UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

Desarrollo y Despliegue de Servicios Web integrados mediante un Servidor de Autenticación único basado en Roles

Enrique Aracil Orduña
Tutor: David Alcázar Llano
Ponente: Javier Aracil Rico

JUNIO 2019

Desarrollo y Despliegue de Servicios Web integrados mediante un Servidor de Autenticación único basado en Roles

AUTOR: Enrique Aracil Orduña
TUTOR: David Alcázar Llano
PONENTE: Javier Aracil Rico

Escuela Politécnica Superior
Universidad Autónoma de Madrid
Junio de 2019

Resumen

Este Trabajo Fin de Grado (TFG en adelante), tiene como objetivo implantar y desarrollar en la nube distintos servicios para uso de UAV Navigation, donde el acceso por parte de los usuarios a estos servicios se unifica en una única plataforma en la que se gestionan todos los usuarios y, según el rol de cada uno, se administra el acceso a estos servicios.

Como se va a distribuir información de manera regular, durante toda la vida del proyecto la seguridad es un tema a tener muy en cuenta, por ello la utilización de protocolos seguros ha sido constante y se ha tenido especial cuidado en los puntos más frágiles del proyecto.

Los servicios que se instalarán en la nube deben soportar y utilizar el servidor de autenticación, tanto los servicios que utilicen programas del sistema, como los *opencode* y los desarrollados por cuenta propia. De este modo se consigue la consistencia, fiabilidad y simplicidad en la gestión de los usuarios administrándolos todos desde un único punto.

Esta memoria recoge la estrategia tomada y el desarrollo a seguir para la cohesión de todos los elementos que el servidor en la nube alberga. Comenzando con una introducción de las tecnologías y herramientas idóneas para el desarrollo del TFG y exponiendo cómo se abordó el proyecto desde el inicio. También se mostrarán y analizarán los datos más relevantes del desarrollo, configuración y testeo de los servicios del servidor.

Este servidor pretende cubrir aquellas necesidades que surgen cuando una empresa tiene gran cantidad de información y necesita almacenarla y gestionarla de forma fácil y segura.

Palabras clave

Nube, Servicio, Rol, Seguridad, Protocolo, Autenticación, *Opencode*, Desarrollo, Gestión, Información.

Abstract

This Bachelor Thesis has the goal of introducing and developing new services in the cloud for the use of UAV Navigation, where the users' access to these services is unified in an only platform in which users are arranged and, according to their role, the access to these services is provided.

Because information is going to be distributed on a regular basis, safety is very important to take into consideration throughout the entire life of the project, that is why the use of safe protocols has been constant and special attention has been paid in the most fragile points of the project.

The services that will be installed in the cloud must support and use the authentication server, services that use system programs like opencode and the developed on each own. This way we get consistency, trustworthiness and simplicity in the management of users, all administered from the same point.

This memory collects the taken strategy and the development to follow for the link of all the elements that the server in the cloud contains. Beginning by an introduction of the fit technologies and tools for the development of the TFG and presenting how the project was approached from the start. The most relevant data for configuration, development and testing of the server services will also be shown and analyzed.

The aim of this server is to cover those necessities that come up when a company has a lot of information that needs to storage and handle in a safe and easy way.

Keywords

Cloud, Server, Role, Security, Protocol, Authentication, Opencode, Development, Management, Information.

Agradecimientos

Quiero dar las gracias en este TFG a mi familia y amigos, por el apoyo incondicional que me han dado siempre.

También agradezco enormemente la confianza que depositó en mí el equipo de UAV Navigation, en especial a David Álcázar y al departamento de IT, que me ha llevado a la realización de este TFG.

INDICE DE CONTENIDOS

INDICE DE FIGURAS

INDICE DE TABLAS

1 Introducción

1.1 Motivación

Desde que el ser humano aprendió a comunicarse entre sí, su evolución intelectual se ha marcado por su necesidad de hacer perdurar lo aprendido en el tiempo, de comunicarse con sus sucesores y enseñarles para que no dediquen su vida entera a ello de nuevo, y así evolucionar más rápido.

Tras miles de años y varias revoluciones tecnológicas relacionadas con la comunicación surgió la comunicación digital, que revolucionó la forma de comunicarse no solo en el tiempo, sino también en el espacio pudiendo de manera casi instantánea comunicarte con alguien en el otro lado del globo como si estuviese a un metro de distancia o menos.

Nuestros días están siendo marcados por la gran cantidad de información que existe y a la que podemos acceder. Mucha cantidad de información que prácticamente cualquier persona con acceso a Internet y en un par de *clicks* puede aprender y utilizar en pro a sus necesidades.

Cuando se habla de tantos miles de billones de bytes de datos que pueden ser accedidos por tantas personas nos debemos preguntar si esos datos son fiables, si la información es lícita, o si Internet es un buen lugar para guardar datos importantes que no se quieran compartir.

Existen muchas compañías que se pueden encargar de almacenar datos digitales, o generarte páginas web de manera automatizada para acceder a ellos, sin embargo, se pierde parte del control de los datos delegando el servicio en terceros, y para asegurar la información debería buscarse lo contrario, tener el control íntegro y servicios a medida, no genéricos.

Debemos preguntarnos cuánto de seguro es tener datos en internet que no quieras compartir con el mundo libremente y si se puede prescindir de compañías que se dediquen a ello, para tener el control íntegro de esos datos.

Cuando llegué a UAV Navigation, compañía a la que me referiré de ahora en adelante como UAVN, me propusieron el traslado de parte de su información, digital y física, a un nuevo servidor en la nube para así poder acceder a ella de forma sencilla y desde cualquier parte. Toda esta información debía estar, al menos, igual de segura que antes, manteniendo en todo momento las propiedades de confidencialidad, integridad, autenticación, accesibilidad y disponibilidad en cada uno de los servicios que se desarrollarían.

1.2 Objetivo

El objetivo principal del proyecto es asegurar una vía de acceso rápida, a medida, duradera y segura a los datos de la empresa UAVN a través de Internet con el desarrollo e implantación de un Cloud Server que albergue varios servicios *opencode* o de propio desarrollo web que sean compatibles con el acceso por parte de los usuarios a través de un servicio LDAP, con el fin de conseguir consistencia, fiabilidad y simplicidad, todo ello administrado desde un único punto, sin más intermediarios la propia compañía.

Además, se debe garantizar la seguridad e integridad de los datos mediante el uso de protocolos de seguridad, así como la garantía de que los datos pueden ser recuperados en caso de problemas externos mediante la programación de *backups* diarios que serán almacenados de forma interna y segura.

Este último punto debe estar presente desde el análisis hasta el despliegue del proyecto, dado que los datos que se van a manejar en el servidor son usados diariamente por empleados de UAVN y son muy sensibles tanto a nivel legal como a nivel histórico de la empresa.

1.3 Organización de la memoria

Esta memoria trata de poner en escena las habilidades desarrolladas y adquiridas con el desarrollo del Trabajo de Fin de Grado, para ello se dispone la información distribuida en los siguientes apartados:

- **Estado del Arte:** Aquí se expondrán las tecnologías usadas o desechadas durante el trabajo, haciendo una descripción directa de sus características y por qué se tuvieron en cuenta para el desarrollo.
- **Análisis y Diseño:** Antes de realizar el desarrollo del trabajo es necesario observar el alcance del mismo y adelantarse a las necesidades o contratiempos futuros en la medida de lo posible. Para ello en este apartado se describen y estudian las especificaciones del proyecto para poder comprender el desarrollo.
- **Desarrollo:** En esta sección se explican los desarrollos de los distintos servicios integrados en el servidor.
- **Pruebas:** Para asegurar la calidad y seguridad del servidor y sus servicios se realizan con el fin de asegurar el correcto comportamiento.
- **Conclusiones y trabajo futuro:** Por último, se recogen las conclusiones tras la realización del proyecto y se exponen posibles evoluciones o integraciones que el servidor, del modo en el que es desarrollado, puede albergar.

2 Estado del arte

En esta sección se van a introducir las distintas herramientas y tecnologías tenidas en cuenta en algún punto del desarrollo del proyecto exponiendo sus características más interesantes.

2.1 Servidor

Un servidor es un equipo informático encargado de suministrar información a una serie de clientes, que pueden ser personas, programas u otros dispositivos conectados a él.

A la hora de desplegar un nuevo servidor es importante analizar sus características físicas, es decir, las especificaciones de la máquina en sí, y el Sistema Operativo, a partir de ahora SO, que corre en esta y que ejecutará el Servidor Web, para así poder escoger la máquina que más se adecúe a las necesidades, presentes y futuras, de los servicios que albergue.

2.1.1 Sistema Operativo

Igual que cualquier equipo informático, un servidor también necesita de un SO para gestionar los programas que se ejecutan en él. Existen varios SO usados en servidores entre los que destacan Windows Server y los SO basados en Linux entre los que se encuentran las distribuciones GNU/Linux cuyo software es de código abierto.

Existen muchas diferencias entre los servidores Windows Server y los GNU/Linux por tanto la elección del SO depende del uso que se le vaya a dar.

	Windows Server	**GNU/Linux**
Tipo de Software	Privativo	De código abierto [*]
Servidor Web	Microsoft IIS	Apache, Nginx
Lenguajes Script	VBScript, ASP.NET	PHP, Python, Ruby, Perl, Bash
Bases de Datos Relacionales	Microsoft SQL Server, Microsoft Access	MySQL, MariaDB, PGSQL

Tabla 2-1 Comparación Windows Server y GNU/Linux

[*] A menudo incorporan aplicaciones o controladores propietarios

2.1.1.1 Windows Server

Figura 2-1 Windows Server

Windows se distingue por tener una estructura muy compleja, punto en el que Microsoft ha centrado su atención desde siempre en ofrecer un SO de manejo sencillo, de ahí que todos los programas tengan una interfaz gráfica de usuario intuitiva. Por lo general, los usuarios tienen el control exclusivo de todos los recursos de hardware, reciben avisos del sistema regularmente y pueden instalar el software de manera independiente.

Esto esconde cierto potencial de errores cuando se modifican los ajustes del sistema o se instalan aplicaciones que suponen un riesgo, lo que produce frecuentes fallos de seguridad. La utilización de las GUI obligatorias requiere la utilización de muchos recursos para procesos simples.

2.1.1.2 GNU/Linux

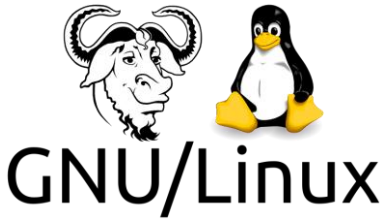

Figura 2-2 Proyecto GNU y Linux

Por su parte, la filosofía de Linux consiste en mantener la estructura del SO lo más sencilla posible y los ajustes del sistema se pueden llevar a cabo en cualquier momento, puesto que al tratarse de código abierto los administradores se benefician de las libertades de la gestión del sistema. Además, los errores de seguridad son poco habituales y se pueden solucionar rápidamente.

Sin embargo, su uso se limita a usuarios expertos, pues, en su mayoría, la configuración y administración del sistema se realiza por líneas de comando, algo que puede desbordar a los usuarios poco experimentados.

Para la implementación del servidor, el SO que mejor se adecúa al uso que va a tener es uno basado en Linux dado que es mucho más robusto en cuanto seguridad y es libre, lo que significa que no se va a necesitar el pago de licencias en el futuro y tiene mucho más margen de acción desde el punto de vista de administración y desarrollo.

GNU/Linux es la combinación de varios proyectos, entre los que destaca el Proyecto GNU, muy partidario del software libre, y el núcleo Linux. La principal característica de los SO

GNU/Linux es que todo su código fuente puede ser utilizado, modificado y redistribuido libremente, bajo ciertos términos de licencias libres.

Existen muchas distribuciones GNU/Linux, también conocidas como *distros*. Para el desarrollo del proyecto se escogió Ubuntu como distribución GNU/Linux por la familiarización con este SO, por la compatibilidad de software que ofrece y por la gran cantidad de documentación existente en la página web oficial. [1]

Figura 2-3 Ubuntu

2.1.2 Cron

Una herramienta muy útil de los sistemas operativos Linux/Unix es el *cron* o *crontab*. Se trata de una herramienta ejecutada en segundo plano por el sistema que permite lanzar procesos de forma regular o agendada. Es útil para lanzamiento de tareas nocturnas que utilizan muchos recursos del sistema, así como ejecución de *scripts* diarios o semanales creados a medida.

2.2 Cloud Server

Un Cloud Server se sustenta sobre la tecnología *cloud computing* que le permite abstraerse totalmente del hardware, donde no sólo la máquina está virtualizada, sino que otros componentes como la red y el almacenamiento también lo están. Está abstracción respecto al hardware significa que un Cloud Server no está localizado en un ordenador físico concreto.

Esto le hace al Cloud Server tolerante a fallos de hardware, es decir, que en caso de que un ordenador físico falle, el servicio seguirá funcionando de forma ininterrumpida o como mucho requerirá reiniciar el servidor en otro nodo de hardware.

Además, un Cloud Server se puede redimensionar de forma inmediata, es decir, se le puede añadir más RAM o más procesadores, se le puede añadir discos o redimensionar los existentes, o incluso aumentar el ancho de banda.

Ello es una gran ventaja frente a un servidor convencional, más si hablamos de una empresa en crecimiento continuo, pues el Cloud Server se puede amoldar a las necesidades del momento y puede ir creciendo si se necesitan más especificaciones en algún momento dado.

2.2.1 Proveedor de Cloud Server

Un Proveedor de Cloud Server, en inglés *Cloud Server Provider*, es una entidad que suministra Cloud Servers además de otros servicios en la nube.

Todos los proveedores de Cloud Server ofrecen herramientas de gestión que permite hacer muchas de las operaciones que se harían en un servidor físico, todo desde la web del proveedor y de manera sencilla e intuitiva.

Existen muchos proveedores de Cloud Servers en internet y la mayoría ofrecen servicios similares. Para el nuevo servidor de UAVN se decidió usar DigitalOcean como proveedor por su simplicidad, por el buen soporte al usuario que posee, por sus manuales de uso e instalación de programas y porque los SO que ofrece para sus servidores son distribuciones GNU/Linux, entre ellos Ubuntu.

Figura 2-4 DigitalOcean

Desde la cuenta de DigitalOcean vía web se puede apagar, reiniciar, redimensionar o destruir el *Droplet*, nombre con el que denominan a la imagen operativa de un Cloud Server, además, se puede acceder a este por medio de una terminal web. Este proveedor ofrece servicios extra como captura de la imagen del *Droplet*, la realización de *backups* o clonar el mismo en otro con una IP distinta.

2.3 Servidor Web

Un servidor web o servidor HTTP es un programa informático que procesa una aplicación del lado del servidor utilizando generalmente el protocolo HTTP, o el protocolo HTTPS basado en este y el que se utilizará en el servidor.

El Servidor web se ejecuta en una máquina manteniéndose a la espera de peticiones por parte de un cliente desde un navegador web y responde a estas peticiones adecuadamente, mediante una página web que se exhibirá en el navegador. El servidor responde al cliente enviando el código, o respuesta, pertinente; el cliente, una vez recibida la respuesta, la interpreta y la exhibe en pantalla, si fuese necesario.

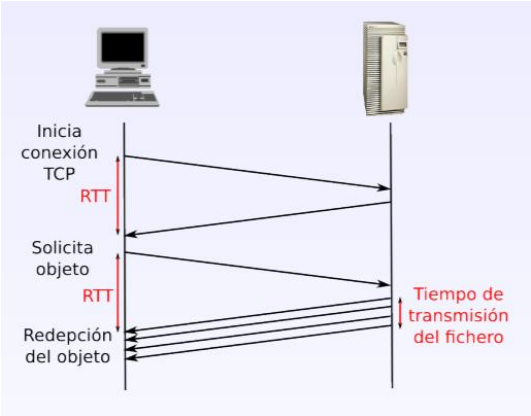

Figura 2-5 Esquema Cliente-Servidor Petición TCP

2.3.1 Apache 2

Figura 2-6 Apache

Apache2 es un servidor web de código abierto multiplataforma. Apache2 destaca por ser altamente configurable y la mayoría de las vulnerabilidades de seguridad descubiertas y resueltas no pueden ser aprovechadas remotamente.

Apache2 ha dominado el mercado de los servidores web y, aunque últimamente ha descendido su uso debido a la aparición del servidor web Nginx, que mejora en rendimiento cuando el número de clientes es muy elevado, en el servidor de UAVN se utilizará Apache2 porque ya se conoce su funcionamiento y es sencillo de utilizar y configurar, y porque no se van a albergar tantas webs y datos como para ver perjudicado el rendimiento.

2.3.2 HTTPS

El protocolo HTTPS utiliza un cifrado basado en SSL/TLS para crear un canal cifrado más apropiado para el tráfico de información sensible que el protocolo HTTP. De este modo se consigue que la información sensible no pueda ser usada por un atacante que haya conseguido interceptar la transferencia de datos de la conexión, ya que lo único que obtendrá será un flujo de datos cifrados.

2.3.3 Certificados SSL/TLS

No solo es importante proteger las webs con el protocolo HTTPS, sino que también es necesario proteger a los usuarios asegurando que sus datos están bien protegidos. Para ello existen los certificados SSL/TLS web, que evalúan la seguridad, la accesibilidad, el cumplimiento de estándares de programación o la adecuación a buscadores de los sitios web. Para ello, una autoridad certificadora, a partir de ahora CA, debe evaluar y asegurar que la web cumple los requisitos de seguridad y que realmente la web es quien dice ser.

Para el servidor se ha utilizado la CA Let's Encrypt dado que es gratis y muy simple de utilizar junto con Certbot que es un cliente que despliega el certificado SSL/TLS simplemente teniendo habilitado el protocolo HTTPS en el servidor web y un DNS activo. Una vez la CA autoriza el certificado, los navegadores web identifican como segura la web a la que apunta el certificado.

Figura 2-7 Certbot

Figura 2-8 Let's Encrypt

2.4 Servidor de Autenticación

El Servidor de Autenticación del servidor, a partir de ahora AS, administra el acceso por parte de los usuarios a los distintos servicios que hay en el servidor de UAVN.

El AS trabaja sobre el protocolo LDAP, siglas de *Lightweight Directory Access Protocol*. LDAP es un protocolo a nivel de aplicación que permite el acceso a un Servicio de Directorio o DS. El DS es una aplicación que almacena y organiza la información de los usuarios de una red y sobre los recursos de red que permite a los administradores gestionar el acceso de usuarios a los recursos de esta.

LDAP almacena la información de autenticación, usuario y contraseña, y es utilizado para ello, aunque también es posible almacenar otra información como datos de usuario, permisos o certificados. LDAP es un protocolo de acceso unificado a un conjunto de información sobre una red.

LDAP tiene una estructura orientada a objetos definida como sigue:

- **Clases**: Se definen los objetos y sus características, por ejemplo, un usuario o un tipo de grupo determinado.
- **Objetos**: Instancias creadas a partir de una clase o más.
- **Atributos**: Campos asociados a cada objeto creado que definen las características de este, por ejemplo, el nombre de un grupo.
- **dn**: Nombre Distinguido o *Distinguished Name*, sirve para identificar unívocamente a un determinado objeto en un directorio. Cada entrada definida es única en todo el directorio.

La siguiente imagen representa gráficamente un árbol de directorios LDAP:

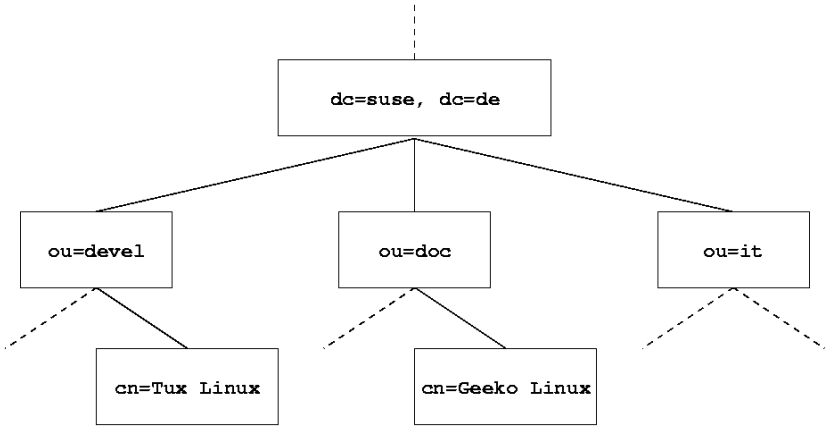

Figura 2-9 Estructura del Árbol de Directorios LDAP

dc indica *domain component* y es el nombre distinguido de la raíz de árbol de directorios, *ou* se refiere a que es un objeto de la clase *organizational unit* y *cn* es el nombre distinguido relativo o *common name*. Con esta información el *dn* del usuario Geeko Linux sería **dn**: *cn*=Geeko Linux, *ou*=doc, *dc*=suse, *dc*=de.

LDAP utiliza un formato de texto determinado, llamado LDIF (*LDAP Data Interchange Format*), para crear, modificar o mostrar entradas de un directorio. En este tipo de fichero la primera línea representa el *dn* de la entrada y viene seguida de distintas líneas que hacen referencia a su clase y atributos. Un ejemplo de fichero con formato LDIF es el siguiente:

```
dn: cn=John Doe,dc=example,dc=com
cn: John Doe
givenName: John
sn: Doe
telephoneNumber: +1 888 555 6789
telephoneNumber: +1 888 555 1232
mail: john@example.com
manager: cn=Barbara Doe,dc=example,dc=com
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: person
objectClass: top
```

Figura 2-10 Ejemplo de Fichero con formato LDIF (*.ldif*)

Existen muchas implementaciones del protocolo LDAP. Windows Server, por ejemplo, usa su propia implementación Active Directory. En el servidor UAVN nos decantamos por la implementación libre OpenLDAP.

2.4.1 OpenLDAP

Figura 2-11 OpenLDAP

Se trata de una implementación libre del protocolo que soporta múltiples esquemas por lo que puede utilizarse para conectarse a cualquier otro servidor LDAP. Además, al ser un protocolo independiente de la plataforma varias distribuciones, como GNU/Linux, lo incluyen.

OpenLDAP ejecuta demonizado el servicio *slapd*, el cual es el componente principal de la aplicación. Se compone además de las bibliotecas que implementan el protocolo y de diversos programas cliente para gestión del DS.

2.4.1.1 phpLDAPadmin

Figura 2-12 phpLDAPadmin

phpLDAPadmin es una herramienta web para la administración de servidores LDAP escrita en PHP. Trabaja en varias plataformas, pudiendo acceder al servidor LDAP desde cualquier lugar usando un navegador web.

2.5 Lenguajes de Programación

Para el desarrollo del servidor se han utilizado los lenguajes web HTML, PHP y JavaScript, así como Ruby para la aplicación Redmine, que se explicará en la sección de desarrollo de esta memoria, mySQL como lenguaje para las Bases de Datos del servidor, DB a partir de ahora, y Bash para scripting en el servidor Linux.

2.5.1 HTML

HTML es un estándar que sirve de referencia del software que conecta con la elaboración de páginas web en sus diferentes versiones, define una estructura básica y un código para la definición de contenido de una página web. La página web contiene solamente texto mientras que recae en el navegador web la interpretación del código.

2.5.2 PHP

PHP es un lenguaje de propósito general y de código abierto especialmente adecuado para el desarrollo web que corre del lado del Servidor y puede ser incorporado directamente en código HTML. Influido por C++ y Java.

2.5.3 JavaScript

JavaScript, abreviado comúnmente como JS, es un lenguaje de programación interpretado, orientado a objetos, imperativo y débilmente tipado. Se inserta en el código HTML, pero a diferencia de PHP, JS es ejecutado normalmente por el navegador del cliente. Influido por C, Java y Python.

2.5.4 CSS

CSS, hojas de estilo en cascada (*Cascading Style Sheets*), es un lenguaje de diseño gráfico para definir y crear la presentación de un documento estructurado. CSS está diseñado principalmente para marcar la separación del contenido del documento y la forma de presentación de este. Cada navegador web usa un motor de renderizado para renderizar páginas web, y el soporte de CSS no es exactamente igual en ninguno de los motores de renderizado. Para asegurar una experiencia consistente para sus usuarios, los desarrolladores web en ocasiones prueban sus sitios en múltiples navegadores, sistemas operativos y versiones de navegadores incrementando el tiempo de desarrollo y la complejidad.

Junto con HTML y JavaScript, CSS es una tecnología usada por muchos sitios web para crear páginas visualmente atractivas, interfaces de usuario para aplicaciones web, y GUIs para muchas aplicaciones móviles.

2.5.5 Ruby

Ruby es un lenguaje de programación interpretado, reflexivo y orientado a objetos. Combina una sintaxis inspirada en Python y Perl, aunque también comparte funcionalidad con Lisp. Ruby es un lenguaje multiparadigma ya que permite programación procedural, orientada a objetos o funcional.

2.5.6 MySQL

SQL es un lenguaje de dominio específico que da acceso a un sistema de gestión de DB relacionales. Una de sus características es que permite efectuar consultas con el fin de recuperar, de forma sencilla, información de las DB, así como hacer cambios en ellas. En el servidor de UAVN se utilizará MySQL como gestor de bases de datos.

2.5.7 Bash

Bash no es propiamente un lenguaje de programación, sino un lenguaje de comandos cuya función consiste en interpretar órdenes. Es un lenguaje de dominio específico utilizado en las Shell de Unix y Linux.

2.6 Programas utilizados para el Desarrollo

Durante el desarrollo del proyecto se han utilizado varios programas para comunicarse entre el sistema local de desarrollo, el servidor de *stage* y el de producción.

2.6.1 Cygwin

Figura 2-13 Cygwin

Cygwin es una colección de herramientas desarrollada por Cygnus Solutions para proporcionar un comportamiento similar a los sistemas Unix en Microsoft Windows. Su objetivo es portar software que ejecuta en sistemas POSIX a Windows con una recompilación a partir de sus fuentes. Se ha utilizado Cygwin para realizar conexiones SSH con el Cloud Server, así como accesos SFTP mediante consola desde fuera del servidor y la manipulación de archivos en el disco de Windows como si de un GNU/Linux se tratase para asegurar compatibilidad.

2.6.2 FileZilla

Figura 2-14 FileZilla

FileZilla es un programa multiplataforma y de código abierto que soporta los protocolos FTP, SFTP y FTP sobre SSL/TLS. Permite a un usuario crear una lista de sitios FTP o SFTP con sus datos de conexión y conectarse a ellos, mediante una interfaz gráfica los usuarios pueden navegar por las carpetas, ver y alterar sus contenidos y arrastrar y soltar archivos entre los ordenadores local y remoto. FileZilla muestra el registro de mensajes intercambiados con el servidor remoto y la cola de transferencia. Todas las pruebas fuera de consola para el servidor de transferencia de archivos se han realizado con este cliente.

2.6.3 SublimeText3

Sublime Text es un editor de texto multiplataforma y gratuito. Desarrollado originalmente como una extensión de Vim, con el tiempo fue creando una identidad propia, por esto aún conserva un modo de edición tipo *vi* llamado Vintage Mode.

2.6.4 Navegadores Web

Los navegadores web interpretan el código que envían los sitios web, lo representan y permiten interactuar con las páginas web. Los navegadores ofrecen herramientas que en tiempo real permiten modificar el aspecto que éste ofrece, lo que hace más ágil el desarrollo gráfico de las páginas web.

Los servicios web implementados en el servidor deben dar soporte a los navegadores Google Chrome y Mozilla Firefox. Estos navegadores en algunas etiquetas utilizan identificadores distintos en las hojas de estilos CSS, *-webkit* para Chrome y *-moz* para Firefox, por ello es muy útil inspeccionar el elemento recibido por el navegador y editarlo para conseguir los diseños deseados más adelante, pues un estilo para un navegador puede ser diferente para la interpretación de otro navegador.

Varias herramientas como *BrowserStack* han sido construidas para reducir la complejidad del mantenimiento de las páginas web.

2.6.5 phpMyAdmin

Se ha trabajado durante el desarrollo del proyecto constantemente con bases de datos, para la manipulación, creación, importación y borrado de bases de datos, así como ejecución de sentencias, para ello se ha utilizado la herramienta phpMyAdmin. Esta herramienta está escrita en PHP y maneja la administración de MySQL a través de una página web.

3 Análisis y Diseño

En este capítulo se describirá el análisis y el diseño llevados a cabo a cabo antes del desarrollo de los servicios del servidor de UAVN. Ya que todos los servicios deben autenticarse mediante el servidor LDAP, se debe centrar la atención en la conectividad de los distintos servicios con este servidor de autenticación.

3.1 Servidor de Autenticación LDAP

El Servidor de Autenticación LDAP es lanzado por la aplicación libre, comentada en el apartado anterior, OpenLDAP. Es importante elegir un buen diseño para la estructura de este servidor ya que es la estructura sobre la que se basan el resto de los servicios que existen en el servidor de UAVN para gestionar la autenticación de usuarios.

Para abordar el problema de la conectividad con LDAP hay que analizar los servicios que se han desarrollado en el servidor y utilizan esta autenticación: el Servidor de Transferencia de Archivos, servicio que no requieren del servidor web; Redmine, aplicación de código abierto lanzada por el servidor web; y, el Flight Log, aplicación web creada a medida para UAVN.

3.2 Servidor SFTP de Transferencia de Archivos

Un Servidor de Transferencia de Archivos es un sistema en el que los usuarios pueden alojar y compartir archivos a través de una red. En UAVN se decidió implementar dicho servicio basado en el protocolo SFTP, que se ejecuta en el servidor SSH del sistema.

A continuación, se explicará la necesidad y ventajas del desarrollo de este servidor y una descripción de cómo ha de ser el funcionamiento del servidor en UAVN.

3.2.1 Necesidad y Ventajas

El Servidor de Transferencia de Archivos provee a UAVN de una vía profesional y segura a la hora de compartir archivos grandes y temporales entre clientes o usuarios de la empresa.

El protocolo SFTP sobre el que funciona el Servidor de Transferencia de Archivos garantiza la seguridad de los archivos que se comparten. A ello se le une la confidencialidad de los usuarios, ya que todos ellos serán administrados por UAVN y no por ninguna otra entidad externa y, la garantía de que cualquier usuario, esté donde esté, tenga acceso a este servicio siempre y cuando exista una conexión a Internet.

3.2.2 Funcionamiento

SFTP, *SSH File Transfer Protocol* o *Secure File Transfer Protocol*, es un protocolo de transferencia de archivos que utiliza SSH, *Secure Shell*, para asegurar los comandos y los datos que se transfieren entre el cliente y el servidor, por lo que dejan de ser vulnerables a escuchas furtivas, interferencias o falsificaciones

No se debe confundir el protocolo SFTP con el protocolo FTPS, ya que SFTP es un protocolo de transferencia de archivos sobre SSH y FTPS es FTP/SSL, es decir, una extensión del protocolo FTP con intercambio seguro (SSL o TLS). SFTP fue construido desde cero y añade la característica de FTP a SSH. Sólo usa un canal de comunicación y envía y recibe los mensajes en binario, y no en formato texto como hace FTP. Por tanto, mientras que el protocolo FTP toma de forma predeterminada el puerto 21, el SFTP toma de forma predeterminada el puerto 22, que es el puerto SSH predeterminado.

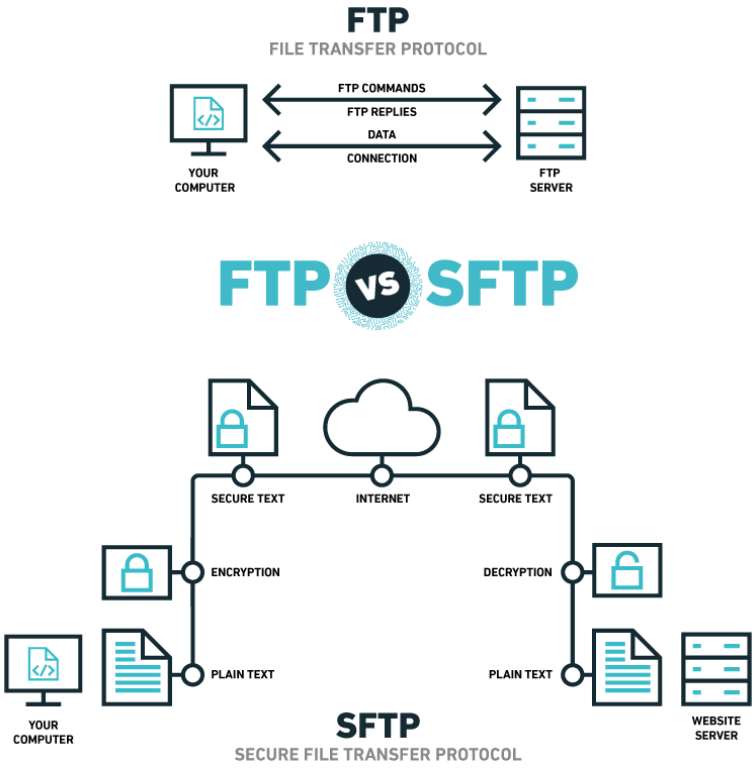

Figura 3-1 FTP vs SFTP

Como se ha dicho, SFTP funciona sobre el protocolo SSH, en las distribuciones GNU/Linux por defecto está instalado el conjunto de aplicaciones OpenSSH que permiten realizar comunicaciones cifradas usando el protocolo SSH. OpenSSH fue creado como una alternativa libre y abierta al programa Secure Shell, que es software propietario.

Por defecto el acceso SSH, por ello el SFTP también, está destinado a usuarios Linux existentes en el SO del servidor, se debe restringir el acceso mediante usuarios locales y habilitar el acceso mediante el servidor LDAP.

OpenSSH puede usar PAM, *Pluggable Authentication Modules* [2], para llevar a cabo la autenticación. PAM es un mecanismo flexible de autenticación que permite abstraer a las aplicaciones del proceso de autenticación sin modificar nada de ellas.

Las ventajas del uso de PAM son que:

- Ofrece un esquema de autenticación común y centralizado.
- Permite a los desarrolladores abstraerse de las labores de autenticación.
- Facilita el mantenimiento de las aplicaciones.
- Ofrece flexibilidad y control tanto para el desarrollador como para el administrador del sistema.

En el apartado de desarrollo de esta memoria se explicará con detalle la configuración de PAM para poder acceder por SSH y SFTP con usuarios LDAP.

Una vez el servidor está configurado para poder acceder a él mediante usuarios LDAP debemos asegurarnos de que sólo los usuarios que tienen permiso puedan acceder a él. Obviamente los usuarios no habilitados no pueden acceder al servidor, sin embargo, para evitar dejar puertas abiertas bastará con aplicar ciertas modificaciones a algunos ficheros de configuración del sistema.

En el próximo capítulo se verán las modificaciones realizadas en el fichero */etc/security/access.conf* [3], el fichero de control de acceso al sistema que actúa como un firewall de usuarios, las realizadas en el fichero de configuración del servidor SFTP */etc/ssh/sshd_config* [4] con el fin de permitir y limitar el acceso al servidor según convenga, y los scripts que agilizan el proceso de administración.

El árbol de directorios del Servidor de Transferencia de Archivos está dividido en dos grupos, una parte para la transferencia de archivos interna entre los empleados de UAVN y otra destinada a la transferencia de archivos entre UAVN y los clientes de la empresa. Las configuraciones del servidor y los permisos en los distintos directorios deben denegar el acceso de los clientes a la zona del servidor de los empleados, así como permitir a los empleados de UAVN modificar en cualquier momento ficheros compartidos por los clientes.

3.3 Redmine

Redmine es una herramienta para la gestión de proyectos que con sus diversas funcionalidades permite a los usuarios de diferentes proyectos realizar el seguimiento y organización de los mismos.

UAVN ya trabajaba con esta herramienta de forma local, por lo que, aparte de instalar la última versión (3.x) en el nuevo servidor, se ha de migrar la base de datos antigua, que trabajaba en la versión 2.x, de modo que los usuarios de la DB antigua sean reconocidos en la nueva y se puedan autenticar mediante LDAP con el mismo *login*.

Redmine ofrece la posibilidad de autenticarse con LDAP, para ello hay que habilitar en el servidor LDAP un tipo de grupo específico (*groupOfNames*), crear los grupos de este tipo que se quiera tengan acceso y añadir los usuarios a este. Cuando un usuario LDAP intente

acceder a Redmine, este preguntará al servidor LDAP si dicho usuario pertenece a algún grupo permitido y si sus credenciales de acceso son válidas.

3.4 Flight Log

En UAVN se realizan vuelos de prueba todas las semanas para probar los autopilotos que se fabrican y que más tarde serán vendidos a los clientes. Según la legislación vigente, el operador de los UAV debe establecer un sistema de registro relativo a dichos vuelos y el tiempo de vuelo, así como la anotación de las incidencias y otros eventos significativos durante el vuelo y el mantenimiento de los UAV [Anexo A]. Este registro se realiza en lo que se denomina un Flight Log.

Durante años en UAVN el Flight Log se ha realizado en formato físico en una plantilla impresa sobre un DIN A4, sin embargo, ello tiene el inconveniente de que tras muchos años el número de hojas de Flight Log llega a ser muy elevado, tanto que el almacenamiento de estos registros puede llegar a ser un problema, ello junto a que las búsquedas de cualquier Flight Log se pueden hacer lentas y tediosas.

En la programación web se utiliza HTML junto con PHP y MySQL en la parte del servidor, y adicionalmente se utiliza JavaScript que se ejecuta en la parte cliente. Las librerías de PHP ofrecen funciones para conectarse a servicios externos como sistemas de bases de datos o servidores LDAP, lo que otorga un gran potencial a los servicios web ya que desde éstos pueden administrarse bases de datos y mantener la seguridad en el acceso a estos servicios.

Por ello, se decidió crear en el nuevo servidor una plataforma web para los Flight Log. Dicha plataforma se ejecuta en el servidor web y el acceso de los usuarios a ella se realiza mediante el servidor de autenticación LDAP. Así, aparte de poder realizar las mismas tareas que en físico, se han podido añadir nuevas funcionalidades como la gestión de los registros o gestión de la DB según el rol de usuario, la unificación de la sección del mantenimiento en esta plataforma y la posible realización de búsquedas concretas en la base de datos de Flight Log. Además, ahora el acceso a todos los registros se puede realizar desde cualquier lugar con conexión a Internet, lo que por una parte es una ventaja, ya que se gana en accesibilidad, por otra parte, hace que la plataforma web deba ser segura ante ataques, pues todos los vuelos realizados por la empresa serán almacenados aquí.

3.5 Respaldo del Servidor

Aunque se comentó en el capítulo anterior que el proveedor del servidor ofrece el servicio de *backups*, lo que interesa no es tener un respaldo constante de todo aquello que existe en el servidor, además que aumentaría el precio ya que va en función del tamaño del disco. Interesa un respaldo de los datos y configuraciones más importantes para no perder ningún dato y poder reestablecer el servidor en caso de algún fallo importante. Así, los ficheros a respaldar diariamente serán los archivos de configuración del servidor, las bases de datos MySQL y LDAP y el código de las aplicaciones web. De este modo se tiene un control sobre qué archivos están siendo respaldados ya que esta copia de seguridad será encriptada y trasladada a otro servidor, por tanto, no puede excederse en tamaño de disco.

4 Desarrollo

En este capítulo se va a comentar el camino de desarrollo seguido durante todo el proyecto para instalar los componentes y desplegar el servidor de forma correcta y segura.

4.1 Configuraciones Iniciales

En la versión de Ubuntu 16.04.X que está instalada en el *Cloud Server* estaban instalados por defecto Apache2, el cual es el servidor web del servidor, y el conjunto de aplicaciones OpenSSH, que implementan el protocolo *SSH* con el que nos conectamos remotamente al servidor y cuyo proceso *sshd* ejecuta el servidor de transferencia de archivos.

Antes de comenzar con el desarrollo de los distintos servicios del servidor se necesitaron dos programas que son imprescindibles en un servidor como este ya que se utilizan en prácticamente todos los contenidos web.

4.1.1 Paquetes de PHP y MySQL

La administración de bases de datos ha sido indispensable durante el desarrollo, por ello era necesario tener un sistema de gestión de bases de datos como es MySQL. Además, Apache2 necesita el módulo de PHP para poder ejecutar los sitios web por parte del servidor que tengan dicho lenguaje en su código.

Por ello hubo que instalar los paquetes de PHP y de MySQL, así como las librerías y extensiones correspondientes para el correcto funcionamiento entre ellos como *mysqli*, una extensión MySQL para PHP, y la aplicación web phpMyAdmin para la administración de bases de datos vía web.

4.1.2 Habilitar HTTPS

A parte, para los servicios web que se implementan en el servidor, es necesario poder asegurar el correcto envío y recibimiento de paquetes, asegurándonos de que esa información esté segura. Para ello es necesario habilitar el protocolo HTTPS en Apache en el lugar del usado por defecto, que es el HTTP.

Para cada uno de los servicios web hay que habilitarlo en los archivos de configuración de Apache que gestionan la conexión.

Hay que hacer dos modificaciones básicas, la primera es que el puerto de escucha debe cambiarse del puerto 80, que es donde escucha el protocolo HTTP al puerto 443, puerto en el que escucha el protocolo HTTPS. La segunda es que hay que proporcionar los certificados electrónicos que firmó la AC y que determinan que nuestro sitio es legítimo.

Se muestra el fichero de configuración de Apache para el servicio de Redmine.

```
# Listening on port 443
<VirtualHost _default_:443>

      DocumentRoot /path_to_redmine/redmine
      Servername redmineserver
      ServerAlias redmine

      # Certificates
      SSLEngine on
      SSLCertificateFile      /path_to_cert/fullchain.pem
      SSLCertificateKeyFile   /path_to_cert /privkey.pem

      <Directory /path_to_redmine/redmine>
            RailsBaseURI /redmine
            PassengerAppRoot /path_to_cert/redmine
            PassengerResolveSymlinksInDocumentRoot on
      </Directory>

</VirtualHost>
```

4.2 Servidor de Autenticación LDAP

Para implementar correctamente el Servidor de Autenticación LDAP se tuvo que instalar en el servidor el programa OpenLDAP, cuyo componente principal es el servicio *slapd*, que una vez configurado correctamente con los datos del servidor se ejecuta demonizado y provee de este servicio.

Junto con la herramienta phpLDAPAdmin, que nos proporciona la interfaz gráfica vía web para así realizar las tareas de administración de LDAP, se generaron las *Organizational Units*, que sirven para estructurar LDAP, y en ellas se crearon los grupos y usuarios correspondientes. Estos usuarios fueron añadidos a los grupos que en un futuro les darían acceso a los servicios del Cloud Server.

Por defecto, las funciones *hash* de encriptado de OpenLDAP no son muy fuertes, por ello se añadió el encriptado SHA512 (SHA-2), que, en comparación con el SHA-1, el utilizado por defecto, mejora en la seguridad del encriptado.

OpenLDAP provee de programas clientes con los que modificar la estructura de árbol de directorio, así como la configuración interna de este directorio. En concreto, para habilitar el nuevo módulo hash hubo que utilizar el programa *ldapadd* [5]. A *ldapadd* se le puede pasar como argumento un fichero *LDIF* con entradas de directorio que LDAP interpreta y añade, si está bien expresado, a su estructura. El fichero LDIF que se pasó al programa *ldapadd* y que habilitó el nuevo módulo hash es el siguiente:

```
dn: cn=module, cn=config
cn= module
objectClass: olcModuleList
olcModuleLoad: pw-sha2
olcModulePath: /usr/lib/ldap
```

Hubo que añadir también la nueva función hash a phpLDAPAdmin para poder actualizar el hash de las contraseñas vía web modificando el fichero PHP */phpldapadmin/lib/functions.php*. Se muestra el código añadido a la función *password_hash_custom($password,$enc_type)* de este fichero:

```
case 'sha512':
if (function_exists('openssl_digest') && function_exists('base64_encode')) {

      $new_value = sprintf('{SHA512}%s,
      base64_encode(openssl_digest($password_clear,
      'sha512', true)));

} else {
      error(_('Your PHP install does not have the openssl_digest() or
            base64_encode() function. Cannot do S2K
      hashes.'),'error','index.php');
}
break;
```

Asimismo, fue necesario habilitar de forma similar al módulo hash un módulo que es imprescindible para que en las consultas LDAP se pudiera filtrar por *memberOf*, lo que sirve para poder determinar si un usuario pertenece a un grupo de tipo *groupOfNames*, un tipo de grupo LDAP que, en vez de almacenar el identificador de usuario como hacen los *PosixGroup*, almacena el *dn* de los miembros del grupo. El tipo de grupo *groupOfNames* será útil especialmente en los servicios web.

En el momento de crear los usuarios LDAP, éstos se crearon con contraseñas aleatorias usando la función hash *SHA512*, que se les dio a conocer para que más adelante la cambiasen por la contraseña que usasen antiguamente.

La siguiente imagen muestra el árbol de LDAP desde el phpLDAPAdmin del servidor una vez realizados los cambios y añadidos los usuarios y grupos. Los grupos añadidos se irán mostrando en función se vayan necesitando a lo largo del desarrollo.

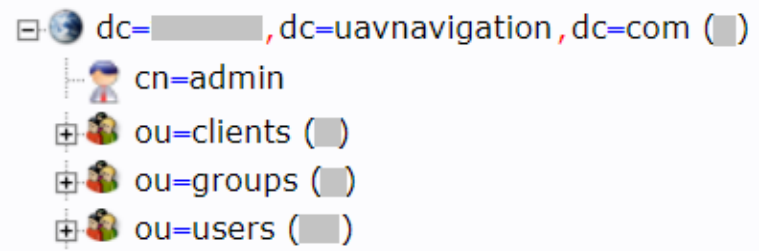

Figura 4-1 Árbol de Directorios LDAP – Vista Básica

4.2.1 Change LDAP Password

Debido a la necesidad de que los usuarios de UAVN siguiesen usando su antigua contraseña o quisieran modificarla sin necesidad de acudir al administrador del sistema, para mantener la privacidad de todos ellos se implementó una sencilla página web llamada *Change LDAP Password* que consiste en un formulario en el que se pide el mail del usuario, su contraseña actual y la nueva contraseña.

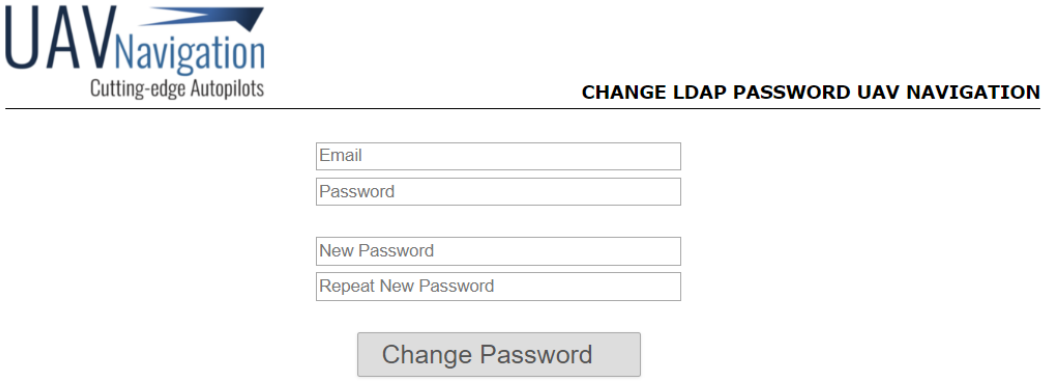

Figura 4-2 Vista de la Página Principal de Change LDAP Password

Los datos son procesados por código PHP, que establece conexión con el servidor LDAP, comprueba que el usuario y la contraseña son correctos y procede a introducir la nueva contraseña como la actual.

Se muestran a continuación fragmentos de código utilizados en el código PHP de esta página:

```
// Establece la Conexión con el Servidor LDAP
$ldapconn = ldap_connect($ldapip);

// Autenticación del usuario en la dirección $ldaprdn con contraseña $ldapass
$ldapbind = ldap_bind($ldapconn, $ldaprdn, $ldappass);

// Modifica la Contraseña del usuario ($newEntry) en la dirección LDAP $ldaprdn
ldap_mod_replace($ldapconn, $ldaprdn, $newEntry);
```

4.3 Servidor SFTP de Transferencia de Archivos

Como se comentó al comienzo de este capítulo, el conjunto de aplicaciones OpenSSH ya estaban instaladas cuando se adquirió el servidor, por lo que no fue necesario instalar ningún paquete adicional que los ya instalados para esta parte del servidor. Sin embargo, sí hubo que modificar su configuración inicial.

4.3.1 Claves SSH

Las claves SSH consisten en la generación de un par de claves que proporcionan dos largas cadenas de caracteres, una pública y otra privada. Las claves SSH proveen más seguridad en el acceso a un servidor con SSH que utilizando sólo contraseña, la cual puede ser descifrada por fuerza bruta.

Normalmente las claves del servidor, a partir de ahora *host keys*, se generan automáticamente cuando se instala OpenSSH o cuando el servidor se inicia por primera vez. Sin embargo, por seguridad se generaron nuevas *host keys* usando el programa *ssh-keygen*.

Las *host keys* son almacenadas en el servidor en los ficheros que comienzan por *ssh_host_** en el directorio de configuración de SSH, */etc/ssh/*.

```
ssh_host_dsa_key            ssh_host_ecdsa_key.pub          ssh_host_rsa_key
ssh_host_dsa_key.pub        ssh_host_ed25519_key            ssh_host_rsa_key.pub
ssh_host_ecdsa_key          ssh_host_ed25519_key.pub
```

Ya que las *host keys* son claves criptográficas, las claves privadas deben ser accesibles sólo por el usuario *root*. Entonces cualquier usuario con acceso *root* puede obtener las claves privadas del servidor, un atacante que consiga tener acceso *root* al servidor puede copiar la clave privada y realizar un ataque *man-in-the-middle* en el servidor. Es importante regenerar periódicamente las *host keys*, además de limitar el acceso *root* sólo a los administradores del servidor para evitar problemas de seguridad.

Por su parte, el cliente debe generar sus claves pública y privada de forma similar. A la hora de la conexión entre cliente y servidor el servidor debe conocer la clave pública del cliente, de este modo, el servidor encripta con la clave pública del cliente un mensaje, si recibe de vuelta el mensaje desencriptado se verifica la identidad del cliente y se continúa con la conexión SSH.

4.3.2 Fichero de Configuración del Servidor SSH

Tras estos pasos la conexión SSH es más estricta, sin embargo, los usuarios del servicio SFTP deben tener únicamente acceso al servidor SFTP y no poder conectarse al servidor mediante SSH, ya que sería un gran problema de seguridad. Por ello, en el archivo de configuración del servidor SSH, */etc/ssh/sshd_config*, hay que realizar ciertos cambios.

El proceso *sshd* es el encargado de lanzar el servidor SSH. Éste lee los datos de configuración del fichero */etc/ssh/sshd_config*. El fichero contiene directivas, o palabras clave, seguidas de valores o listas de valores que determinan el funcionamiento del servidor SSH.

Algunas de las directivas más importantes de este fichero para la correcta configuración del Servidor de Transferencia de Archivos son:

- **PermitRootLogin:** Habilita o deshabilita el acceso sólo por contraseña de los usuarios *root*, también permite habilitar el acceso mediante clave pública/privada.
- **AllowTcpForwarding:** Especifica si el reenvío de paquetes TCP está permitido o no.
- **AllowStreamLocalForwarding:** Especifica si el reenvío de puertos está permitido o no.
- **GatewayPorts:** Especifica si los hosts remotos pueden conectarse a los puertos reenviados por el cliente.
- **PermitTunnel:** Especifica si están permitidos el uso de túneles para acceder al servidor.
- **UsePAM:** Habilita el *Pluggable Authentication Module* (PAM).

- **AllowGroups y AllowUsers:** Especifica que el *login* sólo estará permitido para aquellos grupos o usuarios listados a continuación de la directiva escogida. La directiva *AllowUsers* tiene preferencia sobre *AllowGroups* y ambas se ejecutan por separado.

- **Match Group:** Inicia un bloque condicional, las directivas escritas a continuación de esta sólo serán aplicadas al grupo que se indique.

- **ChrootDirectory:** Enjaula el acceso al servidor a un directorio específico.

- **ForceCommand internal-sftp:** Deshabilita el acceso SSH.

Las cinco primeras directivas aportarán más velocidad de acceso cuanto menor sea la seguridad y viceversa. Puesto que en el servidor lo que buscamos sobre todo es seguridad, la directiva *PermitRootLogin* se configuró para deshabilitar las contraseñas de usuarios *root* y obligar a usar clave pública/privada y autenticación por contraseña, las siguientes cuatro directivas se configuraron para no permitir reenvíos ni túneles.

Para todos los usuarios, excluyendo para el usuario que administra el servidor, el acceso mediante SSH va a estar deshabilitado, *ForceCommand internal-sftp*.

Para poder autenticar a los usuarios de LDAP se habilitará el uso de PAM, cuya configuración y funcionamiento se explicará en el siguiente apartado. La directiva *ChrootDirectory* está muy ligada con la estructura del servidor y se detallará más tarde.

4.3.3 Autenticación SSH/SFTP mediante PAM

Como se explicó en el capítulo anterior, OpenSSH puede usar PAM para llevar a cabo la autenticación mediante LDAP.

Para habilitar esta autenticación hubo que instalar antes determinados paquetes para la correcta comunicación entre LDAP, PAM y NSS. Los paquetes necesarios para proceder fueron *libnss-ldap*, *libpam-ldap* y *libpam-modules*.

Una vez habilitada correctamente la comunicación entre estos tres servicios hubo que configurar NSS y PAM.

4.3.3.1 NSS

NSS, *Name Service Switch*, es el nombre que recibe la interfaz de Linux que permite configurar y acceder a diferentes bases de datos de cuentas de usuarios como */etc/passwd, /etc/group, /etc/hosts* o LDAP.

Al archivo de configuración NSS, */etc/nsswitch.conf*, se le añadió LDAP para la lectura de cuentas de usuario de la forma que sigue en las bases de datos *passwd*, *group* y *shadow*:

```
passwd:                 compat      files       ldap
group:                  compat      files       ldap
shadow:                 compat      files       ldap
gshadow:                files

hosts:                  files       dns
networks:               files

protocols:              db          files
services:               db          files
ethers:                 db          files
rpc:                    db          files

netgroup:               nis
```

4.3.3.2 PAM

Para configurar la autenticación en el servidor mediante PAM se tuvieron que modificar los ficheros */etc/pam.d/common-*.*

La versión de PAM para Linux divide la funcionalidad en diferentes servicios dependiendo en qué parte del proceso se encuentra:

- **Authentication:** Comprueba si el usuario puede aportar credenciales válidas.
- **Account:** Responsable de decidir si la cuenta que está intentando iniciar sesión tiene acceso a los recursos que está solicitando en este momento.
- **Session**: Establece el entorno que se generará y destruirá después de que el usuario inicie o cierre sesión. Los archivos de sesión pueden determinar qué comandos se deben ejecutar para preparar el entorno.
- **Password**: Responsable de actualizar los detalles de autenticación de varios servicios. Si se necesita cambiar una contraseña para un servicio, este módulo puede ayudarlo a comunicarse con el servicio y modificar los valores correctos.

Los dos primeros servicios serán llamados cada vez que un programa utilice PAM para autenticar usuarios. El servicio *Session* se ejecuta sólo si es necesario y al servicio *Password* se accede bajo demanda.

En el directorio */etc/pam.d* existe un archivo de configuración para cada programa que pueda necesitar la autenticación PAM, en caso de no existir ningún archivo asociado al programa se aplica el archivo */etc/pam.d/other*. En los archivos de configuración de cada

programa se realizan llamadas a los ficheros que comienzan por *common-*, que son los archivos de configuración generales de los servicios citados cuyas reglas deben aplicarse en la mayoría de las situaciones.

```
atd       chsh            common-password               cron      other     runuser    su            vmtoolsd
chfn      common-account  common-session                login     passwd    runuser-l  sudo
chpasswd  common-auth     common-session-noninteractive  newusers  polkit-1  sshd       systemd-user
```

Cuando una aplicación requiere la autenticación mediante PAM se lee el archivo de configuración relevante de la aplicación, en el caso actual sería *sshd*, y dentro de este archivo se ejecutan módulos PAM y se realizan llamadas a los servicios PAM mediante la inclusión de los ficheros *common-* según corresponda. Estos archivos contienen una lista de módulos PAM y cómo deben ser operados. Cada módulo es llamado uno a uno y devuelve un resultado de éxito o fallo.

Según el resultado obtenido, el archivo de configuración decide si devolver éxito en la autenticación o devolver un fallo. Si un archivo de configuración devuelve un fallo en un módulo puede estar configurado para pasar al siguiente módulo. Por ejemplo, se puede intentar autenticar un usuario y no existir en los usuarios locales y posteriormente intentar autenticarlo con LDAP y devolver éxito.

Las líneas de cada archivo de configuración son evaluadas de arriba abajo y cada línea puede ser una llamada a un archivo de servicio PAM o una ejecución de un módulo PAM concreto, las cuales siguen la siguiente sintaxis:

```
type      control      module-path      [module-arguments]
```

Donde:

- **Type:** Es el tipo de servicio que se provee. Ha de ser uno de los módulos anteriormente citados, estos son *auth*, *account*, *session* y *password*.

- **Control:** Especifica qué acción realizar cuando se devuelve el retorno del módulo llamado. Puede ser *required*, *requisite*, *sufficient*, *optional*, *include* o *substack*.

- **Module-Path:** Es el nombre del módulo PAM a ejecutar.

- **Module-Arguments:** Son los parámetros adicionales que se le pueden pasar al módulo.

Los distintos módulos que PAM puede ejecutar se pueden visualizar en */lib/*/security*. Para el propósito del servidor de UAVN se utilizó el módulo *pam_ldap.so*.

```
pam_access.so        pam_faildelay.so     pam_ldap.so          pam_motd.so
pam_securetty.so     pam_tally2.so        pam_userdb.so        pam_unix.so
pam_debug.so         pam_filter.so        pam_limits.so        pam_namespace.so
pam_selinux.so       pam_tally.so         pam_warn.so          pam_systemd.so
pam_deny.so          pam_ftp.so           pam_listfile.so      pam_nologin.so
pam_sepermit.so      pam_time.so          pam_wheel.so         pam_rootok.so
pam_echo.so          pam_group.so         pam_localuser.so     pam_permit.so
pam_shells.so        pam_timestamp.so     pam_xauth.so         pam_mkhomedir.so
pam_env.so           pam_issue.so         pam_loginuid.so      pam_pwhistory.so
pam_stress.so        pam_tty_audit.so     pam_lastlog.so       pam_extrausers.so
pam_exec.so          pam_keyinit.so       pam_mail.so          pam_rhosts.so
pam_succeed_if.so    pam_umask.so
```

Para el servidor de SSH, *sshd*, que necesita autenticación LDAP, PAM leerá el fichero */etc/pam.d/sshd*. En este fichero se incluyen llamadas a los ficheros de los servicios *auth*, *account*, *password* y *session*, que devolverán sus respectivos retornos llamando a los distintos módulos y validarán, o no, la autenticación.

A continuación, se muestra parcialmente el archivo de configuración */etc/pam.d/sshd* y el archivo *common-auth*:

```
/etc/pam.d/sshd:

#Ejecución del módulo pam_access.so
auth    required        pam_access.so
#Llamada al servicio PAM common-auth
@include common-auth
…
```

```
/etc/pam.d/common-auth:

auth    [success=2 default=ignore]      pam_unix.so nullok_secure
auth    [success=1 default=ignore]      pam_ldap.so use_first_pass
auth    requisite                       pam_deny.so
auth    required                        pam_permit.so
```

En este último fichero se puede observar como si la autenticación Unix falla se ignora y se pasa a la siguiente línea, que es la autenticación LDAP. Si en este caso la autenticación LDAP también retornara error se ignora y se pasa a la siguiente línea que deniega la autenticación. En caso de éxito en cualquiera de los dos primeros módulos, Unix y LDAP, se observa como en caso de éxito el *control* le comunica a PAM que se salte las dos siguientes o la siguiente línea, respectivamente, para ejecutar el módulo que permite la autenticación.

4.3.4 Estructura de Directorios

La estructura del servidor de transferencia de archivos está dividida en dos directorios principales, un directorio específicamente destinado a compartir archivos entre los usuarios de la empresa y otro en el que se realizarán los envíos de ficheros entre los clientes y UAVN. Según el grupo al que pertenezcan los usuarios que acceden al servidor tienen unos privilegios u otros.

Los usuarios de UAVN pertenecen todos al mismo grupo LDAP al acceder al servidor SFTP, este grupo ha sido llamado *SFTPServerUploads*. Estos usuarios tienen como directorio raíz el mismo directorio raíz que tiene el SFTP y existe un directorio para cada usuario dentro del directorio de usuarios de UAVN.

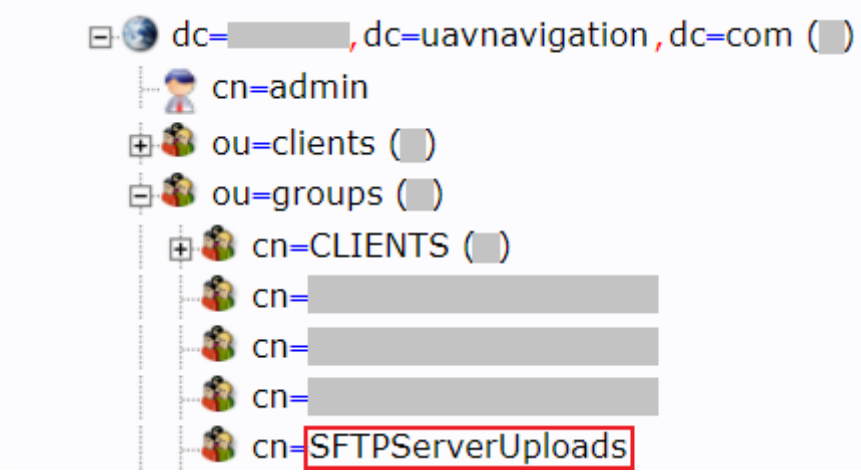

Figura 4-3 Árbol de Directorios LDAP – Localización Grupo SFTPServerUploads

Por otro lado, los clientes están encapsulados en directorios específicos del grupo cliente al que pertenecen dentro del directorio principal destinado a los clientes.

Es aquí donde la directiva ChrootDirectory, comentada anteriormente en el fichero de configuración *sshd_config*, toma importancia. Establecer esta directiva correctamente según el usuario que se conecta es imprescindible para garantizar la seguridad y confidencialidad del servidor.

Los clientes, al estar encapsulados en sus correspondientes directorios, son incapaces de acceder a directorios superiores en el árbol de directorios. De este modo no pueden ver con qué otros clientes UAVN tiene contacto ni pueden acceder al directorio exclusivo para los usuarios de UAVN.

Los usuarios de UAVN también están encapsulados, ya que si se le concediera permiso de acceso SFTP a un usuario no encapsulado podría acceder a todo el contenido del Cloud Server, por tanto, son encapsulados en el directorio raíz el servidor de transferencia de archivos.

El encapsulamiento de los distintos grupos de usuarios sigue la siguiente sintaxis en el fichero de configuración *sshd_config*:

```
/etc/ssh/sshd_config:

…
# Encapsulamiento de los usuarios del grupo cliente ClientGroup
Match Group ClientGroup
      ChrootDirectory /path_to_sftp_server/CLIENTS/ClientGroup
      ForceCommand internal-sftp
…
# Encapsulamiento de los usuarios del grupo SFTPServerUploads
Match Group SFTPServerUploads
      ChrootDirectory /path_to_sftp_server
      ForceCommand internal-sftp
…
```

Además del encapsulamiento de los usuarios es necesario atribuir los permisos de directorio correspondientes para que sólo los usuarios permitidos puedan crear ficheros o subárboles de directorios.

Para mantener una estructura clara y segura, en los grupos de los clientes existen dos directorios, uno destinado a las subidas de UAVN y otro para las subidas de los Clientes, cada uno con los permisos necesarios para que los usuarios de UAVN puedan acceder y borrar todo tipo de archivos nuevos en esta sección y los Clientes sólo tuvieran permisos de escritura en el directorio destinado a ellos.

De este modo, los permisos en los directorios de los clientes quedan de la siguiente manera:

```
drwxrwsrwx+ 2 root ClientGroup         4096 Aug 23 10:31 Customer/
drwxrwsr-x+ 2 root SFTPServerUploads   4096 Aug 23 10:08 Service/
```

4.3.5 Logs del Servidor SFTP

Para que los administradores del servidor puedan conocer qué ha acontecido o qué archivos se han ido subiendo al servidor, es necesario crear un registro o log del servidor SFTP. Un log es un fichero de texto en el que se escriben cronológicamente los acontecimientos que han ido afectando a un sistema informático, en este caso al servidor de transferencia de archivos.

Por defecto, los logs del servidor SFTP no están habilitados. Para habilitarlos es necesario crear un socket de escucha en cada uno de los puntos de encapsulamiento del servidor, estos son el directorio raíz y los directorios destinados a los grupos cliente. Una vez creados, estos sockets enviarán la información al *syslog*, el log del sistema. Sin embargo, la mejor idea para administrar los logs del servidor de transferencia de archivos es tenerlos en un fichero separado, por ello se trasladarán a un nuevo fichero creado en */var/log* llamado *sftp.log*.

Para ello se realizan los siguientes cambios en el fichero de configuración del servidor SFTP y en el servicio responsable de los logs del sistema, *rsyslog*.

```
/etc/ssh/sshd_config:

…
# Agregado en subsistema genearal sftp el método de debug VERBOSE a la facilidad del
syslog LOCAL6
Subsystem sftp internal-sftp -l VERBOSE -f LOCAL6
…
# Agregado en el grupo cliente el método de debug VERBOSE a la facilidad del syslog
LOCAL6
Match Group ClientGroup
      ChrootDirectory /path_to_sftp_server/CLIENTS/ClientGroup
      ForceCommand internal-sftp -l VERBOSE -f LOCAL6
…
# Agregado en el grupo SFTPServerUploads el método de debug VERBOSE a la facilidad del
syslog LOCAL6
Match Group SFTPServerUploads
      ChrootDirectory /path_to_sftp_server
      ForceCommand internal-sftp -l VERBOSE -f LOCAL6
…
```

```
/etc/rsyslog.d/60-sftp.conf

# Se genera un socket por cada punto de encapsulamiento del servidor SFTP
$AddUnixListenSocket /path_to_sftp_server/dev/log
$AddUnixListenSocket /path_to_sftp_server/CLIENTS/ClientGroup/dev/log
```

```
/etc/rsyslog.conf

…
# Redirige el tráfico de la facilidad de syslog LOCAL6 al fichero /var/log/sftp.log
Local6.*    /var/log/sftp.log
```

```
internal-sftp[1287]: session opened for local user earacil from [89.130.129.199]
internal-sftp[1287]: received client version 3
internal-sftp[1287]: realpath "."
internal-sftp[1287]: realpath "/UAV_NAVIGATION/Enrique_Aracil"
internal-sftp[1287]: opendir "/UAV_NAVIGATION/Enrique_Aracil"
internal-sftp[1287]: closedir "/UAV_NAVIGATION/Enrique_Aracil"
internal-sftp[1287]: realpath "/UAV_NAVIGATION/Enrique_Aracil/UAVN_SFTP_Connection.docx"
internal-sftp[1287]: open "/UAV_NAVIGATION/Enrique_Aracil/UAVN_SFTP_Connection.docx" flags WRITE,CREATE,TRUNCATE mode 0666
internal-sftp[1287]: close "/UAV_NAVIGATION/Enrique_Aracil/UAVN_SFTP_Connection.docx" bytes read 0 written 12447
internal-sftp[1287]: realpath "/UAV_NAVIGATION/Enrique_Aracil/."
internal-sftp[1287]: opendir "/UAV_NAVIGATION/Enrique_Aracil"
internal-sftp[1287]: closedir "/UAV_NAVIGATION/Enrique_Aracil"
internal-sftp[1287]: realpath "/UAV_NAVIGATION/Enrique_Aracil/UAVN_SFTP_Connection.docx"
internal-sftp[1287]: stat name "/UAV_NAVIGATION/Enrique_Aracil/UAVN_SFTP_Connection.docx"
internal-sftp[1287]: open "/UAV_NAVIGATION/Enrique_Aracil/UAVN_SFTP_Connection.docx" flags READ mode 0666
internal-sftp[1287]: close "/UAV_NAVIGATION/Enrique_Aracil/UAVN_SFTP_Connection.docx" bytes read 12447 written 0
internal-sftp[1229]: realpath "/UAV_NAVIGATION/Enrique_Aracil"
internal-sftp[1229]: remove name "/UAV_NAVIGATION/Enrique_Aracil/UAVN_SFTP_Connection.docx"
internal-sftp[1287]: session closed for local user earacil from [89.130.129.199]
```

Figura 4-4 Vista del Log del Servidor SFTP

Tras las modificaciones señaladas, el log del servidor SFTP se visualiza como se muestra en la Figura 4-4. Aquí se detallan los movimientos realizados por el usuario *earacil*:

- Inicia sesión en el servidor desde la IP 89.130.129.199.
- Accede al directorio */UAV_NAVIGATION/Enrique_Aracil*, lo que sería su directorio personal dentro del directorio destinado a los usuarios de UAVN.
- Sube un fichero en esta dirección llamado *UAVN_SFTP_Connection.docx*, cuyo tamaño es de 12447 bytes.
- Descarga este mismo fichero.
- Borra del servidor el fichero.
- Cierra sesión en el servidor.

4.3.6 Automatización de los procesos de Administración

Utilizando la herramienta *cron* de Linux/Unix, descrita en el apartado de Estado del Arte, se automatiza un script Bash que se encarga de notificar por correo electrónico a los usuarios administradores haciendo uso del servidor de correo del servidor cuando:

4.3.6.1 Uso del Disco Elevado

Cuando el uso del disco del servidor alcanza una cuota determinada se envía un correo mostrando la cuota de disco alcanzada, así como el uso de disco, el uso por parte del servidor SFTP y el disco libre. A continuación, se muestra un ejemplo de cómo es el formato del correo enviado por el servidor a los administradores.

```
Date: Mon, 10 Jun 2019 08:23:10 +0000 (UTC)
From: root <root@server>

[Mon Jun 10 08:23:10 UTC 2019]

Data Usage of Server has reached 90%. Used 9.0G of 10G, of which, SFTP Server is using
2.0G.

Mail sended by cron. Server.
```

Para obtener el tamaño de disco y el uso actual se puede utilizar el siguiente comando, del cual, se obtiene el disco montado que nos interesa, en este caso *dev1*:

```
# df -h | grep /dev/vda1

/dev/vda1        10G  9G   1G  90% /
```

En el caso de querer obtener el tamaño de un directorio, usaremos el comando:

```
# du -sh /direcotoryorfile

2.0G    /directoryorfile
```

4.3.6.2 Servidor Web no Disponible

En caso de que el servidor web Apache2 se apague o haya sufrido un reinicio inesperado del cual no se pueda levantar, el servidor notificará a los administradores por correo electrónico. Como el servidor de correo no depende del servidor web, se obtendrá el estado del servidor web, y en caso de no estar encendido se notificará:

```
Date: Mon, 10 Jun 2019 08:23:10 +0000 (UTC)
From: root <root@server>

[Mon Jun  10 08:23:10 UTC 2019]

Apache 2 has been stoped in Server server (IP).

If you have not shutted it down you should have a look to the server status.

Mail sended by cron. Server.
```

4.4 Redmine

4.4.1 Proceso de Actualización

Como se comentó en el capítulo 3, además de tener que integrar Redmine con acceso mediante LDAP, hubo que actualizarlo de versión. Para ello fue muy útil la página oficial de Redmine http://www.redmine.org, que incluye mucha información para poder realizar los *upgrades*, entre otras.

El proceso de actualización incluyó respaldo de la DB y de los archivos de los proyectos del programa, actualización del programa a la última versión y, por último, utilizar unas herramientas dedicadas de Redmine escritas en Ruby que migran la DB original para que pueda trabajar con la nueva versión.

4.4.1.1 Migración DB

Tras hacer el adecuado respaldo de la DB de Redmine, fue necesario ejecutar ejecutar una sentencia del *framework Ruby on Rails*, distribuído a partir de las llamadas *gemas* de Ruby.

Este comando modifica la DB de una versión antigua de Redmine para que pueda trabajar normalmente con la versión instalada en el servidor.

```
RAILS_ENV=production bundle exec rake db:migrate
```

4.4.1.2 Migración de Ficheros

Una vez migrada la DB de la versión antigua de Redmine le tocó el turno a los ficheros que se suben a la plataforma para ser compartidos en los proyectos. Como en la DB vienen referenciados y ésta ya está funcionando normalmente con la nueva versión, el proceso es sencillo.

Bastó con mover el respaldo de estos archivos a la nueva ubicación de Redmine y la subcarpeta de la plataforma */files* y proporcionar los permisos adecuados de lectura al usuario apache (usuario que gestiona el servidor web y por tanto las aplicaciones ubicadas en él) para poder visualizar los archivos desde la web.

4.4.2 Autenticación mediante Servidor LDAP

Como Redmine soporta de forma nativa la conexión con LDAP, hay que configurarlo de modo que se conecte con nuestro servidor LDAP a la hora de autenticar usuarios y no con la DB de usuarios de la herramienta.

Desde el panel de Administración de Redmine se accede al módulo 'LDAP authentication', desde aquí se pueden configurar los servidores LDAP que permiten el

acceso de esta herramienta. En este proyecto hubo que configurar el servidor LDAP explicado en el subcapítulo 4.2.

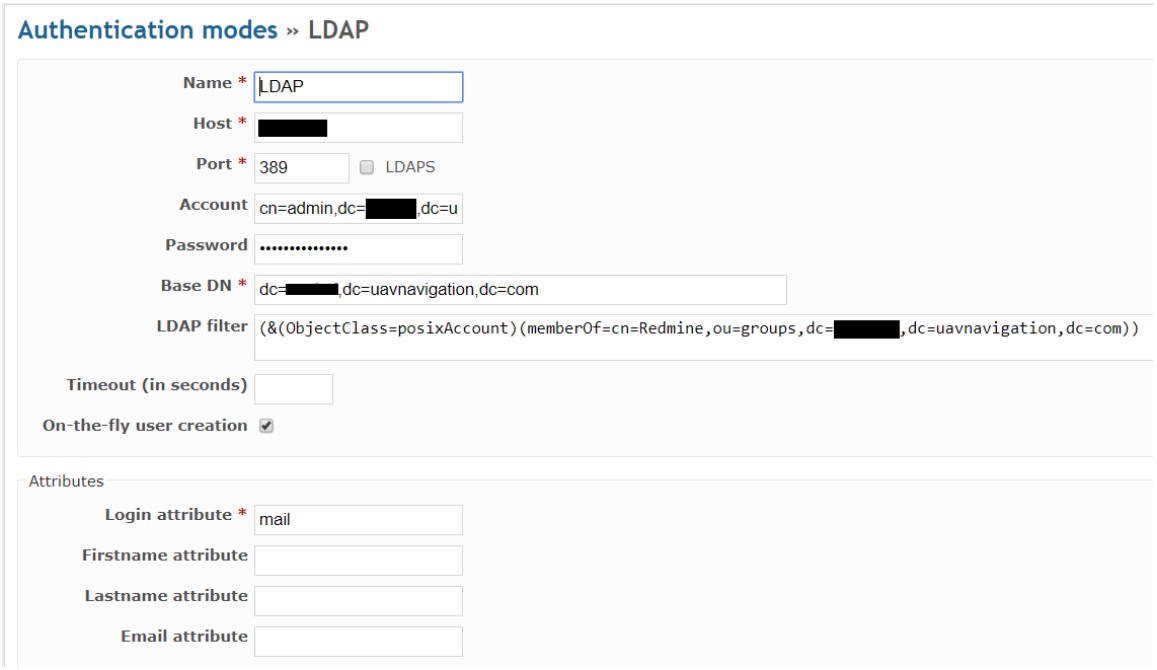

Figura 4-5 Módulo de Redmine para Autenticación con LDAP

Redmine te permite poder elegir qué usuarios de qué grupo de LDAP podrán tener acceso. En este caso, el grupo al que se le está dando acceso es el *cn=Redmine*. el filtro de LDAP obtendrá aquellos usuarios de tipo *posixAccount* de este grupo. Además, para no tener que insertar como login el nombre del usuario de LDAP, se permite poder identificar a estos usuarios con el atributo 'mail' (Login attribute).

Tras rellenar los campos correctamente con los datos de nuestro servidor LDAP, se podrá testear si la conexión y las propiedades de esta son correctas. Una vez comprobado ya se podrá hacer Login en Redmine con los usuarios de LDAP. Esta conexión se añadirá a la DB de Redmine.

En las siguientes imágenes se muestra al usuario *cn=Enrique Aracil* incluido en el grupo LDAP *Redmine* accede a Redmine mediante el servidor de autenticación LDAP.

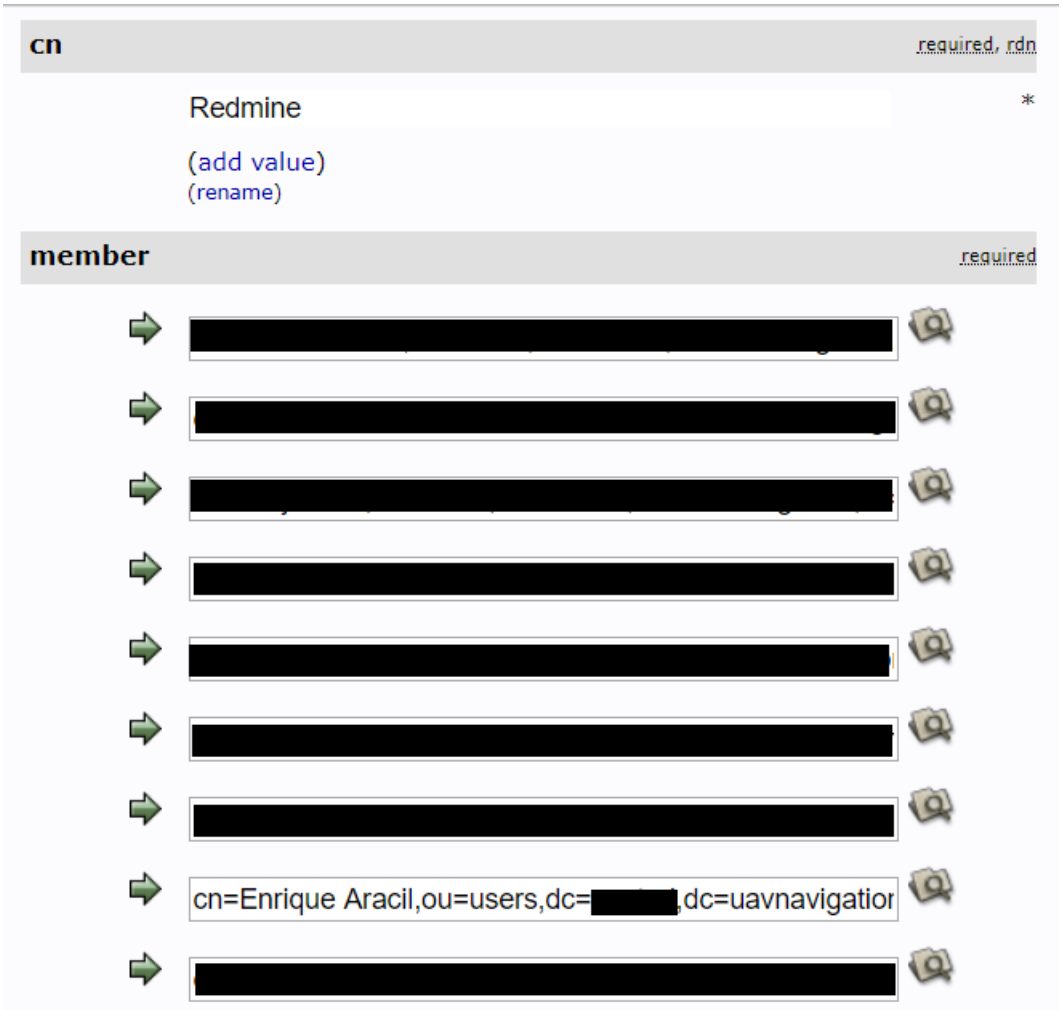

Figura 4-6 Usuario cn=Enrique Aracil en el grupo Redmine de LDAP

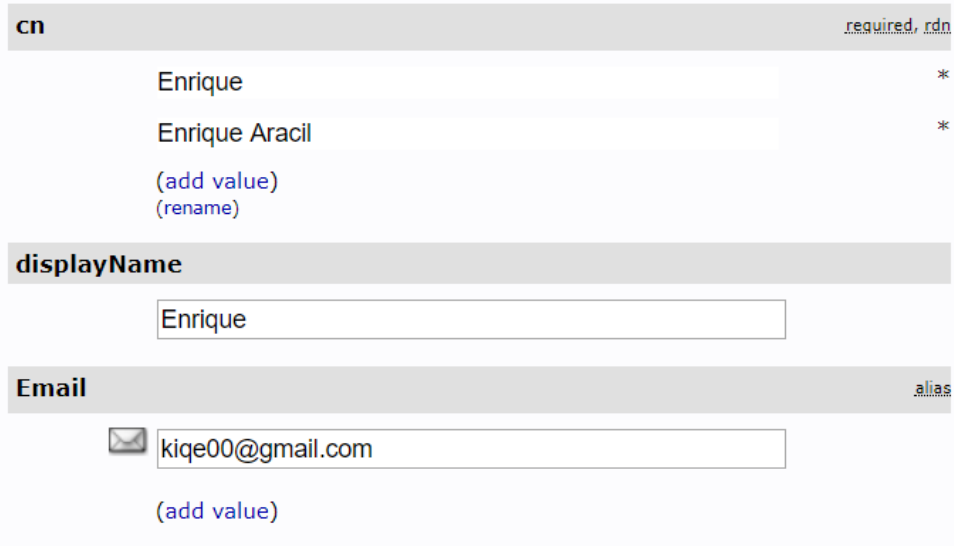

Figura 4-7 Datos del Usuario cn=Enrique Aracil en LDAP

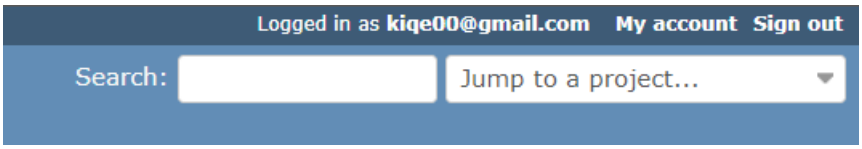

Figura 4-8 Acceso a Redmine del Usuario LDAP con mail kiqe00@gmail.com

Tras la primera conexión a Redmine mediante un servidor de autenticación, este usuario se registrará en la DB de Redmine, teniendo por valor la columna *auth_source_id* el índice de la conexión LDAP creada anteriormente, en este caso es 1. Por tanto, siempre que vaya a intentar acceder este usuario se preguntará al servidor LDAP por las credenciales aportadas.

Figura 4-9 Muestra del Usuario con Login kiqe00@gmail en la DB de Redmine (tabla users)

4.5 Flight Log

Como ya se han explicado las necesidades de la web de Flight Log, el desarrollo de esta herramienta web desde cero se ha basado en suplir estas de forma digital, accesible y unificada.

4.5.1 Login con Servidor LDAP

Para acceder a este sitio web, de la misma manera que el resto de los servicios explicados en esta memoria, fue necesario crear un archivo PHP en el lado del servidor que se comunicase con el servidor LDAP para validar a un usuario.

Antes de continuar con el desarrollo del login es importante conocer la estructura de los grupos que dan acceso a esta herramienta en LDAP.
Se crearon en LDAP dos grupos con acceso al Flight Log, uno con privilegios administración, aquellos usuarios que podrán modificar tablas o eliminar recursos, y otro de acceso sin privilegios, en el que se limitarán a crear, visualizar o modificar las hojas de vuelo.

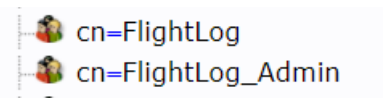

Figura 4-10 Grupos de acceso al Flight Log en LDAP

Ambos grupos tuvieron que crearse de la clase *GroupOfNames* para poder listar todos los usuarios de estos grupos de forma completa.

El aspecto de la página de login a la herramienta del Flight Log es el siguiente:

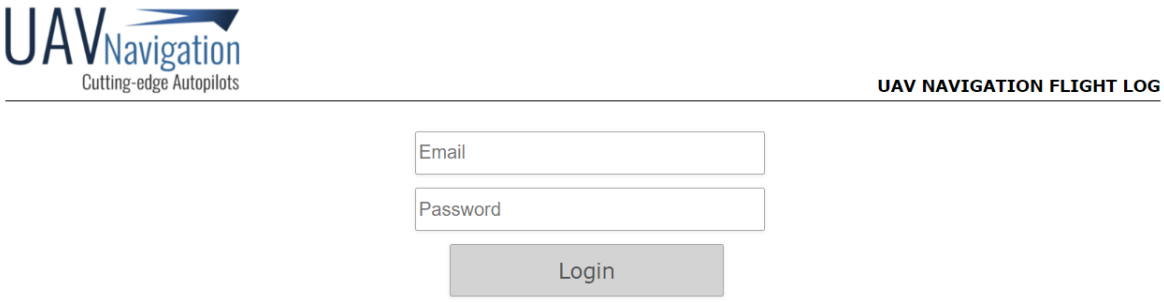

Figura 4-11 Aspecto de la página de login del Flight Log

Esta página desarrollada en HTML y JavaScript, se compone de un *header* superior que compartirá con la mayoría de páginas de esta herramienta y un simple formulario validado con código JavaScript en la parte del cliente compuesto por dos *imputs* y un botón 'Login' que mandará la información introducida a la clase que dará acceso al usuario tras validarlo con LDAP.

```
<form name="flform" action="connectLDAP.php" method="post" onsubmit="return validate()">

    <div class="row_menu">
        <div class = "mail_div">
            <input type="text" class="input_text1" name="usermail" placeholder = "Email" size="39">
        </div>
    </div>
    <div class="row_menu">
        <div class="pass_div">
            <input type="password" class="input_text1" name="password" placeholder = "Password" size="39">
        </div>
    </div>
    <div class="row_menu">
        <div class="button_div">
            <input type="submit" class="button_menu" name="button_send" value="Login">
        </div>
    </div>

</form>
```

Figura 4-12 Fragmento de código HTML de la página del Login del Flight Log

Tras el envío del formulario, en el archivo PHP, se realizan operaciones sobre el Servidor LDAP gracias a las funciones nativas que ofrece PHP.

4.5.1.1 Conexión con un Servidor LDAP

```
// LDAP Server Connection
$ldapconn = ldap_connect($ldap_server_ip)
      or die("Could not connect to LDAP server.");
```

ldap_connect() realiza una conexión con el Servidor LDAP y la mantiene abierta.

4.5.1.2 Obtención del usuario de LDAP mediante el email

```
// Obtaining user dn and CN by mail
$ldapmail = $_POST['usermail'];
$dn="ou=users,dc=x,dc=uavnavigation,dc=com";
$filter="(mail=$ldapmail)";
$sr=ldap_search($ldapconn, $dn, $filter);
$info = ldap_get_entries($ldapconn, $sr);
$ldaprdn = $info[0]["dn"];
```

Donde se especifica la ubicación de LDAP en la que ha de buscar ($dn) el usuario con el mail enviado ($ldapmail) con el filtro especificado ($filter).

Una vez obtenido el usuario en $info, a partir de este array, que devuelve todas las entradas que coinciden con el filtro especificado, puede obtenerse el *dn* del usuario ($ldaprdn), que será necesario a continuación para poder conocer si el usuario pertenece a algún grupo permitido o no.

4.5.1.3 Validar a un usuario en un grupo de LDAP

Para validar a un usuario cuyo *dn* está en la variable $ldaprdn se necesitaron dos funciones, la primera comprueba que la contraseña introducida por el usuario corresponde con el usuario obtenido, y la segunda confirma que el usuario pertenece a alguno de los grupos permitidos:

```
// Validates an user by password and group
function ldapConnectGroupRestriction($ldapconn, $ldaprdn, $ldappass, $ldapgrdn) {
      $ldapbind = ldap_bind($ldapconn, $ldaprdn, $ldappass);
      if ($ldapbind) {
            if (checkGroup($ldapconn, $ldaprdn, $ldapgrdn)){return 0;}
            else{return 1;}
      } else {return 2;}
}
```

Ldap_bind() recibe un *dn* y una contraseña a validar, devuelve true si la contraseña pertenece al usuario con el *dn* indicado.

```
// Validate an user in a group
function checkGroup($ldapconn, $ldaprdn, $ldapgrdn) {
      $attributes = array('members');
      $result = ldap_read($ldapconn, $ldaprdn, "(memberof={$ldapgrdn})", $attributes);
      if ($result === FALSE) { return FALSE; };
      $entries = ldap_get_entries($ldapconn, $result);
      return ($entries['count'] > 0);
}
```

Esta función, incluída en la anterior, obtiene los usuarios de LDAP que son miembros del grupo indicado (mediante el filtro "memberof={$ldapgrdn}") y que tengan el *dn* indicado en *$ldaprdn*. En *$ldapgrdn* se especifica el *dn* de alguno de los grupos permitidos, en ese caso son el *cn=FlightLog* y el *cn=FlightLog_Admin*.

Si la respuesta del Servidor LDAP contiene más de una entrada significa que el usuario que queremos validar existe en el grupo.

Una vez validado el usuario, la página nos redirigirá a la página principal del Flight Log.

4.5.2 DB del Flight Log

La DB del Flight Log se compone de una tabla principal *flight_logs* donde, en esencia, cada entrada es una hoja de vuelo. Esta tabla se divide a su vez en subtablas que componen cada uno de los apartados de una hoja de vuelo. A su vez, estas tablas beben de otras tablas para obtener los datos que se seleccionarán.

A continuación, se listan las columnas de la tabla principal y se proporciona una imagen de una hoja de vuelo antes de ser rellenada para poder asociar las columnas con los apartados de ésta.

- id: Índice y clave primaria de la tabla que identifica cada hoja de vuelo.
- flight_id: Referencia a una entrada de la tabla *flights* donde se indican los datos generales relacionados con el vuelo.
- uav_id: Referencia una entrada de la tabla *uavs* donde se indican los datos relacionados con el UAV.
- autopilot_id: Referencia una entrada de la tabla *autopilots* donde se indican los datos relacionados con el Autopiloto.
- gcs_id: Referencia una entrada de la tabla *gcs* donde se indican los elementos extras que se han utilizado en el vuelo.
- operator_id: Referencia una entrada de la tabla *operators* donde se indican los operadores del vuelo.
- notes_and_actions: Valor en formato de texto donde se añade información adicional del vuelo.
- flight_anomalies: Valor en formato de texto donde se informa de los problemas ocurridos.
- user_name: Mail del usuario que creó la hoja de vuelo.

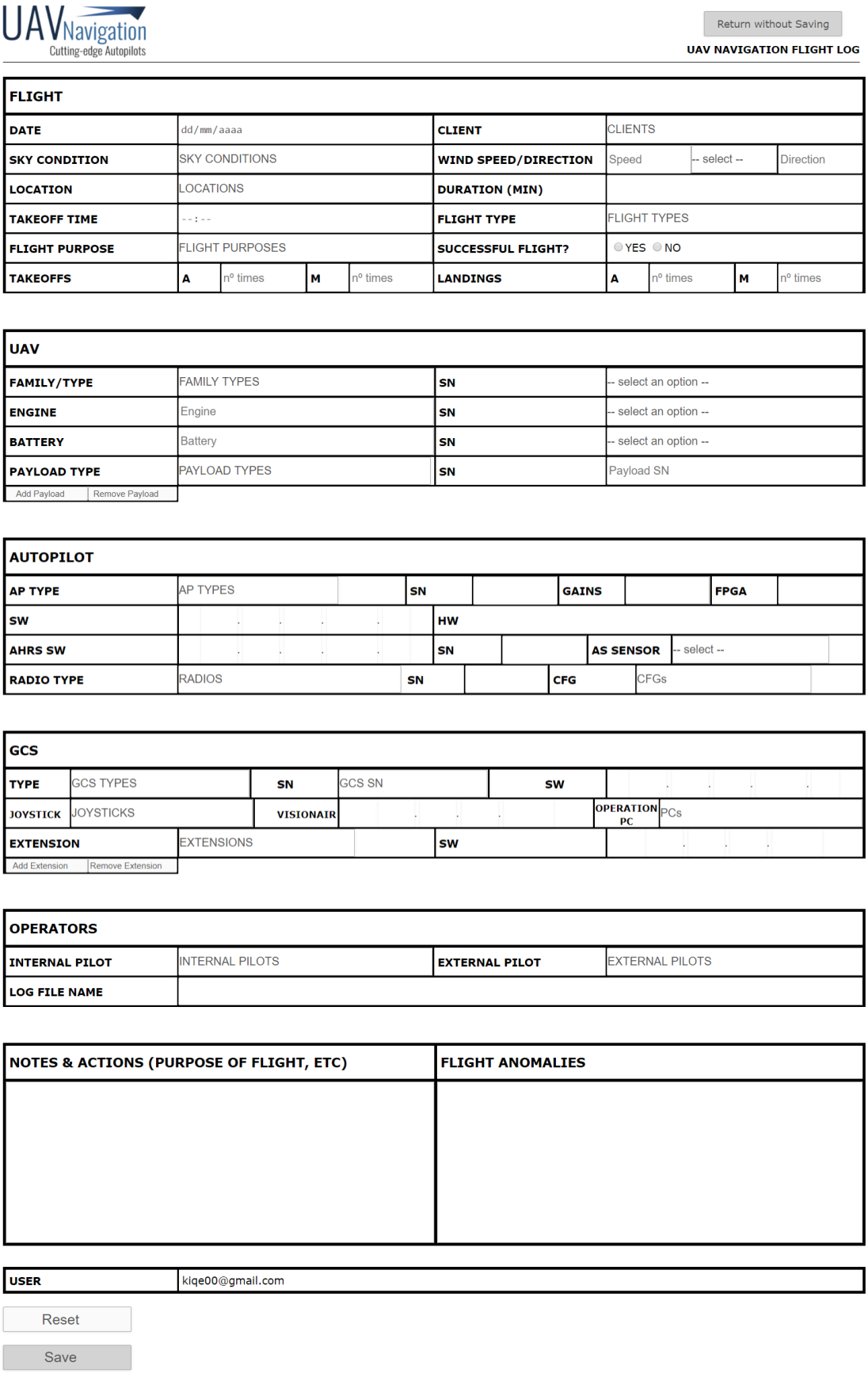

Figura 4-13 Hoja de Vuelo del Flight Log antes de ser rellenada

Para la comunicación entre la DB y el servidor se hace uso de las funciones de PHP, *mysqli*, para manejo de MySQL.

A continuación, se muestran ejemplos de uso con PHP.

4.5.2.1 Conexión con la DB

Para poder comunicarse con la DB del Flight Log, el servidor web debe establecer primero una conexión estable con ella. Para ello, se genera una conexión mediante PHP guardando ésta en un objeto del que más tarde se utilizarán funciones específicas para el manejo de la DB.

```
// DB connect
$mysqli = new mysqli($host, $user, $password, $db, $port);
if ($mysqli->connect_errno) {
    // Manage Connection Errors
}
```

Una vez ejecutada la sentencia anterior, donde *$host* indica la dirección IP donde se encuentra la DB, *$user* y *$password* son el usuario y contraseña de MySQL que tienen permisos sobre la DB *$db* y *$port* es el puerto de escucha del servidor MySQL instalado, por defecto 3306, ya se pueden realizar operaciones sobre la DB cómo se mostrará a continuación.

4.5.2.2 Operaciones sobre la DB

Una vez generada la conexión con la DB del Flight Log en la variable *$mysqli* ya podemos realizar operaciones. Para ello hacemos uso de las sentencias preparadas.

MySQL soporta sentencias preparadas, que tienen dos funcionalidades básicas. La primera es la de poder ejecutar muchas sentencias similares sobre distintas entradas ganando en eficiencia y la segunda es la seguridad a la hora de pasar parámetros a la DB obtenidos directamente desde los formularios de la web.

Las sentencias preparadas de MySQL tienen tres etapas:

4.5.2.2.1 Prepare

La etapa *prepare*, como bien dice el nombre, escribe una sentencia SQL para poder ser reutilizada más adelante. Es decir, genera el esqueleto de una consulta para poder ser rellenado fácilmente a posteriori con los parámetros recibidos.

La peculiaridad de esta función de *mysqli* es que la *query* generada puede contener parámetros variables. Por ejemplo, para obtener una entrada de la tabla *flight_logs* para poder imprimir una hoja de vuelo con id=1 se utilizaría la siguiente sentencia:

```
SELECT * FROM flight_logs WHERE id = 1;
```

Si se quisiera reutilizar esta sentencia SQL, pero con distintos id de la tabla, habría que reescribirla de nuevo, con la carga de trabajo pertinente para ello. Por ello, las sentencias preparadas permiten realizar una *query* 'genérica' y en siguientes etapas aplicar el valor de los parámetros.

La sentencia anterior preparada con PHP quedaría de la siguiente manera:

```
// MySQL Prepare Query
$stmt_select = $mysqli->prepare("SELECT * FROM flight_logs WHERE id = ?");
```

Nótese es símbolo de interrogación, que es el parámetro variable de la consulta. Del mismo modo se pueden añadir tantos parámetros variables como sean necesarios.

En la variable *$stmt_select* quedará almacenada esta sentencia preparada para ser usada en las etapas siguientes.

4.5.2.2.2 Bind_param y execute

En inglés, *bind param* significa enlazar parámetro. Es en esta etapa donde proveemos a la sentencia preparada de los parámetros precisos para que se realice, en este caso, la búsqueda.

Para enlazar un parámetro simplemente hay que indicar el tipo de parámetro que se va a insertar y el valor de éste. Para la sentencia anterior se quiere realizar la búsqueda para el id = 1, por tanto, el enlace de parámetros con PHP se hará de la siguiente manera:

```
// MySQL BindParam and Execute
$id = $_POST['flightlog_id'];
$stmt_select->bind_param('i', $id);
$stmt_select->execute();
$res = $stmt_select->get_result();
```

Como puede observarse, el tipo de parámetro que se pasa viene especificado por el caracter 'i', que indica que el tipo de parámetro ha de ser un entero, *bind_param()* acepta como tipo de dato el entero ya mencionado, double (d), string (s) y *blob* (b) que envía datos binarios en paquetes.

Si el parámetro no fuese un entero, entonces el enlazamiento del parámetro fallará. Aquí entra la segunda funcionalidad de realizar las sentencias preparadas, ya que se dificulta mucho el paso de cadenas de caracteres que puedan hacer que se vulnere la seguridad y produzca una obtención de datos ilícita por parte del servidor MySQL, como una sentencia DELETE dentro de la propia sentencia que produzca daños irreversibles.

Si la sentencia preparada a realizar contuviese más de un parámetro variable, los tipos de los parámetros se indicarían en el primer parámetro que recibe *bind_param()* de forma concatenada y manteniendo el orden de enlazamiento, y los valores de los parámetros a continuación en parámetros distintos:

```
// MySQL BindParam and Execute
$param1 = 'param1';
$param2 = 'param2';
$stmt_select->bind_param('ss', $param1, $param2);
```

Una vez realizado el enlace de parámetros se ejecuta la sentencia preparada con *execute()*. Para obtener los valores devueltos por la consulta será necesario aplicar *get_result()* sobre la sentencia preparada y se obtendrán los resultados en la variable *$res*.

Del mismo modo que se realizan las consultas MySQL pueden ejecutarse otras operaciones sobre la DB, a continuación se muestran ejemplos de sentencias de edición, creación y eliminación:

```
// MySQL Update Prepare Statement
$stmt_insert = $mysqli->prepare("UPDATE TABLE internal_pilots SET internal_pilot = ?
WHERE id = ?");
$stmt_insert->bind_param('si', $pilot, $id);
$stmt_insert->execute();
```

```
// MySQL Insert Prepare Statement
$stmt_insert = $mysqli->prepare("INSERT INTO internal_pilots VALUES (0, ?)");
$stmt_insert->bind_param('s', $pilot);
$stmt_insert->execute();
```

```
// MySQL Delete Prepare Statement
$stmt_insert = $mysqli->prepare("DELETE FROM internal_pilots WHERE id = ?");
$stmt_insert->bind_param('i', $id);
$stmt_insert->execute();
```

4.5.3 Esquema de la Web

Con lo comentado anteriormente operativo queda añadir las distintas funciones que puede realizar un usuario en la web sobre las hojas de vuelo y la gestión de la DB por parte de los usuarios con rol Administrador desde la misma y cómo éstas pueden llevarse a cabo.

4.5.3.1 MVC

Toda la web sigue el patrón de arquitectura MVC (Modelo-Vista-Controlador). Esto quiere decir que la lógica, los datos y la vista están separados.

Como se ha comentado en los apartados anteriores, la lógica, o controlador, que trabaja por parte del servidor se comunica con la DB y con el servidor LDAP, que son los datos con los que se nutre la web.

De igual modo, muchas de las páginas de la web contienen datos de la DB que son obtenidos por el controlador, escrito en código PHP, y mostrados en la parte del cliente gracias al código HTML y JavaScript que recibe.

Por otra parte, los datos introducidos en la parte del cliente por el usuario son mandados al controlador y éste se encarga de modificar los datos en las tablas pertinentes.

4.5.3.2 Vistas

Las vistas que ofrece el servidor al usuario están escritas en código *HTML* para la estructura de los elementos de la web, código *JavaScript* para el manejo de datos en cliente y los estilos fueron creados con *CSS* para dar formato a la web y las tablas y formularios que ésta alberga.

El menú principal tiene la siguiente vista:

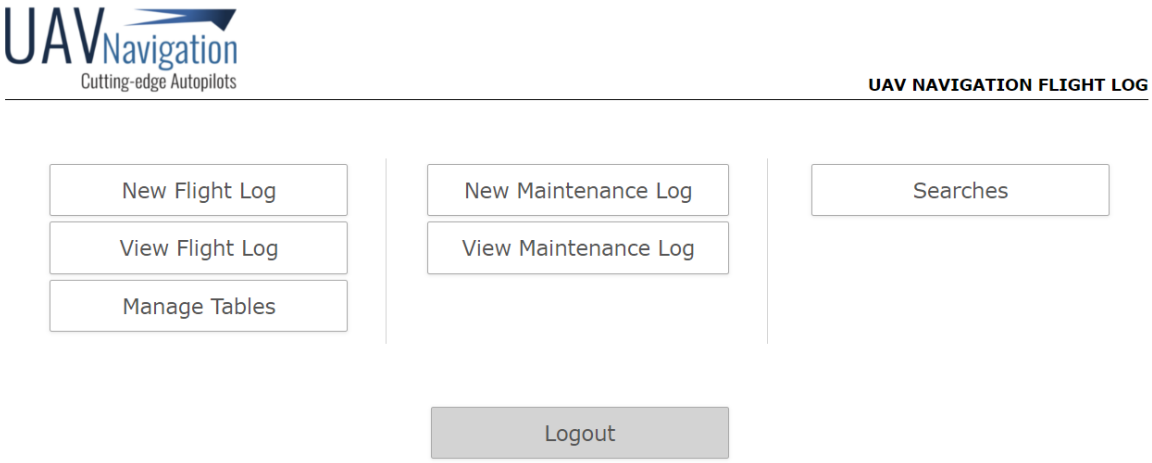

Figura 4-14 Vista del Menú Principal del Flight Log

El menú se divide en tres columnas. La primera columna ofrece la creación de una hoja de vuelo, poder visualizar aquellas hojas de vuelo almacenadas en la DB y la gestión de ciertas tablas de la DB desde la web, este último apartado está limitado a los usuarios Administrador. La segunda columna refiere a acciones de mantenimiento sobre los vehículos o herramientas. La tercera ofrece al usuario poder buscar estadísticas de vuelo sobre los datos de la DB del Flight Log.

4.5.3.2.1 Crear Hoja de Vuelo

La creación de una nueva hoja de vuelo se puede visualizar en la figura 4.13, donde tras rellenarla y guardarla, ésta es enviada al controlador y posteriormente insertada en la DB.

4.5.3.2.2 Visualizar Hoja de Vuelo

Esta pantalla permite visualizar todas las hojas de vuelo creadas en la DB, además, permite modificar sus datos, visualizar las modificaciones realizadas sobre ellas y eliminarlas.

Al acceder se muestra una tabla paginada que muestra el identificador, la fecha de creación y el usuario que creó la hoja de vuelo junto con las opciones a realizar sobre ellas.

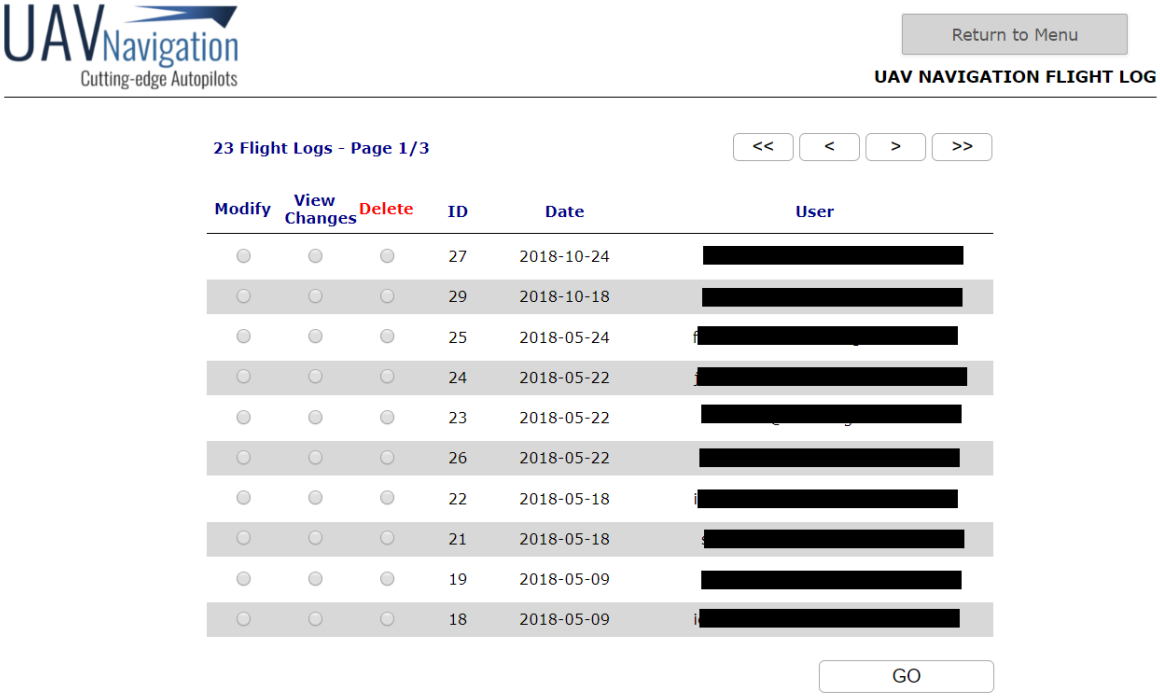

Figura 4-15 Visualizar Hojas de Vuelo

Pinchando sobre una de las hojas de vuelo se mostrará la hoja de vuelo con sus datos, con el formato adecuado para poder ser exportada con PDF, se aporta un ejemplo en el Anexo B.

Si se marcase alguna de las opciones seleccionables, tras pulsar sobre el botón 'go' se realizaría la acción requerida.

Al marcar 'Modify' se abrirá la hoja de vuelo disponible para ser modificada. Los cambios realizados en una hoja de vuelo se registran en un *log* de cambios, disponibles para ver marcando la opción 'View Changes'.

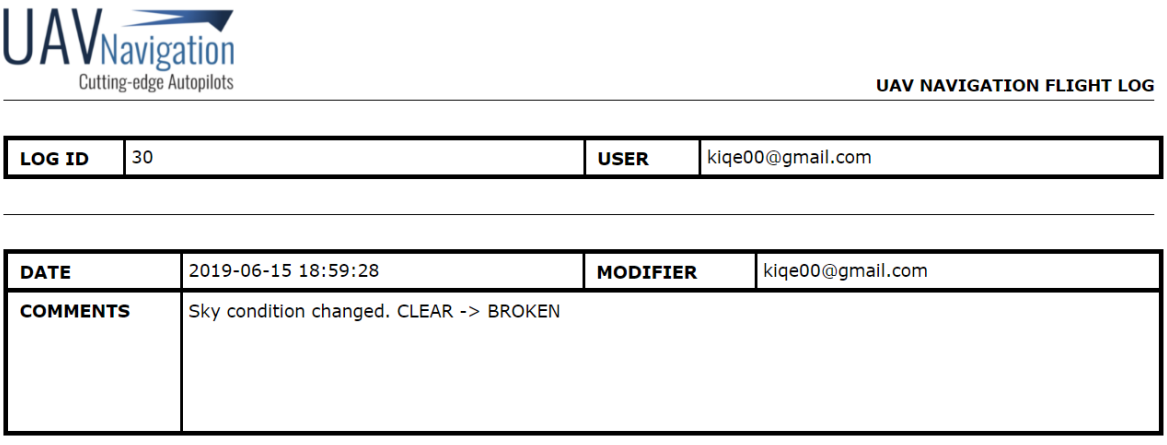

Figura 4-16 Log de Cambios de una Hoja de Vuelo

La opción ‘Delete’ eliminará por completo la hoja de vuelo junto con sus dependencias en la DB del Flight Log.

4.5.3.2.3 Gestión de Tablas

Varias tablas de la DB del Flight Log tienen entradas fijas de las que se nutren los formularios de la web. Estas tablas muchas veces se ven alteradas porque hay entradas que añadir o modificar.

Para ello existe la ventana de gestión de tablas, permite a los usuarios con rol administrador tratar los datos de ciertas tablas de forma sencilla sin necesidad de intervenir directamente con la DB. Se dispone de un desplegable con las posibles tablas a modificar.

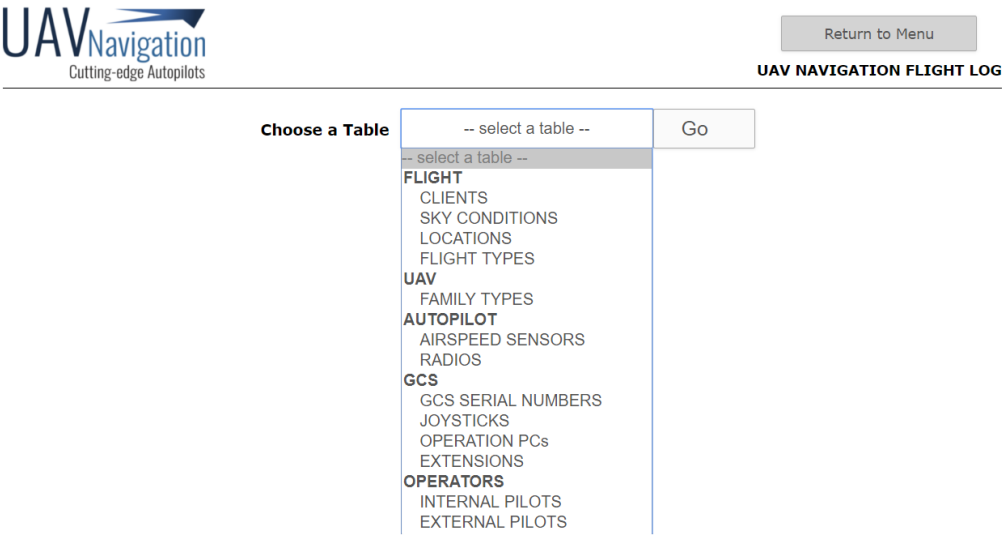

Figura 4-17 Gestión de Tablas – 1

Tras seleccionar una tabla y pulsar sobre ‘go’, se mostrarán los datos existentes en la tabla. Se puede añadir una nueva entrada, rellenando el campo ‘New Row Name’ y pulsando sobre ‘New’ o, se puede modificar una entrada seleccionándola, rellenando el campo ‘Update Row Name’ y pulsando el botón ‘Update’.

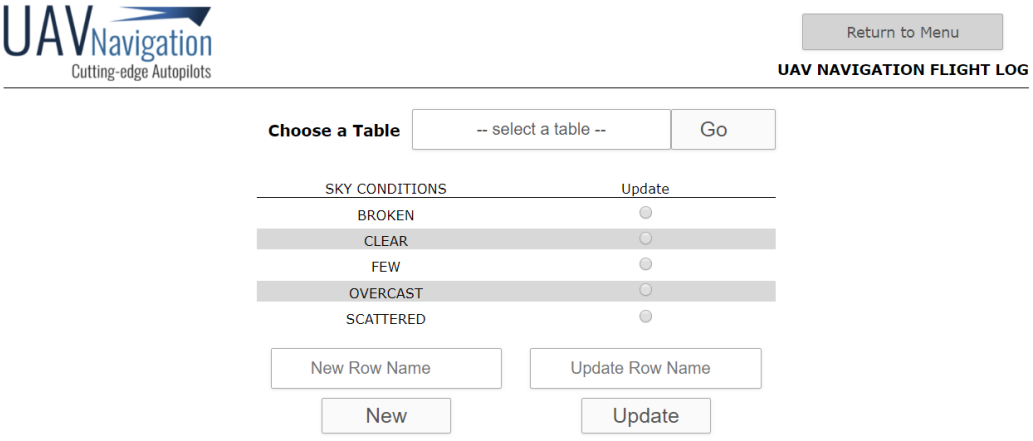

Figura 4-18 Gestión de Tablas – 2

4.5.3.2.4 Crear y Visualizar Mantenimientos

Siguiendo la arquitectura de creación y visualización de hojas de vuelo, se creó el apartado de mantenimiento, que permite almacenar las tareas de mantenimiento realizadas sobre los aparatos. Tras crear un mantenimiento, éste se visualiza accediendo a él desde una tabla en la que se listan aquellas hojas de mantenimiento almacenadas en DB.

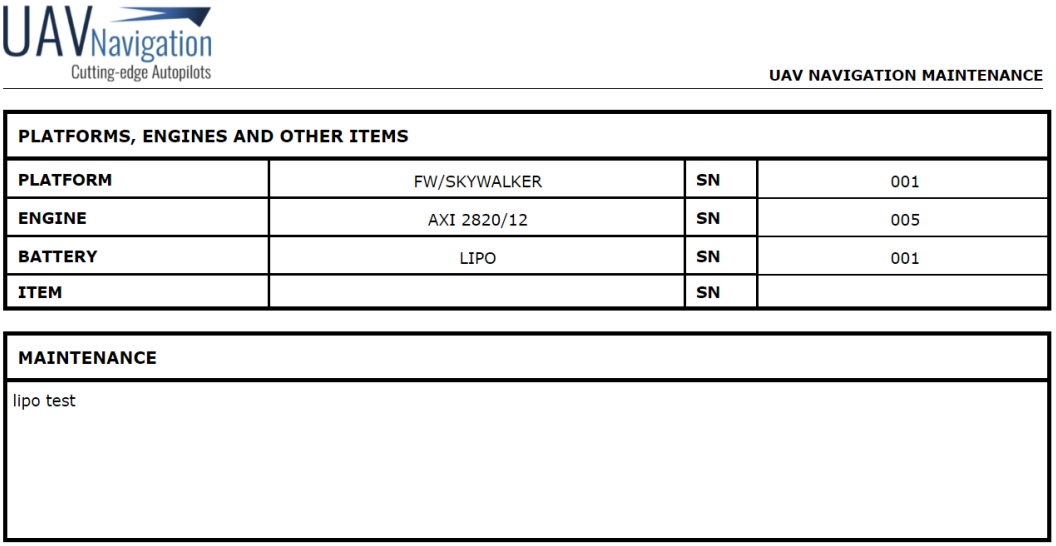

Figura 4-19 Visualización de una Hoja de Mantenimiento

4.5.3.2.5 Búsquedas

Por último, se creó una página destinada a realizar búsquedas acerca de las horas de vuelos realizadas por distintos pilotos, vehículos o elementos.

Al entrar en esta página se ofrecen los datos generales almacenados en la DB del Flight Log extraídos de las hojas de vuelo.

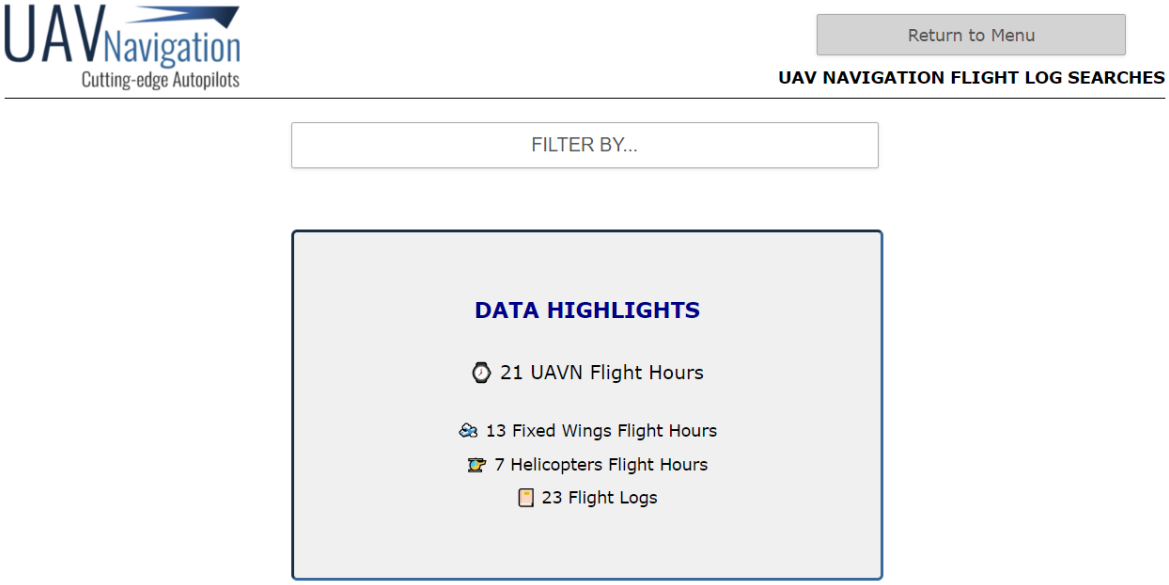

Figura 4-20 Búsquedas – 1

Existe un desplegable 'Filter by…' que permite obtener los resultados seleccionando el tipo de elemento. Al seleccionar el tupo de elemento 'Engines' se le muestra la siguiente vista al usuario. Aquí podrá elegir el tipo de motor así como su número de serie, y podrá optar por mostrar los datos entre dos fechas o los totales además de poder escoger el orden en el que se mostrarán los registros.

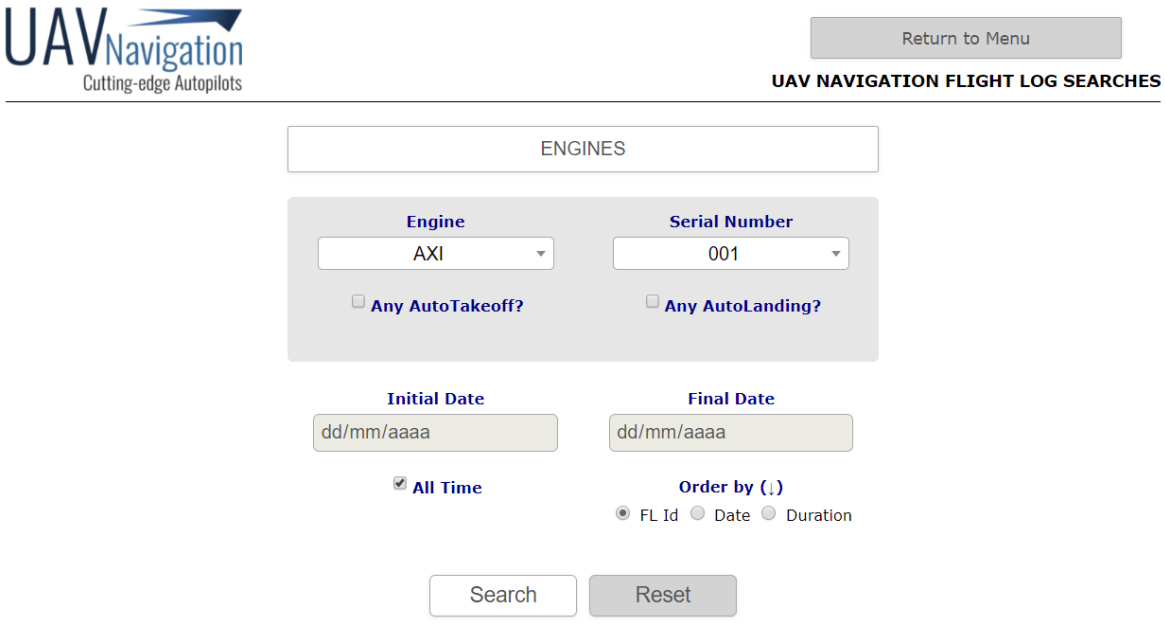

Figura 4-21 Búsquedas – 2

Tras rellenar el formulario y pulsar sobre 'Search', se mostrará una tabla bajo éste con las horas de vuelo realizadas por el motor con número de serie escogido y la lista de las hojas de vuelo que recogen esas horas de vuelo. Si se pulsara sobre una de las hojas de vuelo mostradas en la tabla se visualizaría la hoja de vuelo.

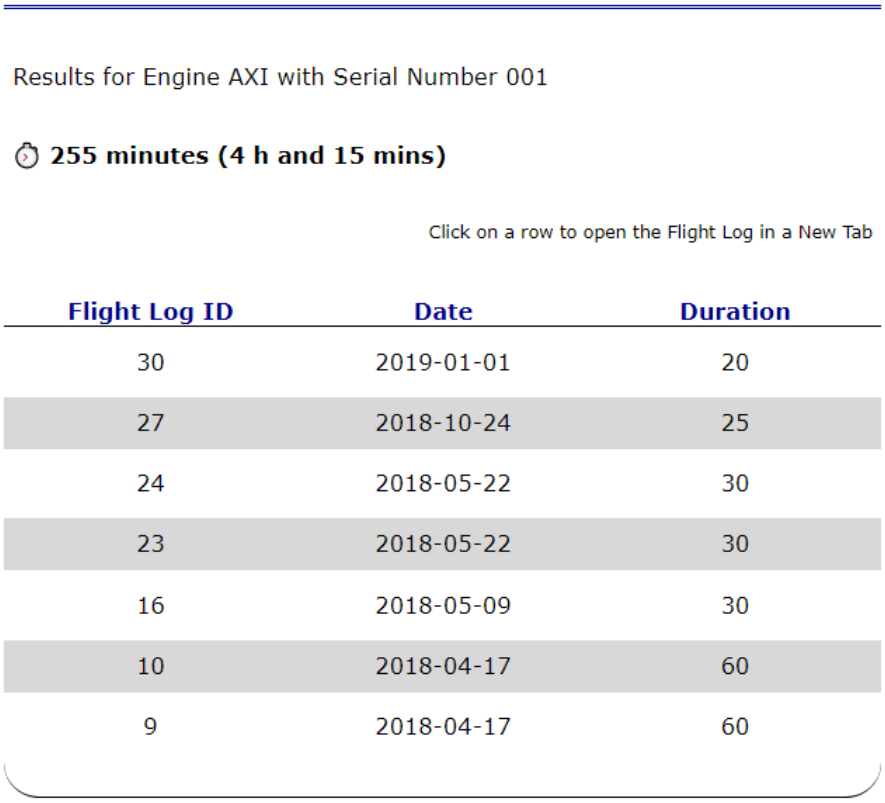

Figura 4-22 Búsquedas – 3

4.6 Respaldo del Servidor

Se realiza un *backup* diario mediante un script en Bash de aquellas DB del sistema; LDAP, Redmine y Flight Log, así como de los códigos propios de Redmine, del Flight Log, los scripts desarrollados para el servidor, y los archivos de configuración del servidor incluidos en */etc*.

Esta copia de seguridad, una vez creada, es comprimida, encriptada y almacenada en el servidor.

Para comprimir y encriptar la copia de seguridad se utilizan los siguientes comandos una vez almacenados todos los archivos necesarios en un mismo directorio:

```
# Compress
gzip data_to_backup

# Encrypt
openssl enc -aes-256-cbc -salt -in data_to_backup -out data_to_backup.enc -k keypassword
```

5 Pruebas

5.1 Pruebas de Login con LDAP

Una vez configurada la conexión para cada uno de los servicios con LDAP, se realizaron pruebas exhaustivas para asegurar el correcto funcionamiento del acceso de los usuarios:

- Usuarios no existentes en LDAP.
- Contraseñas distintas a las de LDAP, para Redmine es importante esta prueba, ya que Redmine almacena en Base de Datos los datos de acceso de los usuarios, y siempre debe hacer login con los datos de LDAP.
- Conexión con usuarios en Grupos LDAP sin acceso a un servicio, si un usuario no pertenece al grupo de acceso no puede acceder al servicio. Del mismo modo, si un usuario no pertenece a un grupo de acceso con un rol determinado, verá limitada su actividad con el servicio al que accede.
- Conexión SSH con usuarios limitados a SFTP.
- Conexión SFTP con usuarios enjaulados.

5.2 PHPUnit

PHPUnit es un Framework de *tests* unitarios para el lenguaje de programación PHP.
Con PHPUnit se realizan pruebas unitarias de ciertas funciones críticas de PHP desarrolladas para el Flight Log como aquellas que gestionan usuarios o recursos de la DB. De este modo, se asume que estas funciones no darán problemas de gestión de datos o seguridad.

5.3 SSLLabs

Se utiliza la página https://www.ssllabs.com/ para comprobar la calidad de la seguridad SSL de las páginas web del servidor, obteniendo la siguiente calificación:

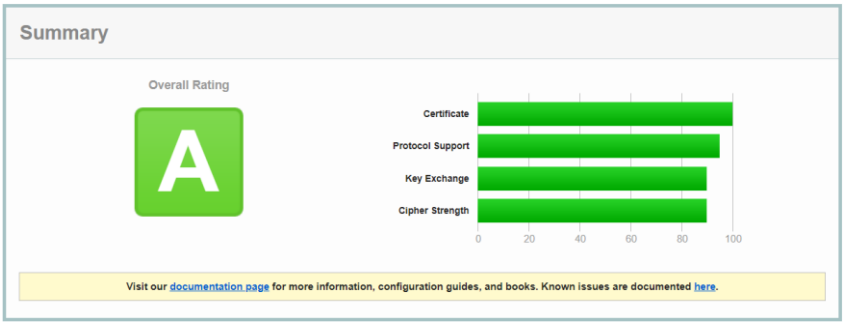

Figura 5-1 Resultado SSLLabs

6 Conclusiones y trabajo futuro

La elaboración de este proyecto consigue que la circulación de la información de UAVN sea más fluida y segura, abriendo la posibilidad de acceso a la información desde el exterior y la gestión de esta. Además, se abren nuevas vías de acción que harán que la productividad se eleve, reduciendo tiempos de búsqueda o de transmisión de información.

El diseño llevado desde el principio del proyecto permite la manipulación y evolución constante del servidor. Todos aquellos servicios que han sido integrados tienen posibilidad de actualizarse a las necesidades futuras y su mantenimiento no es costoso.

La integración de futuras nuevas herramientas, además, está garantizada y, si son compatibles con el servidor de autenticación, queda demostrado con esta memoria que una vez se configura LDAP para ello es sencillo de acoplar nuevas herramientas sin necesidad de modificar el resto de las herramientas o servicios que se albergan en el mismo servidor y la gestión del acceso una vez configurado está muy centralizada y es fácil de gestionar.

Referencias

[1] Ubuntu Members *"Ubuntu Server Guide"*
https://help.ubuntu.com/16.04/serverguide/serverguide.pdf

[2] Imobach González Sosa y Manolo Padrón Martínez *"PAM Documentation"*
http://sopa.dis.ulpgc.es/ii-aso/portal_aso/leclinux/seguridad/pam/pam_doc.pdf

[3] Linux Man Pages *"access.conf"*
https://www.systutorials.com/docs/linux/man/5-access.conf/

[4] Linux Man Pages *"sshd_config"*
https://linux.die.net/man/5/sshd_config

[5] OpenLDAP Manual Pages *"ldapadd"*
http://www.openldap.org/software/man.cgi?query=ldapadd&apropos=0&sektion=0&manpath=OpenLDAP+2.0-Release&format=html

Glosario

AS	Servidor de Autenticación (*Authentication Service*)
Backup	Copia de Seguridad
CA	Autoridad Certificadora (*Certification Authority*)
Cloud Computing	Computación en la Nube. Paradigma que permite ofrecer servicios de computación a través de una red.
Cloud Server	Servidor en la Nube. Servidor virtual basado en *cloud computing* y generado por medio de un sistema operativo virtualizador de alto rendimiento asignando los recursos al servidor de manera dedicada y exclusiva.
Código Fuente	Conjunto de líneas de texto con los pasos que debe seguir la computadora para ejecutar dicho programa.
DB	Base de Datos (*Database*).
De dominio específico	Lenguaje de programación dedicado a resolver un problema en particular, representar un problema específico y proveer una técnica para solucionar una situación particular.
De propósito general	Lenguaje de programación que puede ser usado para varios designios.
Débilmente tipado	Lenguaje de programación en el que se presta poca atención a la definición estricta de los tipos de datos.
DNS	Sistema de Nombres de Dominio (*Domain Name System*).
DS	Servicio de Directorio (*Directory Server*).
Flight Log	Hoja de registro de un vuelo realizado por un piloto.

FTP	Protocolo de Transferencia de Archivos (*File Transfer Protocol*).
FTPS	FTP sobre SSL/TLS.
GNU/Linux	Familia de Sistemas Operativos basados en el proyecto GNU (*GNU is Not Unix*) sobre el núcleo o *kernel* de Linux.
GUI	Interfaz Gráfica de Usuario (*Graphical User Interface*).
Herramienta Web	Medio que facilita la comunicación y el diseño de los contenidos web.
HTTP	Protocolo de Transferencia de Hipertexto (*Hypertext Transfer Protocol*).
HTTPS	Protocolo Seguro de Transferencia de Hipertexto (*Hypertext Transport Protocol Secure*).
Imperativo (Lenguaje)	Lenguaje que describe la programación en términos del estado del programa y sentencias que cambian dicho estado.
Interpretado (Lenguaje)	Lenguaje que consiste en scripts que son interpretados en tiempo real por un intérprete y que por tanto no requieren de un compilador.
IP	Número que identifica a una interfaz en red de un dispositivo que utilice el protocolo de internet.
Kernel	Núcleo. Software que constituye una parte fundamental del sistema operativo.
Login	proceso mediante el cual se controla el acceso individual a un sistema informático mediante la identificación del usuario utilizando credenciales provistas por el usuario.
Multiparadigma (Lenguaje)	Lenguaje que permite crear programas usando más de un estilo de programación.

Opencode	Modelo de desarrollo del software basado en la colaboración abierta con el objetivo de poder modificar la fuente sin restricciones de licencias.
PAM	Pluggable Authentication Modules
POSIX	Interfaz de Sistema Operativo Portable Unix (Portable Operating System Interface Unix)
Reflexivo (Lenguaje)	Lenguaje que dota a un programa capacidad para observar y opcionalmente modificar su estructura de alto nivel.
Servicio Web	Sistema de software designado para dar soporte a la interacción de máquina a máquina interoperativa a través de una red.
SFTP	*SSH File Transfer Protocol*
SO	Sistema Operativo. Conjunto de órdenes y programas que controlan los procesos básicos de una computadora y permiten el funcionamiento de otros programas.
SQL	*Structured Query Language*
SSH	Protocolo *Secure Shell*
SSL/TLS	Protocolos de seguridad para las comunicaciones en una red. Capa de puertos seguros (*Secure Sockets Layer*) y seguridad de la capa de transporte (*Transport Layer Security*), respectivamente.
UAV	Vehículo Aéreo no Tripulado (*Unmanned Aerial Vehicle*)
UAVN	UAV Navigation.
Virtualización	Creación a través de software de una versión virtual de algún recurso tecnológico.

Anexos

A BOE - Responsabilidades en Materia de Mantenimiento

cve: BOE-A-2017-15721 verificable en http://www.boe.es

BOLETÍN OFICIAL DEL ESTADO
Núm. 316 Viernes 29 de diciembre de 2017 Sec. I. Pág. 129621

Sección 3.ª Mantenimiento

Artículo 16. *Responsabilidades en materia de mantenimiento.*

1. El fabricante de una aeronave pilotada por control remoto (RPA) o, en su caso, el titular de su certificado de tipo deberá elaborar y desarrollar un manual o conjunto de manuales que describan su funcionamiento, mantenimiento e inspección. Estos manuales deberán incluir directrices para realizar las tareas necesarias de inspección, mantenimiento y reparación a los niveles adecuados y específicos de la aeronave y sus sistemas asociados (RPAS), y deberán proporcionarse al operador junto con la aeronave.

2. El operador es responsable del mantenimiento y la conservación de la aeronavegabilidad, debiendo ser capaz de demostrar en todo momento que la aeronave pilotada por control remoto (RPA) y sus sistemas asociados conservan las condiciones de aeronavegabilidad con las que fueron fabricados. Además, el operador deberá cumplir con cualquier requisito de mantenimiento de la aeronavegabilidad declarado obligatorio por la Agencia Estatal de Seguridad Aérea.

A estos efectos, el operador deberá establecer un sistema de registro de los datos relativos a:

a) Los vuelos realizados y el tiempo de vuelo.
b) Las deficiencias ocurridas antes de y durante los vuelos, para su análisis y resolución.
c) Los eventos significativos relacionados con la seguridad.
d) Las inspecciones y acciones de mantenimiento y sustitución de piezas realizadas.

En todo caso, el mantenimiento y las reparaciones que procedan deberán realizarse siguiendo las directrices del fabricante o, en su caso, del titular del certificado de tipo RPA.

BOE Responsabilidades en Materia de Mantenimiento

Documento del Boletín Oficial del Estado del que surge la necesidad del servicio web UAVN Flight Log.

B Flight Log - Hoja de Vuelo

UAV Navigation
Cutting-edge Autopilots

UAV NAVIGATION FLIGHT LOG

FLIGHT									
DATE	2019-01-01				**CLIENT**	AAA			
SKY CONDITION	CLEAR				**WIND SPEED/DIRECTION**	2 M/S		20°	
LOCATION	MADRID				**DURATION (MIN)**	20			
TAKE OFF TIME	09:00:00				**FLIGHT TYPE**	DEMO			
FLIGHT PURPOSE	FLIGHT TEST				**SUCCESSFUL FLIGHT?**	yes			
TAKEOFFS	**A**	2	**M**	1	**LANDINGS**	**A**	2	**M**	1

UAV			
FAMILY/TYPE	MULTI-ROTOR	**SN**	001
ENGINE	ENGINE V1	**SN**	001
BATTERY	BATTERY V1	**SN**	
PAYLOAD TYPE	RADAR/SMART MICRO	**SN**	001

AUTOPILOT							
AP TYPE	AP TYPE	**SN**	1	**GAINS**	1	**FPGA**	1
SW	1.1.1.1[a]	**HW**		HWV1			
AHRS SW	1.1.1.1[b]	**SN**	1	**AS SENSOR**	L		
RADIO TYPE	FREEWAVE	**SN**	1	**CFG**	CFG		

GCS					
TYPE	GCS	**SN**	99	**SW**	1.1.1.1[a]
JOYSTICK	102	**VISIONAIR**	1.1.1.1	**OPERATION PC**	FLIGHT CONTROL PC
EXTENSION				**SW**	

OPERATORS			
INTERNAL PILOT		**EXTERNAL PILOT**	
LOG FILE NAME			

NOTES & ACTIONS (PURPOSE OF FLIGHT, ETC)	**FLIGHT ANOMALIES**
OK!	No.

LOG ID	30	**USER**	kiqe00@gmail.com