónimos, deberán entrar con el nombre **anonymous**, sin contraseña y su acceso estará restringido a la carpeta **/srv/ftp** (y subcarpetas). Esta carpeta se crea automáticamente al instalar el **servidor FTP**.

LOCAL_ENABLE

Admite los valores **YES** o **NO**. Para habilitar o negar el acceso a los usuarios del sistema.

```
local_enable=YES
```

Si se habilita el acceso a los usuarios del sistema, deberán entrar con su propio nombre y contraseña. Inicialmente acceden a su propia carpeta personal, pero **Ojo!** porque por defecto **tendrán acceso a todo el sistema y eso supone un problema de seguridad**.

CHROOT_LOCAL_USER

Admite los valores **YES** o **NO**. "**Enjaula**" a los usuarios del sistema en su carpeta personal.

```
chroot_local_user=YES
```

Nota: Con valor **YES** restringimos el acceso de los usuarios del sistema exclusivamente a su carpeta personal, pero por defecto y por motivos de seguridad **vsftpd** niega el acceso de un usuario a una carpeta donde tenga permiso de escritura. Para anular esto tendremos que añadir la directiva:

```
allow_writeable_chroot=YES
```

CHROOT_LIST_ENABLE

Admite los valores **YES** o **NO**.

```
chroot_list_enable=YES
```

Si **chroot_local_user** tiene valor **YES**, poner **YES** en **chroot_list_enable** desenjaula a los usuarios del sistema listados en el fichero **/etc/vsftpd.chroot_list**, es decir, los deja ir a todos lados.

Si **chroot_local_user** tiene valor **NO**, poner **YES** en **chroot_list_enable** enjaula sólo a los usuarios del sistema listados en el fichero **/etc/vsftpd.chroot_list**, es decir, los usuarios **NO LISTADOS** podrán ir a todos lados.

chroot_list_enable	chroot_local_user	
	YES	NO
YES	Los de la lista: son libres El resto: enjaulados en su carpeta	Los de la lista: enjaulados en su carpeta El resto: libres
NO	Todos enjaulados	Todos libres

Nota: La directiva **chroot_list_file=ruta/fichero** permite usar otra ubicación o nombre de fichero para la lista de usuarios, que por defecto es **/etc/vsftpd.chroot_list**.

Este fichero hay que crearlo explícitamente escribiendo un usuario por línea.

LOCAL_ROOT

Para modificar la carpeta a la que acceden los usuarios del sistema, todos a la misma. Por defecto, cada usuario accede a su propia carpeta personal.

```
local_root=ruta/carpetaComun
```

WRITE_ENABLE

Admite los valores **YES** o **NO**. Para habilitar o negar la escritura en el servidor.

```
write_enable=NO
```

Con esta directiva activada, los usuarios del sistema podrán subir ficheros siempre y cuando tengan permiso de escritura en esa carpeta. Los usuarios anónimos necesitan además que la directiva **anon_upload_enable** esté habilitada (y que el usuario **ftp** tenga permiso de escritura en esa carpeta).

ANON_UPLOAD_ENABLE

Admite los valores **YES** o **NO**. Para habilitar o negar que los usuarios anónimos puedan subir contenido al servidor.

```
anon_upload_enable=NO
```

Con esta directiva activada, los usuarios anónimos podrán subir ficheros si se dan otras dos condiciones: que la directiva **write_enable** esté activada y que la carpeta a la que se pretende subir el fichero tenga permiso de escritura para el usuario **ftp**.

Nota: El usuario **ftp** es un usuario que se crea automáticamente al instalar el servicio **FTP** y que representa a los usuarios anónimos que acceden a través del servicio.

ANON_MKDIR_WRITE_ENABLE

Admite los valores **YES** o **NO**. Para habilitar o negar que el usuario anónimo pueda crear carpetas en el servidor.

```
anon_mkdir_write_enable=NO
```

Con esta directiva activada, los usuarios anónimos podrán crear carpetas si se dan otras dos condiciones: que la directiva **write_enable** esté activada y que la carpeta a la que se pretende subir el fichero tenga permiso de escritura para el usuario **ftp**.

LOCAL_UMASK

Para establecer los permisos con los que quedarán los ficheros que suban al servidor los **usuarios locales** (usuarios del sistema). Los permisos se establecen **en base octal y de forma negada**, es decir, el número corresponde a los permisos que se niegan y no a los que se conceden. Para calcular los permisos con los que se crea una carpeta hay que restar (**777-máscara**) y para calcular los permisos con los que se sube un fichero hay que restar (**666-máscara**).

```
local_umask=022
```

En el anterior ejemplo, los ficheros que se suban al servidor quedarán con los permisos:

666-022 = 644 = *rw-r--r--* (lectura y escritura para el propietario, sólo lectura para el resto)

y las carpetas que se creen quedarán con los permisos:

777-022 = 755 = *rwxr-xr-x* (todos los permisos para el propietario, lectura y entrada el resto)

Nota: La nomenclatura usada en esta directiva se corresponde con los permisos de acceso a ficheros **rwrxrwxrwx** que se usan en todos los Sistemas Operativos derivados de **Unix**.

Muy Importante: La máscara siempre debe comenzar por cero para que se considere un número octal, así que si se desea poner una máscara **244** habrá que escribir **local_umask=0244**

ANON_UMASK

Para establecer los permisos con los que quedarán los ficheros que suban al servidor los **usuarios anónimos**. Esta directiva funciona exactamente igual que **local_umask**.

ANON_MAX_RATE

Para limitar la tasa de transferencia de los usuarios anónimos. En este ejemplo se establece en 10Kb:

```
anon_max_rate=10240
```

LOCAL_MAX_RATE

Para limitar la tasa de transferencia de los usuarios autenticados. En este ejemplo se establece en 10Kb:

```
local_max_rate=10240
```

6.5.3 Registro

El servicio **vsftpd** guarda los eventos en el fichero:

```
/var/log/vsftpd.log
```

Este fichero te será muy útil para analizar los problemas que puedan surgir.

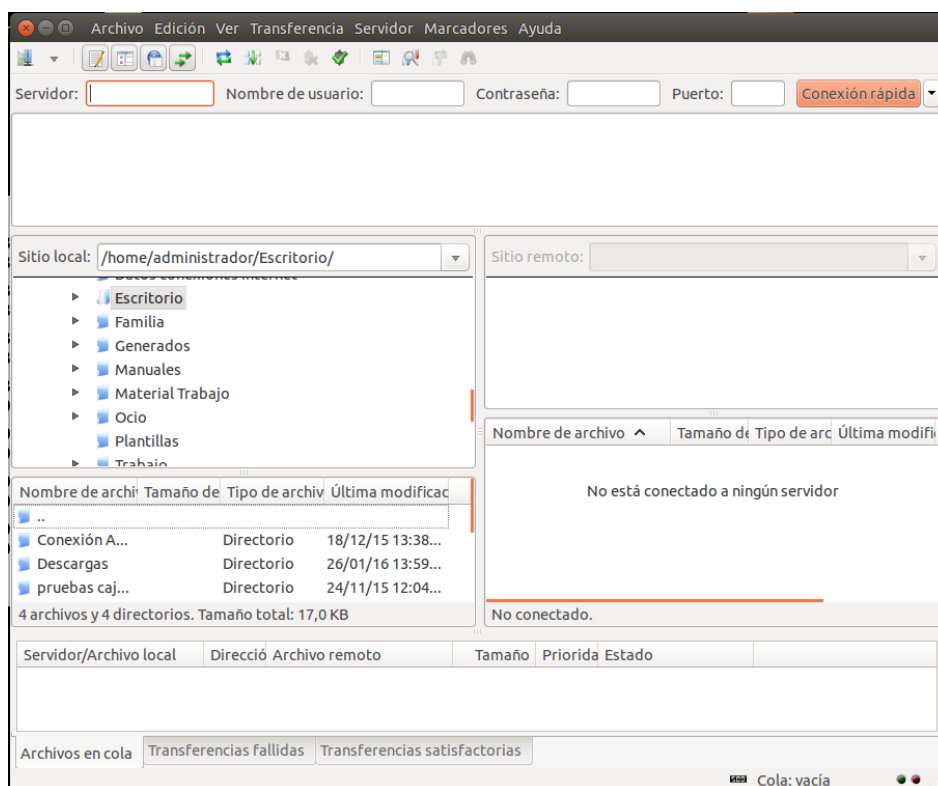
6.5.4 Cliente Filezilla para Windows y para Linux

Hay muchos clientes **FTP**. Podemos destacar **Filezilla** que está disponible tanto para **Windows** (descargándolo de la página oficial) como para **Ubuntu** (instalando el paquete llamado **filezilla**) y también el **gFTP** para **Ubuntu** (instalando el paquete **gftp**).

Con cualquiera de ellos tendremos una ventana dividida en dos partes: la parte de la izquierda representa el equipo local (**cliente FTP**) y la parte derecha representa el equipo remoto (**servidor FTP**).

La siguiente imagen muestra el aspecto de **Filezilla**:

Para acceder a un servidor primero debemos ir a:

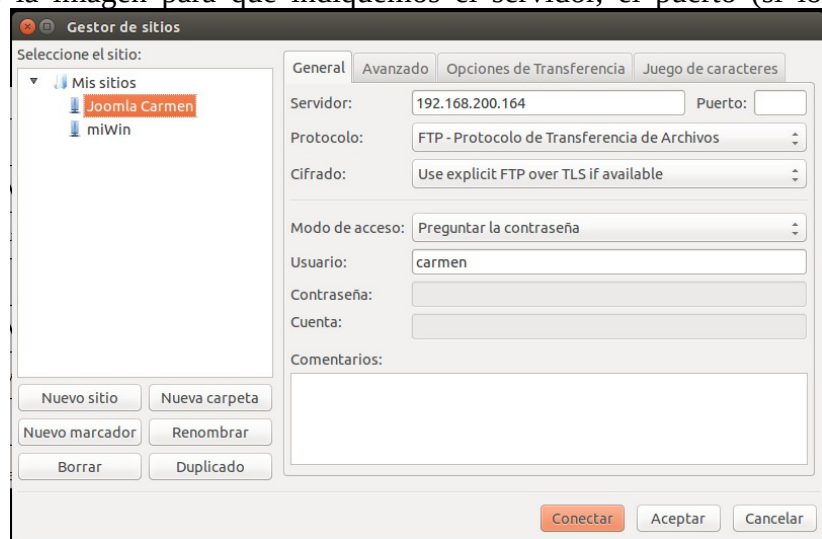


```
Archivo/Gestor de sitios
```

donde aparece la pantalla de la imagen para que indiquemos el servidor, el puerto (si lo dejamos en blanco usará el puerto **21** por defecto), el nombre de usuario y la contraseña.

Tras pulsar conectar, si todo es correcto, en la parte derecha de la parte principal podremos ver las carpetas y ficheros del servidor.

Para bajar o subir ficheros basta con arrastrarlos a la izquierda o derecha respectivamente.



6.6 Servicio SSH

SSH (Secure Shell, terminal seguro) es un protocolo muy similar al **Telnet**. La principal diferencia entre ambos es que **SSH** usa técnicas de cifrado que hacen ilegible la información que viaja entre cliente y servidor. Además **SSH** usa el puerto **22**. También la herramienta que implementa el protocolo se llama **SSH**.

6.6.1 Instalación de SSH en Ubuntu

En **Ubuntu** podemos elegir entre dos implementaciones de **SSH**. Una de ellas necesita que descargues el paquete **ssh** (este paquete incluye tanto el **cliente SSH** como el **servidor SSH**):

```
sudo apt-get update
apt-get install ssh
```

La otra implementación dispone de dos paquetes distintos. Si vamos a usar el cliente descargamos **openssh-client** y si vamos a usar el servidor descargamos **openssh-server**. Naturalmente, si en el mismo equipo necesitamos cliente y servidor, bajamos ambos (aunque por defecto **Ubuntu** ya incluye el cliente):

```
sudo apt-get update
apt-get install openssh-client
apt-get install openssh-server
```

Nota: Puesto que se trata de dos implementaciones distintas del mismo protocolo no es recomendable instalar ambas (**ssh** y **openssh**) al mismo tiempo.

Al instalar el servidor **ssh** se crean los archivos de configuración y se generan dos parejas de claves **RSA**, **DSA** y **ECDSA** en **/etc/ssh/**. Con el comando:

```
ssh-keygen -l -f /etc/ssh/ssh_host_rsa_key
```

obtenemos la huella de la clave, lo que nos sirve para comprobarlo en las conexiones y evitar una suplantación de identidad.

Cuando se hace una conexión a un servidor **ssh** se solicita la aceptación del usuario. Al aceptar, el cliente guarda el **fingerprint** (huella) y no pedirá confirmación en siguientes conexiones.

6.6.2 Iniciar una sesión SSH desde Ubuntu

Para iniciar la comunicación, desde un terminal del equipo cliente:

```
ssh nombre_equipo_remoto
```

O

```
ssh direccionIP_equipo_remoto
```

Si lo hacemos así, el sistema entiende que el nombre de usuario del equipo remoto es el mismo que el nombre de usuario que estamos usando en el equipo local. Por ejemplo, si tecleo esta orden desde un equipo en el que el usuario se llama **admin**, estoy pidiendo conectarme como usuario **admin** del equipo remoto. Naturalmente, en el equipo remoto debe existir ese tal usuario **admin**.

Si la comunicación es posible, lo primero que se nos pedirá será la contraseña de ese usuario en el equipo remoto.

Nota: La primera vez que nos conectamos se nos pedirá confirmar que queremos establecer conexión. Debemos contestar **yes** con todas las letras.

Para conectarse como un usuario distinto del equipo remoto, la orden será:

```
ssh usuarioRemoto@equipoRemoto
```

Por ejemplo, para iniciar una sesión como **root** en el equipo **192.168.10.10**:

```
ssh root@192.168.10.10
```

EJEMPLO DE SESIÓN SSH

```
administrador@Laptop:~$ ssh 192.168.0.111
The authenticity of host '192.168.0.111 (192.168.0.111)' can't be established.
ECDSA key fingerprint is SHA256:GhSJ6pIS5tBWJMIVAZbztawbQL+8BctNNjdRuqIQTlw.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.0.111' (ECDSA) to the list of known hosts.
administrador@192.168.0.111's password:
Welcome to Ubuntu 12.04.5 LTS (GNU/Linux 3.2.0-91-generic-pae i686)

 * Documentation:  https://help.ubuntu.com/

Está disponible la nueva versión «14.04.1 LTS».
Ejecute «do-release-upgrade» para actualizarse a ella.

*** Es necesario reiniciar el sistema ***
Last login: Mon Jan  4 13:14:30 2016 from 192.168.0.109
administrador@virtual-VirtualBox:~$ ls /
bin    dev    initrd.img  libnss3.so  mnt    root  selinux  usr    vmlinuz
boot  etc    initrd.img.old  lost+found  opt    run   srv      var    vmlinuz.old
cdrom  home  lib         media       proc   sbin  sys      webmin-setup.out
administrador@virtual-VirtualBox:~$ exit
logout
Connection to 192.168.0.111 closed.
administrador@Laptop:~$ ssh root@192.168.0.111
root@192.168.0.111's password:
Welcome to Ubuntu 12.04.5 LTS (GNU/Linux 3.2.0-91-generic-pae i686)

 * Documentation:  https://help.ubuntu.com/

Está disponible la nueva versión «14.04.1 LTS».
Ejecute «do-release-upgrade» para actualizarse a ella.

*** Es necesario reiniciar el sistema ***
root@virtual-VirtualBox:~# ls /
bin    dev    initrd.img  libnss3.so  mnt    root  selinux  usr    vmlinuz
boot  etc    initrd.img.old  lost+found  opt    run   srv      var    vmlinuz.old
cdrom  home  lib         media       proc   sbin  sys      webmin-setup.out
root@virtual-VirtualBox:~# exit
logout
Connection to 192.168.0.111 closed.
administrador@Laptop:~$
```

En la imagen de la izquierda tenemos dos sesiones **SSH** sobre el mismo equipo.

En la primera de ellas no indicamos usuario (sólo **IP** del host remoto):

```
ssh 192.168.0.111
```

y se asume por defecto que el nombre del usuario remoto será el mismo que el nombre del usuario local, en este caso **administrador**. Tras escribir la contraseña del usuario administrador del equipo remoto accedemos, ejecutamos el comando **ls**, y salimos con **exit**.

En la segunda conexión indicamos el usuario **root** junto a la dirección **IP** del host remoto:

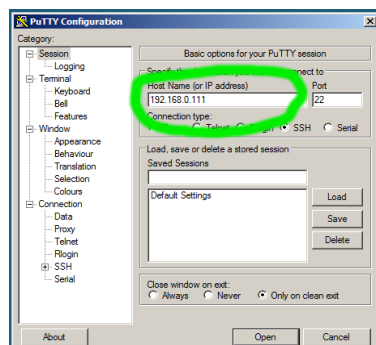
```
ssh root@192.168.0.111
```

Escribimos la contraseña del **root**

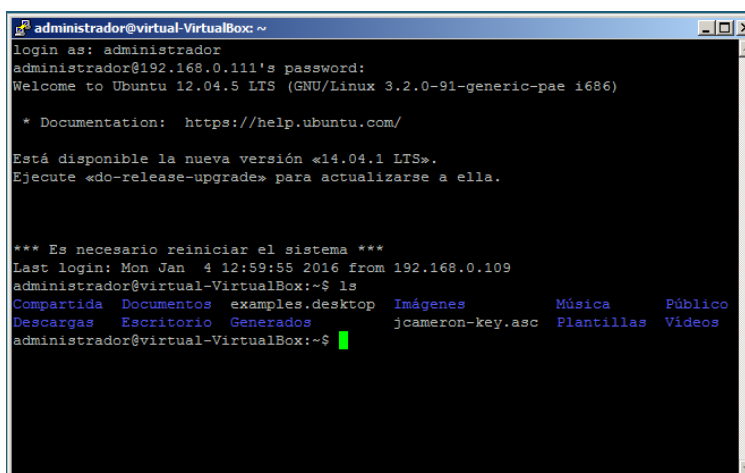
del equipo remoto y accedemos.

6.6.3 Instalación del cliente SSH en Windows

Al contrario que **Ubuntu**, **Windows** no incluye un cliente **SSH**, así que nos bajaremos **PuTTY** que es un cliente **SSH** (con algunas funcionalidades más) de código abierto para **Windows**. Puedes descargarlo desde <http://www.putty.org/>.



Al ejecutar **PuTTY** aparece una ventana como la que ves abajo donde podemos establecer todos los parámetros de conexión. Te bastará con introducir la dirección IP (o el nombre) del equipo remoto y aceptar las advertencias sobre problemas de seguridad.



A continuación se abre un terminal donde solicita nombre de usuario y contraseña (del equipo remoto naturalmente) y se establecerá la conexión.

En la imagen de la derecha aparece un ejemplo de conexión **SSH** desde un equipo **Windows** con **PuTTY** a un equipo **Ubuntu**. Una vez dentro se ejecuta el comando **ls** para listar el contenido de la carpeta actual.

6.6.4 scp: copiar ficheros entre equipos

Ya conocemos el comando **cp** que permite hacer una copia de un fichero:

```
cp ruta_origen/fichero_origen ruta_destino/fichero_destino
```

Este comando hace una copia del **fichero_origen** que se encuentra en **ruta_origen** y la coloca en el **fichero_destino**, dentro de la carpeta **ruta_destino**.

El comando **scp** (**secure cp**, copia segura) va un paso más allá y permite hacer copias de ficheros de un equipo a otro. Necesitamos que el equipo donde lo usamos tenga instalado el **cliente SSH** y que el equipo remoto tenga instalado el **servidor SSH**. Cuando se da esta situación podemos usar el comando **scp** con el mismo formato que el comando **cp**, teniendo en cuenta que si el origen o el destino es remoto habrá que anteponerle la dirección **IP** (o nombre) del equipo y probablemente el nombre del usuario del equipo remoto que puede acceder al fichero.

Veamos algunos ejemplos:

- Copiar el fichero **pruebas.txt** que se encuentra en la carpeta personal del usuario **administrador** a la carpeta **/copias** del equipo remoto:

```
scp /home/administrador/pruebas.txt root@192.168.0.111://copias/pruebas.txt
```

```
administrador@Laptop:~$ scp /home/administrador/pruebas.txt root@192.168.0.111://copias/pruebas.txt
root@192.168.0.111's password:
pruebas.txt                                100%  22    0.0KB/s   00:00
administrador@Laptop:~$
```

En este ejemplo, para acceder al equipo remoto hacemos uso del usuario **root** (suponemos que es el único que puede acceder a la carpeta **/copias**), por tanto, la contraseña que nos solicita es la contraseña del usuario **root** del equipo remoto.

- Copiar el fichero **httpd.conf** que se encuentra en la carpeta **/etc/apache2** del equipo remoto en la carpeta personal del usuario **administrador** del equipo local:

```
scp root@192.168.0.111:/etc/apache2/httpd.conf /home/administrador/httpd.conf
```

```
administrador@Laptop:~$ scp root@192.168.0.111:/etc/apache2/httpd.conf /home/administrador/httpd.conf
root@192.168.0.111's password:
httpd.conf
administrador@Laptop:~$
```

6.6.5 Aplicaciones gráficas con SSH

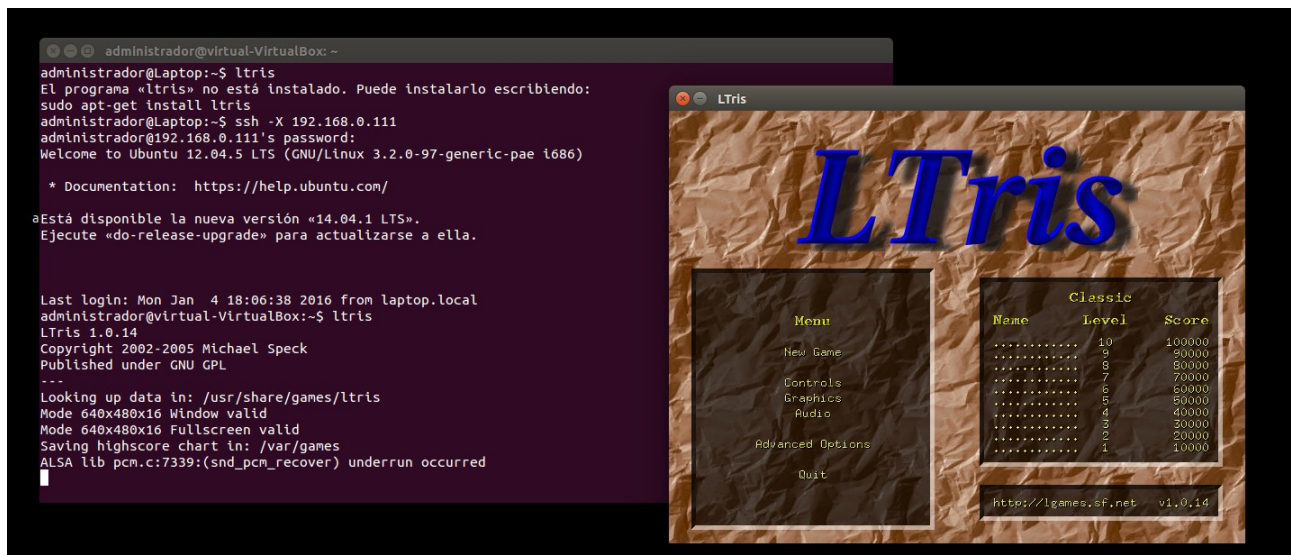
Tal y como hemos visto, con **SSH** podemos manejar un equipo remoto desde una consola de comandos. **SSH** también permite ejecutar aplicaciones gráficas del equipo remoto. Para ello tenemos que tener en cuenta dos cosas:

- El **servidor SSH** debe estar configurado para aceptar el uso de aplicaciones gráficas. Eso se hace estableciendo la directiva **X11Forwarding** con el valor **yes**. El fichero de configuración del servidor **ssh** está en **/etc/ssh/sshd_config**.
- Desde el **cliente SSH** se debe iniciar la sesión con el parámetro **-X**, que indica que se hará uso de aplicaciones gráficas:

```
ssh -X equipo_remoto
```

En el ejemplo que muestra la imagen ejecutamos **Ltris** que es un juego derivado del famoso **Tetris**. El juego no está instalado en nuestro equipo, sino en el equipo remoto.

Si te fijas en el terminal, lo primero que hacemos es intentar ejecutar **Ltris** y recibimos el mensaje de que **el programa no está instalado en nuestro equipo**. A continuación establecemos una conexión **SSH** con el equipo remoto usando el parámetro **-X** que permite usar sus aplicaciones gráficas y una vez conectado volvemos a ejecutar **Ltris**, con lo que aparece la ventana del juego en el escritorio del equipo local.



Nota: No se debe confundir la conexión que permite ejecutar aplicaciones gráficas del equipo remoto (que acabamos de ver) con una conexión de escritorio remoto.

6.7 Despliegue en un servidor remoto

El despliegue en un servidor remoto es igual que un despliegue en local, con la diferencia de que, en este caso, puesto que no tenemos acceso directo a la consola del servidor, tendremos forzosamente que usar las herramientas que hemos visto en apartados anteriores.

Para la gestión de la base de datos en remoto usamos la aplicación web **PHPMYAdmin**.

Para la copia de archivos en las carpetas adecuadas, tendremos que hacer uso del servicio **FTP** o del servicio **SSH**.

6.8 Despliegue en un servidor de hosting

Cuando la aplicación web tenga un propietario que no pueda o no quiera mantener su propio servidor, se puede contratar el alojamiento (**hosting**) del sitio.

Se trata de obtener espacio en el disco de un servidor que, además nos ofrecerá todas las herramientas necesarias para gestionarlo en remoto. Los precios de este alojamiento depende del espacio que reservemos y de los servicios que ofrezca.

También hay servicios de **hosting** gratuito que, generalmente, está acompañado de servidumbres como publicidad en nuestra web.

Los servicios de **hosting** disponen de aplicaciones web para toda la gestión del sitio. Podremos usar **PHPMYAdmin** y **FTP** entre otras muchos.