Sistemas Multimedia Basados en Internet

Especificación y configuración del Back-End

Programming Cloud

(Braineering)

Hito 1

Ututui, George Valentin

Candela Ibáñez, Antonio

Index

Para el Back-End utilizamos un entorno LAMP. Estas son las siglas con las que se identifica a una estructura se servidor para alojar aplicaciones web. Dicho nombre viene de la unión de Linux, Apache, MySQL y PHP que son los componentes correspondientes al sistema operativo, el servidor web, el servidor de las bases de datos y el intérprete de scripts en PHP. La elección de este entrono no ha sido al azar sino por la popularidad del mismo. A día de hoy la mayoría de los servidores que alojan aplicaciones web funcionan utilizando los componentes que hemos mencionado.

A continuación, vamos a mencionar los distintos parámetros, mecanismos y configuraciones que hemos aplicado a cada uno de estos componentes para que se ajusten a nuestras necesidades. Nuestro proyecto consiste en una aplicación web de cursos de programación dedicada a institutos de secundaria en donde interactúan profesores, alumnos e internautas anónimos.

1. Instalación (Apache, MySQL y PHP)

Apache es un servidor web HTTP que es muy configurable ya que permite bases de datos de autenticación y también negociado de contenido. Para la instalación del mismo hemos utilizado el comando: *sudo apt install apache2.*

MySQL es la base de datos relacional open source más utilizada y conocida del mundo que se utiliza sobre todo para entornos web, como es nuestro caso. Para la instalación del MySQL hemos utilizado el comando: *sudo apt install mysql_server.*

1.1 MySQL Tuner

MySQLTuner es un script que nos permite revisar la instalación de MySQL y hacer los ajustes que consideremos necesarios para aumentar el rendimiento y la escalabilidad. La configuración actual y los datos del estado se presentan en un formato breve y esquemático con sugerencias al hacer el comando *mysqltuner.*

En términos de Métricas de Rendimiento el Tuner nos propone habilitar la Cache de nuestra MySQL. Una de las configuraciones que hemos deshabilitado es la Caché, ya que hemos considerado que nuestro proyecto no va a necesitar esta característica.

Para mejores recomendaciones de este Script hemos dejado

en funcionamiento MySQL para ver cómo se comporta y funciona en base a nuestras exigencias de este sistema de gestión de bases de datos.

1.2 PHP

Es un lenguaje muy conocido para el desarrollo web que tiene la característica de poderse incrustar en HTML. En la versión 16 de Ubuntu se emplea PHP7. Para instalar PHP en Ubuntu hemos utilizado el comando:

:# apt-get install php.

Lo siguiente fue instalar el módulo para crear el vínculo entre Apache y PHP:

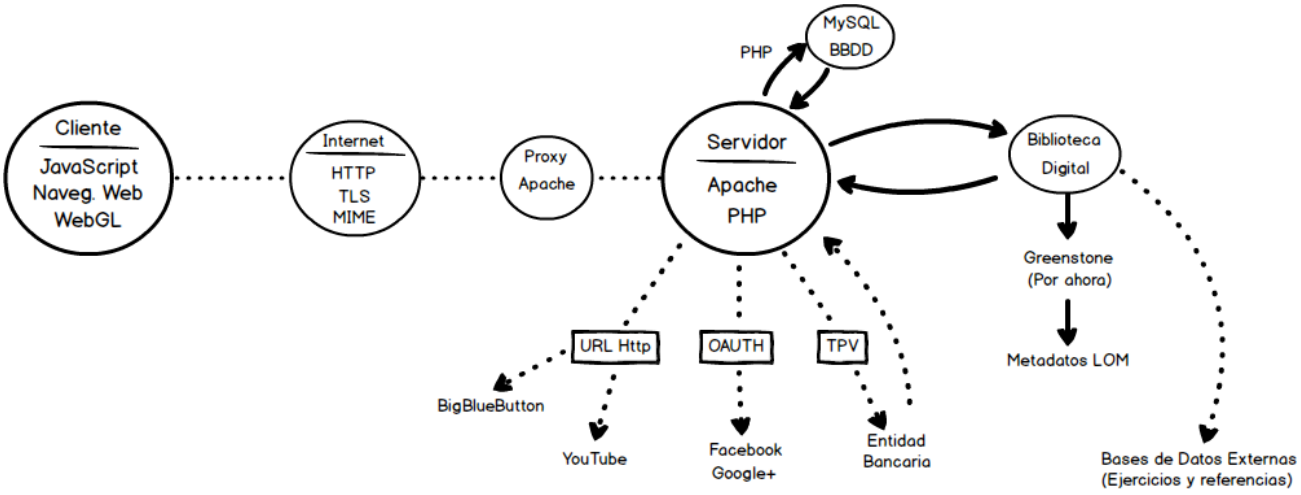
```
:# apt-get install libapache2-mod-php5.
```

2. Especificación del sistema

En los documentos de la asignatura de Proyectos Multimedia se ven reflejados los elementos y las tecnologías de seguridad y autentificación que utilizaremos para la creación de nuestro sistema. También se especifican todos los elementos del back-end, las tecnologías y como se relacionan cada uno de estos componentes entre sí. A continuación, vamos a presentar los elementos más importantes para asentar las bases de nuestro entorno Back-End.

2.1 Arquitectura del sistema

Se puede observar como los dos principales 'actores' en este esquema son el Cliente y el Servidor. Con el servidor se relacionan la mayoría de los sistemas y tecnologías para ofrecer al Cliente el servicio que estamos intentando ofrecer.



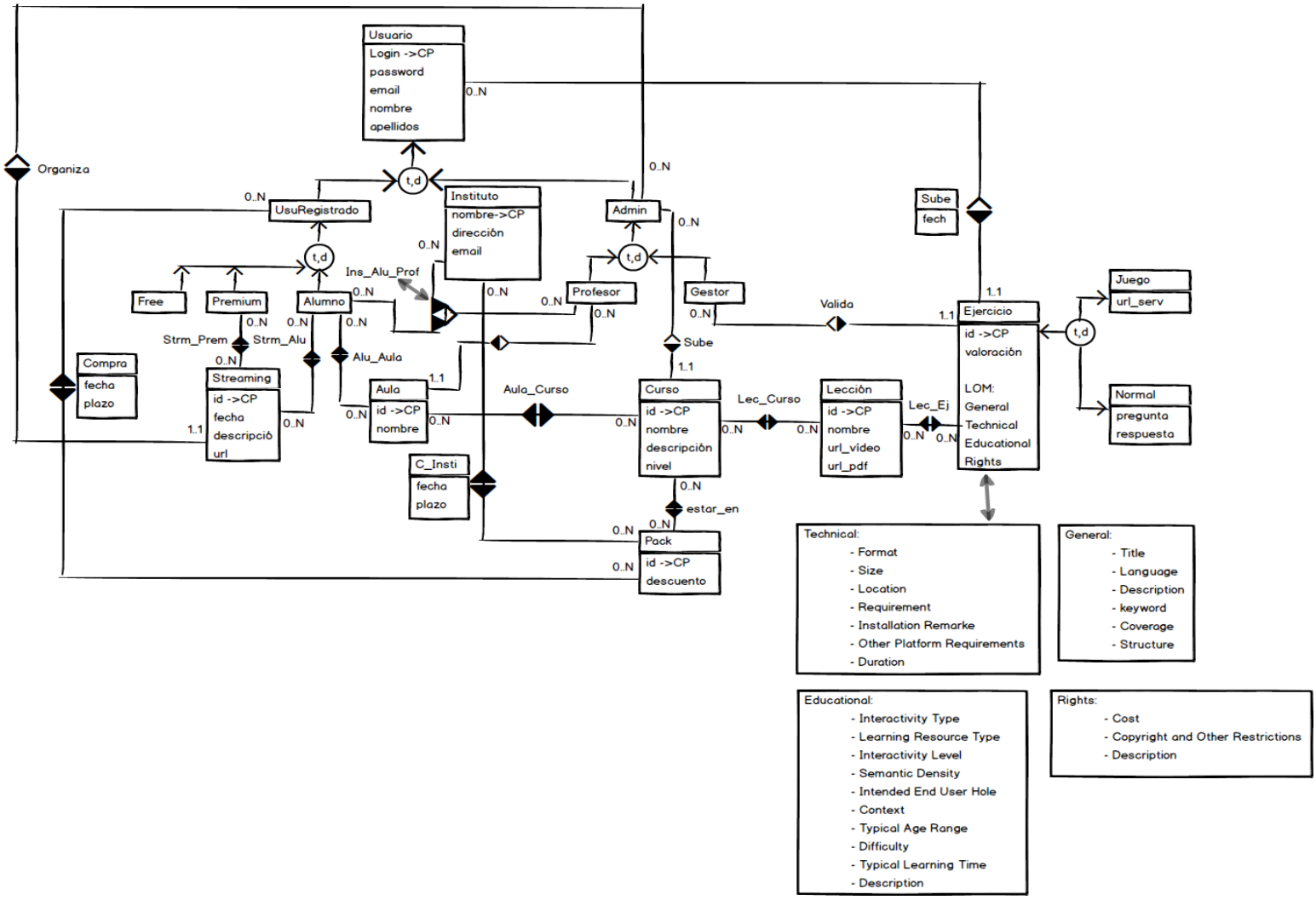
2.2 Esquema ER

Mediante el modelo Entidad-Relación podemos representar las entidades más importantes que conforman nuestro sistema junto con sus relaciones y propiedades.

Aclaraciones del modelo Entidad Relación:

1. Una lección debe tener al menos un curso, pero en el modelo relacional no se puede representar una relación 0..N-1..N, cumpliremos está restricción internamente mediante código.

2. También especificamos mediante código el hecho de que un usuario Premium no puede acceder a los streaming de un profesor.
3. La entidad 'Instituto' también representa academias, por eso un profesor puede estar relacionado con un alumno desde su academia y su instituto en el caso de permanecer ambos a la vez.
4. El LOM representado en el diagrama serán los metadatos que debemos incluir al subir los ejercicios a nuestra biblioteca digital. Esto lo haremos para cumplir es estándar SCROM.
5. Cabe destacar que para catalogar nuestro material educativo vamos a seguir el estándar IEEE LOM



3. Apache

Los principales ficheros que hemos editado son los siguientes: /etc/apache2/apache2.conf, /etc/apache2/mods-enabled/ , etc/apache2/sites-enabled/ y etc/apache2/conf.d/ .

- **Apache2.conf** . Es el fichero de configuración principal de Apache. En él se definen directivas básicas de log, configuración de conexiones máximas y otros. Hay que decir que también se hacen varios include a otros ficheros de configuración.

- **/../mods-enabled/** . En este directorio se encuentran las referencias a los módulos habilitados para Apache. Existen ficheros con terminaciones *.load* y otros con *.conf* . Los *.load* incluyen sentencias con la directiva LoadModule que cargara el módulo que le hemos especificado a Apache. Los ficheros *.conf* sirven para modificar parámetros de la configuración de cada uno de los módulos. Nosotros utilizamos el módulo **Prefork** ya que es el que viene por defecto y nos proporciona una actualización conforme a nuestro proyecto.

- **/../sites-enabled/** . Permite segmentar la configuración por sitios. Cada fichero tiene las directrices de configuración de Apache como VirtualHost, Directory, Allow, etc. Al igual que en los casos de los módulos los dicheros son enlaces simbólicos hacia ficheros contenidos en el directorio /etc/apache2/sites-available. Para habilitar y deshabilitar configuración de estos sitios utilizamos la aplicación **a2ensite.**

- **/../conf.d** . Se dispone de otro directorio donde encontramos otros tipos de configuraciones. El fichero que nosotros editamos dentro de este directorio es el llamado **security.**

3.1 Configuración

Vamos a cambiar configuraciones que afectarán el comportamiento global de Apache y se van a aplicar en los diferentes dicheros que hemos mencionado anteriormente. También hemos hecho uso de la herramienta **a2ensite** para deshabilitar ciertos módulos.

- En primer lugar, hemos hecho el directorio ProgrammingCloud en donde estará toda la información de nuestra web.

- Hemos creado el virtual host para nuestro dominio mediante el comando
  :# sudo cp /../../000-default.conf /../../ProgrammingCloud.conf

- Dentro del archivo **ProgrammingCloud.conf** hemos cambiado las variables de **SrverName**, **DocumentRoot** y **ServerAlias**.

- Hemos habilitado los nuevos Virtual Host mediante: sudo a2ensite **ProgrammingCloud.conf**

3.1.1 Apache2.conf

- Hemos cambiado los valores **Timeout**, **MaxKeepAliveRequest** y **KeepAliveTimeout** a 150,150 y 3 respectivamente.

- Hemos cambiado también la opción de **KeepAlive** a true ya que nuestra web va a hacer muchas peticiones para obtener todos los recursos para el usuario. De esta forma no cerramos y abrimos todo el rato las conexiones.

- A causa de poner el **KeepAlive** en marcha hemos aumentado el número de clientes, **MaxClients** y hemos puesto el **MaxRequestPerChild** a 4000. No lo hemos puesto a 0 (ilimitado) para evitar que procesos que no se quedan sin cerrar gasten memoria.

3.1.2 Deshabilitar módulos innecesarios

Apache viene con mucha información y funcionalidades incluidas al instalar por lo que alguno de los módulos que este tiene para iniciarse por defecto no son necesarios, así que vamos a desactivarlos. Para ellos vamos a utilizar la herramienta **a2ensite**.

:# a2dismod

:# Module autoindex disabled

:# Module cgi disabled

Hemos deshabilitado los módulos **autoindex** y **cgi**. El primero permite listar el contenido de los directorios servidor por Apache cuando no existe un fichero index. El segundo permite el uso de CGI que se ha desactivado por no utilizarlo.

3.2 Seguridad

Para la configuración básica de la seguridad de Apache hemos ejecutado el comando *nano /etc/apache2/conf-enabled/security.conf* . En esté, hemos cambiado las variables **ServerSignature** a OFF y **ServerTokens** a Prod. Con la opción de **ServerSignature** evitamos que se muestre la versión del servidor y del sistema operativo en el pie de página de los documentos generados. Y con **ServerTokens** solo vamos a mostrar las cabeceras de las peticiones HTTP sin mostrar nuestro servidor ni el sistema operativo.

3.2.1 Deshabilitar información ofrecida por el servidor

Cuando se realiza una petición al servidor web Apache, en las cabeceras que devuelve como parte de la respuesta se mostrará información sobre la versión de Apache, versión de PHP, etc. También puede ser que al forzarse un error inexistente en la página el servidor puede mostrar la versión del software y sus principales características.

Para evitar esto vamos a editar algunos parámetros de configuración. En realidad esto no ofrece seguridad como tal, pero sí que dificulta a un potencial atacante la identificación del entorno al que se enfrenta.

En el fichero */etc/apache2/conf.d/security* hay dos directivas llamadas **ServerTokens** y **ServerSignature** que después de cambiarlos han quedado de la siguiente manera:

ServerTokens ProductOnly

ServerSignature Off

A partir de este momento no se distinguirá la versión de Apache ejecutada.

3.2.2 Configuración por contexto

Para mostrar algunas de las configuraciones aplicadas a cada contexto o sitio en Apache se utilizará el sitio habilitado por defecto. Este contenido se encuentra en el fichero */etc/apache2/sites-available/default*, que se encuentra habilitado al existir un enlace simbólico hacia el directorio */etc/apache2/sites-enabled.*

<Directory /urs/lib/cgi-bin>

AllowOverride None

Options +ExecCGI -MultiViews +SymLinksIfOwnerMatch

Order allow,deny

Allow from all

SSLOptions +StdEnvVars

</Directory>

Options

Con la directiva options es posible especificar determinadas funcionalidades del servidor web Apache dentro del contexto Directory en el que se defina.

- SymLinksIfOwnerMatch. Con esta configuración solo se seguirán enlaces simbólicos que tengan el mismo propietario que el recurso enlazado.

- MultiViews. Tratará de ofrecer un recurso indicado si este no dispone de una extensión. Se entregará el recurso que existe probando con las extensiones definidas en */etc/mime.types* .

Access Control

Es posible restringir el acceso al contenido para determinadas direcciones o redes. Esto puede resultar útil cuando el mismo servidor que ofrece contenido público se utiliza para mostrar contenido que solo debería ser accesible por una intranet por ej.

Para conseguir esta funcionalidad se utilizan las directrices Order, Allow, Deny.

Order Allow, Deny

Allow from all

HTaccess

Los ficheros *.htaccess* están dentro de los directorios en donde hemos aplicado una configuración específica para Apache. Es una funcionalidad que ofrece gran flexibilidad en cuanto a configuración se refiere.

Por defecto la opción que determina el uso de los ficheros *.htaccess* es **AllowOverride** que está deshabilitada por defecto. Permanece en este estado ya que no tenemos la necesidad de activarla.

3.2.3 mod_security

Es un módulo que permite filtrar peticiones maliciosas recibidas por el servidor Apache. Mediante los filtros y patrones que implementa podrá detectar ataques SQLi y XSS entre otros, siendo una protección muy eficiente para aplicaciones web.

- En primer lugar hemos preparado el sistema con una serie de dependencias:

:# apt-get install build-essential autoconf automake libapr1-dev apache2-dev libpcre3-dev libxml2-dev liblua5.1-0-dev libcurl4-openssl-dev

- Una vez instalados los paquetes hemos descargado los fuentes de mod_security . Para ellos hemos realizado lo siguiente:

:# wget http://www.modsecurity.org/tarball/2.7.2/modsecurity-apache_2.7.2.tar.gz

:# wget http://www.modsecurity.org/tarball/2.7.2/modsecurity-apache_2.7.2.tar.gz.md5

:# md5sum -c mod_security-apache_2.7.2.tar.gz.md5

modsecurity-apache_2.7.2.tar.gz.md5: Suma Coincide (Mensaje consola)

:#tar xzf modsecurity-apache_2.7.2.tar.gz

- Un vez descargado y extraído, el paquete se instala y compila:

:# cd modsecurity-apache_2.7.2/

:# ./autogen.sh

:# ./configure

:# make

:# make install

:# cp/urs/local/mod_security/lib/mod_security2.so /urs/lib/apache2/modules/

En este momento el modulo está compilando y listo para que Apache pueda utilizarlo. Solo hace falta descargar las reglas y crear los ficheros de configuración.

Preparación Apache

Antes de comenzar con la configuración de Apache es necesario descargar las reglas actualizadas desde la web de mod_security. Para el almacenamiento de todo ello vamos a crear un directorio dentro de la configuración de Apache.

:# mkdir /etc/apache2/modsec

:# wget https://github.com/SpiderLabs/owasp-modsecurity-crs/tarball/master \ -O crs-`date +%Y%m%d`.tgz

.

Resulta que dicho fichero ya no existe por lo que no se ha podido seguir con la configuración del *mod_security* para Apache.

4. MySQL

Una vez instalado el gestor de bases de datos MySQL hay que decir que por defecto la instalación es bastante segura, sin embargo a continuación se verán algunas opciones que hemos modificado para aumentar la seguridad.

4.1 Usuarios BD

Dependiendo de lo que se quiera hacer en la base de datos se va a utilizar un usuario u otro. Para asegurarnos sé que los errores que vamos a tener van a ser justos cada usuario va a tener los mínimos privilegios posibles para las acciones que está destinado.

- Este usuario sirve solo para visualizar contenido en la Base de Datos.

  Usu: **visualizar**

  Pass: **********

  Privilegios: select

- El usuario 'gestionar' sirve para visualizar, insertar y modificar contenido en la Base de Datos.

  Usu: **gestionar**

Pass: **********

Privilegios: select, insert, update

- El usuario 'gestCompleto' tiene todos los posibles privilegios sobre la tabla ProgrammingCloud menos los de Administrador.

Usu: **gestCompleto**

Pass: **********

Privilegios: todo sobre los datos de la BD y sobre la estructura de la misma.

- Los usuarios root son los que vamos a utilizar los componentes de grupo.

Usu: **valen(root)**

Pass: **********

Privilegios: Admin

4.2 Seguridad

Hemos aplicado configuraciones mediante ficheros, otras directamente en la base de datos utilizando el cliente mysql o bien el comando mysqladmin. Las opciones modificadas en la propia consola de MySQL estarán precedidas por *mysql>*.

4.2.1 Dirección de escucha

Se trata de una directiva que por defecto escuchará en local, por lo que no va a aceptar conexiones que provengan de otras máquinas. El parámetro es el siguiente:

Bind-address=127.0.0.1

4.2.2 Carga de ficheros locales

En las aplicaciones web se pueden explotar diversos tipos de ataques. Si un atacante descubriese la forma de exportar un SQLi y quisiera acceder a ficheros del sistema, podría obtener información de los archivos accesibles por el usuario que ejecute el dominio MySQL.

Para deshabilitar la carga de ficheros desde MySQL y evitar una posible fuga de información desde el sistema de ficheros mediante un SQLi, hemos establecido la siguiente configuración en el fichero */etc/mysql/my.cnf.*

Local-infile = 0

Secure-file-priv = /dv/null

Para ver que usuarios tienen el privilegio file_priv ejecutamos la siguiente consulta:

mysql> select user, host, file_priv from mysql.user;

En nuestro caso el único usuario que tiene permisos de archivos es el usuario *root*, además de los usuarios de cada uno los componentes del grupo.

4.2.3 Renombrar el usuario root

No es una medida de seguridad como tal pero hará la labol de un atacante más dificl para encontrar el usuario administrador de la base de datos. Lo hemos hecho de la siguiente forma:

mysql> update mysql.user set user='valen' where user='root';

mysql> flush privileges;

4.2.4 Comprobar existencia de usuarios anónimos

Al instalar MySQL es posible que se hayan creado usuarios anónimos por defecto que se puedan utilizar para extraer información. Para eliminar dicho problema lo haremos del siguiente modo:

mysql> select user usuarios, host from mysql.user where user='';

4.2.5 Comprobar los privilegios de los usuarios

Una de las medidas fundamentales para proteger la privacidad de los usuarios frente a ataques consiste en disponer de diferentes usuarios para cada cometido. Con la siguiente acción podremos obtener los diferentes usuarios y sus privilegios.

mysql> select distinct(grantee) from information_schema.user_privileges;

Esto también se puede hacer desde la interfaz de **phpmyadmin**.

4.2.6 mysql_secure_installation

A través del comando *sudo mysql_secure_installation* aumentamos la seguridad de nuestro MySQL de las siguientes maneras:

- Se ha aumentado el nivel de seguridad de la contraseña y se añadió una contraseña al usuario Root.
- Se ha configurado MySQL de forma que solo se pueda acceder al MySQL desde el localhost.
- Se han eliminado los usuarios anónimos que pueden acceder a MySQL y vienen por defecto.
- Así mismo se ha eliminado la tabla 'test' que viene de serie con MySQL

4.2.7 Evitar MySQL Inyección

La última opción –la preferida– se muestra debajo. En ella simplemente se transfiere a PHP y MySQL la tarea de verificar la sentencia SQL. Se hace uso del método prepare de la clase PDO(PHP Data Object) para preparar la sentencia y de **execute** para ejecutarla. Notemos

el uso de los dos puntos antes de 'name' en la preparación de la sentencia y en la ejecución de la misma. Estos son requisitos que no podemos omitir.

```
static public function getAula($id){
        $query = 'SELECT * FROM Aula WHERE id = :idAula;';
        $ej = getDatabase() -> all($query, array(':idAula' => $id));
}
```

4.3 Copia de Seguridad de las BBDD

Vamos a hacer una copia de seguridad para estar seguros de no perder todos los datos por cualquier imprevisto. Para no tener que hacerlo nosotros manualmente vamos a asignar está tarea a **Cron**.

Cron es una utilidad de programación basada en tiempo en Linux. Se puede encontrar en la mayoría de las distribuciones de Linux y stats con el sistema operativo automáticamente. Lee un archivo de configuración (/etc/crontab) cuando se inicia y sigue funcionando mientras el equipo está en ejecución.

Ejecuta algunos programas en un momento dado. Por ejemplo nosotros creamos un copia de seguridad de MySQL todos los días a las 2:15. Lo hacemos editando el archivo que se encuentra en */etc/crontab* .

```
15 2 * * * root mysqldump -u valen – p ******** --all-databases | gzip > /mnt/disk/database_`data'+%m-%d-%Y'`.sql.gz
```

- El comando se ejecutará con un usuario tipo root.
- La copia se seguridad de guardará de forma comprimida.
- La copia de seguridad se guardará con el siguiente formato: ''nombreBD_MM-DD-YYY.

5. SSL

TLS, o Transport Layer Security, y su predecesor **SSL**, Secure Sockets Layer, son protocolos que sirven para asegurar un tráfico seguro de datos utilizando un envoltorio encriptado para los mismos.

Este protocolo permite que el tráfico sea enviado de forma segura entre partes remotas sin la posibilidad de que el tráfico sea interceptado y leído por alguien en el medio. También son fundamentales en la validación de la identidad de dominios y servidores a través de internet mediante el establecimiento de un servidor de confianza por una autoridad de certificación.

5.1 Activar el módulo SSL

Activamos el módulo utilizando la sentencia:

```
:# sudo a2ensite ssl
```

5.2 Crear un certificado SSL

Empezamos creando un subdirectorio dentro de Apache para situar todos los certificados ssl:

:# sudo mkdir /etc/apache2/ssl

Ahora que tenemos un sitio para situar los certificados podemos crearlos todos de un paso:

:# sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/apache2/ssl/apache.key -out /etc/apache2/ssl/apache.crt

Vamos a analizar la sentencia ejecutada:

- openssl: Este es el comando básico que provee OpenSSL para crear y gestionar certificados, claves, etc.

- req: Esto especifica un subcomando para la gestión de solicitudes de firma de certificados, x509 es un estándar de infrastructura de clave pública de SSL para la gestión de claves y certificados.

- nodes: Esta opción indica a OpenSSL que no queremos proteger nuestro archivo con clave.

- days 365: Esto especifica que el certificado que hemos creado será válido durante un año.

- newkey rsa:2048: Esta opción creará la solicitud del certificado y una nueva clave privada. Esto es necesario ya que no hemos creado una clave privada por adelantado. El RSA: 2048 le dice a OpenSSL que genere una clave RSA de 2048 bits.

Después nos van a preguntar una serie de preguntas:

Country Name (2 letter code) [AU]:ES
State or Province Name (full name) [Some-State]:Comunidad Valenciana
Locality Name (eg, city) []:Alicante
Organization Name (eg, company) [Internet Widgits Pty Ltd]: Braineering
Organizational Unit Name (eg, section) []: .
Common Name (e.g. server FQDN or YOUR name) []: ProgrammingCloud
Email Address []: urui.valentin@gmail.com

La clave y el certificado van a ser creados y situados en */etc/apache2/ssl.*

5.3 Configurar Apache para usar SSL

Ahora que tenemos nuestro certificado y claves disponibles, podemos configurar Apache para usar estos archivos en un archivo de Virtual Host.

En vez de situar nuestro archivo de configuración fuera del archivo */sites-available/000-default.conf* vamos a crear nuestra configuración en el archivo *default-ssl.conf* que contiene alguna configuración SSL por defecto.

Abrimos el archivo:

:# sudo nano /etc/apache2/sites-available/default-ssl.conf

Dentro del archivo vamos a editar lo siguiente:

ServerAdmin valen@localhost

ServerName ProgrammingCloud

ServerAlias www.programmincloud.com

DocumentRoot /var/www/html

SSLCertificateFile ***/etc/apache2/ssl/apache.crt***

SSLCertificateKeyFile ***/etc/apache2/ssl/apache.key***

5.4 Activar el Virtual Host de SSL

Ahora que tenemos configurado el SSL del Virtual Host vamos activarlo:

:#sudo a2ensite default-ssl.conf