



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA



Escola Tècnica
Superior d'Enginyeria
Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Servidor de Logs Centralizado

Proyecto Final de Carrera

Ingeniería Informática de Sistemas

Autor: Juan Carlos Gómez Navarro

Director: Fco. José Gómez Pardo (TVA)
David Picó Vila (ETSINF)

25 septiembre 2014

Resumen

Mi proyecto consiste en un servidor centralizado de logs, es decir un equipo capaz de recoger, almacenar y mostrar los distintos ‘eventos’ o logs generados por otros equipos o dispositivos, para así poder llevar un seguimiento del funcionamiento general de los servicios o analizar posibles incidencias.

Para la realización del proyecto cuento con un equipo con GNU Linux Ubuntu Server 12, sobre el cual configuré un servidor web Apache con soporte PHP y MySQL para la creación y manipulación de bases de datos.

En dicho equipo se instala la herramienta Rsyslog, que permite tanto el envío como la recepción de logs de forma remota. Rsyslog es un software de logging de mensajes que implementa el protocolo básico de Syslog y lo extiende agregando filtros, con una configuración flexible, una herramienta muy potente que trabaja con protocolos TCP y UDP (el utilizado en mi caso). Así mismo este servicio crea sus propias tablas MySQL donde se almacenarán y clasificarán los logs.

Finalmente con una aplicación web creada a partir del framework Yii, podremos visualizar los logs desde cualquier navegador.

Palabras clave: servidor, log, syslog, linux, rsyslog, framework, Yii, PHP



Tabla de contenidos

1. Introducción	8
2. Preparación del equipo	10
3. Rsyslog	13
4. Firewall. (IPtables).....	18
5. Aplicación web. Yii Framework	20
6. Aplicación web. Modelo Log.....	25
7. Aplicación web. Modelo Host.....	31
8. Aplicación web. Alarmas	34
9. CRON.....	39
10. Aplicación web. Gráficas.....	41
11. Posibles ampliaciones. SNMP.....	42
12. Conclusiones.	43

1. Introducción

Empecemos por el principio... ¿Qué es un log?

Un log es un registro oficial de eventos durante un rango de tiempo en particular. En informática es usado para registrar datos o información sobre quién, qué, cuándo, dónde y por qué (who, what, when, where y why) un evento ocurre para un dispositivo en particular o aplicación.

La mayoría de los logs son almacenados o desplegados en el formato estándar, el cual es un conjunto de caracteres para dispositivos comunes y aplicaciones. De esta forma cada log generado por un dispositivo en particular puede ser leído y desplegado en otro diferente.

También se le considera como aquel mensaje que genera el programador de un sistema operativo, alguna aplicación o algún proceso, en virtud del cual se muestra un evento del sistema.

A su vez la palabra log se relaciona con el término evidencia digital. Un tipo de evidencia física construida de campos magnéticos y pulsos electrónicos que pueden ser recolectados y analizados con herramientas y técnicas especiales, lo que implica la lectura del log y deja al descubierto la actividad registrada en el mismo.

En el sistema GNU/Linux Ubuntu (nuestro caso) existen unos ficheros con extensión .log. Estos ficheros, que se encuentran en la carpeta /var/log (en la mayoría de las distribuciones), tienen la función de ir almacenando en su interior todo lo que sucede mientras estamos haciendo uso de linux.

No existe sólo un fichero .log, porque si no sería interminable e inmanejable, para ello están divididos en distintos ficheros y cada uno con su función específica. Por ejemplo:

syslog: se encarga de registrar los mensajes de seguridad del sistema.

kern: se encarga de los mensajes del núcleo (kernel).

messages: archiva los distintos mensajes generales que nos manda el sistema.

debug: mensajes de depuración de los programas.

user.log: información sobre el usuario.

Xorg.0.log: guarda información sobre el entorno gráfico.

auth.log: contiene los accesos al sistema, incluidos los intentos fallidos.

Para poder visualizar el contenido de estos ficheros, tan sólo necesitamos un editor de texto, como puede ser vi, nano, gedit, kwrite...

Tengo que hacer una mención especial sobre estos ficheros, ya que tal vez un usuario novel, desconoce la importancia que pueden llegar a tener, ya que son fuente de información para ver por ejemplo los distintos fallos en el entorno gráfico, o ver si se producen errores en la carga del núcleo (kernel) y quisiera hacer más hincapié en un fichero.

Hoy en día los Administradores de Sistemas debemos estar al pendiente de cientos de logs para asegurarnos de que todo está funcionando bien, en la empresa donde desarrollo mi proyecto hay una importante cantidad de servidores y equipos de comunicación, e ir día tras día por los logs de esos equipos es algo que puede resultar tedioso. Qué tal si pudiéramos contar con un servidor que reciba todos esos logs y lo podamos revisar desde una interfaz centralizada, aquí es donde entra **Rsyslog** y mi aplicación web como Frontend Gráfico.

Es básico en empresas que ofrecen servicios de telecomunicación orientados a usuarios, el control de los equipos en tiempo real para detectar y subsanar posibles averías en el menor tiempo. Es por esto que opté por implementar y añadir a la aplicación, un sistema de alarmas basado en el filtrado de los logs.

Así mismo, como relataré más adelante, implementé un sistema de gráficas donde podremos ver las estadísticas de los logs en base a su procedencia, número de logs diarios... etc. para llevar un control más generalizado de nuestros equipos. Este sistema de alarmas es totalmente configurable y ampliable en función de cuanta información y de qué tipo necesitemos mostrar.



2. Preparación del equipo

Para realizar el proyecto la empresa pone a mi disposición un equipo Supermicro, con un procesador Intel Pentium 4 de 3,00 GHz, un disco duro de 1TB, 2GB de memoria RAM y una arquitectura de 64bits. Sobre este equipo me dispuse a instalar una distribución de Ubuntu versión 12.04.4 LTS.

Para realizar mi aplicación web en dicho equipo, vamos a utilizar un sistema LAMP (Linux, Apache, MySQL y PHP). La distribución 12.04.4 de Linux ya comentada, no me llevó mucho tiempo, ya que lo más “complejo” fue quizás la parte de red, en este caso el equipo tendrá una IP pública fija para que sea siempre la misma, visible y accesible desde cualquier equipo al que le configuremos el envío remoto de Logs. Posteriormente se podrá restringir el acceso a protocolos y equipos mediante el firewall.

Pasamos a instalar Apache como nuestro servidor web mediante el siguiente comando;

```
sudo apt-get install apache2
```

Con esto ya tenemos instalado Apache en nuestro equipo, el servicio se inicia automáticamente pero si necesitáramos pararlo, arrancarlo o reiniciarlo manualmente utilizaré estos comandos:

```
sudo service apache2 stop
sudo service apache2 start
sudo service apache2 restart
```

Una vez instalado el servidor web procedo a instalar PHP utilizando la siguiente línea en mi terminal:

```
sudo apt-get install php5 libapache2-mod-php5 php5-cli php5-mysql
```

Reinicio Apache y creo un pequeño script de prueba para comprobar el correcto funcionamiento de PHP.

```
<?php
    phpinfo();
?>
```

Así abriendo la ruta del script en nuestro servidor web desde cualquier navegador vemos una página con la información sobre la instalación de PHP.

Para terminar la instalación y configuración de mi sistema LAMP, procedo a instalar el servidor de bases de datos MySQL. Utilizo la siguiente línea:

```
sudo apt-get install mysql-server mysql-client libmysqlclient-dev
```

Al igual que con PHP el servidor MySQL se inicia automáticamente. Comprobamos que está en funcionamiento con la siguiente línea:

```
sudo netstat -tap | grep mysql
```

Una vez ejecutado este comando obtengo la siguiente salida, que me confirma que el servidor está en marcha:

```
tcp 0 0 localhost.localdomain:mysql ** LISTEN -
```

Podemos parar, iniciar y reiniciar el servidor MySQL con los siguientes comandos:

```
sudo /etc/init.d/mysql start
```

```
sudo /etc/init.d/mysql restart
```

```
sudo /etc/init.d/mysql stop
```

Hay que tener en cuenta que la contraseña del administrador no está establecida de forma predeterminada, por tanto lo primero que tuve que hacer fue establecer la contraseña de administrador de MySQL. Utilizando las siguientes líneas no hubo complicación:

```
mysqladmin -u root password nuevopassderoot
```

```
mysqladmin -p -u root -h localhost password nuevopassdesqlroot
```

No haré más cambios en la configuración de MySQL pero si fuera necesario habría que dirigirse al archivo `/etc/mysql/my.cnf` para configurar las opciones básicas - archivo de registro, número de puerto, etc. En nuestro caso estas últimas opciones quedan por defecto.

Un elemento más que utilizaré es el servidor FTP, que será necesario para editar y crear archivos para mi aplicación web. Para instalarlo ejecuto la siguiente línea:

```
sudo apt-get install proftpd
```

Defino un usuario que utilizaré para acceder por ftp a los archivos de mi futura aplicación web. ProFTPd nos crea un usuario por defecto llamado ftp. En mi caso el usuario será distinto por razones de seguridad. Lo creo con la siguiente línea y después añadido los datos que nos va a ir pidiendo, tales como contraseña o nombre completo (este último opcional):

```
sudo adduser usuarioftp
```



A continuación edito el archivo */etc/passwd* donde veo que */home* tiene asignado mi usuario. Ese home lo modificare más adelante por el directorio de mi aplicación web para que al conectar, directamente me aparezca ese directorio que es el que de momento me interesa.

```
juanky:x:1001:1001::/home/juanky:/bin/false
```

Ahí nos indica que nuestra home será */home/juanky* (la x es la contraseña que no aparece por estar encriptada). Debemos asegurarnos que al final diga */bin/false* en lugar de */bin/bash*. En caso de que aparezca esto último debemos cambiarlo. Por último modifiqué los permisos del directorio */var/www* a 775, lo que otorga acceso total al dueño y al grupo y limita el permiso de escritura a los demás. Para hacer esto utilicé el comando:

```
chmod 755 /var/www
```

El servicio NTP no se requiere para poder instalar el software, sin embargo, no debemos perder de vista que si no contamos con una correcta precisión, el análisis de los *logs* puede ser bastante caótico. La empresa dispone de un servidor NTP que proporcionará la hora correcta a mi servidor.

3. Rsyslog

A modo introductorio hablaré de Syslog antes de introducir la herramienta Rsyslog.

Syslog es un estándar utilizado para la captura, el procesamiento y el transporte de mensajes de registro del sistema, es decir. los logs del sistema. Es tanto un protocolo de red como a la aplicación o biblioteca compartida que sirve para procesar y enviar los mensajes de registro del sistema.

Los equipos y servicios con soporte para envío de log a este sistema pueden enviar cualquier tipo de mensaje, normalmente suelen enviar información sobre la seguridad del sistema, errores, avisos, etc. aunque pueden contener cualquier información. Junto con cada mensaje se incluye la fecha y hora del envío, el equipo que envía, la prioridad y otros datos adicionales.

El protocolo syslog consiste en un equipo servidor ejecutando el servidor de syslog, conocido como syslogd (demonio de syslog) y clientes que envían un pequeño mensaje de texto (de menos de 1024 bytes) a este servidor.

Los mensajes de syslog se suelen enviar vía UDP, por el puerto 514, en formato de texto plano. Algunas implementaciones del servidor, como syslog-ng, permiten usar TCP en vez de UDP, y también ofrecen Stunnel para que los datos viajen cifrados mediante SSL/TLS.

Aunque syslog tiene algunos problemas de seguridad, su sencillez ha hecho que muchos dispositivos lo implementen, tanto para enviar como para recibir. Eso hace posible integrar mensajes de varios tipos de sistemas en un solo repositorio central.

El mensaje enviado se compone de tres campos, entre todos no han de sumar más de 1024 bytes, pero no hay longitud mínima:

- **Prioridad:** número es un número de 8 bits que indica tanto el *recurso* (tipo de aparato que ha generado el mensaje) como la *severidad* (importancia del mensaje), números de 5 y 3 bits respectivamente. Los códigos de recurso y severidad los decide libremente la aplicación, pero se suele seguir una convención para que clientes y servidores se entiendan.

UN POCO DE HISTORIA...

Syslog fue desarrollado por Eric Allman como parte del proyecto Sendmail, inicialmente (años 1980) sólo para éste proyecto. Sin embargo, se comprobó que era muy útil, y otras aplicaciones empezaron también a usar *syslog*. En el año 2005, *syslog* estaba presente por defecto en casi todos los sistemas Unix y GNU/Linux, y también se encuentran diversas implementaciones de *syslog* para otros sistemas operativos, como Microsoft Windows.



Códigos de recurso: Éstos son los códigos observados en varios sistemas:

- 0 – Mensajes del kernel.
- 1 – Mensajes del nivel de usuario.
- 2 – Sistema de correo.
- 3 – Demonios del sistema.
- 4 – Seguridad/Autorización
- 5 – Mensajes generados internamente por syslogd
- 6 – Subsistema de impresión.
- 7 – Subsistema de noticias sobre la red.
- 8 – Subsistema UUCP.
- 9 – Demonio de reloj.
- 10 – Seguridad/Autorización.
- 11- Demonio de FTP
- 12 – Subsistema de NTP
- 13 – Inspección del registro.
- 14 - Alerta sobre el registro.
- 15 – Demonio del reloj.
- 16 a 23 – Uso local.

Códigos de severidad: Los 3 bits menos significativos del campo *prioridad* dan 8 posibles grados:

- 0 – Emergencia: el sistema está inutilizable.
- 1 – Alerta: se debe actuar inmediatamente.
- 2 – Crítico: condiciones críticas.
- 3 – Error: condiciones de error.
- 4 – Peligro: condiciones de peligro.
- 5 – Aviso: normal, pero condiciones notables.

6 – Información: mensajes informativos.

7 – Depuración: mensajes de bajo nivel.

- **Cabecera:** indica tanto el tiempo como el nombre del ordenador que emite el mensaje. Esto se escribe en codificación ASCII (7 bits), por tanto es texto legible. El primer campo, tiempo, se escribe en formato **Mmm dd hh:mm:ss**, donde Mmm son las iniciales del nombre del mes en inglés, dd, es el día del mes, y el resto es la hora. No se indica el año. Justo después viene el nombre de ordenador (hostname), o la dirección IP si no se conoce el nombre. No puede contener espacios, ya que este campo acaba cuando se encuentra el siguiente espacio.
- **Texto:** lo que queda de paquete syslog al llenar la prioridad y la cabecera es el propio texto del mensaje. Éste incluirá información sobre el proceso que ha generado el aviso, normalmente al principio (en los primeros 32 caracteres) y acabado por un carácter no alfanumérico (como un espacio, ":" o "["). Después, viene el contenido real del mensaje, sin ningún carácter especial para marcar el final.

Rsyslog implementa el protocolo básico de syslog y lo extiende agregando filtros, con una configuración flexible. Es un eficiente y rápido sistema de procesamiento de registros de sistema. Ofrece un diseño modular de alto desempeño y niveles de seguridad apropiados. A diferencia de sus predecesores —*sysklog* y *syslog*— permite ingreso de datos desde diversas fuentes, transformación de datos y salida de resultados hacia varios destinos. Es lo suficientemente versátil y robusto para ser utilizado en entornos empresariales y tan ligero y sencillo que permite utilizarlo también en sistemas pequeños. Permite almacenar las bitácoras en archivos de texto simple o bases de datos MySQL y PostgreSQL, utilizar otros destinos en caso de falla, transporte de *syslog* a través de tcp, control detallado de formatos, etiquetas de tiempo exactas, operaciones en cola de procesamiento y capacidades de filtrado en cualquier parte de los mensajes.

En cuanto a la configuración de Rsyslog, se centra sobre todo en el archivo */etc/rsyslog.conf*. Para convertir el demonio rsyslog en un servidor de logs remoto debe utilizarse el plugin imudp (si se desea utilizar UDP) y/o imtcp (para TCP), en mi caso ambos teniendo en cuenta que habrá equipos muy dispares que queremos monitorizar. Estos plugins reemplazan la opción -r de syslog ya obsoleta. El puerto utilizado por defecto por este servicio, como ya comenté anteriormente, es el 514 para ambos protocolos. Para habilitar estos plugins hay que descomentar unas líneas en el citado archivo de configuración:

```
$ModLoad imudp
$UDPServerRun 514
```



```
$ModLoad imtcp
$InputTCPServerRun 514
```

Una vez hecho esto, reiniciamos el servicio para que los cambios tengan efecto con:

```
service rsyslog restart
```

Con esto hemos habilitado la recepción de logs en nuestro servidor. Vamos con la parte del cliente. Suponemos que nuestro cliente es otro Debian, la configuración es sencilla, simplemente introduciremos en el archivo `/etc/rsyslog.conf` la siguiente línea.

```
*.* @<IP remota>
```

La `@` nos indica que deben enviarse los logs a un servidor remoto. Si aparece una sola `@` será utilizando UDP, colocando dos `@` utilizaremos TCP. Sencillo.

Así, cualquier dispositivo de red puede enviar sus logs al servidor recién configurado, y este los añadirá a los locales del servidor. Claro que lo mejor es no mezclar logs locales con logs remotos, por lo cual se pueden configurar distintas entradas en la configuración que redirijan estos logs a otros archivos. Como decía anteriormente, Rsyslog es capaz de filtrar y reordenar todos estos logs. Las reglas de las que se vale se construyen en base a la “facility” y al nivel de criticidad que ya sabemos que va de de 0 (emergencias) a 7 (debugging). Las facilities permiten indicar el origen del mensaje, y existe un número fijo de ellas. Algunos demonios y procesos del sistema operativo tienen asignados valores de facilities (por ej: kern, mail, daemon, user, etc), y aquellos que no, deben utilizar alguna de las definidas como "uso local" (local0 a local7). El usuario puede utilizar las facilities local0 a local7 para sus logs, y estas son las que se suelen utilizar para separar logs de dispositivos remotos. El formato de las reglas es:

```
<facility>.<criticidad> <archivo log destino (ej /var/log/syslog)>
```

Tanto el facility como la criticidad pueden ser asterisco (*), indicando que abarque todo. Por ejemplo, "kern.*" envía todos los logs del kernel al archivo especificado. En el caso de utilizar asterisco en el facility, es posible utilizar la prioridad none para indicar qué facilities no incluir. Por ejemplo, existe la siguiente regla default en la configuración de rsyslog:

```
*.*;auth,authpriv.none -/var/log/syslog
```

La cual hace que todo se envíe al archivo syslog. En esa misma regla se puede observar que se agregó `auth,authpriv.none` para evitar que los logs de `auth` y `authpriv` se almacenen en dicho archivo. A la regla anterior se puede agregar `local4.none`, para luego utilizar la facility `local4` en dispositivos remotos y así separar esos logs de los locales:

```
*.*;auth,authpriv.none;local4.none -/var/log/syslog
```


Como podemos ver Rsyslog nos da muchas posibilidades pero en mi caso este “filtrado” lo hare por medio de la aplicación web. Por lo que en un principio, todos los logs, locales y remotos, irán almacenados en la base de datos de Rsyslog.

Para utilizar el almacenamiento de base de datos con Rsyslog tenemos que instalar el paquete que nos lo permitirá:

```
sudo apt-get install rsyslog-mysql
```

Al instalarlo nos pregunta si queremos crear la base de datos para que Rsyslog guarde los registros. Le digo que sí y tras poner el password de root de MySQL, creará la base de datos junto con las tablas correspondientes. Para comprobar que se han creado dichas tablas utilizamos la siguiente secuencia, que nos las mostrará:

```
mysql -u root -p
(introducimos password)
mysql> use Syslog
mysql> show tables
```

Comprobamos a su vez que la instalación del módulo para MySQL ha sido correcta, mirando si está en su directorio, el correspondiente archivo de configuración:

```
cat /etc/rsyslog.d/mysql.conf
-----
$ModLoad ommysql
*. *:ommysql:localhost,Syslog,rsyslog,XXXXXX
```

Donde observamos que la primera línea, carga el módulo para MySQL y la segunda línea guarda todos los logs, con cualquier facility y prioridad, en nuestro servidor local de bases de datos (localhost), en la tabla Syslog (creada por Rsyslog anteriormente) y utilizando el usuario rsyslog con su correspondiente contraseña. En este archivo de configuración utilizaré más adelante nuevas líneas con reglas para filtrar determinados logs en base a sus datos. Es decir, los descartaremos antes de almacenarlos según nuestras preferencias.

Ya podemos observar en las tablas de eventos de la base de datos de Rsyslog, como los logs locales están siendo almacenados. Para probar los logs remotos, o bien los configuramos en algún equipo (son numerosos los equipos en la empresa para probar) o bien podemos utilizar cualquier pequeña herramienta software de generación y envío de logs.



4. Firewall. (IPtables)

Para mantener mi equipo protegido y con el fin de que el servidor acepte solo la recepción de logs de aquellos equipos que le indiquemos en la aplicación que realizaré posteriormente, es necesario aplicar una serie de reglas que juntas constituyen mi protección o firewall.

¿Qué es un firewall?

Un firewall es un dispositivo que filtra el tráfico entre redes, como mínimo dos. El firewall puede ser un dispositivo físico o un software sobre un sistema operativo. En general debemos verlo como una caja con DOS o mas interfaces de red en la que se establecen una reglas de filtrado con las que se decide si una conexión determinada puede establecerse o no. Incluso puede ir más allá y realizar modificaciones sobre las comunicaciones, como el NAT.

¿Qué es iptables?

IPtables es un sistema de firewall vinculado al kernel de linux que se ha extendido enormemente a partir del kernel 2.4 de este sistema operativo. iptables está integrado con el kernel, es parte del sistema operativo. ¿Cómo se pone en marcha? Realmente lo que se hace es aplicar reglas. Para ellos se ejecuta el comando iptables, con el que añadimos, borramos, o creamos reglas. Por ello un firewall de iptables no es sino un simple script de shell en el que se van ejecutando las reglas de firewall.

En mi caso, descubro iptables gracias a mi tutor en la empresa y necesito ponerme al día casi desde cero. Encontré diversos manuales e instrucciones en internet y decidí realizar un script con las reglas de iptables que me interesen y que se ejecute al inicio del sistema, este script podrá además detenerse o reiniciarse manualmente.

En primer lugar hay que instalar el paquete para ubuntu de iptables:

```
apt-get install -test iptables
```

A continuación mostraré algunas líneas del código final del script **firewall.sh**, el archivo completo lo adjunto con junto con esta memoria:

```
## Establecemos política por defecto (DROP)
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP
```

Establezco la política por defecto de todo acceso cerrado, para después ir habilitando servicios a nuestra elección. Esta política es más segura que hacerlo al contrario.

```

# El localhost se deja todo
/sbin/iptables -A INPUT -i lo -j ACCEPT
/sbin/iptables -A OUTPUT -o lo -j ACCEPT

# Permitimos que la maquina pueda salir a la web
/sbin/iptables -A INPUT -p tcp -m tcp --sport 80 -m state --state
RELATED,ESTABLISHED -j ACCEPT
/sbin/iptables -A OUTPUT -p tcp -m tcp --dport 80 -j ACCEPT

# Reglas necesarias para FTP pasivo y activo. Se permiten conexiones
entrantes YA establecidas
/sbin/iptables -A INPUT -p tcp -m tcp --sport 20:21 -m state --state
RELATED,ESTABLISHED -j ACCEPT
/sbin/iptables -A OUTPUT -p tcp -m tcp --dport 20:21 -j ACCEPT

##aqui anyadimos las lineas necesarias para permitir a nuestros host
enviar los logs
/sbin/iptables -A INPUT -s 176.57.105.20 -p udp -m udp --dport 514 -
j ACCEPT
/sbin/iptables -A INPUT -s 176.57.105.29 -p udp -m udp --dport 514 -
j ACCEPT

```

Para que este script se ejecute desde el inicio lo añado al directorio /etc/init.d de mi servidor.

El objetivo posterior es modificar este script añadiendo y quitando líneas como las últimas según añadamos o borremos Host en nuestra base de datos. La forma de hacerlo la indicaré más adelante en el punto 7 de esta memoria.



5. Aplicación

Dejando Rsyslog configurado y la creación de mi aplicación voy a introducir primero el framework

Yii es un framework orientado a componentes, PHP y framework como se escribe y es un acrónimo

Algunas de las características:

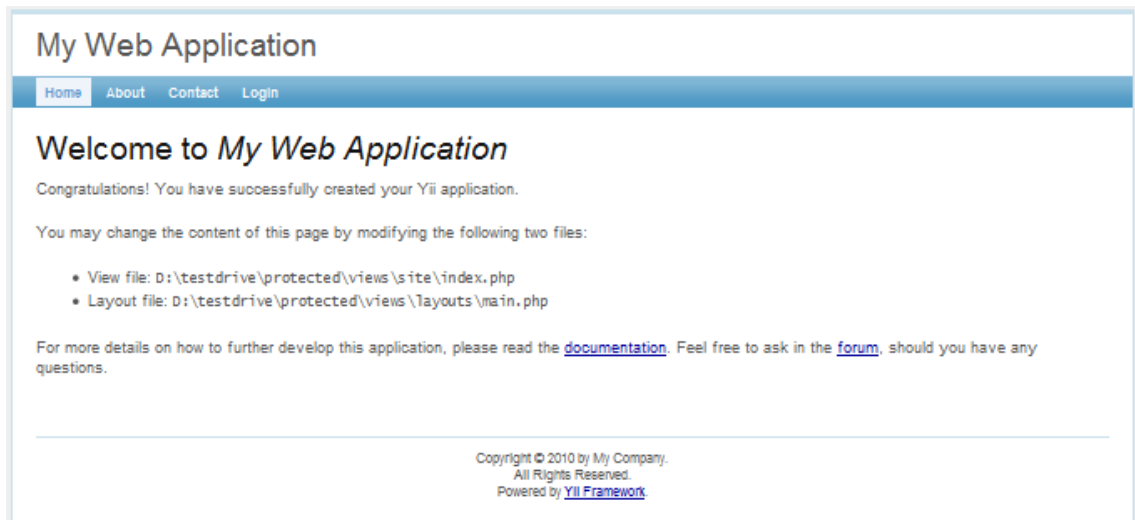
- Patrón de diseño MVC
- Database Access Objects para base de datos.
- Integración con jQuery
- Entradas de Formularios
- Widgets de Ajax, como
- Soporte de Autenticación y based access control
- Generación automática de aplicaciones CRUD

El proyecto Yii tiene un gran número de contribuidos por usuarios.

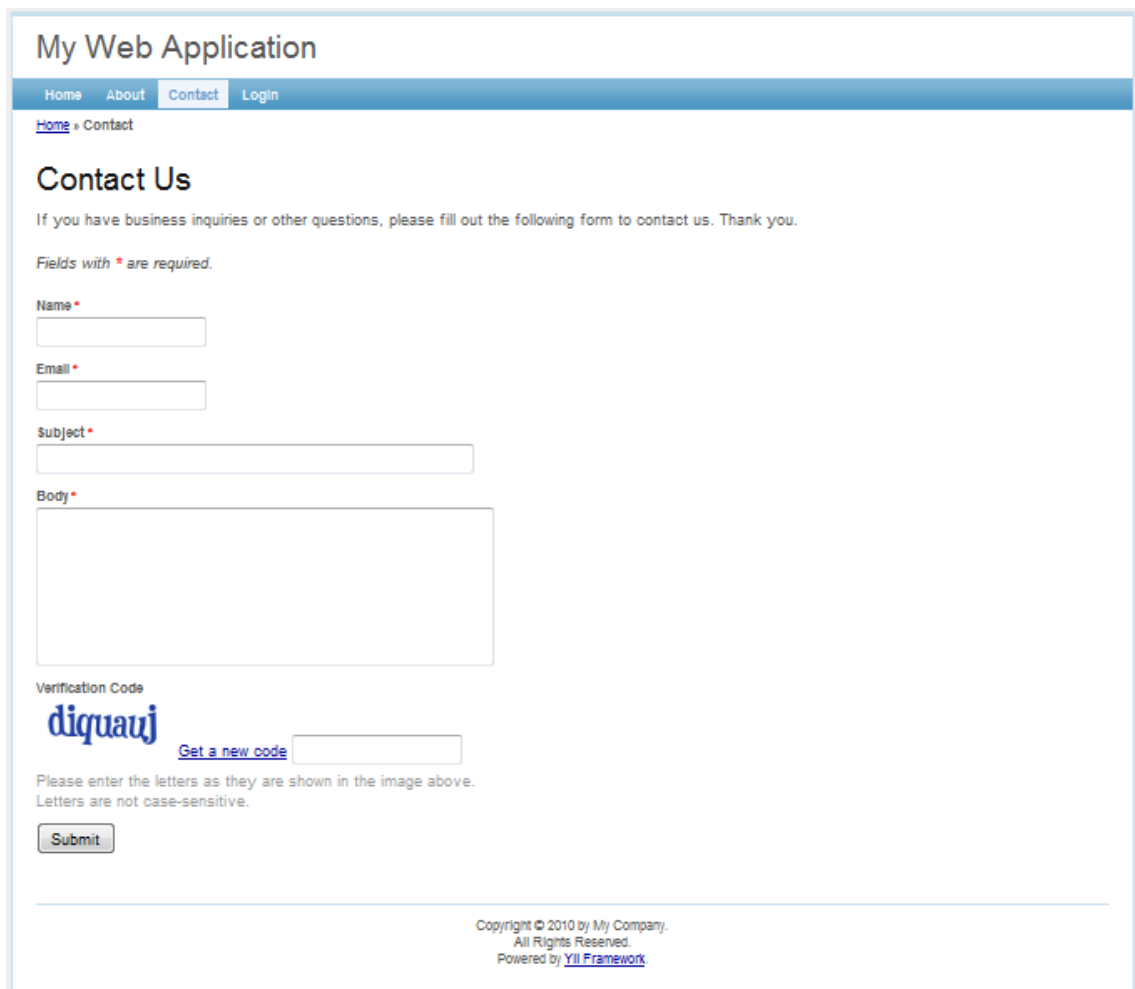
También hay una biblioteca de widgets junto al núcleo del framework como jQuery, etc.

En cuanto a documentación tales como un tutorial para la descripción de cada función y acerca de las propiedades, propiedades,

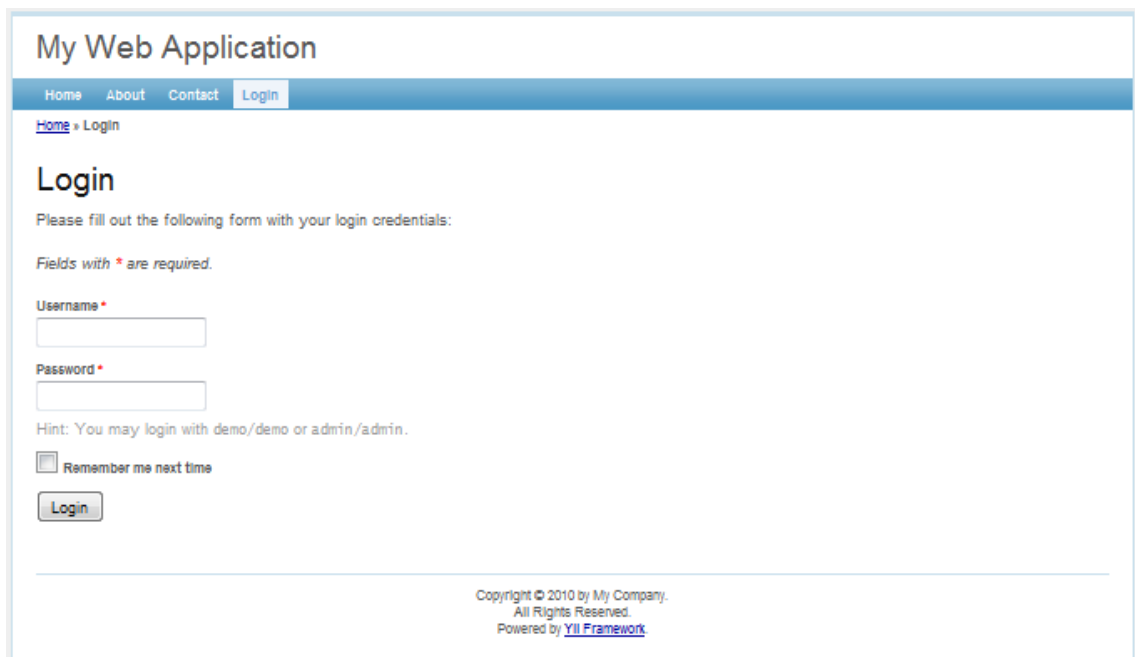
También hay una documentación



La página Home es la página principal de nuestra aplicación, el “index”, la página About es un ejemplo de página estática de la aplicación.



La página de contacto muestra un formulario básico para completar. Los campos se validan mediante Ajax y muestran al momento si la información introducida es correcta o no, bajo mi punto de vista y una de las características más útiles de Yii.



La página de login le permite a un usuario autenticarse antes de acceder a los contenidos privados que requieren ciertos privilegios. El login por defecto es admin/admin.

El siguiente diagrama muestra la estructura de directorios de mi aplicación:

testdrive/

- index.php** archivo de entrada de la aplicación Web
- assets/** contiene archivos de recursos públicos
- css/** contiene archivos CSS
- images/** contiene archivos de imágenes
- themes/** contiene temas de la aplicación
- protected/** contiene los archivos protegidos de la aplicación
- yiic** script de línea de comandos yiic
- yiic.bat** script de línea de comandos yiic para Windows
- commands/** contiene comandos 'yiic' personalizados
- shell/** contiene comandos 'yiic shell' personalizados
- components/** contiene componentes reusables
- MainMenu.php** clase de widget 'MainMenu'
- Identity.php** clase 'Identity' utilizada para autenticación
- views/** contiene los archivos de vistas para los widgets
- mainMenu.php** el archivo vista para el widget 'MainMenu'



config/	contiene archivos de configuración
console.php	configuración aplicación consola
main.php	configuración de la aplicación Web
controllers/	contiene los archivos de clase de controladores
SiteController.php	la clase controlador predeterminada
extensions/	contiene extensiones de terceros
messages/	contiene mensajes traducidos
models/	contiene archivos clase de modeloscontaining model class files
LoginForm.php	el formulario modelo para la acción 'login'
ContactForm.php	el formulario modelo para la acción 'contact'
runtime/	contiene archivos temporarios generados
views/	contiene archivos de vista de controladores y de diseño
layouts/	contiene archivos de diseño
main.php	el diseño default para todas las vistas
site/	contiene archivos vista para el controlador 'site'
contact.php	contiene la vista para la acción 'contact'
index.php	contiene la vista para la acción 'index'
login.php	contiene la vista para la acción 'login'
system/	contiene archivos de vista del sistema

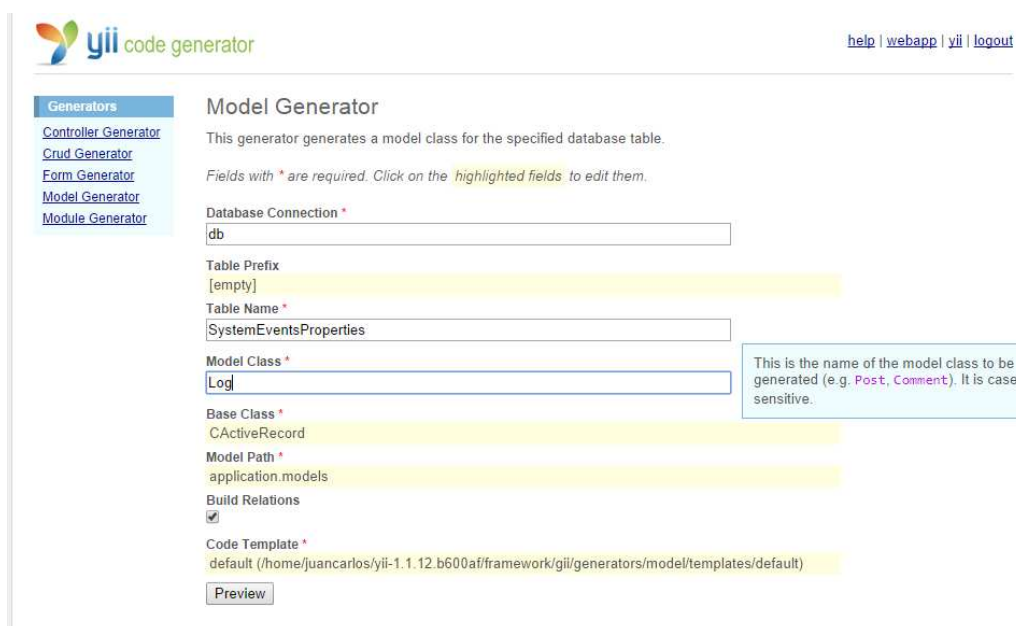
6. Aplicación web. Modelo Log

Una vez me familiaricé con el esquema de archivos era hora de empezar a configurar mi aplicación. Lo primero es conectar con la base de datos de Rsyslog creada anteriormente. Las conexiones de bases de datos en Yii se administran en el archivo de configuración principal de la aplicación `/logstva/protected/config/main.php`

```
'components'=>array(
    'db'=>array(
        'connectionString' => 'mysql:host=IPDELEQUIPO;dbname=Syslog',
        'emulatePrepare' => true,
        'username' => 'root',
        'password' => 'tvalmansa',
        'charset' => 'utf8',
    ),
),
```

Así mismo en este archivo modifiqué parámetros como el nombre de la aplicación. A continuación habilité las líneas para poder utilizar el módulo del generador gráfico de CRUD (crear, leer, actualizar, borrar), que me va a permitir realizar modelos a partir de las tablas de la base de datos, el módulo se llama “gii”.

Para acceder a gii, a través del navegador pongo la URL: `http://ipdelservidor/logstva/index.php?r=gii` con la contraseña y el password editados en el archivo de configuración anterior. Hay que destacar que necesité permisos 777 en los directorios `/protected/models`, `/protected/views` y `/protected/controller`. En gii creo primero el modelo Log, eligiendo la tabla SystemEvents de la BD Syslog.

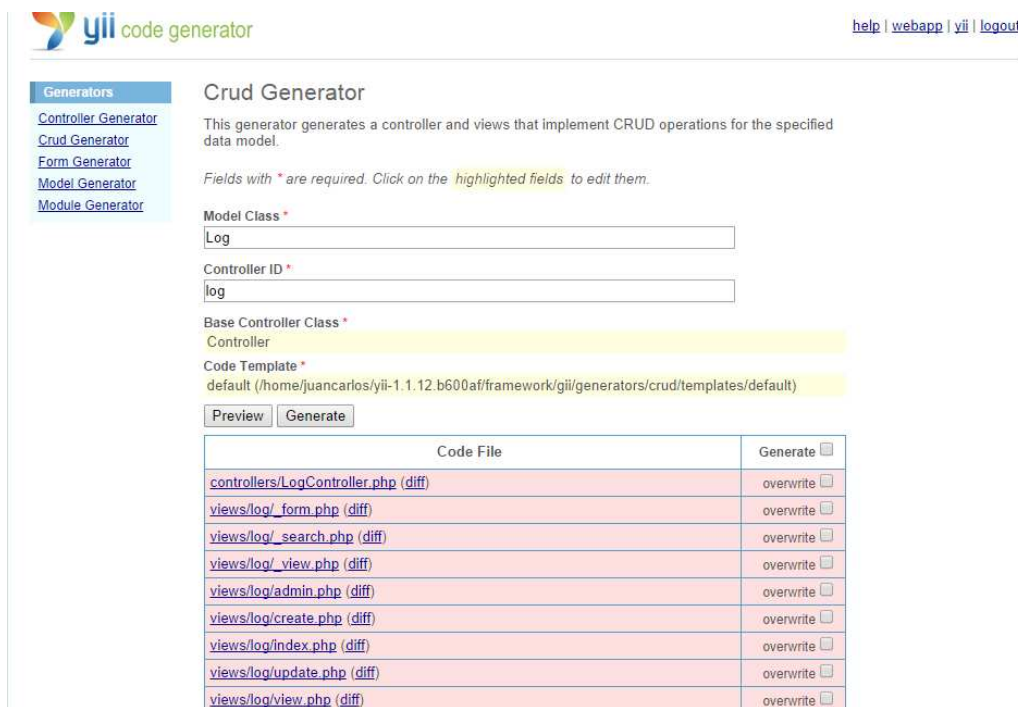


The screenshot shows the 'Model Generator' interface of the Yii Code Generator. On the left, there is a sidebar with a 'Generators' menu where 'Model Generator' is selected. The main area contains the following fields and options:

- Database Connection ***: db
- Table Prefix**: [empty]
- Table Name ***: SystemEventsProperties
- Model Class ***: Log
- Base Class ***: CActiveRecord
- Model Path ***: application.models
- Build Relations**:
- Code Template ***: default (/home/juancarlos/yii-1.1.12.b600af/framework/gii/generators/model/templates/default)

A tooltip next to the 'Model Class' field states: "This is the name of the model class to be generated (e.g. Post, Comment). It is case-sensitive." A 'Preview' button is located at the bottom of the form.

A partir del modelo entrando en “Crud Generator” del menú de la izquierda, creo la plantilla del controlador y las vistas como muestro a continuación.



Una vez tengo mi modelo con sus vistas, puedo visualizar en mi aplicación los logs almacenados en la base de datos, que se listan en un formato tabla, entrando en <http://ipdelservidor/logstva/index.php?r=log/admin>. Las columnas de la tabla todas las de la base de datos, pero podemos elegir cual mostrar y como, modificando la tabla en la view “admin” del modelo. Los parámetros que hay detrás de la letra “r” de la URL indican el directorio del modelo en el que estamos, si la página es del sitio en general, se hará referencia del modo *r=site/login* por ejemplo.

Al principio me costó hacerme a la estructura de Yii, pero una vez entendí el funcionamiento y como acceder a los elementos que quería, modificarlos y trabajar correctamente con ello, todo fue mucho más rápido, el CLASS REFERENCE es completísimo y ofrece una gran cantidad de documentación de las clases y elementos de Yii, así como de sus atributos. Al comienzo me centré en ajustar pequeñas cosas: la pagina de redirección tras el login, ocultar ciertos enlaces en la barra del menú, hacer de la página de login el index de la aplicación, redirigir la acción de logout a la página de login de nuevo... en definitiva ajustar parámetros sobre el comportamiento general de la aplicación modificando sobre todo las “action” del archivo SiteController.php.

Sobre el propio modelo, se pueden ajustar parámetros de la consulta a la base de datos, en Log.php modifique por ejemplo, que las filas de la tabla vengan ordenadas por fecha más reciente y también que se mostrasen al menos 100 filas (logs) por página, ya que el volumen de logs será importante.

Otro cambio que realizo es sustituir el valor de la columna “Priority” y “Facility”, que como vimos son números, por su palabra correspondiente. En éste método utilizo un simple switch, y lo creo en el archivo del modelo Log.php.

```
public function reemplazarPriority($num)
{
    switch($num)
    {
        case 0:
            return 'EMERGENCY';
            break;
        case 1:
            return 'ALERT';
            break;
        case 2:
            return 'CRITICAL';
            break;
        case 3:
            return 'ERROR';
            break;
        case 4:
            return 'WARNING';
            break;
        case 5:
            return 'NOTICE';
            break;
        case 6:
            return 'INFO';
            break;
        default:
            return 'DEBUG';
            break;
    }
}
```

Para diferenciar dichos niveles de prioridad por colores más llamativos cuanto más alto sea el nivel de alarma, utilizo el css para “colorear” el fondo de las celdas de la tabla. Tirando de manual de yii, veo que los elementos de la tabla tienen atributos que permiten establecer el nombre de la “class” css, obviamente en el css debo escribir una clase para cada caso de “Priority” y “Facility”. A continuación pongo el código de la tabla completa.

```

$this->widget('zii.widgets.grid.CGridView', array(
    'id'=>'log-grid',
    'dataProvider'=>$model->search(),
    'filter'=>$model,
    //'pager'=>array('pageSize'=>100), Prueba para mostrar mas registros por pagina, solucion en el search() del
    'columns'=>array(
        array(
            'name'=>'ReceivedAt',
            'value'=>'Log::model()->editarFechaMensajes($data->ReceivedAt)',
            'htmlOptions'=>array('style'=>'width: 150px; text-align: center;'),
        ),
        array(
            'name'=>'Facility',
            'value'=>'Log::model()->reemplazarFacility($data->Facility)',
            'cssClassExpression'=>'Log::model()->reemplazarFacility($data->Facility)',//propiedad que crea una clase
        ),
        array(
            'name'=>'Priority',
            'value'=>'Log::model()->reemplazarPriority($data->Priority)',
            'cssClassExpression'=>'Log::model()->reemplazarPriority($data->Priority)',
        ),
        array(
            'name'=>'FromHost',
            'htmlOptions'=>array('style'=>'width: 100px;'),
        ),
        'SysLogTag',
        array(
            'name'=>'Message',
            'value'=>'substr($data->Message, 0, 100)',//limita la longitud del mensaje de log a 100 caracteres
        ),
        ..
    ),
);

```

Como se puede ver existe otro método llamado “editarFechaMensajes”, que pongo a continuación, con el que edito el formato de la fecha que devuelve Syslog. Lo que hace este método es calcular si la fecha del mensaje corresponde al día de actual para poner la palabra “Hoy”.

```

public function editarFechaMensajes($fechamen)
{
    $sub1=substr($fechamen, 0, 10);
    $sub2=substr($fechamen, 11, 20);
    $fecha=date("Y-m-d");
    if(strcmp($sub1, $fecha)==0)
    {return 'Hoy - '.$sub2;}
    else return $fechamen; //hay que devolver siempre algo, en
este caso la fecha sin modificar
}

```

También modifiqué la longitud a mostrar del mensaje de Log a 100 caracteres con el método de php substr(). El mensaje completo se puede ver pinchando en el icono de la lupa si necesitamos saber la información al completo, corresponde a otra vista del directorio Log. (_view)

Se me planteaba un nuevo problema o reto a la vista, debía conseguir que la página principal, donde se veían los últimos 100 logs, se actualizara periódicamente. A simple vista parece un problema de fácil solución y eso me parecía a mi también...

Lo primero que pensé fue en actualizar la página completamente, para lo que encontré diversas alternativas, una de ellas era a través de código HTML, por ejemplo con la etiqueta meta:

```
<meta http-equiv="refresh" content="600">
```

La tabla que muestra los logs, es un elemento propio de Yii llamado “GridView”, y posee una serie de filtros de búsqueda en la parte superior de cada columna, mi intención es que al refrescarse la página se mantenga cualquier valor que hayamos introducido en esos filtros. (Imaginemos que estamos filtrando los logs de un equipo concreto o la prioridad ‘Alert’ solamente). Por tanto con esta etiqueta no consigo mi objetivo ya que se refresca la página entera y no se mantienen los valores de los filtros.

También vi muy útil la idea de poner un desplegable donde poder elegir una cantidad de segundos tras los cuales queremos que se refresque la información.

Al ver que refrescar la página completa no es la solución, me centro en refrescar solamente la tabla (GridView), por tanto pienso que debo recurrir (muy a mi pesar) a programar con Javascript.

Revisando la documentación de la clase GridView de Yii, descubro que este tipo de elementos posee un método propio que actualiza la tabla y otro método que me devuelve los valores de los filtros que haya en cada momento. Tras este descubrimiento, que llevó su tiempo, la solución se reduce a crear un método Javascript que reciba como parámetro el número de segundos deseado (elegido en un simple desplegable) y que realice una llamada al método que actualiza la tabla periódicamente cada X segundos, eso sí, manteniendo los valores filtrados. Este script se ejecuta en el momento que se elige un valor de segundos en el desplegable, por lo que lo asocio al método onchange() del elemento. A continuación el código del script:

```
var t;

function refresh(numero)
{
    window.clearInterval(t);
    if (numero!='No')
    {
        var tiempo=numero*1000;
        t=setInterval("$.fn.yiiGridView.update('log-grid',
        {url:$.fn.yiiGridView.getUrl('log-grid'), data:
        $(this).serialize()})", tiempo);
    }
}
```

Es importante declarar la variable ‘t’ fuera del método para poder usar correctamente la función ‘window.clearInterval()’, que elimina cualquier intervalo que estuviese activo. Como podemos observar se recibe el valor desde el desplegable, y si es distinto de la palabra ‘No’, se establece un intervalo que irá actualizando la tabla.

Una vez pude mostrar la vista de Logs como quería la convertí en el index de mi aplicación utilizando el método *actionIndex()* del controlador general de la página (/protected/controller/SiteController.php).

La página principal de la aplicación queda como se ve en las capturas siguientes.

Servidor de logs TVA

[Logs](#) | [Host](#) | [Alarma](#) | [Gráficas](#) | [Logout \(admin\)](#)

[Home](#) » [Logs](#) » Administrar

[Advanced Search](#)
 Actualizar página:

Displaying 1-100 of 2188573 results.

Received At	Facility	Priority	From Host	Sys Log Tag	Message
Hoy - 16:47:14		INFO	192.168.152.7	16:	47:14.005 [R.nad-info-updater] INFO com.avenda.tips.snmpserver.snmptasks.NadInfoUpdaterBackgroundTa
Hoy - 16:47:12	SECURITY	INFO	bewall	proftpd:	pam_unix(proftpd:session): session closed for user flptvalmansa
Hoy - 16:47:02	SECURITY	INFO	bewall	proftpd:	pam_unix(proftpd:session): session opened for user flptvalmansa by (uid=0)
Hoy - 16:46:54	SECURITY	INFO	bewall	proftpd:	pam_unix(proftpd:session): session closed for user flptvalmansa
Hoy - 16:46:53	SECURITY	INFO	bewall	proftpd:	pam_unix(proftpd:session): session opened for user flptvalmansa by (uid=0)
Hoy - 16:46:37		ERROR	192.168.152.7	16:	46:37.192 [main SessId R0005ec1d-01-5421879a] ERROR RadiusServer.Radius - reqst_clean_list: Deleting
Hoy - 16:46:37		ERROR	192.168.152.7	16:	46:37.192 [main SessId R0005ec1d-01-5421879a] ERROR RadiusServer.Radius - reqst_clean_list: Packet 7
Hoy - 16:46:37		ERROR	192.168.152.7	16:	46:37.192 [main SessId R0005ec1d-01-5421879a] ERROR RadiusServer.Radius - reqst_clean_list: Packet 8
Hoy - 16:46:37		ERROR	192.168.152.7	16:	46:37.192 [main SessId R0005ec1d-01-5421879a] ERROR RadiusServer.Radius - reqst_clean_list: Packet 7
Hoy - 16:46:37		ERROR	192.168.152.7	16:	46:37.192 [main SessId R0005ec1d-01-5421879a] ERROR RadiusServer.Radius - reqst_clean_list: Packet 8
Hoy - 16:46:37		ERROR	192.168.152.7	16:	46:37.194 [RequestHandler-1-0x7f169596a700 r=auto-865510 h=65 r=R0005ec1d-01-5421879a] ERROR Common.
Hoy - 16:46:37		ERROR	192.168.152.7	16:	46:37.194 [RequestHandler-1-0x7f169596a700 r=auto-865510 h=65 r=R0005ec1d-01-5421879a] ERROR Common.
Hoy - 16:46:37		WARNING	192.168.152.7	16:	46:37.194 [RequestHandler-1-0x7f169596a700 r=auto-865510 h=65 r=R0005ec1d-01-5421879a] WARN Common.

[Home](#) » [Logs](#) » Administrar

[Advanced Search](#)

Actualizar página:

Received At	Facility	Priority	From Host	Sys Log Tag	Message
		5	172.16.20.9		
2014-09-24 23:15:28	SECURITY	NOTICE	172.16.20.9	dropbear[4699]:	pubkey auth succeeded for 'mcuser' with key md5 10:25:54:43
2014-09-23 23:15:23	SECURITY	NOTICE	172.16.20.9	dropbear[25659]:	pubkey auth succeeded for 'mcuser' with key md5 10:25:54:43
2014-09-23 11:15:11	SECURITY	NOTICE	172.16.20.9	dropbear[15054]:	pubkey auth succeeded for 'mcuser' with key md5 10:25:54:43
2014-09-23 11:14:58	SECURITY	NOTICE	172.16.20.9	dropbear[15045]:	pubkey auth succeeded for 'mcuser' with key md5 10:25:54:43
2014-09-23 11:14:45	SECURITY	NOTICE	172.16.20.9	dropbear[15038]:	pubkey auth succeeded for 'mcuser' with key md5 10:25:54:43
2014-09-23 11:14:32	SECURITY	NOTICE	172.16.20.9	dropbear[15030]:	pubkey auth succeeded for 'mcuser' with key md5 10:25:54:43
2014-09-23 11:14:19	SECURITY	NOTICE	172.16.20.9	dropbear[15023]:	pubkey auth succeeded for 'mcuser' with key md5 10:25:54:43
2014-09-23 11:14:13	SECURITY	NOTICE	172.16.20.9	dropbear[15010]:	pubkey auth succeeded for 'mcuser' with key md5 10:25:54:43
2014-09-23 11:14:09	SECURITY	NOTICE	172.16.20.9	dropbear[14963]:	pubkey auth succeeded for 'mcuser' with key md5 10:25:54:43
2014-09-23 11:13:22	USER	NOTICE	172.16.20.9	system:	Stop
2014-09-23 11:12:45	SECURITY	NOTICE	172.16.20.9	dropbear[14504]:	pubkey auth succeeded for 'mcuser' with key md5 10:25:54:43
2014-09-22 23:15:20	SECURITY	NOTICE	172.16.20.9	dropbear[13064]:	pubkey auth succeeded for 'mcuser' with key md5 10:25:54:43

7. Aplicación web. Modelo Host

Con los Logs configurados, me centro en crear otra base de datos en la cual almacenaré los equipos a los que voy a permitir enviar logs a mi servidor. La idea es tener los equipos con sus características en un listado para después permitir el envío de logs solamente a los equipos de esa lista, lo que conseguiré utilizando un firewall basado en IPTABLES en el servidor.

En primer lugar creo la tabla Host, donde almacenaré los equipos. Los campos de la tabla son: ID, Nombre del equipo, Descripción, IP, Prioridad. Establezco como índice el ID que será un “Auto_increment” y también que el campo IP sea único.

Es necesario crear una nueva conexión a la base de datos tal como se hizo con la primera en el archivo /protected/config/main.php.

Una vez definida la tabla, creo el modelo y el CRUD del mismo modo descrito anteriormente con la tabla de logs, quedando el formulario de creación de un Host, del siguiente modo:

The screenshot shows a web application interface for 'Servidor de logs TVA'. At the top, there is a navigation bar with tabs for 'Logs', 'Host' (which is active), 'Alarma', 'Gráficas', and 'Logout (admin)'. Below the navigation bar, there is a breadcrumb trail: 'Home » Hosts » Añadir'. The main heading is 'Añadir Host'. A note indicates 'Campos con * requeridos.' (Fields with * are required). The form contains the following fields: 'Nombre (Hostname) *' (text input), 'Descripcion *' (text area), 'Ip *' (IP address input field), 'Prioridad' (dropdown menu with '-- Select Options --'), 'SNMP Public' (text input with 'public' selected), and 'SNMP Private' (text input with 'private' selected). At the bottom of the form is an 'Añadir' button.

A destacar que el campo IP se almacena en la base de datos como “INT UNSIGNED” y no como “VARCHAR(15)” como se suele hacer. De este modo utilizando el comando INET_ATON de MySQL podremos almacenar la IP codificada en una cadena de 4 bytes, para posteriormente recuperarla con el comando contrario INET_NTOA. Es un modo más seguro de almacenar las IP’s.

NOTA: En la captura aparecen otros campos como SNMP Public y SNMP Private, que se describirán posteriormente en esta memoria.

Una vez se rellena el formulario, el botón invoca a la acción “Create” del controlador del modelo Host. /protected/controllers/HostController.php que se muestra a continuación:

```
public function actionCreate()
{
    $model=new Host;
    if(isset($_POST['Host']))
    {
        /*-----PARTE DE LA IP-----*/
        $cadena=$_POST['ip1'].".".$_POST['ip2'].".".$_POST['ip3'].".".$_POST['ip4'];//Se obtiene la ip formateada del form
        $ipe=ip2long($cadena);// se transforma a entero y si fuera negativo entra en el if a continuacion para transformarlo
        if($ipe<0)
        {
            $ipe=$_POST['ip4']+$_POST['ip3']*256+$_POST['ip2']*65536+$_POST['ip1']*16777216;
        }
        $model->ip=$ipe;//aqui se asigna el valor de la ip insertada al atributo del modelo 'ip'
        /*-----PARTE DE LA IP-----*/

        /*-----PARTE DE PRIORIDAD-----*/
        if(!empty($_POST['Host']['priority2']))
        {
            $priority=$_POST['Host']['priority2'];
            $model->Prioridad=implode(',',$priority);
        }else $model->Prioridad='0,1,2,3,4,5,6,7';
        /*-----PARTE DE PRIORIDAD-----*/
        $model->attributes=$_POST['Host'];
        if($model->save()){
            //Aqui modifico el archivo firewall.sh para permitir los envios del host correspondiente
            $ipformateada=long2ip($model->ip);
            $linea="/sbin/iptables -A INPUT -s ".$ipformateada." -p udp -m udp --dport 514 -j ACCEPT";
            $comandos=Yii::app()->ssh->ejecuta(array("/var/www/logstva/protected/commands/sustituirlinea.sh $linea", "/t
            //print_r ($comandos);
            if(strcmp(trim($comandos[0]), "escrito")!=0)
            {
                $model->delete();
                $comandos=Yii::app()->ssh->ejecuta(array("/var/www/logstva/protected/commands/borrarlinea.sh $linea"
                Yii::app()->user->setFlash('error', "No se ha guardado el Host!");
            }
        }else{
            Yii::app()->user->setFlash('success', "El Host se ha guardado!");
        }
    }
}
```

Si observamos como trato el campo IP, vemos que antes de guardarlo en la base de datos, (`$model->save()`) concateno los campos de la ip para transformarla a la forma xxx.xxx.xxx.xxx y después aplicar el método `ip2long()` (es equivalente al mencionado `inet_aton()` de MySQL) y guardarla como un entero. Si el valor calculado excede el rango, el entero será negativo por lo que lo compruebo y si es así se transforma a positivo multiplicando por su valor correspondiente cada uno de los segmentos de la ip. Guardo finalmente en la variable ip del modelo el resultado final. `$model->ip=...`

La prioridad como se ve en el formulario la gestiono con un desplegable, para hacerlo más visual y cómodo a mi juicio, utilizo la primera extensión de yii. Se utiliza en el `/view/_form`, y me permite almacenar las prioridades seleccionadas como un array. Al enviar el formulario, en el controlador vemos como compruebo la variable `priority2` para ver si está vacía, si lo estuviera le asigno el valor `'0,1,2,3,4,5,6,7'` (todas las prioridades). Si por el contrario la variable no es vacía, se transforma en un string y se guarda en `$model->priority=`

Una parte muy importante de lo que ocurre cuando se crea un Host a través del formulario, es como le permitimos a ese equipo que sus logs sean recibidos por nuestro servidor. Se necesita añadir una línea al iptables para permitir dichas conexiones, para

lo cual utilizo un script que busca un determinado patrón en el archivo firewall.sh y agrega la línea. La ejecución del script se realiza gracias a una extensión de Yii que permite la conexión por ssh al equipo que le digamos en nuestro archivo de configuración general /protected/config/main.php. Las extensiones en Yii las encontramos en la carpeta /protected/extensions y los scripts se agregan en la carpeta /protected/commands. Para la realización de los scripts estuve familiarizandome con el comando 'sed' y realizando numerosas pruebas en local hasta obtener el resultado deseado. A continuación añado el script que se ejecuta al crear un Host y el script contrario que se ejecutaría al borrar un Host y que debe hacer justo lo contrario.

Script sustituirlinea.sh (busca el patron ##aqui)

```
#!/bin/bash

line=$1; #guardo el primer parametro que recibe el script
/bin/sed "s|##aqui|$line\n##aqui|" /etc/init.d/firewall.sh >
/tmp/temp.txt; #sustituyo y redirijo la salida a un archivo de texto
temp.txt
/bin/cat /tmp/temp.txt > /etc/init.d/firewall.sh; #sobreescribo el
archivo deseado
/bin/echo 'escrito'
```

Script borrarlinea.sh (borrará la línea donde aparezca el Host para impedir el acceso y consecuentemente el envío de logs)

```
#!/bin/bash

line=$1; #guardo el primer parametro que recibe el script
##/bin/echo $line; #imprimo para ver que se ha guardado correctamente
/bin/sed "s|$line|##|" /etc/init.d/firewall.sh > /tmp/temp2.txt;
#sustituyo la línea por un espacio y redirijo la salida a un archivo de
texto temp.txt
/bin/cat /tmp/temp2.txt > /etc/init.d/firewall.sh; #sobreescribo el
archivo deseado
/bin/echo 'borrado'
```

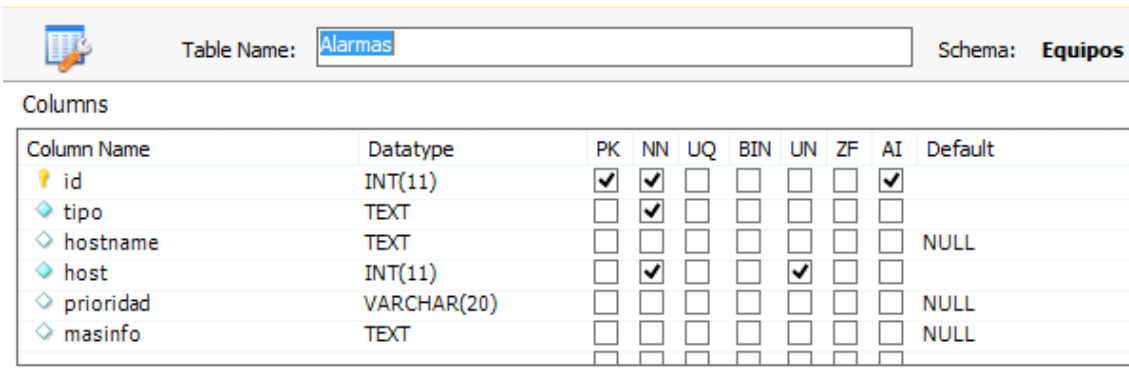
En los scripts algunos de los problemas que tuve estuvieron relacionados con el comando sed. En el script sustituir línea, el comando sed busca el patrón en el archivo firewall.sh y cuando lo encuentra añade la línea que viene en la variable \$line, y la salida del comando la redirijo a un archivo de texto intermedio ya que en sed no puedo dirigirla al mismo archivo en el que se está buscando. Hasta que no descubrí esto, tuve serios problemas para hacer funcionar al script correctamente.

Como se puede ver en la última imagen, se captura la salida de los scripts para notificar que se ha creado bien un Host o no mediante el método setFlash() de yii, que nos muestra un aviso visual en la aplicación.

8. Aplicación web. Alarmas

Una de las implementaciones más útil de mi servidor de logs, es el envío de alarmas por diferentes vías. Mi idea era que en el momento llegase un log desde un equipo que requiera atención inmediata, al instante nos salte una alarma que nos advierta.

Al igual que en los modelos anteriores, creo primero la tabla Alarmas en la base de datos, con los campos id, tipo, host, hostname prioridad y un campo “masinfo”, que al principio no agregué y lo hice posteriormente cuando vi que era necesario. Más adelante explico que quiere decir este campo.



Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
id	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
tipo	TEXT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
hostname	TEXT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
host	INT(11)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
prioridad	VARCHAR(20)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
masinfo	TEXT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL

En la imagen de arriba vemos la estructura de la tabla entrando al servidor MySQL con el programa MySQL Workbench, un programa muy útil e intuitivo para gestionar bases de datos, realizar consultas, etc... que utilizo muy a menudo.

El campo ‘tipo’ será el modo de avisar de la alarma, por ejemplo vía SMS. En el campo ‘host’ tendremos la ip del equipo a monitorizar, en ‘hostname’ el nombre para identificar al equipo, en prioridad tendremos el grado de prioridad para el que debe saltar la alarma y en ‘masinfo’, información adicional, es decir, si el tipo de alarma es SMS en el campo ‘masinfo’ pondremos el número de teléfono, si es vía email, la dirección de correo, etc. Una vez terminada la tabla, creo el modelo y el CRUD en yii.

A continuación hago una serie de modificaciones en la vista del formulario para crear una alarma, como por ejemplo:

- El tipo de alarma se selecciona en un desplegable que limita a: HttpGet, HttpPost, SMS, Email.
- El campo ‘Información adicional’ se completa, bien con un n° de teléfono, un Email o una url.
- El Host se selecciona de un desplegable que limita al listado de Equipos o Host que tenemos creado en ese momento en la aplicación. Estos datos se obtienen de una consulta al modelo ‘Host’
- La prioridad es otro desplegable que limita a elegir entre las prioridades posibles en Syslog.

Servidor de logs TVA



[Home](#) » [Alarmas](#) » [Crear](#)

Crear Alarma

*Campos con * requeridos.*

Tipo *

Información Adicional (Email, telefono o url)

Host *

Prioridad *

Puede parecer que me olvido del campo IP pero como vemos en la siguiente captura, en el desplegable de Host, tenemos concatenados los nombres de los equipos con sus respectivas ips, por tanto solo tengo que utilizar el método explode() de php para guardar cada campo en su lugar correspondiente al almacenar la alarma.

Tipo *

Información Adicional (Email, telefono o url)

Host *

- Select Option --
- host1/111.111.111.111
- host2/111.111.111.112
- Original/192.111.111.111
- logs/89.29.128.37
- Mojinete/89.29.128.44
- SAI Cuesta Castilla/89.29.154.50
- Controlador Virtual/192.168.152.10
- Controlador01/192.168.152.8**
- CPPM/192.168.152.7
- Controlador02/192.168.152.9
- Microtik/192.168.203.2
- Ubiquiti TVAPA1/172.16.20.9
- ALU7342/192.168.4.10
- Calix E7-20/192.168.18.2

El funcionamiento esencial de la alarma, debe ser el siguiente: debe realizarse una consulta a la base de datos de logs filtrando con los datos correspondientes a la alarma (prioridad y hostname). Los logs que se obtengan cumplirán los requisitos para lanzar una alarma. Esto lo consigo con dos métodos cuyo código se muestra a continuación:

```
/*
 * Accion que recibe un id para cargar el modelo alarma y realizar una consulta a la tabla de log(Syslog.SystemEvents)
 para | filtrar los logs que cumplen la condicion impuesta por la alarma
 * Devuelve un array con los datos de los logs, estructurados por etiquetas, que cumplen las condiciones de la alarma.
 * @return array de Logs
 */
public function actionFiltrarAlarma($id)
{
    $model=$this->loadModel($id);

    $rawData=Yii::app()->db->createCommand("SELECT id, ReceivedAt, Facility, Priority, FromHost, SysLogTag, Message
    FROM Syslog.SystemEvents WHERE Priority=".$model->prioridad." AND FromHost='".$model->hostname.'"
    AND TIMESTAMPDIFF(SECOND,ReceivedAt,now())<60")->queryAll();

    $dataProvider=new CActiveDataProvider($rawData, array('pagination'=>false));

    $array=$dataProvider->getData();
    return $array;
}
```

En la acción, que se muestra en la captura superior se ve la consulta a la base de datos que devuelve un array de logs. Como vemos en la consulta se seleccionan solo los logs recibidos en el último minuto. La llamada a este método se realiza en el siguiente como podemos ver:

```
public function actionLanzarAlarma()
{
    $rawData=Yii::app()->db->createCommand("SELECT * FROM Equipos.Alarmas")->queryAll();
    $dataProvider=new CActiveDataProvider($rawData, array('pagination'=>false));
    $array=$dataProvider->getData();//array con las alarmas
    foreach($array as $c => $v)
    {
        $array2=$this->actionFiltrarAlarma($array[$c]['id']);
        if(!empty($array2))
        {
            foreach($array2 as $c2 => $v2)
            {
                if (strcmp($array[$c]['tipo'],'SMS')==0)
                {
                    //llamada al metodo correspondiente alarmaSMS($array2[$c2])
                    Alarma::model()->alarmaSMS($array2[$c2], $array[$c]['host'],$array[$c]['masinfo']);
                }else if (strcmp($array[$c]['tipo'],'Email')==0)
                {
                    Alarma::model()->alarmaEmail($array2[$c2], $array[$c]['host'],$array[$c]['masinfo']);
                }else if (strcmp($array[$c]['tipo'],'HTTP Post')==0)
                {
                    Alarma::model()->alarmaHttpPost($array2[$c2], $array[$c]['host'],$array[$c]['masinfo']);
                }else if (strcmp($array[$c]['tipo'],'HTTP Get')==0)
                {
                    Alarma::model()->alarmaHttpGet($array2[$c2], $array[$c]['host'],$array[$c]['masinfo']);
                }else if (strcmp($array[$c]['tipo'],'SNMP trap')==0)
                {
                    Alarma::model()->alarmaSNMP($array2[$c2], $array[$c]['host'],$array[$c]['masinfo']);
                }
            }
        }
    }
}
```

Resumiendo, este método, recorre las alarmas una a una y realiza el filtrado de los logs que “cumplen” los requisitos llamando al método filtrarAlarma() ya mencionado. Una vez se obtiene el array de logs, se recorre y se lanza la acción correspondiente en base al ‘tipo’ de alarma. Como vemos existe un método para cada tipo de alarma.

Los métodos para los tipos HttpGet y HttpPost, básicamente crean una url mediante la biblioteca curl y pasa los parámetros por POST. La elaboración de estos métodos me permitió aprender sobre CURL y sus posibilidades. Para realizarlos me basé en los manuales de php para curl. A continuación muestro el método HttpPost a modo de ejemplo ya que son muy similares:

```
public function alarmaHttpPost($arrayLog, $ip, $url)
{
    $ch = curl_init();
    $post = '';
    foreach($arrayLog as $name => $value) {
        $post .= $name.'='.$value.'&';
    }
    $post.= 'ip='.long2ip($ip);
    //rtrim($post, '&');
    curl_setopt($ch, CURLOPT_POSTFIELDS, $post);
    curl_setopt($ch, CURLOPT_URL, $url);
    curl_setopt($ch, CURLOPT_HEADER, 0);
    curl_setopt($ch, CURLOPT_POST, 1);
    curl_exec($ch);
    curl_close($ch);
}
```

Para probar cada uno de estos métodos, creo un par de pequeños archivos .php. Cuya ruta utilizo como la url del campo 'masinfo'. Si tomamos como ejemplo este método, cuando un log cumpla la condición, se realizará una llamada a la url de mi script y enviará los datos del log por POST. Es script es el siguiente:

```
<div style="text-align:center">
</div>
<?php print_r ($_POST);?>
```

Para el tipo de alarma SMS utilizo un modem con una tarjeta SIM propiedad de la empresa:

```
public function alarmaSMS($arrayLog, $ip, $tlf)
{
    $mensaje='';

    $priority=Log::model()-
>reemplazarPriority($arrayLog['Priority']);
    $facility=Log::model()-
>reemplazarFacility($arrayLog['Facility']);

    $mensaje.=$facility.'/'.$priority.' en host:
'.$arrayLog['FromHost'].' con ip= '.long2ip($ip).' Mensaje:
'.$arrayLog['Message'];
```

```

        if(strlen($mensaje)>160)
        {
            $mensaje=substr($mensaje, 0, 160);
            $comandos=Yii::app()->ssh2->ejecuta(array("echo
$mensaje | gnokii --sendsms $tlf"));
            }else $comandos=Yii::app()->ssh2->ejecuta(array("echo
$mensaje | gnokii --sendsms $tlf"));
        }
    }

```

Como se puede ver se construye el mensaje y se realiza una llamada por ssh a un equipo que tiene instalado gnokii, un programa para el modem con la tarjeta SIM. Los parámetros son el propio mensaje y el número de teléfono del campo 'masinfo'.

Para el tipo de alarma 'email', se construye el email con sus cabeceras y se utiliza una extensión de Yii que nos permite mandar emails.

```

public function alarmaEmail($arrayLog, $ip, $email)
{
    //ejecutar la extension KEmail para enviar email
    // public function
    send($from,$to,$subject,$body,$additional_headers=array()
        $headers = array(
            'MIME-Version: 1.0',
            'Content-type: text/html; charset=iso-8859-1'
        );
        $from='no-reply@tvalmansa.es';
        $to=$email;

        $priority=Log::model()-
>reemplazarPriority($arrayLog['Priority']);
        $facility=Log::model()-
>reemplazarFacility($arrayLog['Facility']);
        $subject=$priority.' en el Host: '.$arrayLog['FromHost'];

        $body='<html><head><title>'.$subject.'</title></head><body>Detalles de la alarma: <br>-----<br>';
        $body.='Hora: '.$arrayLog['ReceivedAt'].'<br>Host:
'.$arrayLog['FromHost'].'<br>IP: '.long2ip($ip).'

```

Una vez creados estos métodos, surge la necesidad de revisar periódicamente si existen alarmas, es decir, que se ejecute el método lanzarAlarma() con una periodicidad programada. ¿Cómo puedo hacerlo? Es aquí donde aparece CRON.

9. CRON

En Unix, Cron es un administrador regular de procesos en segundo plano (demonio) que ejecuta procesos o guiones a intervalos regulares (por ejemplo, cada minuto, día, semana o mes). Los procesos que deben ejecutarse y la hora en la que deben hacerlo se especifican en el fichero crontab. El nombre cron viene del griego chronos(χρόνος) que significa "tiempo" y que se encuentra en el directorio /etc.

Cron se podría definir como el "equivalente" a Tareas Programadas de Windows.

El formato de configuración de cron es muy sencillo.

- El símbolo almohadilla «#» es un comentario, todo lo que se encuentre después de ese carácter no será ejecutado por *cron*.
- El momento de ejecución se especifica de acuerdo con la siguiente tabla:
 1. Minutos: (0-59)
 2. Horas: (0-23)
 3. Días: (1-31)
 4. Mes: (1-12)
 5. Día de la semana: (0-6), siendo 1=lunes, 2=martes,... 6=sábado y 0=domingo (a veces también 7=domingo)

```
#####
#minuto (0-59), #
#| hora (0-23), #
#| | día del mes (1-31), #
#| | | mes (1-12), #
#| | | | día de la semana (0-6 donde 0=Domingo) #
#| | | | | comandos #
#####
15 02 * * *
```

Para especificar todos los valores posibles de una variable se utiliza un asterisco (*).

- La última columna corresponde a la ruta absoluta del binario o *script* que se quiere ejecutar.

A continuación se puede ver el contenido de mi archivo crontab. Vemos como el método lanzarAlarma() se ejecuta cada minuto, consiguiendo así el objetivo buscado.

```
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# m h dom mon dow user  command
17 * * * * root    cd / && run-parts --report /etc/cron.hourly
25 6 * * * root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6 * * 7 root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6 1 * * root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly
)
* * * * * root    /usr/bin/curl http://89.29.128.37/logstva/index.php?r=Alarma/LanzarAlarma
#
```

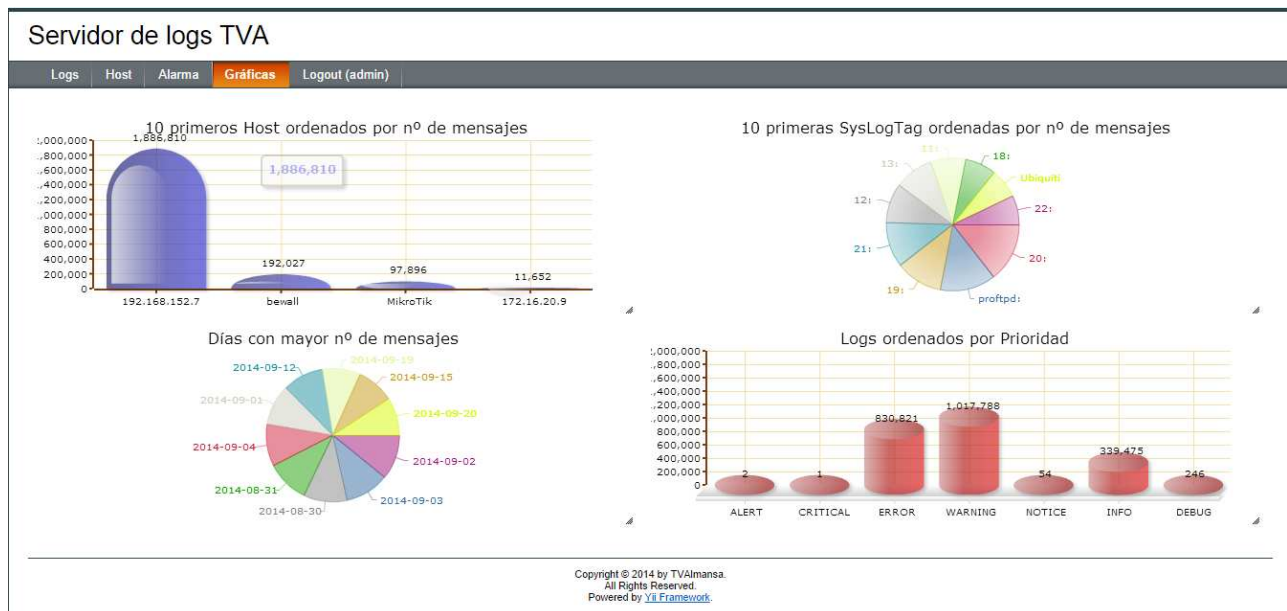

10. Aplicación web. Gráficas.

En una aplicación como esta, resultaba lógico poder hacer un análisis general de los logs recibidos, es decir, poder comprobar de un vistazo qué equipos mandan más logs, qué tipo de logs llegan con más frecuencia o qué días recibimos más logs... es por esto que opto por crear una sección con varias gráficas de análisis de este tipo de datos de mi servidor.

Para mostrar las gráficas recurro al repositorio de extensiones de Yii, confiando en algún widget que permita la creación y configuración de las mismas.

En el modelo Log, /protected/models/Log.php creo un método llamado consultaGráficas(), que realiza 4 consultas a la base de datos de logs (tantas como gráficas distintas se quieran realizar) recibiendo como parámetro de entrada un número (del 0 al 3 en este caso) para elegir la consulta deseada.

En el Controller principal de la página, realizo otro método, apoyándome en la estructura de la extensión de Yii, que tiene sus propias funciones y atributos. Este método llama al anterior y construye una gráfica para cada consulta y le pasa a la vista /protected/views/site/graficas.php, los datos necesarios para “pintar” las gráficas. El resultado final se muestra en la siguiente captura:



11. Posibles ampliaciones. SNMP

Este servidor es un proyecto sujeto a posibles ampliaciones para mejorar utilidad y características. Una de las ideas para mejorar el proyecto consiste en poder obtener datos de los Host que tenemos almacenados vía SNMP.

SNMP o Simple Network Management Protocol, es un protocolo de la capa de aplicación que facilita el intercambio de información de administración entre dispositivos de red. Los dispositivos que normalmente soportan SNMP incluyen routers, switches, servidores, estaciones de trabajo, impresoras, bastidores de módem y muchos más. Permite a los administradores supervisar el funcionamiento de la red, buscar y resolver sus problemas, y planear su crecimiento.

Una Base de Información de Administración (Management Information Base, MIB) es una colección de información que está organizada jerárquicamente. Las MIB's son accedidas usando un protocolo de administración de red, como por ejemplo, SNMP. Por ejemplo, en una determinada MIB podemos acceder al valor de la temperatura del procesador de cierto equipo, o al porcentaje de uso de memoria.

La idea es poder realizar consultas SNMP a equipos para monitorizarlos y detectar posibles averías. Desde el propio servidor y utilizando un cliente snmp para Ubuntu, es posible realizar dichas consultas.

```
sudo apt-get install snmp snmpd
```

La posible ampliación de mi aplicación consiste en aprovechar los datos de los Host que tenemos almacenados para crear acciones o métodos que realicen consultas SNMP. Una manera de hacerlo sería similar al lanzamiento de alarmas, realizando un método con PHP que realice consultas de este estilo. Es importante remarcar que sería necesario disponer de plantillas para acceder a estas MIB's, ya que cada fabricante posee una estructura jerárquica distinta.

```
ID: 4
Nombre (Hostname): logs
Descripción: este servidor
Ip: 89.29.128.37
Prioridad: 0
SNMP:
Percentage of user CPU time: INTEGER: 55%
Percentages of system CPU time: INTEGER: 7%
Percentages of idle CPU time: INTEGER: 32%
Total RAM in machine: INTEGER: 1024076 Bytes
Total RAM used: INTEGER: 86016 Bytes
Total RAM Free: INTEGER: 1107116 Bytes
Total RAM Shared: INTEGER: 0 Bytes
Total RAM Buffered: INTEGER: 8144 Bytes
Available space on the disk: INTEGER: 101204512 Bytes
Used space on the disk: INTEGER: 7096404 Bytes
Percentage of space used on disk: INTEGER: 7%
Percentage of inodes used on disk: INTEGER: 4%
System Uptime: Timeticks: (62257027) 7 days, 4:56:10.27
```

12. Conclusiones.

En general me siento muy satisfecho con este proyecto, me ha servido para ampliar enormemente mis conocimientos y para consolidar, perfeccionar y poner en práctica lo que ya sabía.

He aprendido a trabajar con un framework, que facilita ampliamente la realización de aplicaciones web. He tenido que familiarizarme con su documentación, funciones y elementos propios. Al estar basado en PHP, he perfeccionado la programación en este lenguaje a la vez que he aprendido métodos nuevos.

En resumidas cuentas pienso que es un proyecto muy completo ya que engloba diferentes campos; desde la preparación de un equipo desde cero, configurando un sistema Linux, hasta la programación en diferentes lenguajes, gestión de bases de datos, creación de firewalls y administración de redes, etc.

Debo destacar que en la empresa donde he desarrollado el proyecto, he tenido apoyo en todo momento tanto en infraestructuras como en dudas o problemas que me hayan podido surgir. El SERVIDOR CENTRALIZADO DE LOGS está siendo utilizado a día de hoy por la empresa, algo de lo que me siento orgulloso.



