SAWESO

Sistema de Automatización de Workarounds

y Ejecución de Scripts Ocasionales

TRABAJO FINAL DE CARRERA - Plataforma GNU/Linux

Autor: José Antonio Serrano Rosso

Fecha: 13-01-2013

UOC
Universitat Oberta de Catalunya

Índice de contenido

1. Marco del proyecto

En el funcionamiento diario de una plataforma formada por 3.000 servidores Debian y 500.000 equipos clientes Ubuntu se presentan un gran números de incidencias cuyo proceso de resolución es conocido y esta documentado. Por cada tipología de incidencia, el número de servidores o equipos afectados supera, en la mayoría de los casos, las 20 máquinas. Estas incidencias se solventan mediante el acceso remoto, utilizando un cliente de telnet seguro (SSH), y la ejecución manual de los procedimientos documentados.

Por un lado, el tiempo invertido en el proceso de resolución manual de estas incidencias "máquina por máquina", así como el seguimiento de las mismas, requiere una gran inversión de horas/hombre y disminuye las probabilidades de solventar todas las incidencias en los tiempos establecidos por los SLA (Acuerdos de nivel de servicio) con el cliente.

Por otro lado, en muchas ocasiones se requiere la obtención de datos estadísticos sobre fallos generalizados en las máquinas de la plataforma (servidores o equipos clientes), la recolección de estos datos y su posterior filtrado según las necesidades puntuales del problema que se esta analizando.

2. Alcance del proyecto

Diseñar una herramienta centralizada que controle la ejecución desatendida de comandos y scripts sobre servidores (Debian o Ubuntu) y equipos clientes (Debian o Ubuntu) y que permita, la descarga de los resultados obtenidos, el control del estado de la ejecución y la generación de estadísticas e informes.

3. Características del sistema

Para cumplir con los requisitos utilizaremos las siguientes técnicas:

- ✔ Cliente SSH y certificado "dsa", para el acceso seguro a los servidores de la plataforma.
- ✔ SSH Tunneling y certificado "rsa", para el acceso a los equipos clientes que se encuentran dentro de una intranet remota dotada de un servidor de seguridad.
- ✔ Shell Scripting, para diseñar el motor del sistema que realizará la copia remota ("scp") y ejecución ("bash") de los scripts necesarios.
- ✔ Mailing (mediante "mailx" y "postfix"), para enviar estadísticas e informes generales al acabar la ejecución.
- ✔ Interfaz web mediante Servlets y Java Server Pages (".jsp"), para proporcionar una completa interfaz de usuario.
- ✔ Servidor de aplicaciones web Tomcat, como servidor de sevlets y motor JSP.
- ✔ Base de datos MySQL, para almacenar los datos y estado de la ejecución y para controlar el acceso y los permisos de los diferentes perfiles de usuario que utilizarán la interfaz web.

El diseño del sistema debe ser lo suficientemente flexible para permitir:

- ✔ Seleccionar el script a ejecutar dentro de un directorio de scripts, diferenciando las ejecuciones sobre equipos clientes y servidores.
- ✔ Transporte de ficheros adjuntos hasta el equipo cliente o el servidor (módulos, ficheros de configuración, scripts complejos, ...).
- ✔ Ejecución de shell script en el equipo si se cumplen una serie de condiciones preestablecidas (número de interfaces de red, contenido de un fichero, versión del

sistema operativo, ...).

- ✔ Asegurar la ejecución del script seleccionado y la recogida de los resultados obtenidos.
- ✔ Registro de la actividad desarrollada en el equipo durante la ejecución del script.
- ✔ Registro de la actividad y evolución del proceso general.
- ✔ Empaquetado y descarga de los resultados obtenidos.
- ✔ Generación de un informe en texto plano y su envío mediante correo electrónico al ejecutor.
- ✔ Almacenar el estado de la ejecución en cada momento para permitir recuperar su estado en iteraciones automáticas o manuales.
- ✔ Acceso de administración y de usuario a todo el sistema mediante una interfaz web que permita diferenciar perfiles de usuarios y permisos organizados por grupos.

4. Planificación

4.1. Fases del proyecto

Código	Nombre
1	Plan de trabajo y Planificación
2	Especificación de funcionalidades
3	Diseño Técnico e Implementación
4	Testing y Calidad del Software
5	Documentación y Presentación

4.2. Calendario del proyecto

Fecha	Hito
19/09/2012	Inicio de Fase 1: Plan de trabajo y Planificación
27/09/2012	Propuesta de TFC
30/09/2012	Borrador del plan de trabajo
10/10/2012	Planificación
14/10/2012	Fin de Fase 1: Plan de trabajo y Planificación
15/10/2012	Inicio de Fase 2: Especificación de funcionalidades
04/11/2012	Fin de Fase 2: Especificación de funcionalidades
05/11/2012	Inicio de Fase 3: Diseño Técnico e Implementación
02/12/2012	Fin de Fase 3: Diseño Técnico e Implementación
03/12/2012	Inicio de Fase 4: Testing y Calidad del Software
23/12/2012	Fin de Fase 4: Testing y Calidad del Software
24/12/2012	Inicio de Fase 5: Documentación y Presentación
13/01/2013	Memoria final
16/01/2013	Vídeo presentación
17/01/2013	Fin de Fase 5: Documentación y Presentación

4.3. Descripción de las actividades

Para acotar el alcance de cada una de las fases del proyecto desglosaremos la diferentes actividades que forman cada etapa:

FASE1: Plan de trabajo y Planificación

Estará formado por las siguientes actividades:

- Marco del proyecto, describe el problema inicial que se pretende resolver.
- Alcance del proyecto, resume el objetivo del proyecto.
- Características del sistema, detalla el trabajo concreto que se llevará a cabo.
- Planificación, engloba la descomposición del proyecto en fases, tareas e hitos temporales. En esta actividad se especifican los recursos necesarios para llevar a cabo el proyecto.
    - Fases del proyecto
    - Calendario del proyecto
    - Descripción de las actividades
    - Diagrama de Gantt
    - Recursos e Infraestructura
        - Hardware
        - Software
- Composición de paquetes, se muestran los diferentes paquetes o bloques que formarán el sistema a desarrollar.

FASE2: Especificación de funcionalidades

- Definición de funcionalidades , descripción de las tecnologías que se usarán, de las funcionalidades que permitirá el sistema y comparación con proyectos relacionados.
  - Tecnologías
  - Funcionalidades
  - Proyectos relacionados

FASE3: Diseño Técnico e Implementación

- Diseño , abarca el diseño técnico de los 3 subsistemas (subsistema bash, interfaz web y base de datos).
  - Subsistema bash
    - Diagrama de estados
  - Interfaz web
    - Clases del Sistema
    - Fichas CRC
    - Diagrama de clases
      - Diagrama de clases gestoras, entidades y fronteras
      - Diagrama de clases entidad
  - Modelo de Persistencia
    - Modelo de datos conceptual: Entidad – Relación
    - Modelo de datos lógico

        - Modelo Relacional
        - Descripción de los atributos
    - Script de creación de la base de datos
- Implementación , funcionalidades y manual de usuario.
    - Funcionalidades implementadas
    - Manual de usuario

FASE4: Testing y Calidad del Software

- Manual de instalación , pasos y requisitos para implantar el sistema desarrollado.
    - Requisitos
    - Instalación
- Análisis de Testing , pruebas y análisis del funcionamiento del proyecto.
    - Diseño del testing
    - Entorno de prueba
    - Testing de integración
    - Testing unitario
    - Informe final
- Análisis de Calidad , calidad y eficiencia del software.
    - Corrección
    - Robustez
    - Extensibilidad

- Reutilización
- Compatibilidad
- Eficiencia
- Portabilidad
- Facilidad de uso
- Funcionalidad

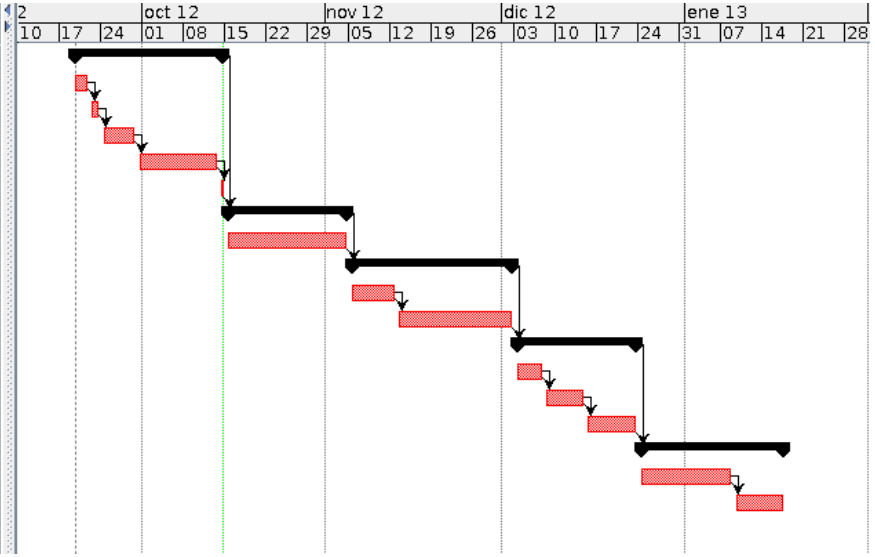
FASE5: Documentación y Presentación

- Memoria, documento formal con el proceso de desarrollo del proyecto y los resultados obtenidos.
- Vídeo presentación, presentación virtual del proyecto.

4.4. Diagrama de Gantt

Utilizando la aplicación ProjectLibre (OpenProj) realizamos el diagrama de Gantt del proyecto con fecha de inicio 19/09/2012, fecha de finalización 17/01/2013 y una estimación total de 363 horas de trabajo repartidas en 121 días.

	Nombre	Duracion	Inicio	Terminado	Predecesores
1	**⊟Fase1: Plan de trabajo y Planificación**	**26 days?**	**19/09/12 17:00**	**14/10/12 20:00**	
2	Marco del proyecto	3 days?	19/09/12 17:00	21/09/12 20:00	
3	Alcance del proyecto	2 days?	22/09/12 17:00	23/09/12 20:00	2
4	Características del sistema	6 days?	24/09/12 17:00	29/09/12 20:00	3
5	Planificación	14 days?	30/09/12 17:00	13/10/12 20:00	4
6	Composición de paquetes	1 day?	14/10/12 17:00	14/10/12 20:00	5
7	**⊟Fase2: Especificación de funcionalidades**	**21 days?**	**15/10/12 17:00**	**4/11/12 20:00**	**1**
8	Definición de funcionalidades	21 days?	15/10/12 17:00	4/11/12 20:00	
9	**⊟Fase3: Diseño Técnico e Implementación**	**28 days?**	**5/11/12 17:00**	**2/12/12 20:00**	**7**
10	Diseño	8 days?	5/11/12 17:00	12/11/12 20:00	
11	Implementación	20 days?	13/11/12 17:00	2/12/12 20:00	10
12	**⊟Fase4: Testing y Calidad del Software**	**21 days?**	**3/12/12 17:00**	**23/12/12 20:00**	**9**
13	Manual de instalación	5 days?	3/12/12 17:00	7/12/12 20:00	
14	Análisis de Testing	7 days?	8/12/12 17:00	14/12/12 20:00	13
15	Análisis de Calidad	9 days?	15/12/12 17:00	23/12/12 20:00	14
16	**⊟Fase5: Documentación y Presentación**	**25 days?**	**24/12/12 17:00**	**17/01/13 20:00**	**12**
17	Memoria final	16 days?	24/12/12 17:00	8/01/13 20:00	
18	Video presentación	9 days?	9/01/13 17:00	17/01/13 20:00	17



4.5. Recursos e Infraestructura

Para elaborar el proyecto se utilizarán los los siguientes recursos.

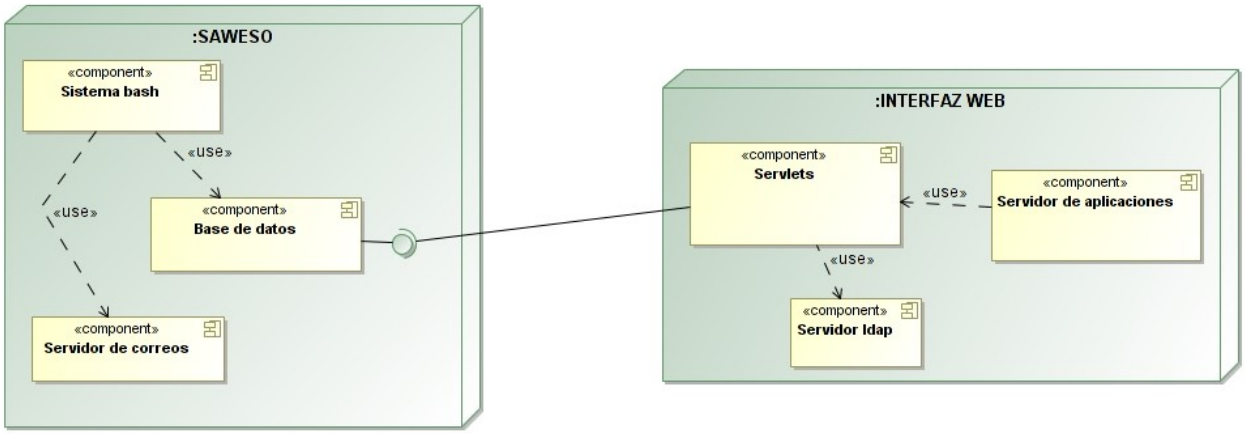
4.5.1. Hardware

- Estación de trabajo con procesador AMD E-350 DIMM de doble núcleo a 1.6 GHz, 8GB de memoria RAM DDR3 1066 MHz, disco duro SSD de 64GB y tarjeta de red RTL8111/8168B PCI Express Gigabit.

4.5.2. Software

- Sistema operativo Ubuntu 12.04.1 LTS (Precise Pangolin).
- Sistema operativo Debian 6.0 (Squeeze).
- Software de virtualización Oracle VirtualBox 4.1.12 o superior.
- Herramienta de planificación de proyectos ProjectLibre (OpenProj) 1.5 beta 5.
- Herramienta CASE de modelado de diagramas MagicDraw UML 17.0 o superior.
- Suite ofimática LibreOffice 3.
- Entorno de desarrollo integrado Eclipse 3.7.2 o superior.
- Servidor de aplicaciones Tomcat 7.0.26 o superior.
- Servidor de bases de datos MySQL 5.5 o superior.
- Paquetería debian adicional: mailx, postfix, openssh-server, openssh-client, ...

5. Composición de paquetes



6. Definición de funcionalidades

6.1. Tecnologías

El SAWESO unifica un conjunto de tecnologías para hacer posible el funcionamiento de un sistema integral que, como veremos en el siguiente apartado, permite entre otras muchas funcionalidades la copia de ficheros, la conexión y ejecución de comandos o scripts remotos y la recuperación de ficheros adjuntos o resultantes de la ejecución.

Entre las tecnologías sobre las que se asienta el funcionamiento del sistema tenemos:

- SSH (Secure Shell ó intérprete de órdenes segura), es el nombre de un protocolo y del programa que lo implementa, y sirve para acceder a máquinas remotas a través de una red. Además de la conexión a otros dispositivos, SSH nos permite copiar datos de forma segura (scp) y pasar los datos de cualquier otra aplicación por un canal seguro tunelizado mediante SSH.
- Certificados DSA y RSA, nos permiten establecer una conexión segura mediante SSH firmando y cifrando las comunicaciones y, por tanto, impidiendo que las claves y el contenido que se intercambian en cada sesión sean interceptados por terceros.
- Bash (Bourne again shell), fue escrito para el proyecto GNU y es el intérprete de comandos por defecto en la mayoría de las distribuciones de Linux.
- SMTP (Simple Mail Transfer Protocol ó Protocolo para la transferencia simple de correo electrónico).
- SQL (Structured Query Language), es un lenguaje de consulta estructurado de acceso a bases de datos relacionales que permite especificar cualquier tipo de operaciones sobre ellas.
- JavaServer Pages (JSP), es una tecnología Java que permite generar contenido dinámico para web, en forma de documentos HTML, XML o de otro tipo.

6.2. Funcionalidades

Para cumplir con los requisitos iniciales, el funcionamiento general del sistema debe permitir las siguientes funcionalidades:

1. Login e identificación del usuario
2. Configuración general del sistema
3. Gestión de permisos, usuarios y grupos
4. Programación de plantillas
5. Gestión de plantillas
6. Ejecución de plantillas
7. Notificación de eventos mediante email
8. Registro de la actividad y consulta de logs

6.2.1. Login e identificación del usuario

Este módulo es el encargado de la gestión de la sesión y del mantenimiento de la misma.

Inicialmente, cuando se accede a la aplicación se solicita un usuario y contraseña. Este primer paso es obligatorio y genérico para todos los usuarios que tengan la intención de acceder al sistema. La herramienta identificará si el usuario es o no un usuario registrado en el sistema. Se procede por lo tanto a validar su autenticidad. Si el usuario se valida correctamente, la aplicación reconocerá el perfil al que pertenece dicho usuario y en función de este le ofrecerá un entorno personalizado de la aplicación permitiéndole el uso de sus diferentes funciones. En caso negativo, se reportará un correspondiente mensaje de error indicando al usuario el posible error cometido e invitando al mismo a la repetición del proceso.

Los distintos perfiles de usuario, que con posterioridad describiremos, deberán introducir sus datos identificativos en los campos habilitados para su posterior validación.

En función del perfil del usuario que se conecta se habilitaran distintas

funcionalidades. Solo podrán ser accesibles por el usuario las funcionalidades asignadas a su perfil (grupo) concreto. Posteriormente se podrán modificar, extendiendo o reduciendo, la posibilidad o acciones de un determinado perfil de usuario.

Existirán distintos perfiles de usuario, los cuales son los habilitados inicialmente para conectarse a la herramienta. Posteriormente, si se desea, se podrá llevar a cabo la inclusión de nuevos perfiles de usuario en el sistema, pero inicialmente no se requiere para la aplicación. Un usuario podrá pertenecer a varios perfiles de forma simultánea, de forma, que tendrá los privilegios acumulativos de cada perfil contando con el acceso a las funcionalidades que le permita cada perfil individual al que pertenezca.

Existen cuatro perfiles de usuario, o grupos, en nuestro sistema: Administrador, Gestor de grupo, Programador, Operador. Las funcionalidades generales de cada grupo de usuarios son:

- **Administrador**, acceso a los menús de configuración de la aplicación. Sus funciones son: establecer los directorios de trabajo, editar los certificados de seguridad (dsa, rsa), datos de conexión con la base de datos MySQL, configurar los datos de dominio para el envío de email, edición del formato de los informes resumen que se enviarán por email y especificación de los rangos de direcciones ips que se consideran válidos para cada tipo de ejecución (servidores o equipos clientes).
- **Gestor de grupo**, acceso a los menús de gestión de la aplicación. Sus funciones son: asignar a cada usuario a uno o varios perfiles (grupos), determinar los módulos y funcionalidades a los que tendrán acceso cada grupo, crear nuevos tipos de plantillas de ejecución y modificar la visibilidad de las plantillas asignándolas a grupos concretos.
- **Programador**, acceso a los menús de programación de plantillas y ejecución avanzada de la aplicación. Sus funciones son: crear nuevas plantillas o editar plantillas de ejecución existentes, ejecutar cualquiera de los tipos de plantillas

(comando, script, búsqueda, recolección, informe, ...) que permite la aplicación y programar tareas.

- **Operador**, acceso a los menús de ejecución básica de la aplicación. Sus funciones son: ejecutar los tipos de plantillas, específicamente asignadas a dicho grupo, que permite la aplicación y programar tareas.

6.2.2. Configuración general del sistema

Este módulo permite configurar los parámetros necesarios para el funcionamiento de la aplicación.

La vista de configuración se muestra en forma de tabla donde se pueden establecer los siguientes parámetros:

- **Directorios de trabajo**, el funcionamiento del sistema necesitará la utilización de varios directorios en la estructura del disco duro del servidor. Estos directorios se utilizarán para almacenar información temporal, almacenar y leer las plantillas (scripts) disponibles y para el registro de logs permanentes. Por defecto el sistema tendrá establecida la siguiente asignación de directorios de trabajo:
  - Directorio temporal , directorio /tmp del servidor donde se aloje el sistema.
  - Directorio para almacenar los scripts, directorio /etc/saweso del servidor donde se aloje el sistema.
  - Directorio para almacenar log permanentes , subdirectorio saweso dentro del directorio de logs (/var/log/saweso) del servidor donde se aloje el sistema.
- **Certificados de seguridad (dsa, rsa)**, para la conexión segura a los servidores y equipos clientes se utilizarán certificados dsa y rsa respectivamente. Mediante esta opción se pueden importar dichos certificados de seguridad para que sean utilizados por el sistema en las conexiones que se establezcan a y desde los hosts

remotos.

- **Datos de conexión con la base de datos MySQL**, se deberá especificar el nombre de usuario y contraseña de un usuario administrador de MySQL que tenga privilegios de creación de nuevas bases de datos. También se debe especificar la dirección ip o nombre del host donde esta corriendo el servicio MySQL que será utilizado por la aplicación.
- **Datos de dominio para el envío de email**, aquí debemos configurar el nombre de dominio al que pertenece el servidor de correo que utilizará la aplicación para el envío de las notificaciones.
- **Edición del formato de los informes**, en un cuadro de texto se indicará el cuerpo general de los informes resumen que se enviarán por email a los usuarios. Estos correos se utilizarán para indicar que ha finalizado la ejecución de la plantilla que estaban ejecutando sobre la muestra de hosts para la que se había programado.
- **Especificación de los rangos de direcciones ips que se consideran válidos para cada tipo de ejecución (servidores o equipos clientes)**. Para cada tipo de host se debe especificar las direcciones de red, y máscaras de subred asociadas, que se considerarán válidas como equipos pertenecientes a la plataforma a la que queremos acceder remotamente.

6.2.3. Gestión de permisos, usuarios y grupos

Este módulo permite la gestión de los usuarios y grupos de la aplicación.

6.2.3.1. Crear o editar un nuevo usuario

Solo se podrá crear o editar nuevos usuarios si hemos seleccionado el modo de autenticación con MySQL. En este caso, mediante un breve formulario, se solicitarán los datos necesarios para la crear un nuevo usuario.

Los datos requeridos son: nombre de usuario, contraseña, nombre completo del usuario y correo electrónico.

Mediante una lista ordenada tendremos la posibilidad de seleccionar cualquier usuario existente o con acceso al sistema. Una vez seleccionado un usuario accederemos a una ventana de propiedades del usuarios donde podremos consultar o editar la información disponible sobre dicho usuario (nombre corto, nombre largo, contraseña y correo electrónico).

6.2.3.2. Asignar usuarios a perfiles (grupos)

Mediante una vista que contiene 3 listas enfrentadas se podrá asignar a cualquier usuario a uno o a varios grupos. Para asignar un usuario a un grupo, en una de las listas, seleccionaremos el grupo en cuestión y, en las otras dos listas, aparecieran los usuarios que pertenecen y que no pertenecen al grupo respectivamente para que podamos desasignar o asignar usuarios al grupo utilizando las flechas de acción existentes.

6.2.4. Programación de plantillas

Este módulo permite la creación de nuevas plantillas (scripts). Las plantillas de ejecución son las utilizadas por el sistema para ejecutar la secuencia de comandos deseada sobre el host remoto.

Todas las posibles ejecuciones de SAWESO se almacenan mediante pequeños scripts encapsulados en una plantilla de ejecución. Una plantilla de ejecución no es más que un script en bash que contiene 6 funciones: ComprobarEntornoEjecucion(), log(), Resultados(), Precondicion(), Script(), Adjuntos().

La **estructura y contenido de una plantilla de ejecución** está formada por un script en bash donde se incorporarán las instrucciones que deseamos ejecutar en la máquina remota. Se divide en un encabezado y 6 funciones que se ejecutarán de secuencialmente: ComprobarEntornoEjecucion(), log(), Resultados(), Precondicion(), Script(), Adjuntos().

Para la **generación de una plantilla de ejecución (script)**, debemos especificar el contenidos de las siguientes 3 funciones en la plantilla base:

- Precondición, especifica la condición que debe cumplirse para que se proceda a ejecutar el script.
- Script, especifica el comando, script o workaorund a ejecutar en el equipo final.
- Adjuntos, si deseamos copiar ficheros del host remoto, en esta sección especificaremos los comandos que producirán los ficheros adjuntos que se desean incorporar en los resultados tras la ejecución del script.

6.2.5. Gestión de plantillas

Este módulo permite la gestión de las plantillas (scripts) que utiliza el sistema para ejecutar la secuencia de comandos deseada sobre el host remoto.

6.2.5.1. Modificar la visibilidad de las plantillas

Por defecto todas las plantillas de ejecución serán visibles y ejecutables por todos los perfiles con permiso de ejecución de plantillas (programador y operador). Si algún usuario con perfil Gestor de grupo desea modificar la visibilidad de alguna plantilla, asignándolas a grupos concretos, para que solo sea ejecutable por un subconjunto de grupos de la aplicación, deberá marcar la visibilidad deseada en cada plantilla mediante la utilización de los checkbox, por cada grupo, disponibles en la vista de gestión de las plantillas.

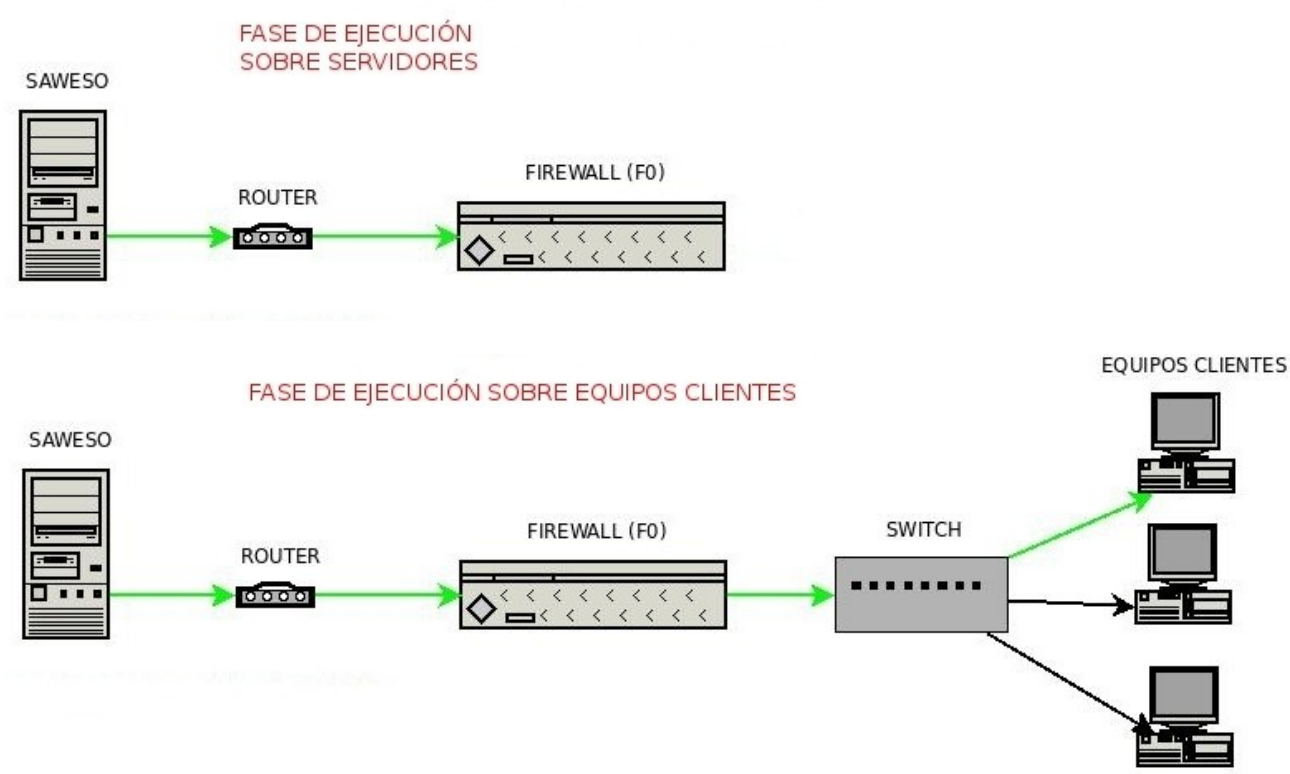
6.2.6. Ejecución de plantillas

La ejecución de las plantillas de ejecución sobre los hosts remotos es muy simple y puede resumirse en 5 grandes fases:

- **Obtención de datos iniciales**, mediante un asistente web nos encargamos de

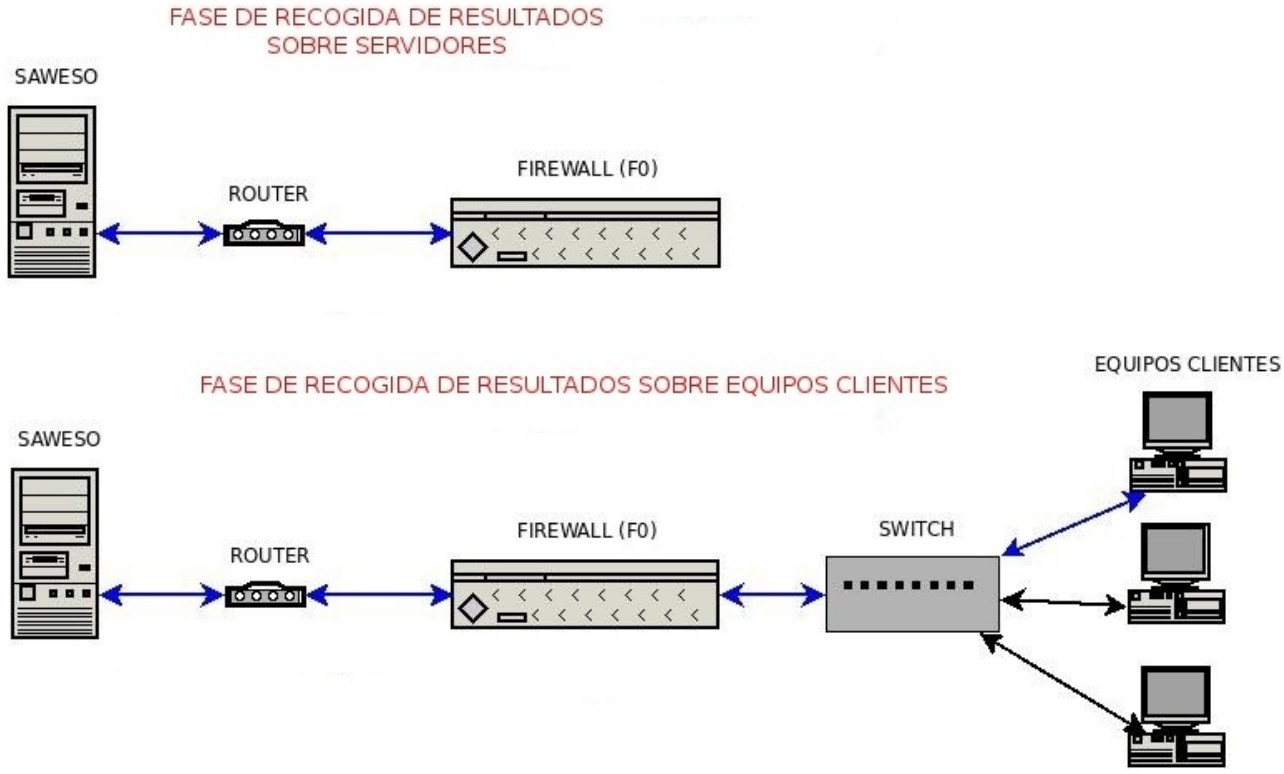
recopilar toda la información necesaria para la ejecución de saweso.

- **Fase de ejecución**, en esta fase vamos creando iterativamente conexiones o túneles ssh hacia los hosts finales especificados en el listado de direcciones ip precargado o proporcionado al sistema. Una vez creada la conexión, la utilizamos para transportar los ficheros necesarios hacia el destino y para lanzar la ejecución del script que hayamos seleccionado. El diseño de los scripts ejecutados en los hosts permitirá que sea el propio script ejecutado el encargado de verificar que se cumplen las condiciones establecidas para la ejecución, de registrar la información generada durante la ejecución del mismo y de crear un archivador con los resultados obtenidos. Hasta que no se haya lanzado la ejecución (ejecución en segundo plano) sobre todos los hosts no se iniciará la recolección de resultados.



- **Fase de recogida de resultados**, durante esta fase nos encargamos de establecer nuevas conexiones hacia los equipos para recoger el archivador (.tar.gz)

con los resultados de la fase de ejecución.



- **Fase de notificación**, durante esta fase generamos un informe resumen con los datos y resultados de la ejecución realizada y lo enviamos mediante email al ejecutor que la programó.
- **Control de estado**, sobre todo el sistema descrito se ejecuta un proceso “padre” que revisa el estado de avance del mismo. Mediante esta supervisión del estado general de la ejecución se puede informar al usuario del número de hosts restantes.

6.2.7. Notificación de eventos mediante email

Como hemos mencionada en la ejecución del plantillas, al finalizar la ejecución y la recogida de resultados se genera un informe resumen y se procede a su notificación al

ejecutor que corresponda. Tras generar un informe resumen en texto, este se envía mediante correo electrónico al propietario de la ejecución y a todos los destinatarios en copia especificados en la introducción de datos inicial.

6.2.8. Registro de la actividad y consulta de logs

Toda la actividad de SAWESO se almacena en la base de datos MySQL y en algunos ficheros de texto (logs), ubicados en las rutas especificadas en los parámetros de configuración general, del servidor donde se está ejecutando el sistema.

A continuación se listan los accesos a los sistemas de registro y de logs que utilizará el algoritmo en cada fase:

OBTENCIÓN DE DATOS INICIALES	
Directorio de scripts	Directorios donde se almacenan los scripts disponibles para la ejecución sobre equipos clientes o sobre servidores.
Base de datos y ficheros temporales	Direcciones ips de los hosts. Datos generales de la ejecución. Programaciones de tareas.
Fichero de log	Fichero de texto donde se reflejará la evolución del proceso.
FASE EJECUCION	
Directorio de scripts	Directorios donde se almacenan los scripts disponibles para la ejecución sobre equipos clientes o sobre servidores.
Base de datos y ficheros temporales	Control de ips ejecutadas. Control de ips restantes.
Fichero de log	Fichero de texto donde se reflejará la evolución del proceso.
FASE DE RECOGIDA DE RESULTADOS	
Base de datos y directorio de registro	Control de todas las ips ejecutadas. Recogida y almacenamiento de resultados obtenidos.
Fichero de log	Fichero de texto donde se reflejará la evolución del proceso.
FASE NOTIFICACION	

Ficheros de registro	Fichero de texto con informe resumen de los resultados de la ejecución (fecha y hora, plantilla ejecutada, total de ips, total de ips ejecutadas, total de ips pendientes, tiempo transcurrido).
Fichero de log	Fichero de texto donde se reflejará la evolución del proceso.
CONTROL DE ESTADO	
Fichero de log	Fichero de texto donde se reflejará la evolución del proceso.

En la utilización de ficheros de texto se utilizará una nomenclatura que use variables que identifiquen de forma única cada conjunto de ficheros pertenecientes a una ejecución.

6.3. Proyectos relacionados

A continuación destacaremos el proyecto de software libre que tienen las características y funcionalidades que más se asemejan al proyecto SAWESO.

6.3.1. MASSH

El proyecto Massh permite realizar las siguientes funciones sobre un grupo múltiple de hosts remotos:

- Ejecutar comandos
- Copiar scripts y ejecutarlos
- Copias ficheros hacia los hosts
- Copiar ficheros desde los hosts

Massh realiza estas acciones estableciendo conexiones paralelas sobre el conjunto de objetivos remotos especificados. El sistema permite establecer agrupaciones lógicas de hosts.

Durante la ejecución de los comandos o scripts remotos podemos mostrar la salida completa de las acciones realizadas en cada hosts o un resumen del resultado obtenido (éxito o fracaso).

Para la administración de Massh se dispone de una línea de comandos simplificada para facilitar la gestión.

6.3.2. WASSH2

El proyecto WASSH2 define conceptualmente la implementación de un sistema con funciones de ejecución de comandos sobre un listado de hosts objetivos especificados mediante un listado de hosts separados por comas o especificados mediante un fichero de texto. Entre sus funciones permite:

- Organización de los hosts en grupos o subgrupos
- Ejecución paralela personalizable (timeout y numero de conexiones simultáneas)
- Posibilidad de especificar opciones a utilizar en las conexiones SSH (nombre de usuario, quiet, …)
- Formateo de la salida de las ejecuciones

6.3.3. Conclusiones

Analizando las características de los proyectos relacionados podemos concluir que el sistema SAWESO incorpora todas las funcionalidades que permiten el resto de los proyectos relacionados añadiendo las siguientes funcionalidades:

- Interfaz web para la administración y uso de todo el sistema.
- Control total sobre el acceso y la ejecución de comandos y scripts mediante el módulo de gestión de permisos, usuarios y grupos.
- Acceso a equipos clientes, ubicados en una zona militarizada, y que estén accesibles a través de un servidor de seguridad centralizado.
- Persistencia de la información al utilizar un sistema mixto con base de datos MySQL y ficheros de logs en directorios persistentes.
- Resumen de las ejecuciones y estadísticas de resultados.

- Validación de las direcciones ip de los objetivos.
- Estructuras de control en las plantillas de ejecución que permiten un sistema de ejecución condicional en cada host.
- Control del la ejecución mediante la generación de logs individuales por cada host.
- Informes y notificación por email.

7. Diseño

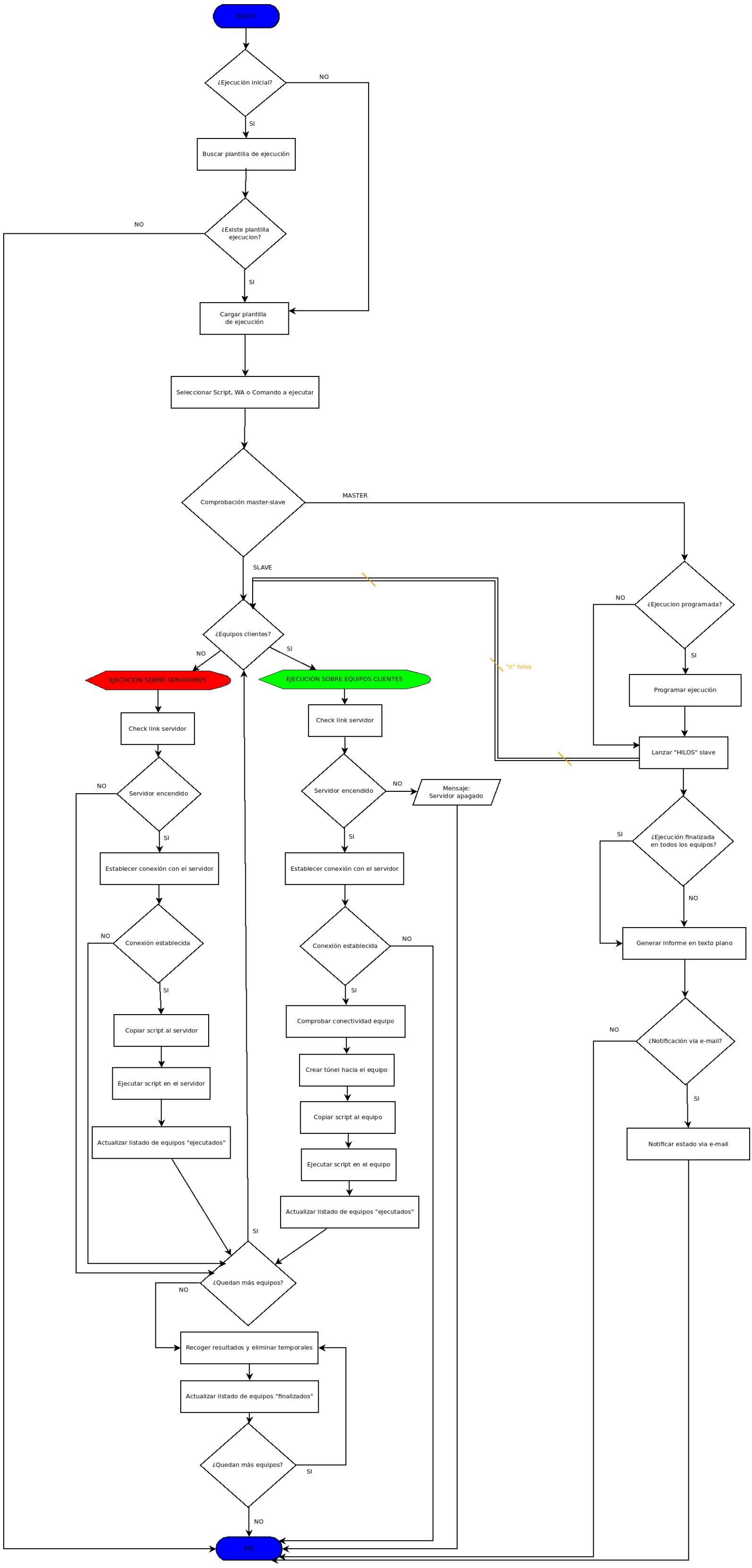
7.1. Subsistema Bash

Aunque el sistema está compuesto por varios subsistemas y vistas web, la ejecución de las conexiones y la función principal de la aplicación se desarrolla desde un script escrito en Bash.

Bash (Bourne again shell) es un programa informático cuya función consiste en interpretar órdenes. Está basado en la shell de Unix y es compatible con POSIX (Portable Operating System Interface; la X viene de UNIX como seña de identidad de la API). El interprete Bash fue escrito para el proyecto GNU y es el intérprete de comandos por defecto en la mayoría de las distribuciones de Linux.

7.1.1. Diagrama de estados

El diagrama de estados que describe el subsistema bash es el siguiente:


INICIO
¿Ejecución inicial?
NO
SI
Buscar plantilla de ejecución
¿Existe plantilla ejecucion?
NO
SI
Cargar plantilla de ejecución
Seleccionar Script, WA o Comando a ejecutar
Comprobación master-slave
MASTER
SLAVE
¿Equipos clientes?
NO
SI
EJECUCIÓN SOBRE SERVIDORES
EJECUCIÓN SOBRE EQUIPOS CLIENTES
"n" hilos
Check link servidor
Check link servidor
NO
Servidor encendido
SI
Establecer conexión con el servidor
Establecer conexión con el servidor
NO
Conexión establecida
SI
Copiar script al servidor
Ejecutar script en el servidor
Actualizar listado de equipos "ejecutados"
Servidor encendido
NO
Mensaje: Servidor apagado
SI
Conexión establecida
NO
SI
Comprobar conectividad equipo
Crear túnel hacia el equipo
Copiar script al equipo
Ejecutar script en el equipo
Actualizar listado de equipos "ejecutados"
SI
¿Quedan más equipos?
NO
Recoger resultados y eliminar temporales
Actualizar listado de equipos "finalizados"
¿Quedan más equipos?
SI
NO
FIN
NO
¿Ejecucion programada?
SI
Programar ejecución
Lanzar "HILOS" slave
SI
¿Ejecución finalizada en todos los equipos?
NO
Generar informe en texto plano
NO
¿Notificación via e-mail?
SI
Notificar estado via e-mail

7.2. Interfaz web

7.2.1. Clases del Sistema

A continuación se presenta una relación de las clases que componen el sistema:

ENTIDADES	GESTORES	FRONTERAS	EXCEPCIONES	UTILES
Plantilla	GestorBD	Login	Excepcion	Uploader
Usuario	GestorEjecucionTareas	PantConsulta	EAcceso	AccesoMySQL2
Grupo	GestorFichero	PantEjecucionAvanzada	EAplicacion	AccesoTxt
Ejecucion	GestorPermisos	PantEjecucionBasica	EFaltaFichero	Aleatorios
Tarea	GestorUsuariosGrupos	PantPlantillas		Filename
Host	GestorPlantillas	PantUsuariosGrupos		Cerrar
		Cabecera		Inicio
		Pie		
		Menu		
		PantConfiguracion		
		PantAyuda		
		PaginaError		
		PaginaErrorLogin		

7.2.2. Fichas CRC

Plantilla	
Descripción	Clase para crear las plantillas de ejecución de SAWESO
Tipo	Clase Principal
Características	Concreta, persistente
Responsabilidades	Identificar cada plantilla y almacenar su contenido (tipo, subtipo, autor, modificacion, descripcion)
Colaboraciones	GestorPlantillas, Grupo, GestorEjecucionTareas
Constructores	
+Plantilla(nombre:String, tipo:String, subtipo:String, autor:String, fechaModificacion:Date, descripcion:String) +Plantilla(nombre:String, tipo:String, subtipo:String, autor:String)	
Atributos	
-nombre: String -tipo: String -subtipo: String -autor: String -modificacion: Date	

-descripcion: String
Métodos
+getNombre(): String +setNombre(nombre: String) +getTipo(): String +setTipo(tipo: String) +getSubtipo(): String +setSubtipo(subtipo: String) +getAutor(): String +setAutor(autor: String) +getModificacion(): Date +setModificacion(modificacion: Date) +getDescripcion(): String +setDescripcion(descripcion: String)

Usuario	
Descripción	Clase para implementar los usuarios que tendran acceos a la aplicación
Tipo	Clase Principal
Características	Concreta, persistente
Responsabilidades	Identifica al usuario
Colaboraciones	Grupo
Constructores	
+Usuario(nombre:String, clave:String)	
Atributos	
-nombre: String -clave: String	
Métodos	
+getNombre(): String +setNombre(nombre: String) +getClave(): String +setClave(clave: String)	

Grupo	
Descripción	Clase para crear los distintos tipos de perfiles que tendran acceso a la aplicación
Tipo	Clase Principal
Características	Concreta, persistente
Responsabilidades	Identificar el grupo al que pertenece cada usuario. Determina los privilegios que tendrá cada usuario.
Colaboraciones	GestorUsuariosGrupos, Usuario, Plantilla
Constructores	
+Grupo(nombre:String, descripcion:String)	
Atributos	
-nombre: String -descripcion: String	

Métodos
+getNombre(): String +setNombre(nombre: String) +getDescripcion(): String +setDescripcion(descripcion: String)

Ejecucion	
Descripción	Clase para cursar las ejecuciones programadas por cada usuario
Tipo	Clase Principal
Características	Concreta, persistente
Responsabilidades	Relaciona la tarea a ejecutar y el listado de hosts sobre los que se ejecutará
Colaboraciones	GestorEjecucionTareas, Tarea, Host
Constructores	
+Ejecucion(codigo:int , inicio:Date , email:String)	
Atributos	
-codigo: int -inicio: Date -email: String -informe: String	
Métodos	
+getCodigo(): int +setCodigo(codigo: int) +getInicio(): Date +setInicio(inicio: Date) +getEmail(): String +setEmail(email: String) +getInforme(): String +setInforme(informe: String)	

Tarea	
Descripción	Clase para programar las tareas
Tipo	Clase Principal
Características	Concreta, persistente
Responsabilidades	Almacenar la información para programar en el sistema las tareas a realizar de forma inmediata, periódica o diferida.
Colaboraciones	GestorEjecucionTareas, Host, Ejecucion
Constructores	
+Tarea(id:int , minuto:int , hora:int , diaMes:int , mes:int , diaSemana:int) +Tarea(id:int , minuto:int , hora:int)	
Atributos	
-id: int -minuto: int -hora: int -diaMes: int -mes: int -diaSemana: int	

Métodos
+getId(): int +setId(id: int) +getMinuto(): int +setMinuto(minuto: int) +getHora(): int +setHora(hora: int) +getDiaMes(): int +setDiaMes(diaMes: int) +getMes(): int +setMes(mes: int) +getDiaSemana(): int +setDiaSemana(diaSemana: int)

Host	
Descripción	Clase para crear cada uno de los host que componen una ejecución
Tipo	Clase Principal
Características	Concreta, persistente
Responsabilidades	Almacenar el estado en el que se encuentra cada host (servidor o cliente) durante las diferentes fases que componene una ejecución
Colaboraciones	GestorEjecucionTareas, Tarea, Ejecucion
Constructores	
+Host(ip:String , centro:String , descripcion:String)	
Atributos	
-ip: String -centro: String -descripcion: String	
Métodos	
+getIp(): String +setIp(ip: String) +getCentro(): String +setCentro(centro: String) +getDescripcion(): String +setDescripcion(descripcion: String)	

GestorBD	
Descripción	Clase para gestionar las conexiones con la base de datos
Tipo	Interacción
Características	Concreta, compuesta
Responsabilidades	Consultar y actualizar los registros de la base de datos MySQL utilizados por el resto de clases del sistema
Colaboraciones	GestorPlantillas, GestorUsuariosGrupos, GestorEjecucionTareas, GestorPermisos
Constructores	
+GestorBD()	
Atributos	

+accesoBD: AccesoMySQL2 +txt: AccesoTxt -conexion: Connection
Métodos
+desconectarBD() +checkLogin(usuario: String, clave: String): boolean +getGrupos(usuario: String): String[ ]

GestorEjecucionTareas	
Descripción	Clase para gestionar todos los subsistema que participan en la ejecución de las tareas
Tipo	Interacción
Características	Concreta, compuesta
Responsabilidades	Programar las tareas con los parámetros necesarios para realizar la ejecución especificando los hosts objetivo.
Colaboraciones	GestorBD, GestorFichero, Plantilla, Ejecucion, Tarea, Host
Constructores	
+GestorEjecucionTareas()	
Atributos	
-request: HttpServletRequest -context: ServletContext -carpeta: String -tipoHost: String -tipoEjecucion: String -rutaRelativa: String	
Métodos	
+upload (request: HttpServletRequest, context: ServletContext): String -parseRequest(request: HttpServletRequest) -seekFileFields(items: List)	

GestorFichero	
Descripción	Clase para gestionar la lectura y escritura en los ficheros de texto utilizados por la aplicación
Tipo	Interacción
Características	Concreta, compuesta
Responsabilidades	Consultar y actualizar los ficheros de texto utilizados por el resto de clases del sistema
Colaboraciones	GestorEjecucionTareas, GestorPlantillas
Constructores	
+GestorFichero()	
Atributos	
+txt: AccesoTxt	
Métodos	

GestorPermisos	
Descripción	Clase para gestionar la autenticación del usuario y el grupo al que pertenece
Tipo	Interacción
Características	Concreta, compuesta
Responsabilidades	Verificación y autenticación del login de cada uno de los 4 perfiles de usuarios (Administrador, Gestor de grupo, Programador y Operador)
Colaboraciones	GestorBD
Constructores	
Atributos	
Métodos	

EAcceso hereda de Exception	
Descripción	Clase que gestiona los errores que puedan producirse durante el acceso a la base de datos o a los ficheros de texto
Tipo	Auxiliar
Características	Concreta, compuesta
Responsabilidades	Gestiona los errores derivados de los accesos a la base de datos MySQL o a los ficheros de texto plano
Colaboraciones	GestorBD, GestorFichero, AccesoMySQL2
Constructores	
+EAcceso (mensaje: String)	
Atributos	
Métodos	

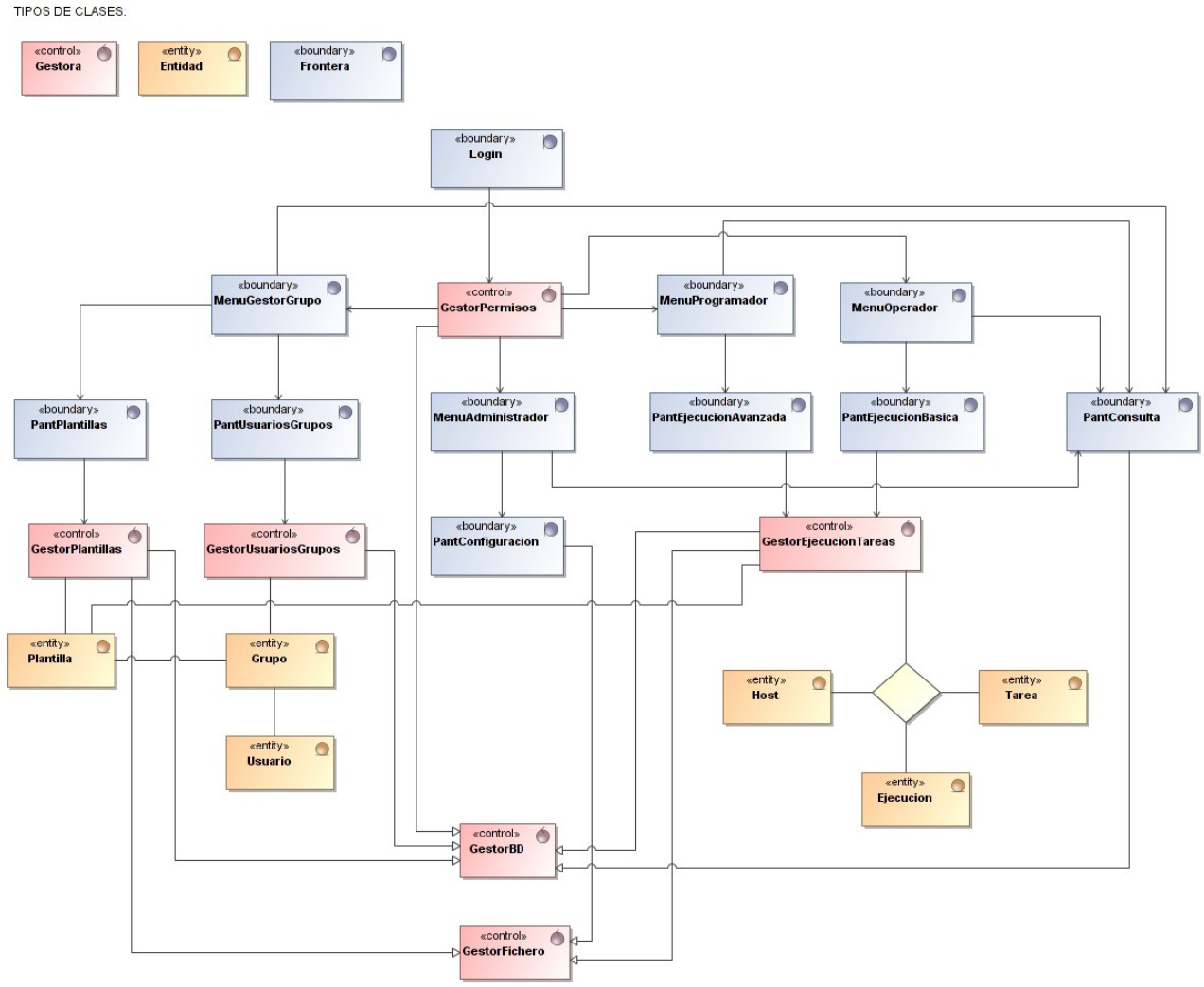
EAplicacion hereda de Exception	
Descripción	Clase que gestiona los errores que puedan producirse durante el funcionamiento de los subsistemas vitales de la aplicación
Tipo	Auxiliar
Características	Concreta, compuesta
Responsabilidades	Gestiona los errores derivados del funconamiento de la base de datos, la utilización de los ficheros de texto plano y el control de la autenticación y privilegios
Colaboraciones	GestorBD, GestorFichero, GestorPermisos
Constructores	

+EAplicacion (mensaje: String)
Atributos
Métodos

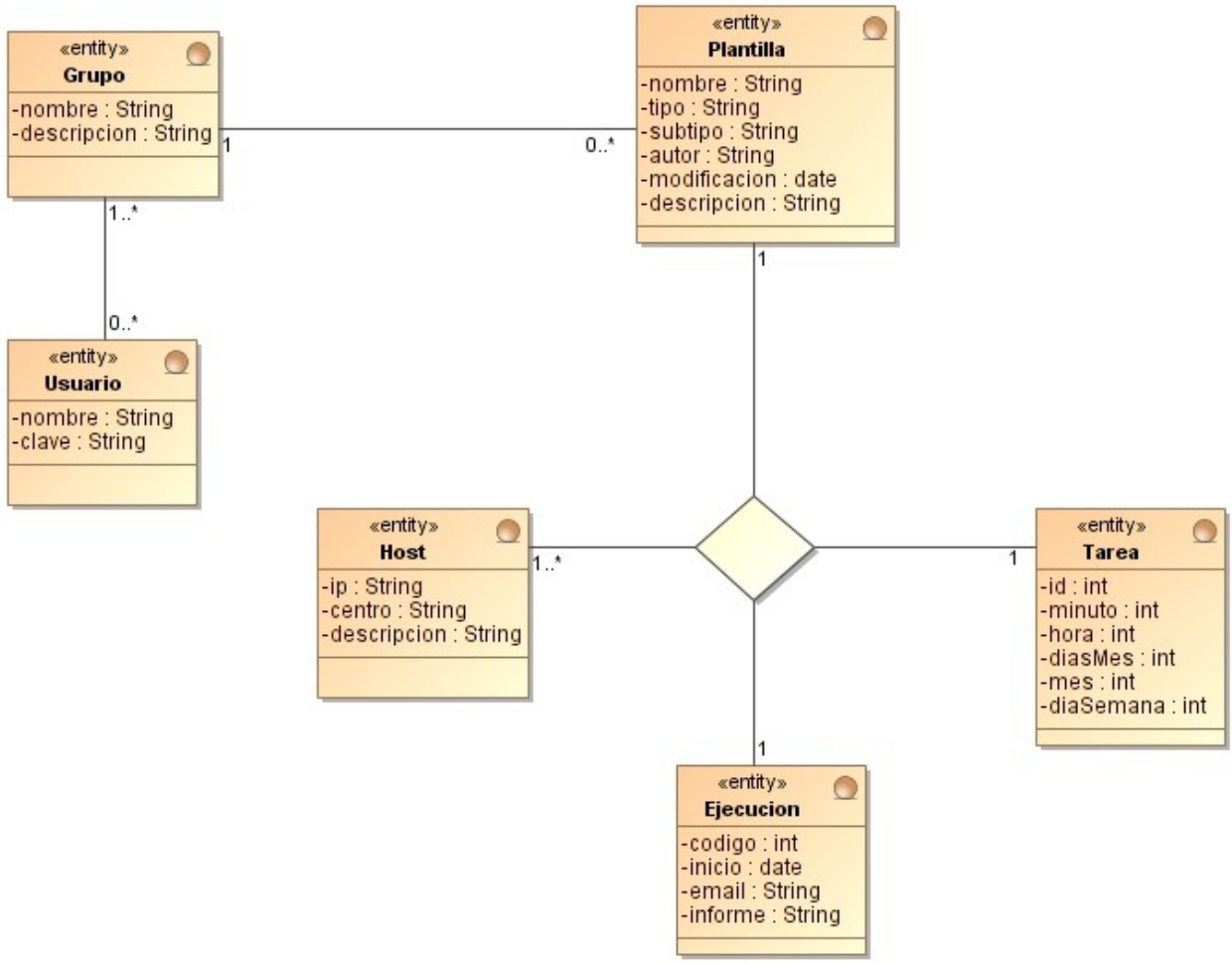
EFaltaFichero hereda de Exception	
Descripción	Clase que gestiona los errores que puedan producirse durante la búsqueda o apertura de los ficheros de texto
Tipo	Auxiliar
Características	Concreta, compuesta
Responsabilidades	Gestiona los errores provocados por la ausencia o corrucción de los ficheros de texto plano
Colaboraciones	GestorBD, AccesoTxt
Constructores	
+EFaltaFichero (mensaje: String)	
Atributos	
Métodos	

7.2.3. Diagrama de clases

7.2.3.1. Diagrama de clases gestoras, entidades y fronteras

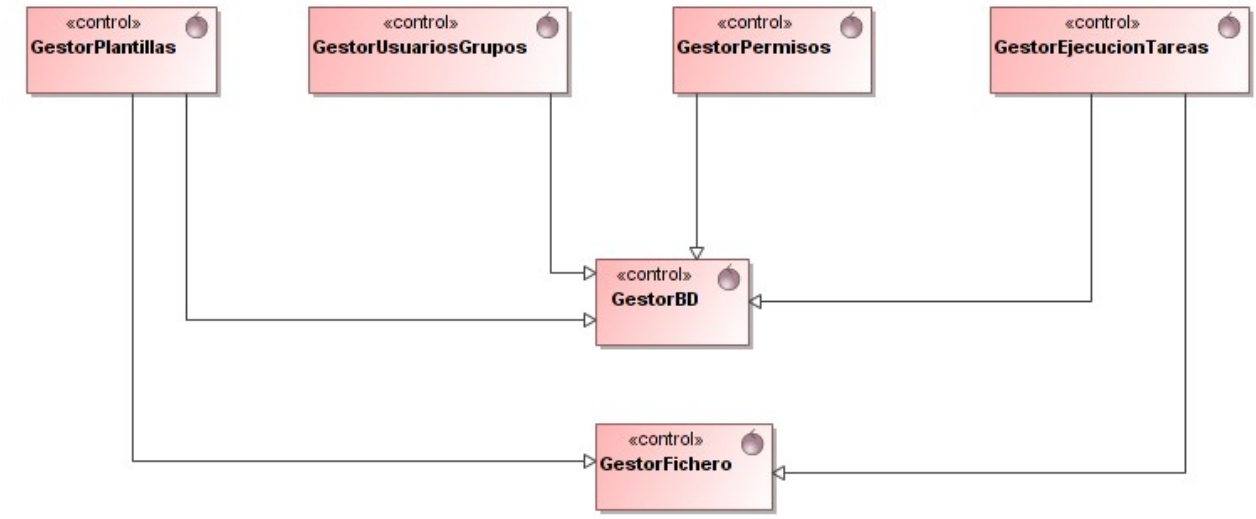


7.2.3.2. Diagrama de clases entidad

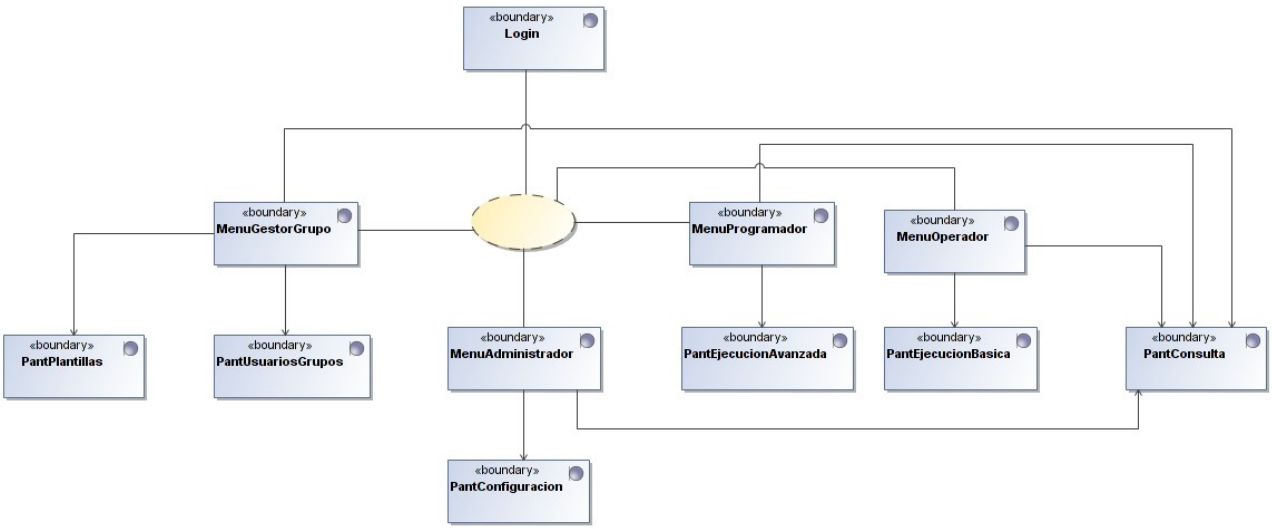


7.2.4. Diagrama de jerarquías

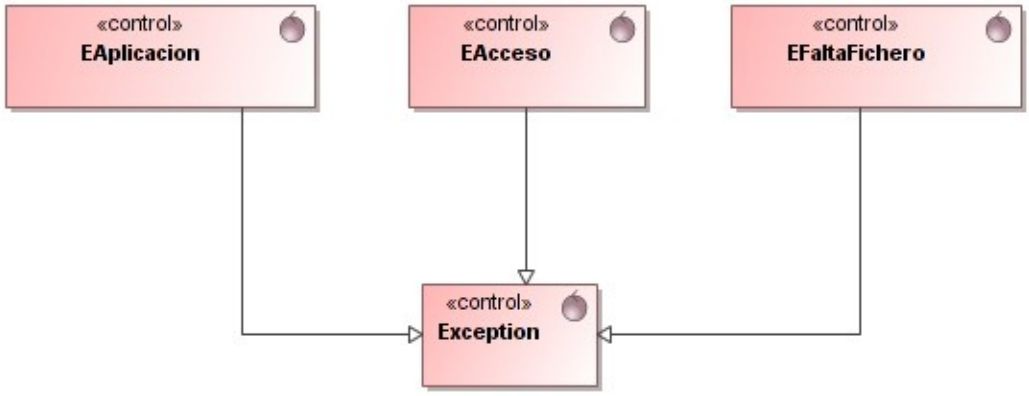
7.2.4.1. Gestores del subsistema



7.2.4.2. Pantallas del subsistema



7.2.4.3. Excepciones del subsistema

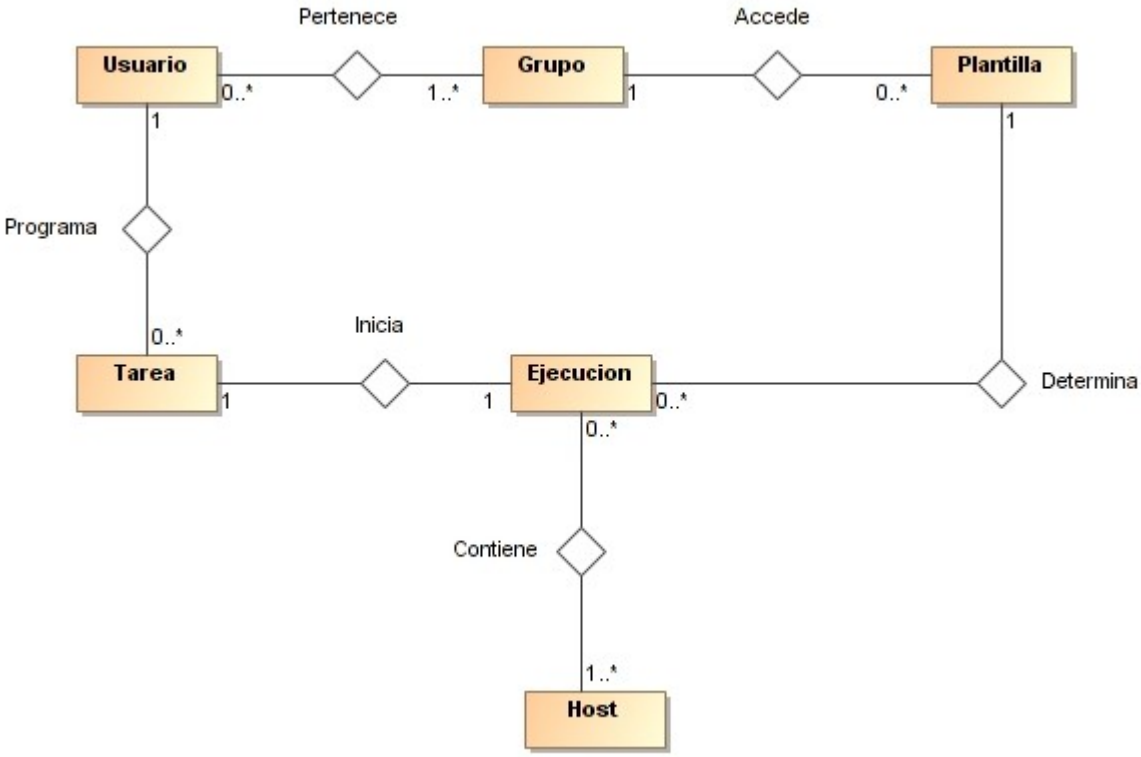


7.3. Modelo de Persistencia

El modelo de persistencia tiene un papel determinante en la definición de la estructura y organización de los datos en nuestro sistema gestor de base de datos.

7.3.1. Modelo de datos conceptual: Entidad - Relación

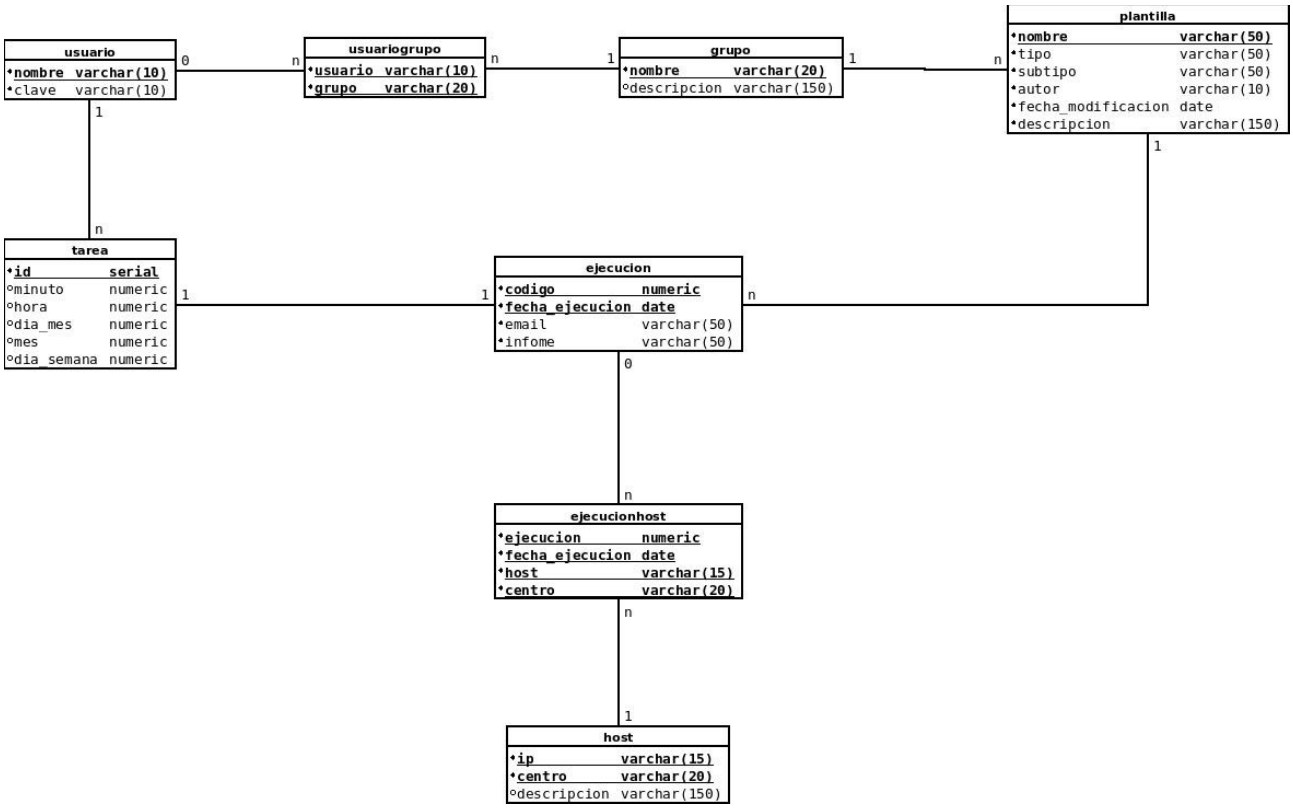
A continuación se muestra el modelo Entidad - Relación en base a los requisitos de la aplicación.



7.3.2. Modelo de datos lógico

El Modelo de datos lógico describe la forma que utilizaremos para organizar la información en nuestro sistema gestor de base de datos. Inicialmente describiremos la estructura con el Modelo relacional y posteriormente procederemos a la descripción de cada uno de los atributos de las distintas entidades.

7.3.2.1. Modelo Relacional



7.3.2.2. Descripción de los atributos

A continuación se describen las tablas y atributos que formarán el modelo de datos lógico:

Tabla "usuario"			
Campo	**Clave**	**Tipo**	**Descripción**
nombre	Primaria	Texto (10)	Identificador único del usuario
clave		Texto (10)	Contraseña del usuario encriptada en MD5

Tabla "grupo"			
Campo	**Clave**	**Tipo**	**Descripción**
nombre	Primaria	Texto (20)	Nombre del grupo. Existirán 4 tipos de grupos predefinidos: 1. administrador 2. gestor

			3. programador 4. operador
descripcion		Texto (150)	Breve descripción de las características y privilegios del grupo

Tabla "usuariogrupo"

Campo	Clave	Tipo	Descripción
usuario	Primaria	Texto (10)	Identificador único del usuario
grupo	Primaria	Texto (20)	Nombre del grupo
usuario	Foránea (usuario.nombre)	Texto (10)	
grupo	Foránea (grupo.nombre)	Texto (20)	

Tabla "tarea"

Campo	Clave	Tipo	Descripción
id	Primaria	Número (10)	Identificador secuencial de la tarea
minuto		Número (10)	Minuto en el que queremos que se ejecute la tarea
hora		Número (10)	Hora en la que queremos que se ejecute la tarea
dia_mes		Número (10)	Día del mes en el que queremos que se ejecute la tarea
mes		Número (10)	Mes en el que queremos que se ejecute la tarea
dia_semana		Número (10)	Día de la semana en el que queremos que se ejecute la tarea
usuario	Foránea (usuario.nombre)	Texto (10)	

Tabla "plantilla"

Campo	Clave	Tipo	Descripción
nombre	Primaria	Texto (50)	Identificador único de la plantilla de ejecución
tipo		Texto (50)	Tipo de host para el que se ha diseñado la plantilla. Puede contener los siguientes valores: 1. servidor 2. equipo
subtipo		Texto (50)	Categorización de la plantlla de ejecución. Puede contener los siguientes valores: 1. busqueda 2. comando 3. informe 4. recoleccion 5. script 6. workaround

autor		Texto (10)	Nombre del usuario que ha programado la plantilla
fecha_modificacion		Fecha	Fecha de la última actualización de la plantilla
descripcion		Texto (150)	Descipción de las funcionalidades de la plantilla
grupo	Foránea (grupo.nombre)	Texto (20)	

Tabla "ejecucion"

Campo	Clave	Tipo	Descripción
codigo	Primaria	Número (10)	Código único asignado a cada ejecución
fecha_ejecucion	Primaria	Fecha	Fecha de inicio de la ejecución
email		Texto (50)	Dirección de correo para notificaciones sobre la ejecución
informe		Texto (50)	Resumen de las estadísticas obtenidas tras la ejecución
plantilla	Foránea (plantilla.nombre)	Texto (50)	
tarea	Foránea (tarea.id)	Número (20)	

Tabla "host"

Campo	Clave	Tipo	Descripción
ip	Primaria	Texto (15)	Dirección ip de cada host (servidor o equipo cliente) objetivo
centro	Primaria	Texto (20)	Centro asociado a cada host para identificar su pertenencia y evitar duplicidades en direcciones ip lan (192.168.0.*, ...)
descripcion		Texto (150)	Descripción de cada host

Tabla "ejecucionhost"

Campo	Clave	Tipo	Descripción
ejecucion	Primaria	Número (10)	Código único asignado a cada ejecución
fecha_ejecucion	Primaria	Fecha	Fecha de inicio de la ejecución
host	Primaria	Texto (15)	Dirección ip de cada host (servidor o equipo cliente) objetivo
centro	Primaria	Texto (20)	Centro asociado a cada host para identificar su pertenencia y evitar duplicidades en direcciones ip lan (192.168.0.*, ...)
ejecucion	Foránea (ejecucion.codigo)	Número (10)	
host	Foránea (host.ip)	Texto (15)	

7.3.3. Script de creación de la base de datos

Como primer paso crearemos 2 ficheros de texto plano con el nombre, extensión y contenido que se especifica a continuación:

Fichero con la estructura de las tablas de la base de datos (tablas_BD.txt)

```
DROP TABLE IF EXISTS usuario;
CREATE TABLE usuario (
        nombre character varying(10) NOT NULL,
        clave character varying(10) NOT NULL,
        PRIMARY KEY (nombre)
);

DROP TABLE IF EXISTS grupo;
CREATE TABLE grupo (
        nombre character varying(20) NOT NULL,
        descripcion character varying(150),
        PRIMARY KEY (nombre)
);

DROP TABLE IF EXISTS usuariogrupo;
CREATE TABLE usuariogrupo (
        usuario character varying(10) NOT NULL,
        grupo character varying(20) NOT NULL,
        PRIMARY KEY (usuario, grupo),
        FOREIGN KEY (usuario) REFERENCES usuario(nombre),
        FOREIGN KEY (grupo) REFERENCES grupo(nombre)
);

DROP TABLE IF EXISTS tarea;
CREATE TABLE tarea (
        id serial NOT NULL,
        minuto numeric,
        hora numeric,
        dia_mes numeric,
        mes numeric,
        dia_semana numeric,
        PRIMARY KEY (id),
        usuario character varying(10) NOT NULL,
        FOREIGN KEY (usuario) REFERENCES usuario(nombre)
);

DROP TABLE IF EXISTS plantilla;
CREATE TABLE plantilla (
        nombre character varying(50) NOT NULL,
        tipo character varying(50) NOT NULL,
        subtipo character varying(50) NOT NULL,
        autor character varying(10) NOT NULL,
        fecha_modificacion date NOT NULL,
        descripcion character varying(150) NOT NULL,
        PRIMARY KEY (nombre),
        grupo character varying(20) NOT NULL,
        FOREIGN KEY (grupo) REFERENCES grupo(nombre)
);

DROP TABLE IF EXISTS ejecucion;
CREATE TABLE ejecucion (
        codigo numeric NOT NULL,
        fecha_ejecucion date NOT NULL,
```

```
	email character varying(50) NOT NULL,
	informe character varying(50) NOT NULL,
	PRIMARY KEY (codigo, fecha_ejecucion),
	plantilla character varying(50) NOT NULL,
	FOREIGN KEY (plantilla) REFERENCES plantilla(nombre),
	tarea bigint(20) unsigned NOT NULL,
	FOREIGN KEY (tarea) REFERENCES tarea(id)
);

DROP TABLE IF EXISTS host;
CREATE TABLE host (
	ip character varying(15) NOT NULL,
	centro character varying(20) NOT NULL,
	descripcion character varying(150),
	PRIMARY KEY (ip, centro)
);

DROP TABLE IF EXISTS ejecucionhost;
CREATE TABLE ejecucionhost (
	ejecucion numeric NOT NULL,
	fecha_ejecucion date NOT NULL,
	host character varying(15) NOT NULL,
	centro character varying(20) NOT NULL,
	PRIMARY KEY (ejecucion, fecha_ejecucion, host, centro),
	FOREIGN KEY (ejecucion) REFERENCES ejecucion(codigo),
	FOREIGN KEY (host) REFERENCES host(ip)
);
```

tablas_BD.txt

Fichero con la carga de datos iniciales para el funcionamiento de la aplicación (datos_iniciales.txt)

```
INSERT INTO usuario VALUES ('admin', 'admin');
INSERT INTO usuario VALUES ('gestor', 'gestor');
INSERT INTO usuario VALUES ('programador', 'programador');
INSERT INTO usuario VALUES ('operador', 'operador');
INSERT INTO usuario VALUES ('saweso', 'saweso');

INSERT INTO grupo VALUES ('administrador', 'Grupo de Administradores');
INSERT INTO grupo VALUES ('gestor', 'Grupo de Gestores de grupo');
INSERT INTO grupo VALUES ('programador', 'Grupo de Programadores');
INSERT INTO grupo VALUES ('operador', 'Grupo de Operadores');

INSERT INTO usuariogrupo VALUES ('admin', 'administrador');
INSERT INTO usuariogrupo VALUES ('gestor', 'gestor');
INSERT INTO usuariogrupo VALUES ('programador', 'programador');
INSERT INTO usuariogrupo VALUES ('operador', 'operador');
INSERT INTO usuariogrupo VALUES ('saweso', 'administrador');
INSERT INTO usuariogrupo VALUES ('saweso', 'gestor');
INSERT INTO usuariogrupo VALUES ('saweso', 'programador');
INSERT INTO usuariogrupo VALUES ('saweso', 'operador');
```

datos_iniciales.txt

A continuación adjuntamos el script de creación de la base de datos **saweso**, que será válido para su ejecución desde una terminal linux:

Script para ejecutar desde consola mediante el comando: bash script_creacion_BD.sh

```
txtrst=$(tput sgr0) # Text reset
txtred=$(tput setaf 1) # Red

echo "${txtred} 1. Borrando BD saweso existente ... ${txtrst}"
mysqladmin -uroot -p"holamundo" drop saweso

echo "${txtred} 2. Creando nueva BD saweso ... ${txtrst}"
mysqladmin -uroot -p"holamundo" create saweso

echo "${txtred} 3. Cargando estructura de tablas ... ${txtrst}"
mysql -u root -p"holamundo" saweso < tablas_BD.txt

echo "${txtred} 4. Cargando datos iniciales en las tablas ... ${txtrst}"
mysql -u root -p"holamundo" saweso < datos_iniciales.txt
```

script_creacion_BD.sh

Todo este proceso ha sido automatizado en un script de instalación global de la aplicación SAWESO (*Ver punto 9: Manual de instalación*)

8. Implementación

8.1. Funcionalidades implementadas

8.1.1. Login e identificación del usuario

Para el diseño del login e identificación del usuario se ha utilizado la siguiente relación de clases que realizan las funcionalidades que se especifican:

- **Login.jsp**, mediante un formulario captura el nombre de usuario y contraseña y se lo envía la clase que realiza la verificación de la autenticación.
- **GestorPermisos.jsp**, recibe el usuario y la contraseña desde la pantalla de login y determina si es un usuario válido en el sistema. Si las credenciales del usuario no son correctas, se devuelve el flujo a la pantalla de login que mostrará un mensaje de error. Si las credencias son correctas, consulta los grupos al que pertenece el usuario y establece las variables de sesión necesarias para cargar el menú con todas las opciones correspondientes.
- **Menu.jsp**, lee la variable de sesión establecida y carga las opciones de menú correspondientes.

La interfaz gráfica que se utiliza para la funcionalidad de login es la siguiente:

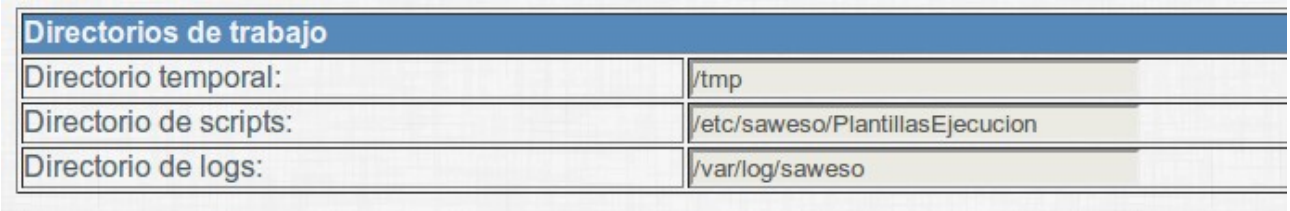
Prototipo de interfaz web para el login del usuario

8.1.2. Configuración general del sistema

Solo los usuarios que pertenezcan al perfil de “Administrador” tendrán disponible esta funcionalidad.

Mediante una tabla se pueden consultar los diferentes directorios que, junto a la base de datos, se utilizan en el funcionamiento de la aplicación.

La interfaz gráfica que se utiliza para permitir esta funcionalidad es la siguiente:



Prototipo de interfaz web para la configuración general del sistema

La instalación de la aplicación establece un fichero de configuración (/etc/saweso/saweso.conf) donde se almacena el contenido establecido desde la configuración general del sistema junto con los datos de acceso a la base de datos MySQL:

```
#Datos de acceso a la BD
schemaBD = saweso
usernameBD = root
passwordBD = holamundo

#Directorios de trabajo
tmp = /tmp
scrips = /etc/saweso
logs = /var/log/saweso
```

Ejemplo de fichero de configuración /etc/saweso/saweso.conf

8.1.3. Gestión de permisos, usuarios y grupos

Esta funcionalidad estará disponible únicamente para los usuarios con perfil Gestor de grupo.

Permite crear nuevos usuarios en el sistema asignándolos a uno o a varios grupos.

Permite buscar y editar la información (clave, email y asignación a grupos) de cada usuario.

8.1.4. Programación de plantillas

La **nomenclatura** de las plantillas de ejecución sigue el siguiente estándar:

- Para plantillas que no transporten ficheros adjuntos:
  saweso.<nombre_identificativo>.sh

  Ejemplo: saweso.apt_sources.sh

- Para plantillas que transporten ficheros adjuntos:

  saweso.<nombre_identificativo>.tar.gz

  Ejemplo: saweso.WA42_flash_firefox.tar.gz

La **estructura y contenido de una plantilla de ejecución** está formada por un script en bash donde se incorporarán las instrucciones que deseamos ejecutar en la máquina remota. Se divide en un encabezado y 6 funciones que se ejecutarán de secuencialmente: ComprobarEntornoEjecucion(), log(), Resultados(), Precondicion(), Script(), Adjuntos().

- Encabezado: en el encabezado aparece el intérprete de órdenes para el que ha sido programada la plantilla así como un conjunto de variables generales necesarias para la ejecución en el host remoto.

```
#!/bin/bash

#Obtenemos la fecha en formatos dd-mm-yyyy y la hora en formato hh:mm:ss
FECHA="$(date +"%d-%m-%Y")"
HORA="$(date +"%H_%M_%S")"

#Variables
declare -a ADJUNTOS
INCIDENCIA=$1
IP=$2

#Ficheros
```

```
ENCABEZADO=$INCIDENCIA.$FECHA.$HORA.$IP   #Encabezado para todos los ficheros
LOG=$ENCABEZADO.saweso.log                #Log de ejecucion en el equipo final
RESULTADOS=$ENCABEZADO.tar.gz             #Archivado que se creará para transportar los resultados
SALIDA=salida.log                         #Si se redirecciona la salida de algún comando a salida.log,
                                            esta se incluirá en la notificación por e-mail
```

- Funciones: la plantilla ejecutará las siguientes funciones en el orden secuencial en el que se listan a continuación:
  - ComprobarEntornoEjecucion(), comprueba el entorno en el host de destino para asegurar la correcta ejecución del script.
  - log(), registra la actividad de la ejecución del script en el directorio de registro.
  - Resultados(), empaqueta los resultados en .tar.gz para su posterior transporte hacia el sistema saweso.
  - Precondicion(), especifica la condición que debe cumplirse para que se proceda a ejecutar el script.
  - Script(), comando, script o workaorund a ejecutar en el equipo final.
  - Adjuntos(), copia al directorio de resultados los ficheros adjuntos que deseamos incorporar en los resultados tras la ejecución del script.

El contenido y estructura inicial de las plantillas de ejecución, que se utilizará como modelo base en la programación de nuevas plantillas, es la siguiente:

```
#!/bin/bash

#Obtenemos la fecha en formatos dd-mm-yyyy y la hora en formato hh:mm:ss
FECHA="$(date +"%d-%m-%Y")"
HORA="$(date +"%H_%M_%S")"

#Variables
declare -a ADJUNTOS
INCIDENCIA=$1
IP=$2
```

```
#Ficheros
ENCABEZADO=$INCIDENCIA.$FECHA.$HORA.$IP          #Encabezado para todos los ficheros
LOG=$ENCABEZADO.saweso.log                       #Log de ejecucion en el equipo final
RESULTADOS=$ENCABEZADO.tar.gz                    #Archivado que se creará para transportar los resultados
SALIDA=salida.log                                #Si se redirecciona la salida de algún comando a salida.log, esta se
                                                  incluirá en la notificación por e-mail

#Directorios de trabajo
DIR_REGISTRO="/tmp"                                                    #Directorio de trabajo
DIR_RESULTADOS="$DIR_REGISTRO/$INCIDENCIA/$FECHA.$HORA"                #Directorio para almacenar los resultados de la
                                                                        ejecucion


##################################################################### DEFINICION DE FUNCIONES
#####################################################################
#NOMBRE: ComprobarEntornoEjecucion - Comprueba el entorno en el host de destino para asegurar la correcta ejecución del script
#SINOPSIS: ComprobarEntornoEjecucion
#SALIDA: produce la salida con errores (exit 1) si encuentra algún parámetro que no permita la ejecucion del script
ComprobarEntornoEjecucion()
{
if [ -z $INCIDENCIA ] && [ -z $IP ];then
        logger "SAWESO.SH: No se ha especificado código de incidencia o dirección ip del equipo"
    exit 1
elif [ ! -d $DIR_REGISTRO ] && [ ! -r $DIR_REGISTRO ] && [ ! -w "$DIR_REGISTRO" ] && [ ! -x "$DIR_REGISTRO" ];then
        logger "SAWESO.SH: No se puede escribir en el directorio de registro $DIR_REGISTRO"
    exit 1
else
        logger "SAWESO.SH: Ejecución de saweso (Incidencia: $INCIDENCIA; DIR_RESULTADOS: $DIR_RESULTADOS)"
fi
}


#NOMBRE: log - Registra la actividad de la ejecucion del script en el directorio de registro
#SINOPSIS: log <mensaje de texto>
#SALIDA: produce la salida con errores (exit 1) sino puede registrar el mensaje
log()
{
mensaje=$1
encabezado="`date +"%d-%m-%Y"` `date +"%T"`"


if [ -n "$INCIDENCIA" ] && [ -n "$INCIDENCIA" ]; then
    if [ -d "$DIR_REGISTRO/$INCIDENCIA" ] && [ -r "$DIR_REGISTRO/$INCIDENCIA" ]; then
                mkdir $DIR_RESULTADOS 2> /dev/null
                chmod 700 $DIR_RESULTADOS
        echo "$encabezado: $mensaje" >> $DIR_RESULTADOS/$LOG
    elif [ -d "$DIR_REGISTRO" ] && [ -r "$DIR_REGISTRO" ];then
        mkdir $DIR_REGISTRO/$INCIDENCIA 2> /dev/null
        chmod 700 $DIR_REGISTRO/$INCIDENCIA
                mkdir $DIR_RESULTADOS 2> /dev/null
        chmod 700 $DIR_RESULTADOS
        echo "$encabezado: $mensaje" >> $DIR_RESULTADOS/$LOG
    else
        logger "SAWESO.SH: No se ha podido registrar el mensaje $mensaje. No existe el directorio de log $DIR_REGISTRO"
        exit 1
    fi
else
    logger "SAWESO.SH: No se ha podido registrar el mensaje en el fichero de log. Falta código de incidencia o el mensaje esta
vacío"
    exit 1
fi
}
```

```
#NOMBRE: Resultados - empaqueta los resultados en .tar.gz para su posterior transporte hacia el sistema saweso
#SINOPSIS: Resultados
Resultados()
{
cd $DIR_RESULTADOS
tar -cpzf $RESULTADOS * 2> /dev/null
mv $RESULTADOS $DIR_REGISTRO/$INCIDENCIA
}


################################################################ FUNCIONES MODIFICABLES
################################################################
#NOMBRE: Precondicion - Especifica la condicion que debe cumplirse para que se proceda a ejecutar el script (funcion Script)
#SINOPSIS: Precondicion
#SALIDA: devuelve 1 - Cumple la condicion; 0 - No cumple la condicion
Precondicion()
{
#EJEMPLO DE PRECONDICION. MODIFICAR SEGUN NECESIDADES
#Solo se ejecutará la funcion Script si estamos en un servidor de contenidos (c0) bajo Debian Etch (4.0)
coincidencia=`hostname -f | awk -F"." '{print $1}'`

if [ "$coincidencia" == "c0" ];then
        distro=`cat /etc/debian_version`
        if [ "$distro" == "4.0" ];then
                return 1
        else
                return 0
        fi
else
        return 0
fi
}


#NOMBRE: Script - comando, script o workaorund a ejecutar en el equipo final
#SINOPSIS: Script
#SALIDA: si deseamos redireccionar la salida o salidas a ficheros de texto debemos tener en cuenta que:
#- El fichero de salida debe almacenarse en $DIR_RESULTADOS
#- El nombre del fichero debe comenzar con la cadena "$ENCABEZADO"
#- Si deseamos ademas que la salida se adjunte en el informe mediante correo electrónico, el nombre del fichero deberá finalizar en
"$SALIDA"
#Ejemplo: uname -a >> $DIR_RESULTADOS/$ENCABEZADO.uname.$SALIDA
Script()
{
#EJEMPLO DE SCRIPT. MODIFICAR SEGUN NECESIDADES
#Establecemos el contenido del fichero /etc/tipo_centro a TIC:0809:DELL y comprobamos el cambio efectuado
[ -r /etc/tipo_centro ] && mv /etc/tipo_centro /etc/tipo_centro.$FECHA.$HORA.backup
echo "TIC:0809:DELL" >> /etc/tipo_centro
tipo=`cat /etc/tipo_centro`

if [ "$tipo" == "TIC:0809:DELL" ];then
echo -e "TEST OK (`hostname -f`): Fichero /etc/tipo_centro correcto = $tipo" >> $DIR_RESULTADOS/$ENCABEZADO.tipo_centro.
$SALIDA
else
echo -e "TEST FAIL (`hostname -f`)" >> $DIR_RESULTADOS/$ENCABEZADO.tipo_centro.$SALIDA
fi
}


#NOMBRE: Adjuntos - Carga el array ADJUNTOS con los comandos que producirán los ficheros adjuntos que se desean incorporar
en los resultados tras la ejecución del script
#SINOPSIS: Adjuntos
```

```
#SALIDA: las salidas de los comandos que generarán los adjuntos deben cumprir las siguientes condiciones:
#- El fichero de salida debe almacenarse en $DIR_RESULTADOS
#- El nombre del fichero debe comenzar con la cadena "$ENCABEZADO"
#- Si deseamos ademas que la salida se adjunte en el informe mediante correo electrónico, el nombre del fichero deberá finalizar en
"$SALIDA"
Adjuntos()
{
#EJEMPLO DE CAPTURA DE ADJUNTOS. MODIFICAR SEGUN NECESIDADES
#Recogemos la siguiente información del equipo
lspci >> $DIR_RESULTADOS/$ENCABEZADO.lspci
uname -ra >> $DIR_RESULTADOS/$ENCABEZADO.uname
cp /var/log/syslog $DIR_RESULTADOS/$ENCABEZADO.syslog

#Notificamos por email de la información recogida
echo "Adjuntos capturados en el equipo: lspci, uname -ra y syslog" >> $DIR_RESULTADOS/$ENCABEZADO.script.$SALIDA
}


############################################################## FIN FUNCIONES MODIFICABLES
#################################################################
############################################################## CUERPO DEL SCRIPT
#################################################################

ComprobarEntornoEjecucion
log "Entorno de ejecucion comprobado OK"
Precondicion
if [ $? -eq 1 ];then
        log "##################### INICIO EJECUCION $0 ($FECHA.$HORA) ####################"
    log "Precondicion para ejecucion de $0 OK"
        Script
        log "Script $0 ejecutado con exito"
        Adjuntos
        log "Se han recopilado los ficheros adjuntos solicitados"
        log "*) Fichero saweso.log con el log de la ejecucion"
        log "*) Ficheros adjuntos solicitados"
        log "*) Fichero .txt con la salida del script $0"
        log "##################### FIN EJECUCION $0 ($FECHA.$HORA) ####################"
        Resultados
else
    log "##################### INICIO EJECUCION $0 ($FECHA.$HORA) ####################"
        log "Precondicion para ejecucion de $0 FAIL. No se ejecutará nada"
        log "##################### FIN EJECUCION $0 ($FECHA.$HORA) ####################"
        Resultados
        exit 1
fi
```

8.1.5. Gestión de plantillas

Los almacenes de plantillas de ejecución se ubican en una carpeta local del servidor donde esta alojado el sistema. Este carpeta almacén contiene una jerarquía principal de 2 directorios:

- **Directorio equipo**: almacena todos las plantillas de ejecución que se pueden ejecutar sobre equipos clientes. Las plantillas de ejecución se organizan en 6 subdirectorios para facilitar su búsqueda e indicar la finalidad para la que han sido diseñadas.
    - Subdirectorio Busqueda: plantillas de ejecución destinadas a localizar patrones concretos sobre los equipos clientes donde sean ejecutadas. *Ejemplo: localizar el mensaje “server c0 not responding" en los log del equipo, …*
    - Subdirectorio Comando: plantillas de ejecución destinadas a ejecutar comandos simples en los equipos clientes. *Ejemplo: dmesg | grep -i warning, uptime, uname -a, …*
    - Subdirectorio Informe: plantillas de ejecución destinadas a obtener un informe formateado con información concreta sobre los equipos.
    - Subdirectorio Recolección: plantillas de ejecución destinadas a copiar determinados ficheros desde los equipos clientes. *Ejemplo: copiar los ficheros /var/log/syslog*, …*
    - Subdirectorio Script: plantillas de ejecución destinadas a ejecutar scripts con una finalidad especifica. *Ejemplo: script que detecte si existen problemas de actualización en el equipo y que fuerce la actualización del mismo comprobando al finalizar que la actualización ha terminado correctamente, …*
    - Subdirectorio Workaround: plantillas de ejecución destinadas a ejecutar Workaorund conocidos y previamente documentados.

- **Directorio servidor**: almacena todos las plantillas de ejecución que se pueden ejecutar sobre servidores. Las plantillas de ejecución se organizan en los mismos 6 subdirectorios, y con las mismas finalidades, descritos para el directorio de equipo.

8.1.6. Ejecución de plantillas

Esta funcionalidad solo estará disponible para los usuarios con perfil de “Programador” o de “Operador”.

Se utiliza una vista que permite la navegación por los diferentes tipos de host (servidor o equipo cliente) y las distintas categorías de plantillas de ejecución existentes (búsqueda, comando, informe, recolección, script, workaround) hasta seleccionar el tipo de ejecución deseada para la que queremos subir una nueva plantilla o lanzar una ejecución.

Si queremos subir una nueva plantilla (utilizando la plantilla base modificada según nuestras necesidades), tendremos que cargarla y almacenarla definitivamente en el servidor mediante el formulario disponible.

Si queremos ejecutar una plantilla, tendremos que proporcionar el listado de direcciones ip (separadas por un retorno de carro) sobre las que queremos operar.

La interfaz gráfica que se utiliza para permitir esta funcionalidad es la siguiente:

Prototipo de interfaz web para la ejecución de plantillas

Una vez iniciada la ejecución, el subsistema bash se encargará, entre otras, de realizar las siguientes funciones:

- Validar direcciones IP
- Comprobar la visibilidad con el host
- Comprobar la conectividad con el host
- Crear túneles con los equipos clientes
- Conectar mediante SSH

- Recoger los resultados obtenidos
- Generar el informe final

8.1.7. Notificación de eventos mediante email

La notificación de eventos se realizará a la dirección de correo electrónico proporciona al realzar el alta de cada usuario.

Al finalizar cada ejecución se enviará un informe resumen en texto plano a dicha dirección de correo.

8.1.8. Registro de la actividad y consulta de logs

En la pantalla correspondiente al menú Vistazo se centralizará el acceso a toda la información del estado, resultados y logs que se genere durante el funcionamiento de la aplicación.

8.2. Manual de usuario

Para acceder a la aplicación, desde cualquier navegador web, introduciremos la siguiente URL en la barra de direcciones http://IP_SERVIDOR:8080/SAWESO/.

La aplicación cuenta con un **usuario por defecto (usuario: saweso, contraseña: saweso)** que pertenece a todos los grupos y, por tanto, permite la gestión integral del aplicativo.

8.2.1. Login

Inicialmente, cuando se accede a la aplicación, se solicita un usuario y una contraseña mediante la siguiente ventana:

Tras introducir los datos, la aplicación identificará si el usuario es o no un usuario registrado en el sistema validando su autenticidad. La validación de los datos introducidos es “case sensitive”, es decir, sensible al uso de mayúsculas y minúsculas.

Si los datos del usuario no son correctos, se mostrará el siguiente mensaje de error:

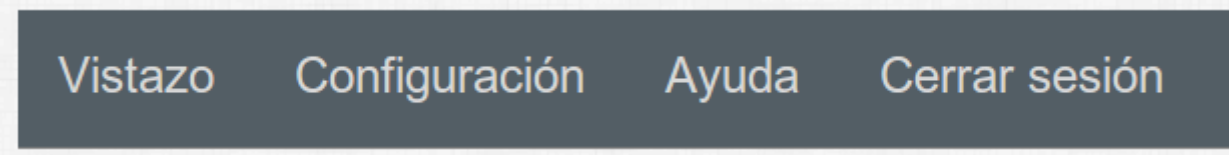
Si el usuario se valida correctamente, la aplicación reconocerá el perfil al que pertenece dicho usuario y en función de este le ofrecerá un entorno personalizado de la aplicación permitiéndole el uso de sus diferentes funciones. En función del perfil del usuario que se conecta se habilitarán distintas funcionalidades. Solo podrán ser accesibles por el usuario las funcionalidades asignadas a su perfil (grupo) concreto.

Existen 4 perfiles de usuario, o grupos: Administrador, Gestor de grupo, Programador, Operador. Las funcionalidades generales de cada grupo son:

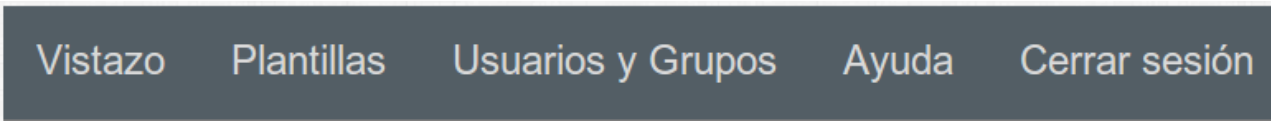
- **Administrador**, acceso a los menús de configuración de la aplicación (directorios de trabajo).
- **Gestor de grupo**, acceso a los menús de gestión de la aplicación. Sus funciones son: crear nuevos usuarios, asignar a cada usuario a uno o varios perfiles (grupos), modificar la visibilidad de las plantillas de ejecución asignándolas a grupos concretos y deshabilitar plantillas de ejecución.
- **Programador**, acceso a los menús de programación de plantillas y ejecución avanzada de la aplicación. Sus funciones son: crear nuevas plantillas de ejecución y ejecutar cualquier plantilla pertenecientes a cualquiera de los 6 tipos de plantillas existentes en el sistema (comando, script, búsqueda, recolección, informe y workaround).
- **Operador**, acceso a los menús de ejecución básica de la aplicación. Sus funciones son: ejecutar los tipos de plantillas, específicamente asignadas al grupo de operadores, que permite la aplicación.

Los menús visibles para cada uno de los 4 perfiles de usuario, o grupos, serán los siguientes:

Menú para usuario con perfil “Administrador”



Menú para usuario con perfil “Gestor de grupo”



Menú para usuario con perfil “Programador”

Menú para usuario con perfil “Operador”

Un usuario podrá pertenecer a varios perfiles de forma simultánea, de forma, que tendrá los privilegios acumulativos de cada perfil contando con el acceso a las funcionalidades que le permita cada perfil individual al que pertenezca.

8.2.2. Menú Ejecución PRO

La pantalla de Ejecución PRO nos ofrece dos funcionalidades:

- Subir nuevas plantillas de ejecución
- Ejecutar cualquier plantilla de ejecución

8.2.2.1. Subir una nueva plantilla de ejecución

Esta funcionalidad está implementa en la sección superior de la pantalla:

Para crear y dar de alta una nueva plantilla de ejecución debemos partir de los modelos de plantillas proporcionados por la aplicación. En la parte superior de la sección tenemos 2 enlaces donde podemos descargar un modelo para cada uno de los 2 tipos de ejecuciones que podemos programar:

- Ejecución sin adjuntos (saweso.modelo_script.sh) → en el host remoto se ejecutará la plantilla de ejecución.
- Ejecución con ficheros adjuntos (saweso.modelo_script_conAdjuntos.tar.gz) → en

el host remoto se ejecutará la plantilla de ejecución, contenida en el archivador .tar.gz, y además tendremos la posibilidad de trasportar en el mismo archivador todos los ficheros adjuntos (ficheros de configuración de servicios, módulos para componentes hardware, …) que necesitemos en el host remoto para ejecutar la operación para la que se ha diseñado la plantilla.

Para incorporar nuevas ejecuciones a SAWESO debemos crear una nueva plantilla de ejecución para el tipo de ejecución que deseemos: Servidor o Equipo cliente
El protocolo a seguir para generar una nueva plantilla de ejecución se divide en los siguientes pasos:

1. Descargar la plantilla de ejemplo deseada desde el servidor de SAWESO a una ruta fuera de cualquier entorno de explotación, por ejemplo el directorio /tmp de nuestra propia máquina.
2. Cambiar al directorio donde hayamos copiado la plantilla de ejemplo y editarla con cualquier editor de texto plano (vi, vim, nano, gedit, …).
3. Modificamos el encabezado por defecto por el encabezado final, describiendo claramente para que sirve la plantilla que estamos programando.

```
#Autor: <Nombre desarrollador>
#Actualizado: <fecha actualización>
#Descripcion: <Descripcion del script>
```

   A modo de ejemplo lo cambiamos por:

```
#Autor: José Antonio Serrano Rosso
#Actualizado: 12-01-2013
#Descripcion: Comprueba si el esquema de cableado en los centros TIC 2008-2009 está correcto
```

4. Modificar las funciones de la plantilla, sea cual sea el tipo de de ejecución que deseamos programar **solo debemos modificar estas 3 funciones**:
   - **Precondicion()**, es esta función especificamos que condición debe cumplirse en el host remoto para continuar con la ejecución del script, es

decir, de la función Script. *Ejemplo:*

```
#NOMBRE: Precondicion - Especifica la condicion que debe cumplirse para que se proceda a ejecutar el script (funcion Script)

#SINOPSIS: Precondicion

#SALIDA: devuelve 1 - Cumple la condicion; 0 - No cumple la condicion

Precondicion()

{

#EJEMPLO DE PRECONDICION. MODIFICAR SEGUN NECESIDADES

#Solo se ejecutará la funcion Script si estamos en un servidor de contenidos (c0) bajo Debian Etch (4.0)

coincidencia=`hostname -f | awk -F"." '{print $1}'`


if [ "$coincidencia" == "c0" ];then
        distro=`cat /etc/debian_version`
        if [ "$distro" == "4.0" ];then
                return 1
        else
                return 0
        fi
else
        return 0
fi
}
```

- **Script()**, esta función es la encargada de realizar todo el trabajo en el host remoto mediante un bloque de instrucciones bash que, en el caso de ejecuciones con adjuntos, puede utilizar otros ficheros o scripts adjuntos que transporte la plantilla. Ejemplo:

```
#NOMBRE: Script - comando, script o workaorund a ejecutar en el equipo final

#SINOPSIS: Script

#SALIDA: si deseamos redireccionar la salida o salidas a ficheros de texto debemos tener en cuenta que:

#- El fichero de salida debe almacenarse en $DIR_RESULTADOS

#- El nombre del fichero debe comenzar con la cadena "$ENCABEZADO"

#- Si deseamos ademas que la salida se adjunte en el informe mediante correo electrónico, el nombre del fichero deberá finalizar en "$SALIDA"
```

```
#Ejemplo: uname -a >> $DIR_RESULTADOS/$ENCABEZADO.uname.$SALIDA

Script()

{

#EJEMPLO DE SCRIPT. MODIFICAR SEGUN NECESIDADES

#Establecemos el contenido del fichero /etc/tipo_centro a TIC:0809:DELL y comprobamos el cambio efectuado

[ -r /etc/tipo_centro ] && mv /etc/tipo_centro /etc/tipo_centro.$FECHA.$HORA.backup

echo "TIC:0809:DELL" >> /etc/tipo_centro

tipo=`cat /etc/tipo_centro`


if [ "$tipo" == "TIC:0809:DELL" ];then

echo -e "TEST OK (`hostname -f`): Fichero /etc/tipo_centro correcto = $tipo" >> $DIR_RESULTADOS/
$ENCABEZADO.tipo_centro.$SALIDA

else

echo -e "TEST FAIL (`hostname -f`)" >> $DIR_RESULTADOS/$ENCABEZADO.tipo_centro.$SALIDA

fi

}
```

En cualquier punto del bloque de instrucciones bash se pueden incorporar salidas a ficheros de texto que posteriormente serán recogidos por el sistema. Si deseamos redireccionar la salida o salidas a ficheros de texto debemos tener en cuenta:

- El fichero de salida debe almacenarse en $DIR_RESULTADOS
- El nombre del fichero de salida debe comenzar con la cadena $ENCABEZADO
- Si deseamos, además, que la salida se adjunte en el informe mediante correo electrónico, el nombre del fichero deberá finalizar en "$SALIDA"

- **Adjuntos()**, aquí podemos indicar si deseamos o no capturar algún fichero adjunto del host remoto (syslog, passwd, httpd.conf, ...). La recogida de adjuntos permite las mismas salidas que la ejecución del script.

```
#NOMBRE: Adjuntos - Carga el array ADJUNTOS con los comandos que producirán los ficheros adjuntos que se desean
incorporar en los resultados tras la ejecución del script

#SINOPSIS: Adjuntos
```

```
#SALIDA: las salidas de los comandos que generarán los adjuntos deben cumprir las siguientes condiciones:

#- El fichero de salida debe almacenarse en $DIR_RESULTADOS

#- El nombre del fichero debe comenzar con la cadena "$ENCABEZADO"

#- Si deseamos ademas que la salida se adjunte en el informe mediante correo electrónico, el nombre del fichero deberá finalizar en "$SALIDA"

Adjuntos()

{

#EJEMPLO DE CAPTURA DE ADJUNTOS. MODIFICAR SEGUN NECESIDADES

#Recogemos la siguiente información del equipo

lspci >> $DIR_RESULTADOS/$ENCABEZADO.lspci

uname -ra >> $DIR_RESULTADOS/$ENCABEZADO.uname

cp /var/log/syslog $DIR_RESULTADOS/$ENCABEZADO.syslog


#Notificamos por email de la información recogida

echo "Adjuntos capturados en el equipo: lspci, uname -ra y syslog" >> $DIR_RESULTADOS/$ENCABEZADO.script.
$SALIDA

}
```

Si utilizamos la función de adjunto debemos descomentar la llamada a dicha función que se realiza en las últimas líneas de la plantilla de ejecución

```
#¡¡DESCOMENTAR LAS SIGUIENTES LINEAS SI QUEREMOS UTILIZAR LA FUNCION DE ADJUNTOS!!

#Adjuntos

#log "Se han recopilado los ficheros adjuntos solicitados"

#log "*) Fichero saweso.log con el log de la ejecucion"

#log "*) Ficheros adjuntos solicitados"

#log "*) Fichero .txt con la salida del script $0"
```

5. Comprobar el funcionamiento de la plantilla generada copiándola al directorio /tmp de un equipo o un servidor de muestra, dependiendo del tipo de ejecución, y ejecutándola manualmente.

   Copiamos el modelo de plantilla modificado a una máquina remota que coincida con los parámetros que hemos establecido en la precondición.

```
scp -p /tmp/saweso.modelo_script.sh root@<ip_maquina_remota>:/tmp
```

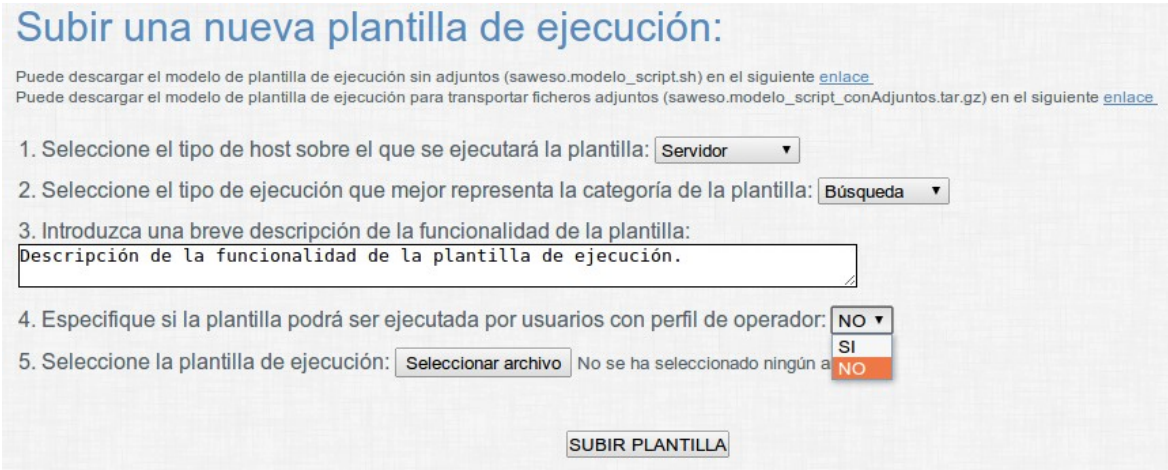
Ejecutamos la plantilla en la máquina remota simulando los parámetros que le proporcionará la aplicación SAWESO. A modo de ejemplo utilizamos el código de ejecución “1” e ip de la máquina remota “2”:

```
HOST:/tmp# bash saweso.modelo_script.sh 1 2
```

Comprobamos que se ejecuta correctamente y sin errores de sintaxis.

6. Verificamos que que los resultados de la ejecución, los adjuntos capturados y las salida en ficheros de texto son las esperadas. Para ello comprobamos que ha realizado correctamente la función del script y verificamos el contenido del directorio de resultados: /tmp/<codigo_ejecucion>, en este caso hemos elegido la ejecucion número 1 a modo de prueba, por tanto debemos acceder a: /tmp/1
   El contenido del directorio de resultados muestra un directorio con la fecha de ejecución (12-01-2013.14_42_34) y un archivador resultado de la compresión automática que realiza de este directorio (12-01-2013.14_42_34.tar.gz).
   Si accedemos al directorio veremos los resultados de la ejecución del script: ficheros adjuntos solicitados (.lspci, .syslog, .uname), fichero .salida.log que contiene la información para la notificación vía mail y un fichero .saweso.log con el log de la ejecución de la plantilla.
7. Si la prueba ha sido satisfactoria procedemos a renombrar la plantilla modificada con el nombre definitivo, y a subirla al servidor de SAWESO en el tipo de host para el que está diseñada la plantilla y en el tipo de ejecución que mejor represente la categoría de la plantilla.

8. Al subir la plantilla a la aplicación deberemos introducir una breve descripción de la funcionalidad de la plantilla (cuadro de texto de 150 caracteres) e indicar si la plantilla podrá ser ejecutada, o no, por usuarios con perfil de Operador.



9. Tras seleccionar la plantilla de ejecución a subir y pulsar el botón “SUBIR PLANTILLA” se mostrará el mensaje de confirmación: “Nueva plantilla subida al sistema”. **A partir de este momento la nueva plantilla de ejecución estará disponible para todos los usuarios con perfil Programador y para los usuarios con perfil Operador dependiendo de la visibilidad seleccionada.**

Transporte de ficheros adjuntos y Ejecución de scripts adjuntos

Como hemos mencionado al principio de este apartado, un caso especial en la generación de plantillas se da cuando deseamos transportar junto con el script de la plantilla de ejecución uno o varios ficheros adjuntos. Estos ficheros pueden ser: scripts complejos que ya han sido desarrollados previamente, ficheros de configuración, ficheros de texto, módulos nuevos para el hardware soportado por la distribución, …

Para permitir el transporte de adjuntos tenemos que añadir algunos pasos extra al proceso descrito anteriormente antes de subir la plantilla de ejecución al servidor SAWESO. Análogamente al modelo para ejecución con ficheros adjuntos (saweso.modelo_script_conAdjuntos.tar.gz) empaquetaremos la plantilla de ejecución generada siguiendo estos pasos:

1. Con la plantilla previamente generada, siguiendo los pasos descritos anteriormente, empaquetamos en un archivador .tar.gz todos los archivos que deseamos transportar, añadiendo un archivo inicio.sh que se encargará de ejecutar el script en el host remoto. El contenido del script inicio.sh será:

   ```
   bash saweso.<nombre_identificativo>.sh $1 $2
   ```

   Donde <nombre_identificativo> debe sustituirse por el nombre de la plantilla que hemos generado.

   Por ejemplo: en nuestra máquina, creamos un archivador para transportar dos ficheros adjuntos "fichero_adjunto1" y "fichero_adjunto2" junto con una plantilla de ejecución denominada "saweso.modelo_script.sh". El nombre del archivador será "saweso.modelo_script_conAdjuntos.tar.gz"

   ```
   root@elrond:/tmp/prueba# ls
   fichero_adjunto1
   fichero_adjunto2
   inicio.sh
   saweso.modelo_script.sh
   ```

```
root@elrond:/tmp/prueba# cat inicio.sh
bash saweso.modelo_script.sh $1 $2

root@elrond:/tmp/prueba# tar -czf saweso.modelo_script_conAdjuntos.tar.gz *

root@elrond:/tmp/prueba# ls
 fichero_adjunto1
fichero_adjunto2
inicio.sh
saweso.modelo_script_conAdjuntos.tar.gz
saweso.modelo_script.sh
```

Para utilizar estos ficheros adjuntos (fichero_adjunto1 y fichero_adjunto2) en el host remoto solo tenemos que saber que estarán disponibles durante la plantilla de ejecución en la ruta /tmp/<nombre_fichero_adjunto>.

Ejemplo: queremos utilizar un fichero adjunto, denominado "e1000.ko", que transportamos junto con la plantilla de ejecución y el fichero inicio.sh. En concreto deseamos copiar este nuevo módulo (e1000.ko) para una tarjeta de red e1000 de un servidor de seguridad con núcleo 2.6.18-6-amd64. Definimos las condiciones en la precondición (f0 y versión del núcleo) y durante la ejecución del script (función Script) copiamos dicho módulo a su ruta definitiva como se indica en el ejemplo.

```
Script()
{
if [ -e  /lib/modules/2.6.18-6-amd64/kernel/drivers/net/e1000/e1000.ko ];then
mv /lib/modules/2.6.18-6-amd64/kernel/drivers/net/e1000/e1000.ko /lib/modules/2.6.18-6-amd64/kernel/drivers/net/e1000/e1000.ko.backup
install -m 644 -c /tmp/e1000.ko /lib/modules/2.6.18-6-amd64/kernel/drivers/net/e1000/
depmod -a
modprobe
echo "Módulo e1000.ko copiado con éxito en el servidor `hostname -f`" >> $DIR_RESULTADOS/$ENCABEZADO.e1000.$SALIDA
else
```

```
echo “No existe el módulo e1000 en este servidor `hostname -f`” >> $DIR_RESULTADOS/
$ENCABEZADO.e1000.$SALIDA

fi

}
```

Para ejecutar un script adjunto en el host remoto procederos de la misma forma que en el caso anterior.

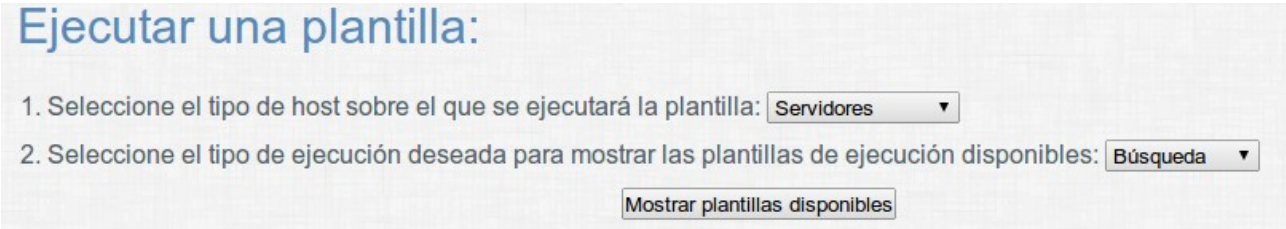
Ejemplo: queremos ejecutar un script denominado “ldap_script.sh” adjunto que transportamos junto con la plantilla de ejecución y el fichero inicio.sh. Además queremos que la salida del script se notifique en el informe mediante correo electrónico.

```
Script()

{

echo -e “Salida de la ejecución del script ldap_script.sh en el servidor `hostname -f`:\n” >> $DIR_RESULTADOS/
$ENCABEZADO.ldap.$SALIDA

bash /tmp/ldap_script.sh >> $DIR_RESULTADOS/$ENCABEZADO.ldap.$SALIDA

}
```

2. Análogamente al proceso descrito anteriormente, procedemos a subir el archivador creado al servidor de SAWESO en el tipo de host para el que está diseñada la plantilla y en el tipo de ejecución que mejor represente la categoría de la plantilla. **A partir de este momento la nueva plantilla de ejecución estará disponible para todos los usuarios con perfil Programador y para los usuarios con perfil Operador dependiendo de la visibilidad seleccionada.**

8.2.2.2. Ejecutar una plantilla

Esta funcionalidad está implementa en la sección inferior de la pantalla:

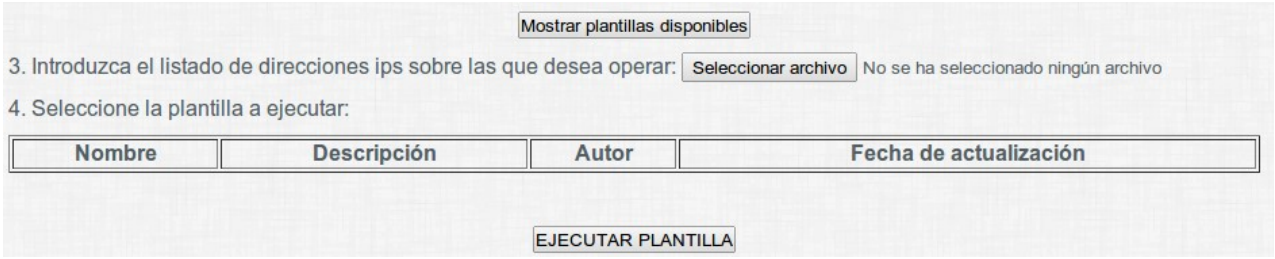


La pantalla de Ejecución PRO ofrece la posibilidad de ejecutar cualquier plantilla de ejecución que no se encuentre deshabilitada.

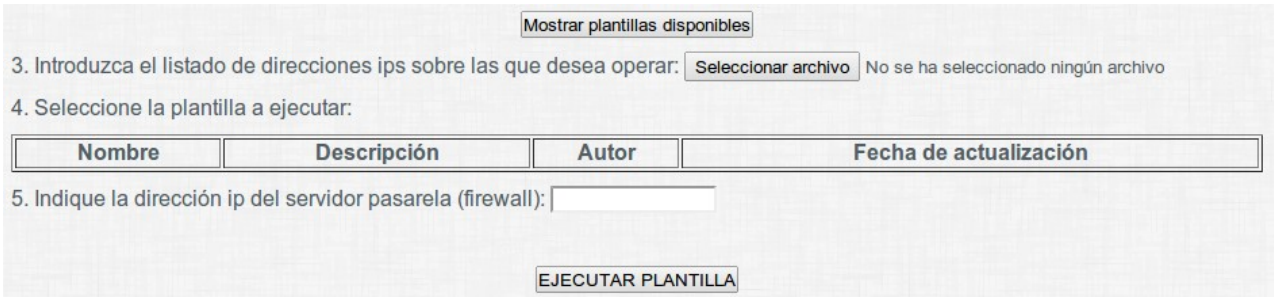
Para lanzar una nueva ejecución debemos seleccionar el tipo de host sobre el que queremos lanzar la ejecución (Servidores o Equipos clientes) y, dentro del tipo de host seleccionado, el tipo de ejecución (Búsqueda, Comando, Script, Workaround, Informe y Recolección) deseada.

Tras pulsar el botón “Mostrar plantillas disponibles” se añadirá, dependiendo del tipo de host seleccionado, el siguiente formulario:

Host tipo “Servidores”



Host tipo “Equipos clientes”



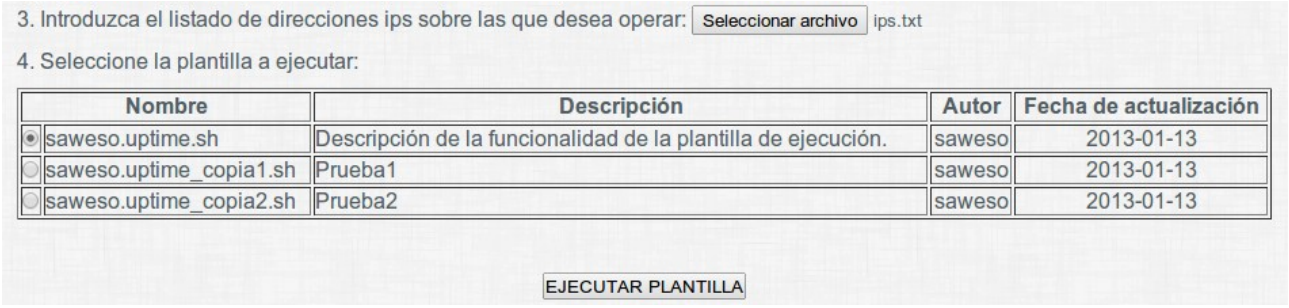
En el **punto 3**, se solicita un fichero de texto que contenga las direcciones ip de los hosts donde se desea lanzar la ejecución. Cada dirección ip debe aparecer en una línea diferente deparada por un retorno de carro:

```
170.26.0.58
170.26.0.59
170.26.0.60
```

170.26.0.61
170.26.0.62
170.26.0.63
170.26.0.64
170.26.0.65
170.26.0.66
170.26.0.67
170.26.0.68
170.26.0.69
170.26.0.70
170.26.0.71
170.26.0.72
170.26.0.73
170.26.0.74
170.26.0.75

La aplicación validará la coherencia de todas las direcciones ips introducidas mostrando un mensaje de error en caso de encontrar ips incorrectas en el fichero y especificando que ips considera erróneas.

En el **punto 4**, se solicita la selección de la plantilla que se desea ejecutar sobre el listado de direcciones ips proporcionado.



En el **punto 5 (solo Equipos clientes)**, se solicita la dirección ip del servidor (pasarela) del que “cuelgan” los equipos clientes.

En cualquiera de los casos, tras pulsar el botón “EJECUTAR PLANTILLA” la aplicación mostrará el mensaje de confirmación con el código asignado a la ejecución y lanzará la ejecución de la plantilla sobre los hosts remotos:

> Procesando ejecución, por favor espere ...
>
> Ejecucion lanzada con codigo: 2

8.2.3. Menú Ejecución OP

La pantalla de Ejecución OP ofrece la posibilidad de ejecutar **solo** las plantillas de ejecución que tengan visibilidad para usuarios con perfil Operador y que no se encuentren deshabilitadas.

El resto del comportamiento de la pantalla de Ejecución OP es idéntica a la funcionalidad de “Ejecutar una plantilla” de la pantalla de ejecución PRO.

8.2.4. Menú Vistazo

La pantalla Vistazo ofrece una panorámica de todas las ejecuciones existentes en la aplicación.

Los usuario con perfil de Operador o Programador solo podrán visualizar sus propias ejecuciones.

Los usuario con perfil de Gestor de grupo o Administrador verán todas las ejecuciones de todos los usuarios.

En una tabla se muestra un listado completo de todas las ejecuciones existentes indicándonos el estado actual de cada una y permitiéndonos el acceso a toda la información vinculada a la ejecución.

Ejecución	Estado	Fecha	Informe	Estadística	Plantilla	Log	Usuario
2	FINALIZADA	2013-01-13	Informe final	Estadística actual	saweso.modelo_script.sh	Log actual	saweso
1	FINALIZADA	2013-01-13	Informe final	Estadística actual	saweso.uptime.sh	Log actual	saweso

Desde esta pantalla podemos consultar:

- **Ejecución**, código secuencia que se asigna a cada ejecución
- **Estado**, estado actual de la ejecución. Una ejecución puede encontrarse en los siguientes estados:
  - EJECUTANDO, la ejecución está en curso.
  - GENERANDO INFORME, se está generando el informe en texto plano que se enlazará en el campo Informe.
  - ENVIANDO EMAIL, se está enviando el informe en texto plano generado a la dirección email del usuario que ha lanzado la ejecución.
  - FINALIZADA, la ejecución ha finalizado.
- **Fecha**, indica la fecha de inicio de la ejecución.
- **Informe**, al finalizar la ejecución disponemos en esta campo de un enlace a un informe resumen en texto plano con los resultados de la ejecución. Este mismo fichero será el que se notificará al email del usuario que ha lanzado la ejecución.

  Un ejemplo de este tipo de informe sería:

EJECUCION DE SAWESO (Fecha: 13-01-2013 Hora: 22_23_03)

Codigo de ejecucion: 1
Ejecucion sobre: servidor
Script ejecutado: /etc/saweso/PlantillasEjecucion/servidor/script/saweso.uptime.sh
E-mail de notificacion: saweso@dominio.com
Directorio de registro: /var/log/saweso/1
Directorio de resultados: /var/log/saweso/1/Resultados

RESUMEN ESTADO ACTUAL

Numero de equipos pendientes de ejecucion: 11
Numero de equipos donde se ha ejecutado el script: 7
Numero de equipos donde se ha ejecutado el script y se ha descargado el resultado de la ejecucion: 7

RESULTADOS DE LA EJECUCION POR EQUIPOS

IP: 170.26.0.59
(1.13-01-2013.22_38_31.170.26.0.59.uptime.salida.log):
22:38:31 up 10:40, 0 users, load average: 0.00, 0.00, 0.00

IP: 170.26.0.64
(1.13-01-2013.22_38_31.170.26.0.64.uptime.salida.log):
22:38:31 up 10:40, 1 user, load average: 0.00, 0.00, 0.00

IP: 170.26.0.65
(1.13-01-2013.22_38_31.170.26.0.65.uptime.salida.log):
22:38:31 up 10:40, 0 users, load average: 0.16, 0.03, 0.01

IP: 170.26.0.66
(1.13-01-2013.22_38_31.170.26.0.66.uptime.salida.log):
22:38:31 up 10:40, 0 users, load average: 0.00, 0.00, 0.00

IP: 170.26.0.67
(1.13-01-2013.22_38_31.170.26.0.67.uptime.salida.log):
22:38:31 up 10:39, 0 users, load average: 0.02, 0.02, 0.00

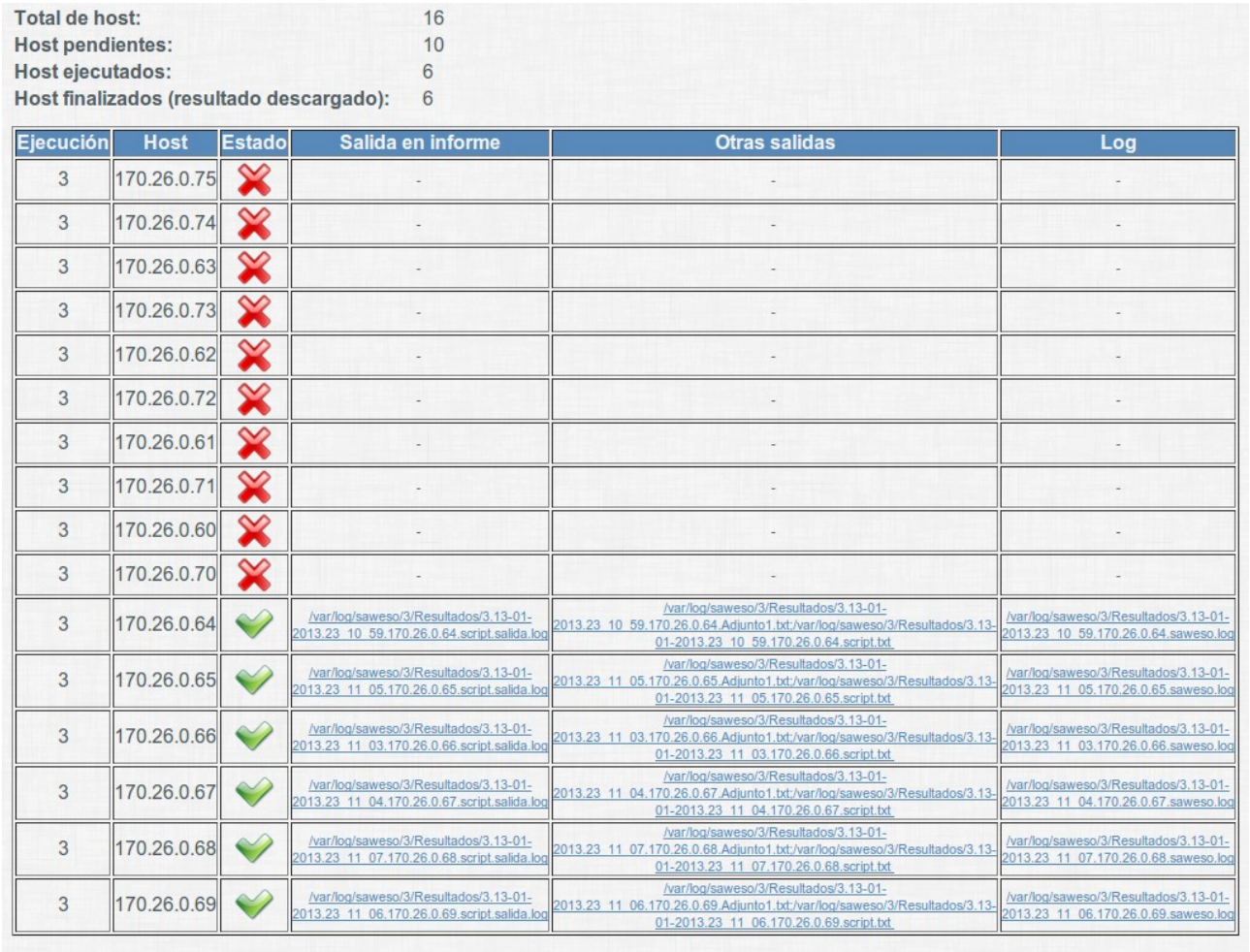
IP: 170.26.0.68
(1.13-01-2013.22_38_33.170.26.0.68.uptime.salida.log):
22:38:33 up 10:39, 0 users, load average: 0.00, 0.00, 0.00

IP: 170.26.0.69
(1.13-01-2013.22_38_34.170.26.0.69.uptime.salida.log):
22:38:34 up 10:39, 0 users, load average: 0.00, 0.00, 0.00

- **Estadística**, durante todo el proceso de la ejecución se muestra un enlace donde

se puede consultar un resumen del estado de evolución de la ejecución y todos los detalles de cada uno de los hosts que forman parte de la ejecución.

Para cada uno de los host podemos consultar: direccción ip, estado, salida que se ha incorporado el informe que se notificará por email, salida de cualquier fichero adjunto que la plantilla de ejecución haya generado y un fichero de log detallado de la evolución de la ejecución en el host remoto.



Total de host: 16
Host pendientes: 10
Host ejecutados: 6
Host finalizados (resultado descargado): 6

Ejecución	Host	Estado	Salida en informe	Otras salidas	Log
3	170.26.0.75	✗	-	-	-
3	170.26.0.74	✗	-	-	-
3	170.26.0.63	✗	-	-	-
3	170.26.0.73	✗	-	-	-
3	170.26.0.62	✗	-	-	-
3	170.26.0.72	✗	-	-	-
3	170.26.0.61	✗	-	-	-
3	170.26.0.71	✗	-	-	-
3	170.26.0.60	✗	-	-	-
3	170.26.0.70	✗	-	-	-
3	170.26.0.64	✓	/var/log/saweso/3/Resultados/3.13-01-2013.23_10_59.170.26.0.64.script.salida.log	/var/log/saweso/3/Resultados/3.13-01-2013.23_10_59.170.26.0.64.Adjunto1.txt;/var/log/saweso/3/Resultados/3.13-01-2013.23_10_59.170.26.0.64.script.txt	/var/log/saweso/3/Resultados/3.13-01-2013.23_10_59.170.26.0.64.saweso.log
3	170.26.0.65	✓	/var/log/saweso/3/Resultados/3.13-01-2013.23_11_05.170.26.0.65.script.salida.log	/var/log/saweso/3/Resultados/3.13-01-2013.23_11_05.170.26.0.65.Adjunto1.txt;/var/log/saweso/3/Resultados/3.13-01-2013.23_11_05.170.26.0.65.script.txt	/var/log/saweso/3/Resultados/3.13-01-2013.23_11_05.170.26.0.65.saweso.log
3	170.26.0.66	✓	/var/log/saweso/3/Resultados/3.13-01-2013.23_11_03.170.26.0.66.script.salida.log	/var/log/saweso/3/Resultados/3.13-01-2013.23_11_03.170.26.0.66.Adjunto1.txt;/var/log/saweso/3/Resultados/3.13-01-2013.23_11_03.170.26.0.66.script.txt	/var/log/saweso/3/Resultados/3.13-01-2013.23_11_03.170.26.0.66.saweso.log
3	170.26.0.67	✓	/var/log/saweso/3/Resultados/3.13-01-2013.23_11_04.170.26.0.67.script.salida.log	/var/log/saweso/3/Resultados/3.13-01-2013.23_11_04.170.26.0.67.Adjunto1.txt;/var/log/saweso/3/Resultados/3.13-01-2013.23_11_04.170.26.0.67.script.txt	/var/log/saweso/3/Resultados/3.13-01-2013.23_11_04.170.26.0.67.saweso.log
3	170.26.0.68	✓	/var/log/saweso/3/Resultados/3.13-01-2013.23_11_07.170.26.0.68.script.salida.log	/var/log/saweso/3/Resultados/3.13-01-2013.23_11_07.170.26.0.68.Adjunto1.txt;/var/log/saweso/3/Resultados/3.13-01-2013.23_11_07.170.26.0.68.script.txt	/var/log/saweso/3/Resultados/3.13-01-2013.23_11_07.170.26.0.68.saweso.log
3	170.26.0.69	✓	/var/log/saweso/3/Resultados/3.13-01-2013.23_11_06.170.26.0.69.script.salida.log	/var/log/saweso/3/Resultados/3.13-01-2013.23_11_06.170.26.0.69.Adjunto1.txt;/var/log/saweso/3/Resultados/3.13-01-2013.23_11_06.170.26.0.69.script.txt	/var/log/saweso/3/Resultados/3.13-01-2013.23_11_06.170.26.0.69.saweso.log

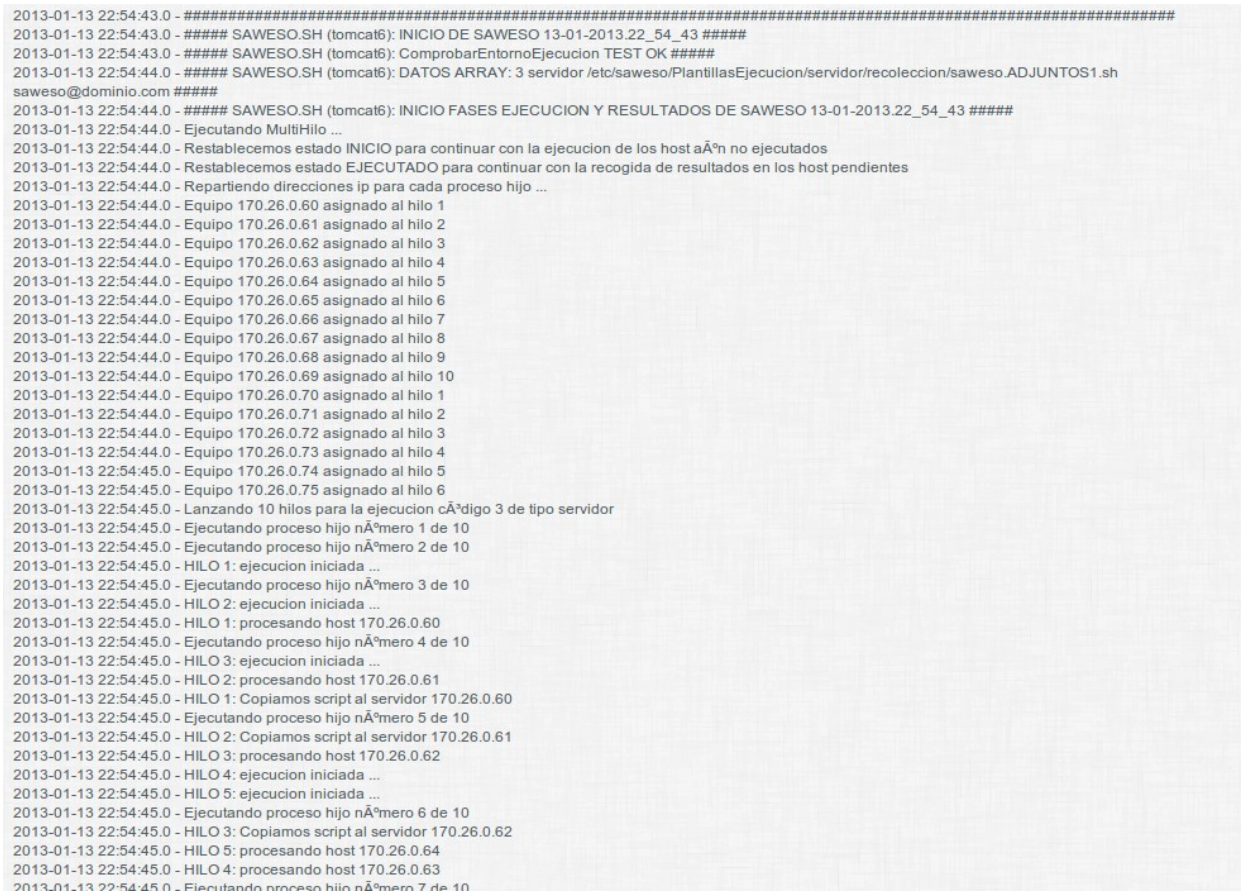
- **Plantilla**, enlace que nos conduce a otra página donde se muestran todos la información de la plantilla ejecutada.

Nombre:	saweso.uptime.sh
Tipo de host:	servidor
Categoría:	script
Autor:	saweso
Fecha Actualización:	2013-01-13
Descripción:	DescripciÃ³n de la funcionalidad de la plantilla de ejecuciÃ³n.
Grupo:	operador

PLANTILLA

```
#!/bin/bash
#Autor: José Antonio Serrano Rosso (CGA)
#Actualizado: 05-08-2008
#Descripcion: Muestra el tiempo que lleva levantado el sistema

#Obtenemos la fecha en formatos dd-mm-yyyy y la hora en formato hh:mm:ss
FECHA="$(date +"%d-%m-%Y")"
HORA="$(date +"%H_%M_%S")"

#Variables
declare -a ADJUNTOS
INCIDENCIA=$1
IP=$2

#Ficheros
ENCABEZADO=$INCIDENCIA.$FECHA.$HORA.$IP
LOG=$ENCABEZADO.saweso.log #Log de ejecucion en el equipo final
RESULTADOS=$ENCABEZADO.tar.gz #Archivado que se creará para transportar los resultados
SALIDA=salida.log #Si se redirecciona la salida de algún comando a salida.log, esta se incluirá en la notificación por e-mail

#Directorios de trabajo
DIR_REGISTRO="/tmp" #Directorio de trabajo
DIR_RESULTADOS="$DIR_REGISTRO/$INCIDENCIA/$FECHA.$HORA" #Directorio para almacenar los resultados de la ejecucion
```

- **Log**, enlace que nos conduce a otra página donde se muestra el log detallado de la ejecución.



```
2013-01-13 22:54:43.0 - ##########################################################################################################################
2013-01-13 22:54:43.0 - ##### SAWESO.SH (tomcat6): INICIO DE SAWESO 13-01-2013.22_54_43 #####
2013-01-13 22:54:43.0 - ##### SAWESO.SH (tomcat6): ComprobarEntornoEjecucion TEST OK #####
2013-01-13 22:54:44.0 - ##### SAWESO.SH (tomcat6): DATOS ARRAY: 3 servidor /etc/saweso/PlantillasEjecucion/servidor/recoleccion/saweso.ADJUNTOS1.sh
saweso@dominio.com #####
2013-01-13 22:54:44.0 - ##### SAWESO.SH (tomcat6): INICIO FASES EJECUCION Y RESULTADOS DE SAWESO 13-01-2013.22_54_43 #####
2013-01-13 22:54:44.0 - Ejecutando MultiHilo ...
2013-01-13 22:54:44.0 - Restablecemos estado INICIO para continuar con la ejecucion de los host aÃºn no ejecutados
2013-01-13 22:54:44.0 - Restablecemos estado EJECUTADO para continuar con la recogida de resultados en los host pendientes
2013-01-13 22:54:44.0 - Repartiendo direcciones ip para cada proceso hijo ...
2013-01-13 22:54:44.0 - Equipo 170.26.0.60 asignado al hilo 1
2013-01-13 22:54:44.0 - Equipo 170.26.0.61 asignado al hilo 2
2013-01-13 22:54:44.0 - Equipo 170.26.0.62 asignado al hilo 3
2013-01-13 22:54:44.0 - Equipo 170.26.0.63 asignado al hilo 4
2013-01-13 22:54:44.0 - Equipo 170.26.0.64 asignado al hilo 5
2013-01-13 22:54:44.0 - Equipo 170.26.0.65 asignado al hilo 6
2013-01-13 22:54:44.0 - Equipo 170.26.0.66 asignado al hilo 7
2013-01-13 22:54:44.0 - Equipo 170.26.0.67 asignado al hilo 8
2013-01-13 22:54:44.0 - Equipo 170.26.0.68 asignado al hilo 9
2013-01-13 22:54:44.0 - Equipo 170.26.0.69 asignado al hilo 10
2013-01-13 22:54:44.0 - Equipo 170.26.0.70 asignado al hilo 1
2013-01-13 22:54:44.0 - Equipo 170.26.0.71 asignado al hilo 2
2013-01-13 22:54:44.0 - Equipo 170.26.0.72 asignado al hilo 3
2013-01-13 22:54:44.0 - Equipo 170.26.0.73 asignado al hilo 4
2013-01-13 22:54:45.0 - Equipo 170.26.0.74 asignado al hilo 5
2013-01-13 22:54:45.0 - Equipo 170.26.0.75 asignado al hilo 6
2013-01-13 22:54:45.0 - Lanzando 10 hilos para la ejecucion cÃ³digo 3 de tipo servidor
2013-01-13 22:54:45.0 - Ejecutando proceso hijo nÃºmero 1 de 10
2013-01-13 22:54:45.0 - Ejecutando proceso hijo nÃºmero 2 de 10
2013-01-13 22:54:45.0 - HILO 1: ejecucion iniciada ...
2013-01-13 22:54:45.0 - Ejecutando proceso hijo nÃºmero 3 de 10
2013-01-13 22:54:45.0 - HILO 2: ejecucion iniciada ...
2013-01-13 22:54:45.0 - HILO 1: procesando host 170.26.0.60
2013-01-13 22:54:45.0 - Ejecutando proceso hijo nÃºmero 4 de 10
2013-01-13 22:54:45.0 - HILO 3: ejecucion iniciada ...
2013-01-13 22:54:45.0 - HILO 2: procesando host 170.26.0.61
2013-01-13 22:54:45.0 - HILO 1: Copiamos script al servidor 170.26.0.60
2013-01-13 22:54:45.0 - Ejecutando proceso hijo nÃºmero 5 de 10
2013-01-13 22:54:45.0 - HILO 2: Copiamos script al servidor 170.26.0.61
2013-01-13 22:54:45.0 - HILO 3: procesando host 170.26.0.62
2013-01-13 22:54:45.0 - HILO 4: ejecucion iniciada ...
2013-01-13 22:54:45.0 - HILO 5: ejecucion iniciada ...
2013-01-13 22:54:45.0 - Ejecutando proceso hijo nÃºmero 6 de 10
2013-01-13 22:54:45.0 - HILO 3: Copiamos script al servidor 170.26.0.62
2013-01-13 22:54:45.0 - HILO 5: procesando host 170.26.0.64
2013-01-13 22:54:45.0 - HILO 4: procesando host 170.26.0.63
2013-01-13 22:54:45.0 - Ejecutando proceso hijo nÃºmero 7 de 10
```

- **Usuario**, nos informa del usuario que ha lanzado la ejecución.

8.2.5. Menú Plantillas

La pantalla de plantillas muestra un resumen general de todas las plantillas de ejecución existentes y nos permite descargarlas mediante un enlace en el nombre de cada plantilla.

Permite establecer la visibilidad de cada plantilla de ejecución asignándola al grupo de programadores (Solo Programadores), operadores y programadores (Operadores y Programadores) o a ninguno (Ninguno). En caso de seleccionar la visibilidad a ninguno, los usuarios con perfiles de ejecución en el sistema (operador y programador) no podrán ver estas plantillas.

Tipo host	Tipo ejecución	Nombre plantilla	Descripción	Autor	Fecha creación	Visibilidad
servidor	recoleccion	saweso.ADJUNTOS.sh	Prueba recolección de adjuntos desde servidores.	saweso	2013-01-13	Solo Programadores
servidor	recoleccion	saweso.ADJUNTOS1.sh	Adjuntos	saweso	2013-01-13	Solo Programadores
servidor	workaround	saweso.authorized.tar.gz	Prueba trasporte de adjuntos, junto con plantilla de ejecución, a servidores.	saweso	2013-01-13	Solo Programadores
servidor	script	saweso.uptime.sh	Descripción de la funcionalidad de la plantilla de ejecución.	saweso	2013-01-13	Operadores y Programadores
servidor	script	saweso.uptime_copia1.sh	Prueba1	saweso	2013-01-13	Solo Programadores
servidor	script	saweso.uptime_copia2.sh	Prueba2	saweso	2013-01-13	Solo Programadores
equipo	comando	saweso.halt.sh	Prueba1 clientes	saweso	2013-01-13	Ninguno
equipo	busqueda	saweso.modelo_script.sh	Prueba clientes	saweso	2013-01-13	Solo Programadores

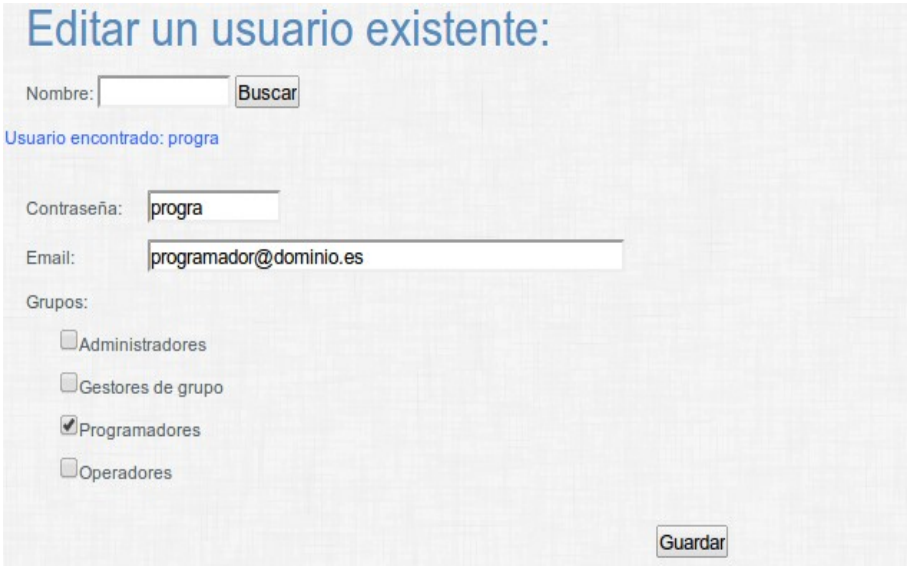
Guardar

8.2.6. Menú Usuarios y Grupos

En esta pantalla tenemos 2 funcionalidades diferentes:

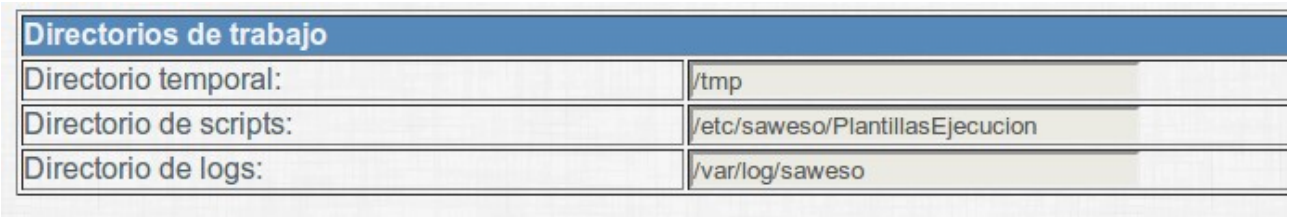
- En la sección superior podemos crear un nuevo usuario en la aplicación asignándolo a uno o a varios grupos simultáneamente.

- En la sección inferior podemos buscar y editar la información (clave, email y asignación a grupos) de cada usuario.



8.2.7. Menú Configuración

En esta pantalla solo podemos consultar los directorios de trabajo que está utilizando la aplicación.



9. Manual de instalación

La instalación y funcionamiento de la aplicación SAWESO ha sido comprobada bajo el sistema operativo Debian Squeeze (6.0.6), por tanto, se recomiendo el uso de la misma versión del sistema operativo para su instalación.

La instalación debe ejecutarse en el servidor, con sistema operativo Debian Squeeze, que albergará la aplicación y con el usuario root.

9.1. Requisitos pre-instalación

- Desde una terminal del servidor, instalamos toda la paquetería necesaria ejecutando la siguiente secuencia de comandos. Pulsaremos la tecla “Entrar” en la opción por defecto frente a cualquier pregunta que se realice durante el proceso de instalación de los paquetes:

```
apt-get update
apt-get -y install openjdk-6-jre openjdk-6-jdk
apt-get -y install tomcat6 tomcat6-docs tomcat6-examples tomcat6-admin
apt-get -y install mysql-server
apt-get -y install postfix
apt-get -y install ssh autossh
apt-get -y install netcat-openbsd
```

- Desde una terminal del servidor, establecemos la contraseña de root al servicio MySQL (por ejemplo: “holamundo”)

```
mysqladmin -h localhost -u root password "holamundo"
```

9.2. Instalación

El siguiente proceso de instalación automatizado establecerá un usuario administrador (user: “admin”, pass: “admin”) en el fichero de configuración de usuarios de Apache Tomcat.

Para realizar la instalación de SAWESO debemos ejecutar los siguientes pasos:

- Copiar el archivador del instalador (saweso_1.0.1.tar.gz) a cualquier ruta del servidor.
- Desde una terminal del servidor, descomprimimos el archivador (tar -xvvzf saweso_1.0.1.tar.gz). Tras la descompresión tendremos:
  - Carpeta “saweso_templates” → contiene todos los ficheros a instalar en el sistema Debian, así como el fichero “.war” que debemos desplegar manualmente en el servidor de aplicaciones Tomcat.
  - Script “install_saweso.sh” → instalador automatizado que realiza la instalación de la aplicación junto con la creación y carga inicial de la base de datos. A continuación mostramos el contenido del fichero install_saweso.sh:

```
#!/bin/bash
#Autor: José Antonio Serrano Rosso
#Actualizado: 08-01-2013
#Descripción: Instalador de SAWESO

#Obtenemos la fecha en formatos dd-mm-yyyy y la hora en formato hh:mm:ss
FECHA="$(date +"%d-%m-%Y")"
HORA="$(date +"%H_%M_%S")"

#Formateo de la salida por pantalla del instalador
txtrst=$(tput sgr0) # Text reset
txtred=$(tput setaf 1) # Red

#Directorios de trabajo
DIR_TEMPLATES="./saweso_templates"

#Variables
USER_SAWESO=tomcat6

######################################### REQUISITOS PREVIOS
#########################################
#La instalación del sistema SAWESO ha sido comprobada bajo el Sistema Operativo Debian Squeeze (6.0.6).

##Instalar la paquetería necesaria , desde una terminal ejecutamos:
echo "${txtred} Instalando la paquetería necesaria ... ${txtrst}"
```

```
apt-get -y install openjdk-6-jre openjdk-6-jdk
apt-get -y install tomcat6 tomcat6-docs tomcat6-examples tomcat6-admin
apt-get -y install mysql-server
apt-get -y install postfix
apt-get -y install ssh autossh
apt-get -y install netcat-openbsd

##Establecemos la contraseña de root al servidor MySQL (por ejemplo: "holamundo")
echo "${txtred} Establecemos la contraseña de root (holamundo) al servidor MySQL ${txtrst}"
mysqladmin -h localhost -u root password "holamundo"


######################################### SCRIPT DE INSTALACION
#########################################
#Comprobaciones iniciales
usuario=`whoami`
if [ "$usuario" != "root" ];then
        echo "FAIL: El script $0 debe ser ejecutado por el usuario root"
        exit 1
fi


#Instalacion de saweso
if [ -d $DIR_TEMPLATES ];then
        #Limpiamos instalaciones de saweso anteriores
        echo "${txtred} Limpiando datos de instalaciones anteriores ... (¡Se guardará una backup en
/var/backups/saweso.backup.$FECHA.$HORA!) ${txtrst}"
        mkdir -p /var/backups/saweso.backup.$FECHA.$HORA/etc.saweso.PlantillasEjecucion
        mkdir -p /var/backups/saweso.backup.$FECHA.$HORA/var.log.saweso
        cp -a /etc/saweso/PlantillasEjecucion /var/backups/saweso.backup.$FECHA.
$HORA/etc.saweso.PlantillasEjecucion
        rm -rf /etc/saweso/PlantillasEjecucion
        cp -a /var/log/saweso /var/backups/saweso.backup.$FECHA.$HORA/var.log.saweso
        rm -rf /var/log/saweso
        #Instalacion de ficheros y directorios
        echo "${txtred} Copiando directorio de scripts a /etc/saweso ... ${txtrst}"
        mkdir -p /etc/saweso
        cp -a $DIR_TEMPLATES/etc.saweso/* /etc/saweso
        echo "${txtred} Copiando script saweso.sh a /usr/bin ... ${txtrst}"
        cp -a $DIR_TEMPLATES/usr.bin/saweso.sh /usr/bin
        echo "${txtred} Creando directorio de log /var/log/saweso ... ${txtrst}"
        mkdir -p /var/log/saweso
        #Añadimos un usuario administrador (por ejemplo: “admin”, “admin”) al fichero de configuración de usuarios de
Apache Tomcat
        /etc/init.d/tomcat6 stop
        cp $DIR_TEMPLATES/etc.tomcat6/tomcat-users.xml /etc/tomcat6/
        chown root:tomcat6 /etc/tomcat6/tomcat-users.xml
        chmod 640 /etc/tomcat6/tomcat-users.xml
        /etc/init.d/tomcat6 start
        #Establecemos permisos correctos para el usuario y grupo que ejecutará el sistema
        chown -R $USER_SAWESO:$USER_SAWESO /etc/saweso
        chown $USER_SAWESO:$USER_SAWESO /usr/bin/saweso.sh
        chmod 755 /usr/bin/saweso.sh
        chown -R $USER_SAWESO:$USER_SAWESO /var/log/saweso

        #Creacion y carga de la base de datos
        PASSBD=""
        TESTBD=1
        until [ -n "$PASSBD" ] && [ $TESTBD -eq 0 ]; do
                echo -e "\nIntroduzca la contraseña de root de MySQL:"
                read PASSBD
                mysql -u"root" -p"$PASSBD" -D "mysql" -e "use mysql;" -N
                if [ $? -eq 0 ];then
```

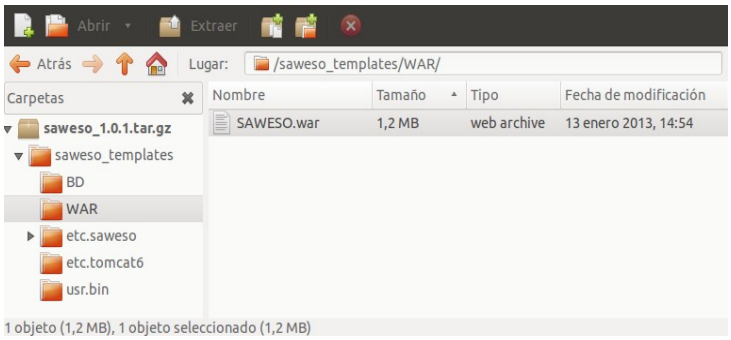
```
                                TESTBD=0
                        fi
                done

                echo "${txtred} Actualizamos la contraseña de la BD en el fichero de configuración (/etc/saweso/saweso.conf) $
{txtrst}"
                sed -i -e 's/'holamundo'/'$PASSBD'/' /etc/saweso/saweso.conf

                echo "${txtred} Borrando BD saweso existente ... ${txtrst}"
                mysqladmin -uroot -p$PASSBD drop saweso

                echo "${txtred} Creando nueva BD saweso ... ${txtrst}"
                mysqladmin -uroot -p$PASSBD create saweso

                echo "${txtred} Cargando estructura de tablas ... ${txtrst}"
                mysql -u root -p$PASSBD saweso < $DIR_TEMPLATES/BD/tablas_BD.txt

                echo "${txtred} Cargando datos iniciales en las tablas ... ${txtrst}"
                mysql -u root -p$PASSBD saweso < $DIR_TEMPLATES/BD/datos_iniciales_BD.txt
else
                echo "FAIL: no existe el directorio de templates $DIR_TEMPLATES en la ruta donde se está ejecutando $0"
fi


######################################### REQUISITOS MANUALES
#########################################
# Copiar el certificado id_dsa en el directorio /etc/saweso
# Borrar clave del certificado para evitar que la solicite en cada conexion
## ssh-keygen -p -P "<CLAVE>" -N "" -f /etc/saweso/id_dsa
## chown -R $USER_SAWESO:$USER_SAWESO /etc/saweso
## chmod 600 /etc/saweso/id_dsa

# Copiar certificado id_rsa en el directorio /etc/saweso
# Borrar clave del certificado para evitar que la solicite en cada conexion
## ssh-keygen -p -P "<CLAVE>" -N "" -f /etc/saweso/id_rsa
## chown -R $USER_SAWESO:$USER_SAWESO /etc/saweso
## chmod 600 /etc/saweso/id_rsa

# Revisar el fichero de configuracion general (/etc/saweso/saweso.conf) para comprobar que los nombres de los
certificados coinciden
```

- Desde una terminal del servidor, ejecutamos el instalador install_saweso.sh (bash install_saweso.sh). Mostramos la salida de dicho proceso:

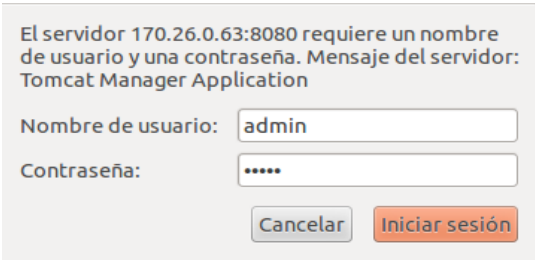
```
root@SqueezeBase:~# bash install_saweso.sh
 Comprobamos la instalanción de la paquetería necesaria ...
 ….....................................................
0 actualizados, 0 se instalarán, 0 para eliminar y 0 no actualizados.
 Establecemos la contraseña de root (holamundo) al servidor MySQL
 Limpiando datos de instalaciones anteriores ... (¡Se guardará una backup en /var/backups/saweso.backup.13-01-
2013.16_02_28!)
 Copiando directorio de scripts a /etc/saweso ...
 Copiando script saweso.sh a /usr/bin ...
 Creando directorio de log /var/log/saweso ...
Stopping Tomcat servlet engine: tomcat6.
Starting Tomcat servlet engine: tomcat6.

Introduzca la contraseña de root de MySQL:
holamundo
```

```
Actualizamos la contraseña de la BD en el fichero de configuración (/etc/saweso/saweso.conf)
Borrando BD saweso existente ...
Dropping the database is potentially a very bad thing to do.
Any data stored in the database will be destroyed.

Do you really want to drop the 'saweso' database [y/N] y
mysqladmin: DROP DATABASE saweso failed;
error: 'Can't drop database 'saweso'; database doesn't exist'
Creando nueva BD saweso ...
Cargando estructura de tablas ...
Cargando datos iniciales en las tablas ...
root@SqueezeBase:~#
```

- Desde nuestro PC de escritorio, copiamos y descomprimimos el archivador del instalador (saweso_1.0.1.tar.gz) para poder acceder al fichero “saweso.war” ubicado en la ruta “/saweso_templates/WAR/”

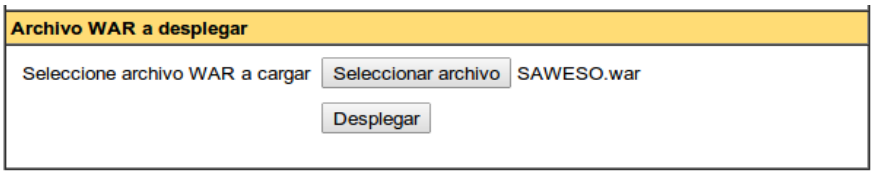


- Desde un navegador web en nuestro PC de escritorio, accedemos a la interfaz de administración del servidor Apache Tomcat, introduciendo en la barra de direcciones la URL http://IP_SERVIDOR:8080/manager/html. El servidor Tomcat nos solicitará el usuario (“admin”) y contraseña de acceso (“admin”).



- Tras acceder a la interfaz de administración de Tomcat, nos vamos a la sección “Desplegar” → “Archivo WAR a desplegar”, seleccionamos el fichero “saweso.war”,

ubicado en la ruta “/saweso_templates/WAR/”, y pulsamos el botón “Desplegar”.



- Tras finalizar este proceso, ya podremos acceder a la aplicación, desde cualquier navegador web, utilizando la URL http://IP_SERVIDOR:8080/SAWESO/

9.3. Requisitos post-instalación

Tal y como hemos descrito en apartados anteriores (6. Definición de funcionalidades), el ámbito de aplicación de SAWESO requiere el acceso remoto mediante ssh y certificados de seguridad (dsa / rsa) a una infraestructura de servidores y/o equipos clientes. Esta infraestructura forma el conjunto de host a los que accederá la aplicación para ejecutar las plantillas de ejecución, que componen el objetivo y funcionalidad principal de la aplicación.

Cada empresa u organización utiliza sus propios certificados de seguridad dsa / rsa para el acceso ssh a los host que administra remotamente, por tanto, no se ha ligado ningún certificado de seguridad específico al funcionamiento de la aplicación para, en su lugar, poder utilizar los certificados existentes.

Supongamos que nuestra organización utiliza un certificado DSA para la conexión con los servidores de su plataforma y un certificado RSA para la conexión con los equipos clientes que “cuelgan” de cada uno de estos servidores.

Para incorporar a la aplicación SAWESO los certificados de seguridad pre-existentes en nuestra empresa u organización debemos seguir los siguientes pasos.

9.3.1. Certificado de seguridad DSA para conexión a servidores

- Copiar nuestro certificado DSA (fichero “id_dsa”) en el directorio /etc/saweso / del servidor donde hemos instalado la aplicación.
- Desde una terminal del servidor SAWESO, borrar la solicitud de nuestra clave (<CLAVE>) del certificado DSA. Con esto evitamos que solicite la clave en cada conexión y, por tanto, impida la ejecución desatendida de la aplicación:

```
root@SqueezeBase:~# ssh-keygen -p -P "<CLAVE>" -N "" -f /etc/saweso/id_dsa
Key has comment '/etc/saweso/id_dsa'
Your identification has been saved with the new passphrase.
```

- Desde una terminal del servidor SAWESO, establecer los permisos necesarios del del certificado DSA para su correcta utilización por el servidor de aplicaciones Tomcat:

```
root@SqueezeBase:~# chown tomcat6:tomcat6 /etc/saweso/id_dsa
root@SqueezeBase:~# chmod 600 /etc/saweso/id_dsa
```

9.3.2. Certificado de seguridad RSA para conexión a equipos clientes

- Copiar nuestro certificado RSA (fichero “id_rsa”) en el directorio /etc/saweso / del servidor donde hemos instalado la aplicación.
- Desde una terminal del servidor SAWESO, borrar la solicitud de nuestra clave (<CLAVE>) del certificado RSA. Con esto evitamos que solicite la clave en cada conexión y, por tanto, impida la ejecución desatendida de la aplicación:

```
root@SqueezeBase:~# ssh-keygen -p -P "<CLAVE>" -N "" -f /etc/saweso/id_rsa
Key has comment '/etc/saweso/id_rsa'
Your identification has been saved with the new passphrase.
```

- Desde una terminal del servidor SAWESO, establecer los permisos necesarios del

del certificado RSA para su correcta utilización por el servidor de aplicaciones Tomcat:

```
root@SqueezeBase:~# chown tomcat6:tomcat6 /etc/saweso/id_rsa
root@SqueezeBase:~# chmod 600 /etc/saweso/id_rsa
```

9.3.3. Fichero de configuración general de la aplicación

Para finalizar la instalación de la aplicación debemos modificar los parámetros que identifican los ficheros de los certificados de seguridad DSA y RSA en el fichero de configuración general de la aplicación SAWESO (**/etc/saweso/saweso.conf**).

```
root@SqueezeBase:~# cat /etc/saweso/saweso.conf
#Datos de acceso a la BD
schemaBD = saweso
usernameBD = root
passwordBD = holamundo

#Certificados
ssh_setup_SERVIDOR = id_dsa
ssh_setup_CLIENTE = id_rsa

#Directorios de trabajo
#tmp = /tmp
#scrips = /etc/saweso
#logs = /var/log/saweso
```

Para modificar los valores (id_dsa, id_rsa) deberemos utilizar un editor en texto plano y respetar los espacios en blanco existentes en el fichero.

10. Análisis de Testing

Para organizar el testing se ha estructurado en las siguientes secciones:

- Diseño del testing, aclaraciones relevantes sobre el entorno, la estructura y los contenidos del análisis de testing realizado.
- Testing de integración, para cada subsistema se realizan pruebas detalladas de las principales funcionalidades, así como pruebas detalladas cuyo ciclo involucre a más de un subsistema, y se cumplimentan las respectivas fichas de testing.
- Testing unitario, se programa un test unitario de las operaciones remotas.
- Informe final, valoración general y conclusiones obtenidas tras el análisis de testing.

10.1. Diseño del testing

En los siguientes subpuntos se indican algunas aclaraciones relevantes sobre el entorno, la estructura y contenidos del análisis de testing realizado.

10.1.1. Entorno de prueba

Las características del entorno de prueba utilizado son:

- Sistema Operativo Ubuntu 12.04.1 LTS (Precise Pangolin).
- Sistema SAWESO instalado según las especificacionel del punto 9 (Manual de instalación) de este documento.
- Versión de Java 1.6 (openjdk-6-jre)
- Fichero de configuración (/etc/saweso/saweso.conf) editado con el usuario y contraseña de MySQL y el nombre de la base de datos.

```
#Datos de acceso a la BD
schemaBD = saweso
usernameBD = root
passwordBD = holamundo
```

- Para realizar el testing de integración utilizaremos el superusuario por defecto (user "saweso" y password "saweso").

10.1.2. Testing de integración

Se han definido las pruebas en base a cada subsistema principal indicando en cada ficha de testing, si existen, el resto de subsistemas involucrados (campo "**Otros subsistemas implicados**").

El campo "**Referencia"** de la ficha de testing indica la ruta del menú o funcionalidad principal que se está testeando.

La **valoración** realizada al final de cada ficha de testing se realiza de forma numérica sobre una calificación total de 10 puntos y con redondeo de 2 decimales.

10.1.3. Testing unitario

Para realizar el test unitario se ha creado una clase .java, dentro del **package servidor**, con la siguiente nomenclatura: **Testing.java**

En esta clase se ha añadido un método main dentro del cual se programan las diferentes pruebas sobre los métodos de la clase **servidor.GestorBD.java.**

10.2. Testing de integración

Subsistema principal:	Autenticación y mantenimiento de la conexión
Otros subsistemas implicados:	Gestión de permisos
Referencia:	**Login**
Propósito:	Autenticación del usuario

Código	Acciones a Verificar	Resultado Esperado	Verificación
01	Acceder al sistema con un usuario con perfil “Operador”.	Aparecen habilitados los menús correspondientes al tipo de usuario: -Vistazo -Ejecucion OP -Ayuda -Cerrar sesión	
02	Acceder al sistema con un usuario con perfil “Programador”.	Aparecen habilitados los menús correspondientes al tipo de usuario: -Vistazo -Ejecucion PRO -Ayuda -Cerrar sesión	
03	Acceder al sistema con un usuario con perfil “Gestor de grupo”.	Aparecen habilitados los menús correspondientes al tipo de usuario: -Vistazo -Plantillas -Usuarios y grupos -Ayuda -Cerrar sesión	
04	Acceder al sistema con un usuario con perfil “Administrador”.	Aparecen habilitados los menús correspondientes al tipo de usuario: -Vistazo -Configuracion -Logs -Ayuda -Cerrar sesión	
05	Acceder al sistema con un usuario con perfl mixto y que pertenece a los siguientes grupos: “Operador” ”Programador”	Aparecen habilitados los menús acumulativos correspondientes a cada uno de los diferentes tipos de usuarios a los que pertenece.	
06	Acceder al sistema con un usuario con perfl mixto y que pertenece a los siguientes grupos: “Operador” ”Gestor de grupo”	Aparecen habilitados los menús acumulativos correspondientes a cada uno de los diferentes tipos de usuarios a los que pertenece.	
07	Acceder al sistema con un usuario con perfl mixto y que pertenece a los siguientes grupos: “Operador” ”Administrador”	Aparecen habilitados los menús acumulativos correspondientes a cada uno de los diferentes tipos de usuarios a los que pertenece.	
08	Acceder al sistema con un usuario con perfl mixto y que pertenece a los siguientes grupos: “Programador” ”Gestor de grupo”	Aparecen habilitados los menús acumulativos correspondientes a cada uno de los diferentes tipos de usuarios a los que pertenece.	

09	Acceder al sistema con un usuario con perfl mixto y que pertenece a los siguientes grupos: “Programador” ”Administrador”	Aparecen habilitados los menús acumulativos correspondientes a cada uno de los diferentes tipos de usuarios a los que pertenece.	
10	Acceder al sistema con un usuario con perfl mixto y que pertenece a los siguientes grupos: ”Gestor de grupo” “Administrador”	Aparecen habilitados los menús acumulativos correspondientes a cada uno de los diferentes tipos de usuarios a los que pertenece.	
11	Acceder al sistema con diferentes combinaciones de nombres de usuarios y contraseñas incorrectas: mayúsculas, minúsculas, espacios en blanco, ...	Mensaje de “Login incorrecto”.	
12	Acceso a los diferentes menús que aparecen habilitados para cada tipo de usuario.	Al pulsar sobre el menú la vista web cambia mostrándo el contenido del subsistema o funcionalidad seleccionada.	
13	Cerrar la sesión inciada.	Al pulsar sobre el submenú “Cerrar sesión”, el usuario se desloguea perdiendo el acceso a la aplicación y se vuelve a mostrar la pantalla de Login inicial.	
14	Recuperar la actividad tras un largo periodo de tiempo sin actuar sobre la aplicación.	La aplicación debe desloguear al usuario y mostrar la pantalla de Login inicial para que vuelva a iniciar una nueva sesión.	 1

Observaciones
(1) Se produce una excepción mostrando la siguiente causa raíz:

Pruebas Correctas	Errores	Valoración
13	1	9,28

Subsistema principal:	Visualización de ejecuciones
Otros subsistemas implicados:	
Referencia:	**Vistazo**
Propósito:	Visualización de todas las ejecuciones existentes en el sistema

Código	Acciones a Verificar	Resultado Esperado	Verificación
01	Visualización de las ejecuciones para un usuario con perfil “Operador” o “Programador”.	Se muestra una tabla que contiene únicamente las ejecuciones pertenecientes al usuario logueado.	
02	Visualización de las ejecuciones para un usuario con perfil “Gestor de grupo” o “Administrador”.	Se muestra una tabla que contiene todas las ejecuciones existentes en el sistema independientemente del usuario a que pertenezcan.	
03	Mostrar todos los atributos que determinan el estado de una ejecución.	Para cada una de las ejecuciones listadas se puede consultar la información de los campos: -Estado -Ejecución	

		-Fecha -Informe -Estadística -Plantilla -Usuario	
04	Acceder al informe de la ejecución	Al pulsar sobre el enlace existente en el campo “Informe” de la fila que contiene la información de la ejecución nos redirige a otra página donde se muestra el informe completo de la ejecución.	
05	Acceder a las estadísticas de la ejecución	Al pulsar sobre el enlace existente en el campo “Estadística” de la fila que contiene la información de la ejecución nos redirige a otra página donde se muestran las estadísticas generales de la ejecución y el estado detallado de cada host individualmente.	
06	Acceder a la plantilla de la ejecución	Al pulsar sobre el enlace existente en el campo “Plantilla” de la fila que contiene la información de la ejecución nos redirige a otra página donde se muestra la información (nombre, categoría, autor, descripción, ...) y contenido de la plantilla de ejecución utilizada.	

Observaciones

Pruebas Correctas	Errores	Valoración
6	0	10

Subsistema principal:	Ejecución avanzada
Otros subsistemas implicados:	Gestión de ejecucón de tareas
Referencia:	**Ejecución PRO**
Propósito:	Subir nuevas plantillas de ejecución o lanzar nuevas ejecuciones

Código	Acciones a Verificar	Resultado Esperado	Verificación
01	Subir nueva plantilla – Sin introducir ninguna descripción.	Tras pulsar el botón “Subir”, se muestra el mensaje de error “No se ha especificado la descripción”	
02	Subir nueva plantilla – Sin seleccionar ningún fichero.	Tras pulsar el botón “Subir”, se muestra el mensaje de error “No se ha especificado un fichero plantilla válido”	
03	Subir nueva plantilla – Seleccionando un fichero con extensión diferente a .sh o .tar.gz	Tras pulsar el botón “Subir”, se muestra el mensaje de error “No se ha especificado un fichero plantilla válido”	
04	Subir nueva plantilla – Escribiendo manualmente en el campo descripción hasta el límite que nos permite el campo de texto.	Se limita el contenido del texto a 150 caracteres.	

05	Subir nueva plantilla – Pegando desde el portapapeles en el campo descripción un texto mayor de 150 caracteres.	Se limita el contenido del texto a 150 caracteres.	✗ 1
06	Subir nueva plantilla – Seleccionando un fichero válido (extensión .sh o .tar.gz) y que no se haya dado de alta previamente.	Tras pulsar el botón “Subir”, se registra correctamente la nueva plantilla de ejecución.	✓
07	Subir nueva plantilla – Seleccionando un fichero válido (extensión .sh o .tar.gz) que ya haya sido dado de alta previamente.	Tras pulsar el botón “Subir”, se muestra el mensaje de error “La plantilla ya existe en la BD, debe poner otro nombre al fichero”	✓
08	Subir nueva plantilla – Seleccionando el tipo de host “Servidor” con cualquiera de las categorías existentes (Búsqueda, Comando, Script, Workaround, Informe o Recolección).	Tras pulsar el botón “Subir”, se registra correctamente la nueva plantilla de ejecución.	✓
09	Subir nueva plantilla – Seleccionando el tipo de host “Equipo cliente” con cualquiera de las categorías existentes (Búsqueda, Comando, Script, Workaround, Informe o Recolección).	Tras pulsar el botón “Subir”, se registra correctamente la nueva plantilla de ejecución.	✓
10	Subir nueva plantilla – Especificando que puede ser ejecutada por usuarios con perfil de operador.	Tras pulsar el botón “Subir”, se registra correctamente la nueva plantilla de ejecución.	✓
11	Subir nueva plantilla – Especificando que no puede ser ejecutada por usuarios con perfil de operador.	Tras pulsar el botón “Subir”, se registra correctamente la nueva plantilla de ejecución.	✓
12	Ejecutar plantilla – Mostrando las plantillas que se pueden ejecutar para el tipo de host “Servidores” con cualquiera de los tipos de ejecución, o categorías, existentes (Búsqueda, Comando, Script, Workaround, Informe o Recolección).	Tras pulsar el botón “Mostrar Plantillas”, se listan todas las plantillas existentes para el tipo y categoría seleccionado.	✓
13	Ejecutar plantilla – Mostrando las plantillas que se pueden ejecutar para el tipo de host “Equipos clientes” con cualquiera de los tipos de ejecución, o categorías, existentes (Búsqueda, Comando, Script, Workaround, Informe o Recolección).	Tras pulsar el botón “Mostrar Plantillas”, se listan todas las plantillas existentes para el tipo y categoría seleccionado.	✓
14	Ejecutar plantilla – Sin seleccionar el nombre de la plantilla	Tras pulsar el botón “Ejecutar Plantilla”, se muestra el mensaje de error “No se ha especificado la plantilla que se desea ejecutar, vuelva a intentarlo”	✓
15	Ejecutar plantilla – Sin seleccionar el fichero con las direcciones ips	Tras pulsar el botón “Ejecutar Plantilla”, se muestra el mensaje de error “No se ha especificado el fichero de ips, vuelva a intentarlo”	✓
16	Ejecutar plantilla – Seleccionando un fichero que contenga direcciones ips erróneas o mal formadas. *Por ejemplo: dsadasd y 15.125.14.78*	Tras pulsar el botón “Ejecutar Plantilla”, se muestra el mensaje de error “Existen las siguientes ips incorrectas en el fichero de ips: [dsadasd, 15.125.14.78]”	✓
17	Ejecutar plantilla – Seleccionando la plantilla y un fichero que contenga ips válidas.	Tras pulsar el botón “Ejecutar Plantilla”, se muestra un mensaje de confirmación indicando el número de ejecuciño asignado “Ejecucion lanzada con codigo: <NUM>”	✓

Observaciones
(1) El texto se limita a 150 caracteres y el sistema devuelve un mensaje de confirmación indicando que la plantilla se ha

subido correctamente pero se puede observar en la consola el siguiente mensaje de error: *com.mysql.jdbc.MysqlDataTruncation: Data truncation: Data too long for column 'descripcion' at row 1* Y podemos comprobamos que la plantilla no se ha dado de alta correctamente en el sistema.

Pruebas Correctas	Errores	Valoración
16	1	9,41

Subsistema principal:	Ejecución básica
Otros subsistemas implicados:	Gestión de ejecucón de tareas
Referencia:	**Ejecución OP**
Propósito:	Lanzar nuevas ejecuciones de plantillas con visibilidad para usuarios con perfil de Operador

Código	Acciones a Verificar	Resultado Esperado	Verificación
01	Ejecutar plantilla – Mostrando las plantillas que se pueden ejecutar para el tipo de host “Servidores” con cualquiera de los tipos de ejecución, o categorías, existentes (Búsqueda, Comando, Script, Workaround, Informe o Recolección).	Tras pulsar el botón “Mostrar Plantillas”, se listan todas las plantillas existentes, para el tipo y categoría seleccionado, con visibilidad para usuarios con perfil de Operador.	
02	Ejecutar plantilla – Mostrando las plantillas que se pueden ejecutar para el tipo de host “Equipos clientes” con cualquiera de los tipos de ejecución, o categorías, existentes (Búsqueda, Comando, Script, Workaround, Informe o Recolección).	Tras pulsar el botón “Mostrar Plantillas”, se listan todas las plantillas existentes, para el tipo y categoría seleccionado, con visibilidad para usuarios con perfil de Operador.	
03	Ejecutar plantilla – Sin seleccionar el nombre de la plantilla	Tras pulsar el botón “Ejecutar Plantilla”, se muestra el mensaje de error “No se ha especificado la plantilla que se desea ejecutar, vuelva a intentarlo”	
04	Ejecutar plantilla – Sin seleccionar el fichero con las direcciones ips	Tras pulsar el botón “Ejecutar Plantilla”, se muestra el mensaje de error “No se ha especificado el fichero de ips, vuelva a intentarlo”	
05	Ejecutar plantilla – Seleccionando un fichero que contenga direcciones ips erróneas o mal formadas. *Por ejemplo: dsadasd y 15.125.14.78*	Tras pulsar el botón “Ejecutar Plantilla”, se muestra el mensaje de error “Existen las siguientes ips incorrectas en el fichero de ips: [dsadasd, 15.125.14.78]”	
06	Ejecutar plantilla – Seleccionando la plantilla y un fichero que contenga ips válidas.	Tras pulsar el botón “Ejecutar Plantilla”, se muestra un mensaje de confirmación indicando el número de ejecuciń asignado “Ejecucion lanzada con codigo: <NUM>”	

Observaciones

Pruebas Correctas	Errores	Valoración
6	0	10

10.3. Testing unitario

Clase de implementación:	**servidor.GestorBD.java**
Clase de testing:	**Testing.java**

Código	Método a Verificar	Resultado Esperado	Verificación
01	checkLogin	Comprueba si los datos de login son corretos verificando si el usuario y la contraseña existen en la base de datos.	✓
02	getGrupos	Devuelve un array String[ ] con todos los grupos a los que pertenece el usuario.	✓
03	altaPlantilla	Realiza el alta de una nueva plantilla en la base de datos.	✓ 1
04	existePlantilla	Comprueba si existe el nombre de la plantilla en la base de datos.	✓
05	ObtenerPlantillas	Cargar un array con todas las plantillas existentes del tipo y subtipo especificados.	✓
06	getPlantilla	Devuelve un objeto de la clase Plantilla.	✓
07	altaHostEjecucion	Registra el host en la lista de host que formarán parte de la ejecucion indicada. Utiliza el método “existeHost ” para comprobar si el host está dado de alta en la base de datos general de host. Utiliza el método “existeHostEjecucion” para comprobar si el host ya esta incluido en la ejecucion en la que queremos incluirlo y evitar duplicados.	✓
08	ObtenerEjecuciones	Cargar un array con todas las ejecuciones existentes para el usuario especificado.	✓
09	getEjecucionHost	Obtener los datos de la ejecucion con los estados detallados por cada host individual.	✓

Observaciones
(1) Falla si no se especifica un nombre de grupo existente en la aplicación (operador, programador, gestor, administrador)

Pruebas Correctas	Errores	Valoración
9	0	10

Método main()
public static void main(String[] args) **throws** EAplicacion {

```
		try{
		//############ TESTING GESTORBD.JAVA ############
		GestorBD gestorBD = new GestorBD();

		//Método checkLogin
		System.out.println("\nMétodo checkLogin");
		System.out.println("Login correcto (saweso, saweso): " + gestorBD.checkLogin("saweso", "saweso"));
		System.out.println("Login incorrecto (Saweso, saweso): " + gestorBD.checkLogin("Saweso", "saweso"));
		System.out.println("Login incorrecto (saweso, saWeso): " + gestorBD.checkLogin("saweso", "saWeso"));
		System.out.println("Login incorrecto (saweso, ): " + gestorBD.checkLogin("saweso", ""));
		System.out.println("Login en blanco ( , ): " + gestorBD.checkLogin("", ""));

		//Método getGrupos
		System.out.println("\nMétodo getGrupos");
		String grupos[];
		String grupo;
		grupos = gestorBD.getGrupos("o");
		for(int i = 0; i < grupos.length; i++){
			grupo=grupos[i];
			if (grupo != "-") System.out.println("Usuario con nombre 'o' pertenece al grupo: " + grupo);
		}
		grupos = gestorBD.getGrupos("p");
		for(int i = 0; i < grupos.length; i++){
			grupo=grupos[i];
			if (grupo != "-") System.out.println("Usuario con nombre 'p' pertenece al grupo: " + grupo);
		}
		grupos = gestorBD.getGrupos("g");
		for(int i = 0; i < grupos.length; i++){
			grupo=grupos[i];
			if (grupo != "-") System.out.println("Usuario con nombre 'g' pertenece al grupo: " + grupo);
		}
		grupos = gestorBD.getGrupos("a");
		for(int i = 0; i < grupos.length; i++){
			grupo=grupos[i];
			if (grupo != "-") System.out.println("Usuario con nombre 'a' pertenece al grupo: " + grupo);
		}
		grupos = gestorBD.getGrupos("indi");
		for(int i = 0; i < grupos.length; i++){
			grupo=grupos[i];
			if (grupo != "-") System.out.println("Usuario con nombre 'indi' pertenece al grupo: " + grupo);
		}
		grupos = gestorBD.getGrupos("saweso");
		for(int i = 0; i < grupos.length; i++){
			grupo=grupos[i];
			if (grupo != "-") System.out.println("Usuario con nombre 'saweso' pertenece al grupo: " + grupo);
		}
		grupos = gestorBD.getGrupos("noexiste");
		for(int i = 0; i < grupos.length; i++){
			grupo=grupos[i];
			System.out.println("Usuario inexistente con nombre 'noexiste' pertenece al grupo: " + grupo);
		}

		//Método altaPlantilla
		System.out.println("\nMétodo altaPlantilla");
		System.out.println("Alta de nueva plantilla (nombre): " + gestorBD.altaPlantilla("nombre", "tipo", "subtipo", "autor", "descripcion", "operador"));
		System.out.println("Alta de nueva plantilla (plantilla1): " + gestorBD.altaPlantilla("plantilla1", "tipo", "subtipo", "autor", "descripcion", "operador"));
		System.out.println("Alta de plantilla cuyo nombre ya existe (plantilla1): " + gestorBD.altaPlantilla("plantilla1", "tipo", "subtipo", "autor", "descripcion",
"operador"));
		System.out.println("Alta de plantilla cuyo nombre ya existe (plantilla1): " + gestorBD.altaPlantilla("plantilla1", "tipo", "subtipo", "autor", "descripcion",
"programador"));
		System.out.println("Alta de nueva plantilla (plantilla2): " + gestorBD.altaPlantilla("plantilla2", "tipo", "subtipo", "autor", "descripcion", "programador"));
		//System.out.println("Alta de nueva plantilla con grupo no existente (grupono): " + gestorBD.altaPlantilla("plantilla3", "tipo", "subtipo", "autor",
"descripcion", "grupono"));

		//Método existePlantilla
		System.out.println("\nMétodo existePlantilla");
		System.out.println("¿Existe plantilla 'nombre'?: " + gestorBD.existePlantilla("nombre"));
		System.out.println("¿Existe plantilla 'plantilla1'?: " + gestorBD.existePlantilla("plantilla1"));
		System.out.println("¿Existe plantilla 'plantilla2'?: " + gestorBD.existePlantilla("plantilla2"));
		System.out.println("¿Existe plantilla 'plantilla3'?: " + gestorBD.existePlantilla("plantilla3"));
		System.out.println("¿Existe plantilla 'plantilla4'?: " + gestorBD.existePlantilla("plantilla4"));
		System.out.println("¿Existe plantilla ' '?: " + gestorBD.existePlantilla(" "));

		//Método ObtenerPlantillas
		System.out.println("\nMétodo ObtenerPlantillas");
		ArrayList<Plantilla> plantillas;
		Plantilla plantillaux;
		Iterator i=null;
		plantillas=gestorBD.ObtenerPlantillas("servidor", "busqueda");
		i = plantillas.iterator();
		while(i.hasNext()){
			plantillaux=(Plantilla)i.next();
			System.out.println(plantillaux.getNombre());
		}
		plantillas=gestorBD.ObtenerPlantillas("equipo", "recoleccion");
```

```
		i = plantillas.iterator();
		while(i.hasNext()){
			plantillaux=(Plantilla)i.next();
			System.out.println(plantillaux.getNombre());
		}

		//Método getPlantilla
		System.out.println("\nMétodo getPlantilla");
		System.out.println(gestorBD.getPlantilla("sawe.sh"));
		System.out.println(gestorBD.getPlantilla("plantilla4"));

		//Método altaHostEjecucion
		System.out.println("\nMétodo altaHostEjecucion");
		System.out.println(gestorBD.altaHostEjecucion("192.168.1.100", 1));
		System.out.println(gestorBD.altaHostEjecucion("192.168.2.250", 1));
		System.out.println(gestorBD.altaHostEjecucion("192.168.2.250", 1));

		//Método ObtenerEjecuciones
		System.out.println("\nMétodo ObtenerEjecuciones");
		ArrayList<Ejecucion> ejecuciones;
		Ejecucion ejecucionaux;
		Iterator j=null;
		ejecuciones=gestorBD.ObtenerEjecuciones("p");
		i = ejecuciones.iterator();
		while(i.hasNext()){
			ejecucionaux=(Ejecucion)i.next();
			System.out.println("CODIGO: " + ejecucionaux.getCodigo() + " ESTADO: "+ ejecucionaux.getEstado());
		}

		//Método getEjecucionHost
		System.out.println("\nMétodo getEjecucionHost");
		ArrayList matrix = gestorBD.getEjecucionHost("1");
		for(int a = 0; a < matrix.size();a++){
			//fila i, columna 1 y columna 2
			System.out.println("HOST: " + ((ArrayList)matrix.get(a)).get(0) + " ESTADO: " + ((ArrayList)matrix.get(a)).get(1));
		}

		}catch(Exception e){
				throw new EAplicacion(e.getMessage());
		}
}
```

INPUTS / OUTPUTS

Método checkLogin

INPUTS:
gestorBD.checkLogin("saweso", "saweso")
gestorBD.checkLogin("Saweso", "saweso")
gestorBD.checkLogin("saweso", "saWeso")
gestorBD.checkLogin("saweso", "")
gestorBD.checkLogin("", "")

OUTPUTS:
Login correcto (saweso, saweso): true
Login incorrecto (Saweso, saweso): false
Login incorrecto (saweso, saWeso): false
Login incorrecto (saweso,): false
Login en blanco (,): false

Método getGrupos

INPUTS:
grupos = gestorBD.getGrupos("o");
grupos = gestorBD.getGrupos("p");
grupos = gestorBD.getGrupos("g");
grupos = gestorBD.getGrupos("a");
grupos = gestorBD.getGrupos("indi");
grupos = gestorBD.getGrupos("saweso");
grupos = gestorBD.getGrupos("noexiste")

OUTPUTS:
Usuario con nombre 'o' pertenece al grupo: operador
Usuario con nombre 'p' pertenece al grupo: programador
Usuario con nombre 'g' pertenece al grupo: gestor

Usuario con nombre 'a' pertenece al grupo: administrador
Usuario con nombre 'indi' pertenece al grupo: administrador
Usuario con nombre 'indi' pertenece al grupo: gestor
Usuario con nombre 'indi' pertenece al grupo: operador
Usuario con nombre 'indi' pertenece al grupo: programador
Usuario con nombre 'saweso' pertenece al grupo: administrador
Usuario con nombre 'saweso' pertenece al grupo: gestor
Usuario con nombre 'saweso' pertenece al grupo: operador
Usuario con nombre 'saweso' pertenece al grupo: programador
Usuario inexistente con nombre 'noexiste' pertenece al grupo: -
Usuario inexistente con nombre 'noexiste' pertenece al grupo: -
Usuario inexistente con nombre 'noexiste' pertenece al grupo: -
Usuario inexistente con nombre 'noexiste' pertenece al grupo: -

Método altaPlantilla

INPUTS:
gestorBD.altaPlantilla("nombre", "tipo", "subtipo", "autor", "descripcion", "operador")
gestorBD.altaPlantilla("plantilla1", "tipo", "subtipo", "autor", "descripcion", "operador")
gestorBD.altaPlantilla("plantilla1", "tipo", "subtipo", "autor", "descripcion", "operador")
gestorBD.altaPlantilla("plantilla1", "tipo", "subtipo", "autor", "descripcion", "programador")
gestorBD.altaPlantilla("plantilla2", "tipo", "subtipo", "autor", "descripcion", "programador")
System.out.println("Alta de nueva plantilla con grupo no existente (grupono): " + gestorBD.altaPlantilla("plantilla3", "tipo", "subtipo", "autor", "descripcion", "grupono"));

OUTPUTS:
Alta de nueva plantilla (nombre): true
Alta de nueva plantilla (plantilla1): true
Alta de plantilla cuyo nombre ya existe (plantilla1): false
Alta de plantilla cuyo nombre ya existe (plantilla1): false
Alta de nueva plantilla (plantilla2): true
com.mysql.jdbc.exceptions.jdbc4.MySQLIntegrityConstraintViolationException: Cannot add or update a child row: a foreign key constraint fails (`saweso`.`plantilla`, CONSTRAINT `plantilla_ibfk_1` FOREIGN KEY (`grupo`) REFERENCES `grupo` (`nombre`))

Método existePlantilla

INPUTS:
gestorBD.existePlantilla("nombre")
gestorBD.existePlantilla("plantilla1")
gestorBD.existePlantilla("plantilla2")
gestorBD.existePlantilla("plantilla3")
gestorBD.existePlantilla("plantilla4")
gestorBD.existePlantilla(" ")

OUTPUTS:
¿Existe plantilla 'nombre'?: true
¿Existe plantilla 'plantilla1'?: true
¿Existe plantilla 'plantilla2'?: true
¿Existe plantilla 'plantilla3'?: false
¿Existe plantilla 'plantilla4'?: false
¿Existe plantilla ' '?: false

Método ObtenerPlantillas

INPUTS:
gestorBD.ObtenerPlantillas("servidor", "busqueda")
gestorBD.ObtenerPlantillas("equipo", "recoleccion")

OUTPUTS:
Cargar un array con todas las plantillas existentes del tipo: servidor y subtipo: busqueda
plantilla1.sh
plantilla3.sh
sawe.sh

Cargar un array con todas las plantillas existentes del tipo: equipo y subtipo: recoleccion
plantilla2.sh

Método getPlantilla

INPUTS:
//Plantilla existente en el sistema
gestorBD.getPlantilla("sawe.sh")

//Plantilla no existente en el sistema
gestorBD.getPlantilla("plantilla4")

OUTPUTS:
INFORMACION DE LA PLANTILLA:
Nombre: sawe.sh
Tipo de host: servidor
Categoria: busqueda
Autor: p
Fecha actualización: 2012-12-20
Descripción: ewaeaeawe

null

Método altaHostEjecucion

INPUTS:
gestorBD.altaHostEjecucion("192.168.1.100", 1)
gestorBD.altaHostEjecucion("192.168.2.250", 1)
gestorBD.altaHostEjecucion("192.168.2.250", 1)

OUTPUTS:
Consultar si existe un host
Consultar si existe un host en la ejecucion especificada
Registro de un nuevo host en ejecucionhost
true

Consultar si existe un host
Consultar si existe un host en la ejecucion especificada
Registro de un nuevo host en ejecucionhost
true

Consultar si existe un host
Consultar si existe un host en la ejecucion especificada
true

Método ObtenerEjecuciones

INPUTS:
gestorBD.ObtenerEjecuciones("p")

OUTPUTS:
Cargar un array con todas las ejecuciones existentes para el usuario: p
CODIGO: 3 ESTADO: EJECUTANDO
CODIGO: 2 ESTADO: EJECUTANDO
CODIGO: 1 ESTADO: EJECUTANDO

Método getEjecucionHost

INPUTS:
```
ArrayList matrix = gestorBD.getEjecucionHost("1");
for(int a = 0; a < matrix.size();a++){
        System.out.println("HOST: " + ((ArrayList)matrix.get(a)).get(0) + " ESTADO: " + ((ArrayList)matrix.get(a)).get(1));
}
```

OUTPUTS:

```
Consultar los datos del la ejecucion: 1
¡Array de ejecuciones y host cargado y devuelto!
HOST: 192.168.1.100 ESTADO: INICIO
HOST: 192.168.2.250 ESTADO: INICIO
HOST: 192.168.1.1 ESTADO: EJECUTADO
HOST: 192.168.1.2 ESTADO: EJECUTADO
HOST: 192.168.1.7 ESTADO: ASIGNADO 5
HOST: 192.168.1.6 ESTADO: ASIGNADO 4
HOST: 192.168.1.5 ESTADO: ASIGNADO 3
HOST: 192.168.1.4 ESTADO: ASIGNADO 2
HOST: 192.168.1.3 ESTADO: ASIGNADO 1
```

10.4. Informe final

Antes de exponer las conclusiones del análisis de testing, resumiremos los resultados obtenidos en las fases de testing de integración y testing unitario.

Resultados del testing de integración

Como podemos apreciar, la evaluación global del testing de integración es de:

(9,28 + 10 + 9,41 + 10) / 4 = 9,67 puntos sobre 10

Resultados del testing unitario

Como podemos apreciar, la evaluación del testing unitario es de: 10 puntos sobre 10

Para realizar la valoración global del proyecto asignamos una ponderación, basada en porcentajes, a cada una de las etapas del testing. De esta forma, asignamos un peso del 55% al testing de integración y un 45% al testing unitario.

Por tanto, la valoración final del proyecto es de:

(9,67 x 0,55) + (10 x 0,45) = 9,82 puntos sobre 10

De cara a futuras fases de mantenimiento o calidad del proyecto, conviene destacar los errores más relevantes detectados para analizar e implementar las correcciones necesarias:

- Login → revisar el control de la sesión para evitar excepciones tras largos periodos de inactividad del usuario.
- Ejecución PRO → mejorar el control de las entradas en los campos de texto.

11. Análisis de Calidad

Para el análisis de la calidad del software de la aplicación SAWESO hemos tenido en cuenta los siguientes factores de calidad del software:

11.1. Corrección

La corrección representa la capacidad del software implementado para llevar a cabo todas las funcionalidades especificadas para la aplicación. Se trata del principal factor de calidad del software, y se ve favorecido con el uso de las técnicas de depuración y prueba.

En la análisis de testing realizado en el punto 10 de este documento hemos comprobado como el software SAWESO, en líneas generales, presenta una respuesta eficaz a las pruebas, mostrando un alto nivel de corrección.

11.2. Robustez

La robustez se define como la capacidad del software para reaccionar ante situaciones excepcionales. A diferencia de la corrección, que mide la correcta implementación de las especificaciones, la robustez mide la respuesta ante lo que se encuentra fuera de las especificaciones. Para asegurar la robustez de una aplicación es recomendable la inclusión en el mismo de un buen sistema de tratamiento de excepciones.

En la presente aplicación encontramos una baja complejidad que proporciona cierta facilidad a la hora de realizar el testing de la aplicación. Durante la fase de pruebas se han detectado algunas excepciones no controladas por el sistema de gestión de excepciones, por tanto, podemos considerar un valor medio de robustez del software.

11.3. Extensibilidad

La extensibilidad es la facilidad que el software tiene para adaptarse a cambios en la especificación. Para que una aplicación tenga un alto nivel de extensibilidad se precisan cumplir dos principios: simplicidad del diseño y descentralización.

En SAWESO encontramos que se cumplen los dos principios: por un lado contamos con un diseño sencillo en términos generales, salvo en casos aislados de clases con una gran complejidad. Por otro lado podemos encontrar cierta tendencia a la centralización en algunas clases. No obstante, la mayoría de las clases son descentralizadas.

Como conclusión podemos decir que, si bien es recomendable la simplificación de ciertas clases y la descentralización de otras, se trata de partes muy concretas del software, mientras que en la gran mayoría de las clases existe una simplificación y una descentralización suficientes para dotar de una extensibilidad alta a la aplicación .

11.4. Reutilización

Podemos definir la reutilización como la capacidad que tienen los componentes o elementos de un software para ser utilizados en otras aplicaciones. A mayor similitud en la arquitectura de varias aplicaciones mayor será la capacidad de reutilización de los componentes de uno en los otros.

En la presente aplicación la principal fuente de elementos reutilizables se encontrará en el contenido de las clases del package utiles.

11.5. Compatibilidad

La compatibilidad de un software es su facilidad para combinar sus elementos con los de otro software.

Dado el lenguaje de programación en el que ha sido implementada (JAVA) y el gestor

de BBDD utilizado como almacén de datos (MySQL), podemos concluir que se trata de una aplicación con un nivel óptimo de compatibilidad.

11.6. Eficiencia

La eficiencia es la cualidad del software que utiliza pocos recursos (tiempo de procesador, espacio de memoria, etc.) para poder funcionar. A menor cantidad de recursos utilizados mayor eficiencia del software.

No se han realizado pruebas exhaustivas de rendimiento para medir la eficiencia del software SAWESO pero, si podemos afirmar, que tanto en el desarrollo de la aplicación web como en el funcionamiento del motor en bash (Shell Scripting) se han dimensionado las conexiones y uso de los recursos para asegurar su funcionamiento óptimo en entornos con requisitos medios de memoria RAM y CPU.

11.7. Portabilidad

La portabilidad es la capacidad de un software para ser transportado a diferentes entornos de software y hardware (sistemas operativos y computadoras).

Dado que la aplicación ha sido implementada principalmente en el lenguaje de programación JAVA, al ser éste un lenguaje multiplataforma, la aplicación sería perfectamente portable a diferentes tipos de computadoras y sistemas operativos.

11.8. Facilidad de uso

La facilidad de uso es la facilidad con la que personas con formación y aptitudes diferentes pueden aprender a utilizar la aplicación. En este factor de la calidad del software se incluyen el nivel de facilidad de instalación, operación y supervisión del

software.

En lo que se refiere a la funcionalidad de operación del software, el sistema de vistas y menús de la aplicación SAWESO hacen muy fácil el uso de la misma, resultando bastante intuitivo. Esta característica se ve facilitada gracias al control de los menús dispnibles en función de los perfiles de usuario, quedando muy bien delimitadas las funciones que cada tipo de usuario puede llevar a cabo.

Respecto a la funcionalidad de instalación, la aplicación consta de un instalador que automatiza el proceso de instalación, creación de la base de datos y carga inicial de datos en la misma pero, requiere el despliegue manual de la aplicación web en el servidor de aplicaciones Apache Tomcat. Por ello, el usuario instalador deberá tener unos ciertos conocimientos específicos que imposibilitan la instalación a usuarios sin nivel técnico.

Por tanot, si bien la facilidad de uso de la aplicación es alta, la dificultad a la hora de instalar correctamente la misma hace que el nivel de facilidad de usabilidad sea medio.

11.9. Funcionalidad

La funcionalidad se define como el conjunto de operaciones que un determinado software permite realizar.

El software SAWESO está dotado de un conjunto de operaciones muy completo que se adapta prácticamente a la totalidad de las especificaciones marcadas. Hay que señalar que durante la fase de implementación no se ha desarrollado la funcionalidad de login con servidor ldap, motivo por el cual la funcionalidad de la aplicación puede ser considerada media-alta.

12. Bibliografía

A continuación se recogen los enlaces y documentos más importantes que se han utilizado para el diseño y desarrollo de la aplicación.

Titulo web o Libro	URL o Editorial
Debian Administration	http://www.debian-administration.org/article/SSH_dynamic_port_forwarding_with_SOCKS
Crysol	http://crysol.org/es/node/1355
Blog Vicente Navarro	http://www.vicente-navarro.com/blog/2009/05/24/creando-tuneles-tcpip-port-forwarding-con-ssh-los-8-escenarios-posibles-usando-openssh/ http://www.vicente-navarro.com/blog/2009/06/13/reenvio-dinamico-de-puertos-montar-un-servidor-socks-con-ssh/
Aplicaciones web. Un enfoque práctico	Año 2010 ISBN: 978-84-7897-957-8 RA-MA EDITORIAL
JAVA 2. Manual de usuario y tutorial	3ª Edición ISBN: 84-7897-517-9 RA-MA EDITORIAL