

Proyecto Plataforma De Gestión Corporativa de Usuarios

Manual Técnico



Alumno: Carlos Márquez Calero
Curso: ASIR 2

ÍNDICE:

1. Introducción a este Proyecto:	2
2. Objetivo de este Proyecto:	2
3. Estructura General del Proyecto:	2
4. Diseño de la app web:	3
4.1 Estructura interna y diseño del contenedor Apache-php:	3
4.2 Estructura interna y diseño del contenedor Openldap:	5
4.3 Estructura y Diseño de el fichero Docker-Compose:	6
5. Despliegue de Infraestructura y App Web:	9
5.1 Herramientas necesarias para la creación de la infraestructura:	9
1. Despliegue en docker-desktop:	9
6. Funciones de la App Web:	11
1. Creación de Usuarios:	11
2. Listar Usuarios en el Servidor LDAP:	13
3. Creación de Grupos:	13
4. Listar Grupos:	15
5. Crear usuarios mediante CSV:	15
6. Revisión de logs:	17
7. Hacer Copias de Seguridad (Idif):	17
7. Explicación de Código Destacado:	18
1. Organización del código:	18
2. Explicación de código a destacar:	22
8. Conclusión:	23
9. Mejoras a Futuro:	24
10. WebGrafía y enlaces de Interés:	24

4. Diseño de la app web:

4.1 Estructura interna y diseño del contenedor Apache-php:

En cuanto a la estructura del contenedor apache cabe destacar que este contenedor cuenta con un **Dockerfile** en el cual se descargan e instala todas las dependencias necesarias para su correcto funcionamiento en conjunto al contenedor Openldap:

```
# Instalar dependencias del sistema y extensiones PHP necesarias para LDAP
RUN apt-get update && apt-get install -y \
    libldap2-dev \
    libzip-dev \
    unzip \
    libsasl2-dev \
    libssl-dev \
    && docker-php-ext-configure ldap \
    && docker-php-ext-install ldap zip \
    && a2enmod rewrite \
    && apt-get clean \
    && rm -rf /var/lib/apt/lists/*
```

Dentro de este Dockerfile se usa una imagen de Apache con php ya instalado, en concreto **php:8.3-apache**:

```
# Usar la imagen base de PHP con Apache
FROM php:8.3-apache
```

Además de las dependencias ya instaladas anteriormente, descargamos e instalamos [docker.io](https://docs.docker.com/engine/install/) y [docker.compose](https://docs.docker.com/compose/install/). Esto será de gran ayuda más adelante para usar ciertos controladores de la app:

```
# Actualización de Dependencias (De nuevo)
RUN apt-get update

RUN apt-get upgrade -y

# Permisos para script copia de seguridad
RUN apt-get install docker.io -y

RUN apt-get install docker-compose -y
```

Por último copiamos el archivo de configuración de nuestra app web, la cual tiene la siguiente configuración:

```
#Fichero de la app de gestion
#Escucha las peticiones que vengan por el puerto 80 desde cualquier dirección de red
<VirtualHost *:80>
    #Ruta donde se alojará el documento index.html o index.php de la app
    DocumentRoot /var/www/html/

    #Bloque de configuración de la ruta de la app web
    <Directory /var/www/html>
        #Muestra un listado de archivos si no hay index.php o .html
        #Por otro lado permite seguir enlaces simbolicos dentro del directorio
        Options Indexes FollowSymLinks
        #Permite cualquier petición de cualquier cliente IP
        Require all granted
    </Directory>

    #Ruta del archivo donde se registrarán posibles errores de la app
    ErrorLog ${APACHE_LOG_DIR}/error.log
    #Ruta del archivo donde se registrará todos los accesos a la app
    CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

Exponemos el **puerto 80** del contenedor fuera de este para poder acceder a la app web desde fuera y asignamos los permisos necesarios para acceder a el directorio “/var/www/html”:

```
# Establecer permisos correctos
RUN chown -R www-data:www-data /var/www/html/
RUN chmod -R 755 /var/www/html/

# Copiar configuración del vhost si la tienes
COPY app_gestion.conf /etc/apache2/sites-available/000-default.conf

# Exponer puerto 80
EXPOSE 80
```

4.2 Estructura interna y diseño del contenedor Openldap:

En cuanto a la estructura interna del contenedor Openldap podemos destacar el uso de un fichero Dockerfile, el cual contiene la configuración necesaria para poder hacer un uso correcto de este contenedor. La imagen usada para este Dockerfile es **osixia/openldap:1.5.0**:

```
FROM osixia/openldap:1.5.0
```

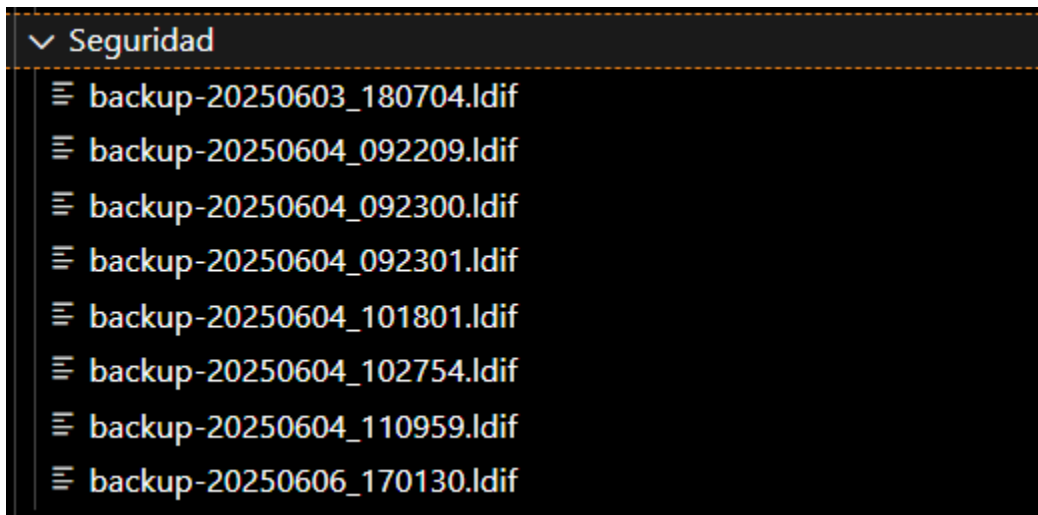
Además de contener la imagen de Openldap, este Dockerfile contiene una línea la cual copia el contenido del subdirectorio ldif para una vez se despliegue el contenedor, se despliegue la configuración de nuestro dominio:

```
ldap > ldif > 01-nuevo.ldif
1  # Unidad organizativa de usuarios
2  dn: ou=Usuarios,dc=carlete,dc=sl
3  objectClass: organizationalUnit
4  ou: Usuarios
5
6  # Unidad organizativa de grupos
7  dn: ou=Grupos,dc=carlete,dc=sl
8  objectClass: organizationalUnit
9  ou: Grupos
10
11 # Usuario Carlos
12 dn: cn=carlos,ou=Usuarios,dc=carlete,dc=sl
13 objectClass: inetOrgPerson
14 cn: carlos
15 sn: Martinez
16 uid: carlos
17 userPassword: carlos123
18
19 # Grupo de administradores
20 dn: cn=admins,ou=Grupos,dc=carlete,dc=sl
21 objectClass: groupOfNames
22 cn: admins
23 member: cn=administrador,ou=Usuarios,dc=carlete,dc=sl
24
25 # Usuario administrador
26 dn: cn=administrador,ou=Usuarios,dc=carlete,dc=sl
27 objectClass: inetOrgPerson
28 cn: administrador
29 sn: Principal
30 uid: administrador
31 userPassword: admin123
32
```

(Fichero .ldif con la configuración inicial del dominio)

```
# Copiar LDIFs personalizados al directorio de carga inicial de OpenLDAP  
COPY ./ldif/ /container/service/slapd/assets/config/bootstrap/ldif/custom/
```

Por último cabe destacar la presencia de un subdirectorío el cual está montado dentro del contenedor Openldap, este se encarga de guardar los ldifs que se crean a modo de copia de seguridad del dominio:



4.3 Estructura y Diseño de el fichero Docker-Compose:

El fichero docker-compose es un fichero escrito en un lenguaje yaml, un lenguaje creado y diseñado para que sea legible para cualquier tipo de usuario. La versión usada para la ejecución de la infraestructura completa dentro de el fichero, es la “**version 3.3**”:

```
version: "3.3"
```

Además de especificar la versión que se usará de docker-compose, también especificamos la sección de servicios o services (en inglés). Dentro de esta sección se encuentra toda la info específica sobre la configuración externa de los dos contenedores usados en este proyecto.

El primer contenedor que se muestra en la sección services es el contenedor Openldap:

```
▷Run All Services
services:
  ▷Run Service
  ldap:
    build:
      context: ./ldap
      dockerfile: Dockerfile
    container_name: openldap
    env_file:
      - .env
    ports:
      - "389:389"
    volumes:
      - ./ldap/Seguridad/:/ldap/Seguridad/
    networks:
      red_gestion:
        ipv4_address: 192.168.10.2
```

Entre la configuración de este contenedor cabe destacar, el uso de un Dockerfile (el cual como se ha explicado en puntos anteriores, contiene la configuración interna del contenedor, que necesita para ser desplegado y funcionar correctamente), especificando que este se encuentra al mismo nivel que el docker-compose dentro del directorio ldap.

Además se especifica un nombre, en este caso “**Openldap**”, un fichero .env que contiene variables de entorno, una redirección de puertos, punteando el puerto 389 de mi máquina física con el del contenedor. Un volumen, el cual se encarga de guardar las copias de seguridad y una red fija dentro de el direccionamiento interno de docker.

El segundo contenedor especificando en el apartado services es el contenedor Apache-php:

```
► Run Service
apachephp:
  build:
    context: ./apache-php
    dockerfile: Dockerfile
  container_name: apache-php
  ports:
    - "80:80"
  volumes:
    - ./public:/var/www/html
    - //var/run/docker.sock:/var/run/docker.sock
  networks:
    red_gestion:
      ipv4_address: 192.168.10.10
  depends_on:
    - ldap
```

Al igual que el contenedor Openldap, contamos con el uso de un Dockerfile encargado de cargar la configuración interna, una redirección de puertos que puntea el puerto 80 físico con el del contenedor, necesario para usar el método http y poder acceder a la app web. Dos volúmenes compartidos, en uno pintaremos la carpeta donde se encuentra todo el código de la app web y en el otro punteamos el socket de docker para poder usar comandos docker dentro del contenedor.

En cuanto a la red, el contenedor dispone (al igual que el contenedor openldap) de una dirección ip fija dentro de la red interna de docker. Por último para que ese contenedor funcione depende directamente del contenedor ldap.

5. Despliegue de Infraestructura y App Web:

En este punto explicaremos cómo configurar la infraestructura y cómo desplegar dentro de esta la app web de gestión de usuarios ldap.

5.1 Herramientas necesarias para la creación de la infraestructura:

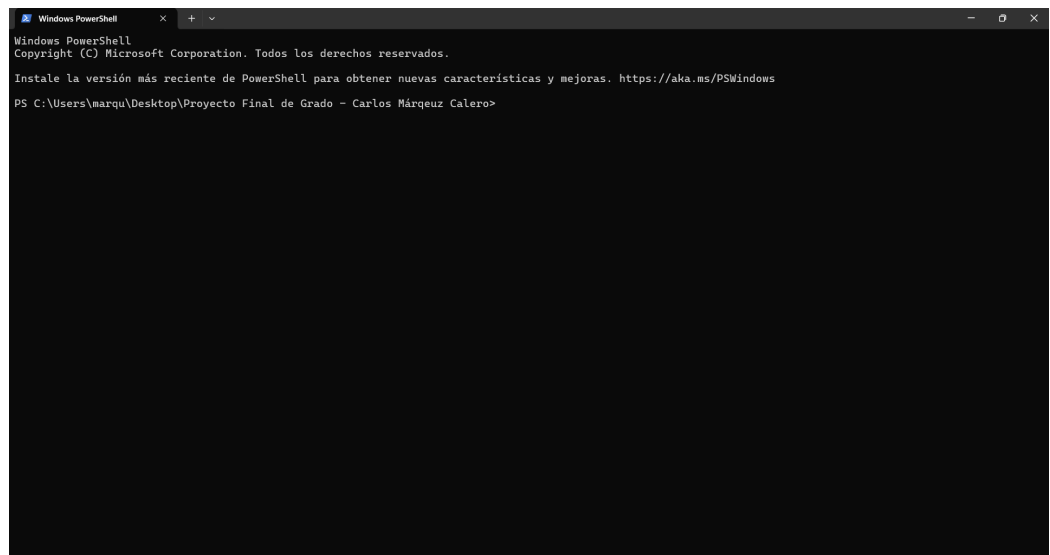
Para crear la infraestructura donde desplegamos la app web de gestión de usuarios ldap, necesitaremos las siguientes herramientas:

- **Docker-Desktop:** Esta herramienta será útil para poder desplegar nuestro proyecto en local y poder acceder mediante una interfaz de red puente desde el cliente de ubuntu hacia la app.

5.2 Proceso de Despliegue de la infraestructura:

1. Despliegue en docker-desktop:

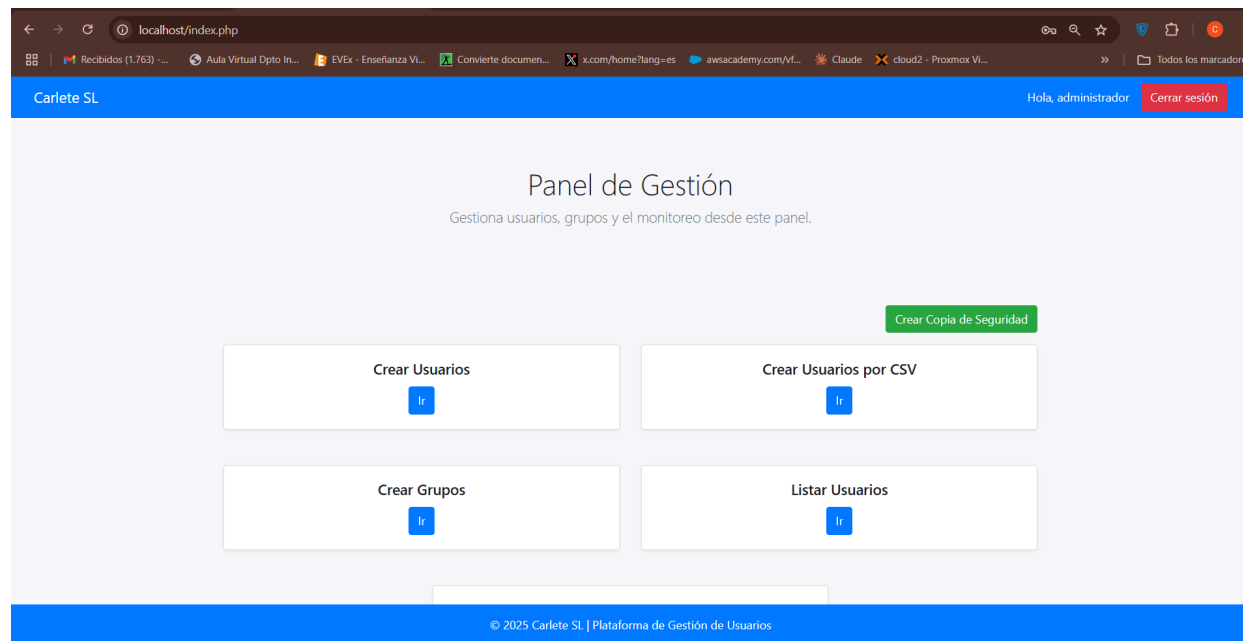
Una vez tengamos docker-desktop, lo único que necesitaremos es dirigirnos a la carpeta donde se encuentra nuestras app y abrir una sesión de terminal:



Una vez abierta usaremos el comando “docker-compose up -d” para levantar la infraestructura:

```
PS C:\Users\marqu\Desktop\Proyecto Final de Grado - Carlos Márquez Calero> docker-compose up -d
time="2025-06-09T12:16:25+02:00" level=warning msg="C:\\Users\\marqu\\Desktop\\Proyecto Final de Grado - Carlos Márquez Calero\\docker-compose.yml: the attribute 'version' is obsolete, it will be ignored, please remove it to avoid potential confusion"
[+] Running 3/3
✔ Network projectofinaldegrado-carlosmrquezcalero_red_gestion Created 0.1s
✔ Container openldap Started 0.5s
✔ Container apache-php Started 0.8s
PS C:\Users\marqu\Desktop\Proyecto Final de Grado - Carlos Márquez Calero> docker ps
CONTAINER ID   IMAGE                                     COMMAND                  CREATED        STATUS        PORTS
AMES          3857414ad010   projectofinaldegrado-carlosmrquezcalero-apachephp   "docker-php-entrypoi..." 6 seconds ago  Up 5 seconds  0.0.0.0:80->80/tcp
a             2a99bdaedde1   projectofinaldegrado-carlosmrquezcalero-ldap        "/container/tool/run"     6 seconds ago  Up 5 seconds  0.0.0.0:389->389/tcp, 636/tcp
benldap
PS C:\Users\marqu\Desktop\Proyecto Final de Grado - Carlos Márquez Calero>
```

Comprobamos que la infraestructura está operativa en local:



Y listo ya estaría nuestra infraestructura desplegada y lista para ser usada.

6. Funciones de la App Web:

En este punto explicaré al detalle todas las funciones y utilidades que dispone la app de gestión de usuarios vía LDAP

1. Creación de Usuarios:

Una de las utilidades de la cual dispone la app web de gestión de empresa, es la de crear usuarios en el servidor LDAP, para ello haremos uso de la vista llamada “formulario_usuarios.php”, mediante el uso de un botón llamado “Crear Usuarios”:

The screenshot displays a web application interface for creating users. At the top, a light blue box contains the text 'Crear Usuarios' in bold, with a blue button labeled 'Ir' below it. Below this, a blue header bar shows 'Carlete SL' on the left and 'Hola, administrador' with a 'Cerrar sesión' button on the right. The main content area is titled 'Crear Nuevo Usuario LDAP' and includes the instruction 'Rellena los campos para registrar un nuevo usuario en el servidor LDAP.' The form consists of several input fields: 'Nombre de usuario (UID)' with a note 'Solo letras minúsculas y números, sin espacios.', 'Nombre completo (CN)', 'Apellidos (SN)', 'Correo Electrónico' with a note 'Ingresa una dirección de correo válida.', 'Contraseña' with a note 'Mínimo 6 caracteres.', 'Grupo' with a dropdown menu labeled 'Seleccionar grupo (opcional)' and a note 'Selecciona el grupo al que pertenecerá el usuario.', and 'Unidad Organizativa (OU)' with a dropdown menu labeled 'Usuarios'. A footer bar at the bottom reads '© 2025 Carlete SL | Plataforma de Gestión de Usuarios'.

El funcionamiento es simple, para crear un usuario debemos introducir un uid(identificador de usuario), un nombre, unos apellidos, una cuenta de correo, una contraseña, un grupo (opcional) y especificar la unidad organizativa. Ahora mostraré un ejemplo:

Crear Nuevo Usuario LDAP

Rellena los campos para registrar un nuevo usuario en el servidor LDAP.

Nombre de usuario (UID)

Solo letras minúsculas y números, sin espacios.

Nombre completo (CN)

Apellidos (SN)

Correo Electrónico

Ingresa una dirección de correo válida.

Contraseña

Mínimo 6 caracteres.

Grupo

Selecciona el grupo al que pertenecerá el usuario.

Unidad Organizativa (OU)

Por defecto es "Usuarios", cambia si usas otra OU.

[Crear Usuario](#) [Cancelar](#)

Una vez le demos a crear, nos deberá salir un mensaje de verificación mostrando que el usuario se a añadido al servidor ldap:

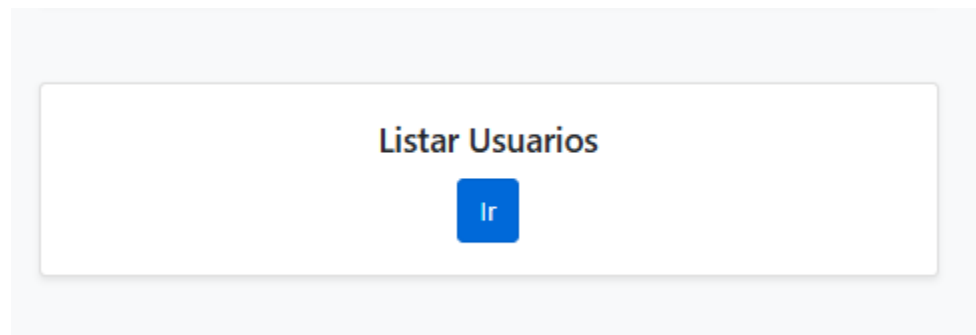
Crear Nuevo Usuario LDAP

Rellena los campos para registrar un nuevo usuario en el servidor LDAP.

Usuario 'x611188' creado correctamente.

2. Listar Usuarios en el Servidor LDAP:

Otra de las funcionalidades de las que dispone la app web de gestión, es la de listar usuarios del sistema, para ello en el panel de gestión nos dirigiremos a la opción que pone “listar usuarios”:



Una vez le demos se nos mostrará una tabla con los diferentes usuarios que existen en el servidor LDAP.

A screenshot of a web application interface showing a table of users. The table has five columns: ID Usuario, Nombre, Grupo, Correo Electrónico, and Opciones. There are three rows of user data. The interface includes a blue header bar with 'Carlote SL' on the left and 'Hola, administrador' and 'Cerrar sesión' on the right. Below the table is a 'Volver al Panel' button.

ID Usuario	Nombre	Grupo	Correo Electrónico	Opciones
carlos	carlos	Sin grupo	N/A	Eliminar
administrador	administrador	Sin grupo	N/A	Eliminar
x611188	Carlos	Sin grupo	marquezcalerocarlos@gmail.com	Eliminar

[Volver al Panel](#)

Adicionalmente, hay una opción para eliminar usuarios del servidor si fuese necesario.

3. Creación de Grupos:

La creación de grupos es otra de las utilidades de las que dispone la plataforma de gestión de usuarios ldap, esta se encarga de crear objetos dentro de la unidad organizativa de “Grupos”, al igual que lo hacía la creación de usuarios dentro de la unidad organizativa “Usuarios”.

Para ello nos dirigiremos al cuadro donde pone Crear Grupos y le daremos click en Ir, tras esto se nos mostrará una vista con un formulario:

Carlete SL rol: administrador Crear sesión

Crear Nuevo Grupo LDAP

Rellena los campos para registrar un nuevo grupo en el servidor LDAP.

Nombre del grupo (CN)

Solo letras minúsculas y números, sin espacios.

Unidad Organizativa (OU)

Grupos

Por defecto es "Grupos", cambia si usas otra OU.

Miembros del grupo

carlos

administrador

Carlos

Mantén pulsada Ctrl (o Cmd en Mac) para seleccionar varios usuarios.

Crear Grupo Cancelar

© 2015 Carlete SL | Plataforma de Gestión de Usuarios

En este formulario, introduciremos el nombre del grupo que queremos crear, la unidad organizativa (por defecto grupos) y una lista de usuarios que queremos añadir al grupo nada más crearlo:

Crear Nuevo Grupo LDAP

Rellena los campos para registrar un nuevo grupo en el servidor LDAP.

Nombre del grupo (CN)

Solo letras minúsculas y números, sin espacios.

Unidad Organizativa (OU)

Por defecto es "Grupos", cambia si usas otra OU.

Miembros del grupo

carlos

administrador

Carlos

Mantén pulsada Ctrl (o Cmd en Mac) para seleccionar varios usuarios.

Crear Grupo Cancelar

Una vez le demos a crear grupo, nos aparecerá un mensaje, validando la creación del grupo y el usuario o usuarios añadidos a este:

Crear Nuevo Grupo LDAP

Rellena los campos para registrar un nuevo grupo en el servidor LDAP.

Grupo 'clase1' creado exitosamente. Se agregó el usuario '' como miembro inicial.

4. Listar Grupos:

Al igual que listar usuarios, esta opción se encargará de cargar una vista con una tablet con una serie de valores como el nombre del grupo, los miembros, la cantidad de miembros y la opción de poder eliminar el grupo del servidor:

Carlete SL

Hola, administrador

Cerrar sesión

Listado de Grupos

Nombre del Grupo	Miembros	Cantidad de Miembros	Opciones
admins	Sin miembros	0	<button>Eliminar</button>
clase1	x011100	1	<button>Eliminar</button>

Volver al Panel

5. Crear usuarios mediante CSV:

Para crear más de un usuario a la vez, la app web de gestión de usuarios LDAP dispone de una opción para crear usuarios mediante el uso de un archivo csv, como por ejemplo:

usuarios-01.csv

```
1 uid,cn,sn,password,mail
2 x12345,Juan,Carlos,clave123,juan.carlos@carlete.sl
3 x23456,Ana,López,pass456,ana.lopez@carlete.sl
4 x34567,Pedro,García,secure789,pedro.garcia@carlete.sl
5 x45678,Luis,Ramírez,luisspass,luis.ramirez@carlete.sl
6 x56789,Carmen,Martín,carmen123,carmen.martin@carlete.sl
7
```


En este caso lo unico que habria que hacer sería subir el fichero a la app web y darle al botón de subir y crear:

Carlete SL Hola, administrador Cerrar sesión

Crear Usuarios desde Archivo CSV

Sube un archivo CSV con los datos de los usuarios que deseas crear en el servidor LDAP.

Selecciona archivo CSV

usuarios-01.csv

El archivo debe tener las columnas uid, cn, sn, password, email, UID

Una vez se crean lo usuarios saldrá una pequeña vista emergente verificando la creación de estos últimos:

Resultado del proceso:

✅ Usuarios creados correctamente: 5

[Volver](#)

Listado de Usuarios				
ID Usuario	Nombre	Grupo	Correo Electrónico	Opciones
carlos	carlos	Sin grupo	N/A	<input type="button" value="Eliminar"/>
administrador	administrador	Sin grupo	N/A	<input type="button" value="Eliminar"/>
x611188	Carlos	clase1	marquezcalerocarlos@gmail.com	<input type="button" value="Eliminar"/>
x12345	Juan	Sin grupo	juan.carlos@carlete.sl	<input type="button" value="Eliminar"/>
x23456	Ana	Sin grupo	ana.lopez@carlete.sl	<input type="button" value="Eliminar"/>
x34567	Pedro	Sin grupo	pedro.garcia@carlete.sl	<input type="button" value="Eliminar"/>
x45678	Luis	Sin grupo	luis.ramirez@carlete.sl	<input type="button" value="Eliminar"/>
x56789	Carmen	Sin grupo	carmen.martin@carlete.sl	<input type="button" value="Eliminar"/>

6. Revisión de logs:

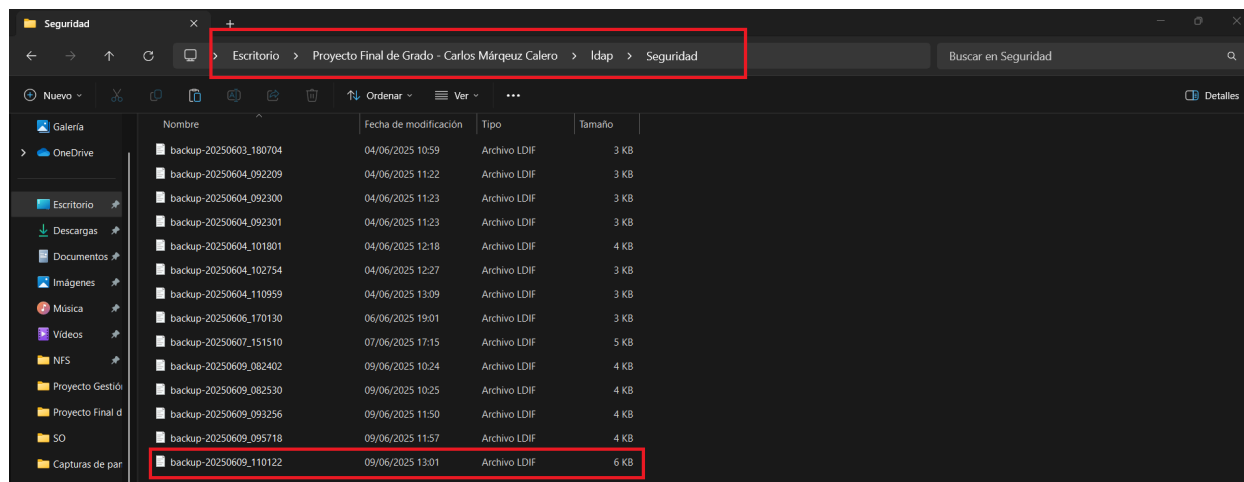
Esta es una opción por la cual podremos revisar los logs del contenedor de Openldap, ordenados según su función, como lo es BIND (Inicio de Sesión), ADD (añadir un usuario o grupo), SEARCH (hacer una búsqueda en el servidor ldap) y otros Logs. Ejemplo de Logs de ADD listados:

Operaciones de ADD (alta)	
Fecha	Contenido
2025-06-09T10:28:44.960167511	2025-06-09T10:28:44.960167511Z 6846b75c conn=1002 op=1 ADD dn="uid=x611188,ou=Usuarios,dc=carlete,dc=sl"
2025-06-09T10:40:22.172374912	2025-06-09T10:40:22.172374912Z 6846ba16 conn=1006 op=7 ADD dn="cn=clase1,ou=Grupos,dc=carlete,dc=sl"
2025-06-09T10:41:40.913437400	2025-06-09T10:41:40.913437400Z 6846ba64 conn=1009 op=7 ADD dn="cn=clase2,ou=Grupos,dc=carlete,dc=sl"
2025-06-09T10:42:53.623687338	2025-06-09T10:42:53.623687338Z 6846baad conn=1021 op=7 ADD dn="cn=clase1,ou=Grupos,dc=carlete,dc=sl"
2025-06-09T10:50:44.095728439	2025-06-09T10:50:44.095728439Z 6846bc84 conn=1024 op=1 ADD dn="uid=x12345,ou=Usuarios,dc=carlete,dc=sl"
2025-06-09T10:50:44.101823812	2025-06-09T10:50:44.101823812Z 6846bc84 conn=1024 op=3 ADD dn="uid=x23456,ou=Usuarios,dc=carlete,dc=sl"
2025-06-09T10:50:44.106841948	2025-06-09T10:50:44.106841948Z 6846bc84 conn=1024 op=5 ADD dn="uid=x34567,ou=Usuarios,dc=carlete,dc=sl"
2025-06-09T10:50:44.110290843	2025-06-09T10:50:44.110290843Z 6846bc84 conn=1024 op=7 ADD dn="uid=x45678,ou=Usuarios,dc=carlete,dc=sl"
2025-06-09T10:50:44.115026791	2025-06-09T10:50:44.115026791Z 6846bc84 conn=1024 op=9 ADD dn="uid=x56789,ou=Usuarios,dc=carlete,dc=sl"

7. Hacer Copias de Seguridad (Idif):

Como última opción en nuestra app web, hacer copias de seguridad se encarga de generar un fichero Idif con todos las unidades organizativas, el dominio, los usuarios y los grupos que esten activos en el momento de hacerla, esta se guarda en un volumen compartido entre mi maquina local y el contenedor openldap, llamado “/ldap/seguridad”:



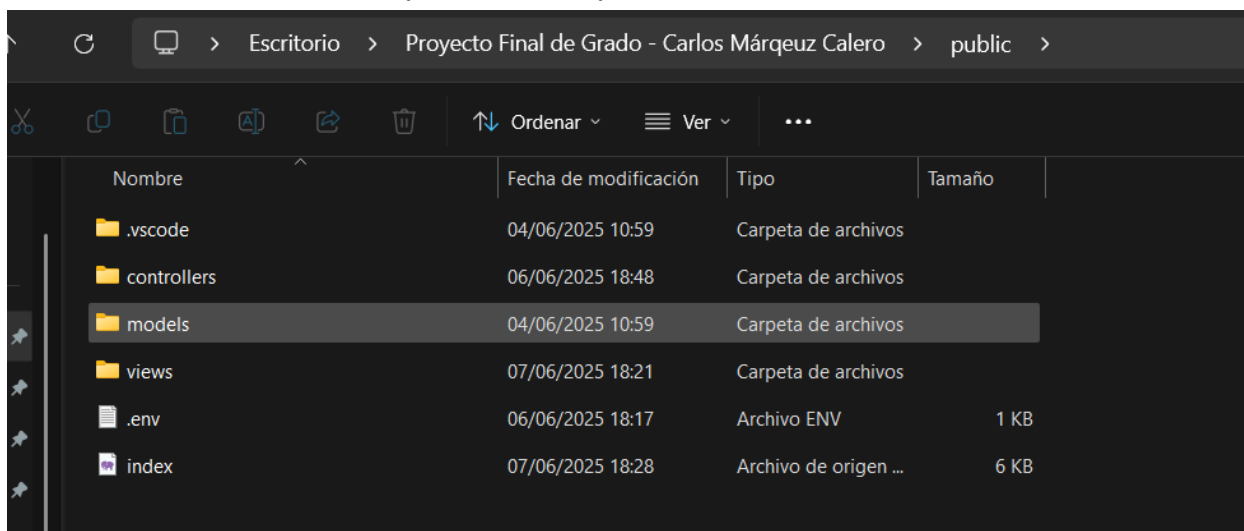


7. Explicación de Código Destacado:

La App Web de Gestión de Usuarios vía LDAP, está desarrollada usando el lenguaje orientado a objetos llamado PHP. En este punto explicaré la organización del código, es decir como estan dividido los ficheros y en que subdirectorios se encuentran y explicare también una parte del código de forma destacada.

1. Organización del código:

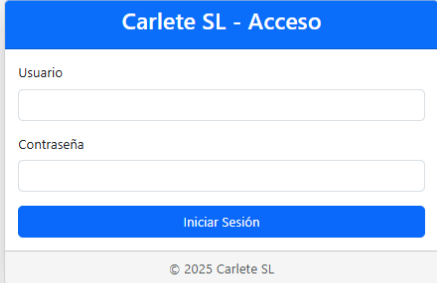
En cuanto a la organización del código cabe destacar lo siguiente, la app web se encuentra dentro de una carpeta llamada public:



Dentro de esta carpeta tenemos la siguiente organización:

- **Index.php:** fichero principal de la app web y al que llamaremos al principio para iniciar sesión en la plataforma de gestión, una vez iniciada la sesión, la vista llamada “gestión.php” parará a ser el index hasta que se cierre la sesión:

1. Vista:



2. Código:

```
// =====  
// PROCESAMIENTO DEL FORMULARIO DE LOGIN  
// =====  
if ($_SERVER["REQUEST_METHOD"] === "POST" && isset($_POST["usuario"], $_POST["contrasena"])) {  
    $user = $_POST["usuario"];  
    $password = $_POST["contrasena"];  
  
    // Creamos una instancia del modelo LDAP  
    $ldap = new ModeloLDAP();  
  
    // Intentamos autenticar al usuario  
    if ($ldap->authenticate($user, $password)) {  
        // Si la autenticación es exitosa, iniciamos sesión  
        $_SESSION["usuario"] = $user;  
  
        // Verificamos si el usuario tiene permisos de administrador  
        if ($ldap->isAdmin($user)) {  
            $_SESSION["es_admin"] = true;  
        }  
  
        // Redirigimos a la página de gestión  
        View::show("views/gestion.php");  
        exit;  
    } else {  
        // Si falla la autenticación, mostramos error  
        $error = "Usuario o contraseña incorrectos.";  
    }  
}
```

- **models:** Dentro de este directorio encontraremos un fichero llamado ModeloLDAP.php, el cual contendrá todos los métodos que nos permitirán, por ejemplo:

Abrir una conexión con LDAP:

```
// =====
// CONSTRUCTOR Y CONFIGURACIÓN INICIAL
// =====

// Constructor: se ejecuta automáticamente al crear un objeto de esta clase
public function __construct() {
    // Establecemos conexión con el servidor LDAP que está en Docker
    $this->ldapconn = ldap_connect("ldap://openldap");

    // Configuramos la versión 3 del protocolo LDAP (la más actual)
    ldap_set_option($this->ldapconn, LDAP_OPT_PROTOCOL_VERSION, 3);
}

// Función auxiliar para autenticarse como administrador
private function bindAdmin() {
    $adminDn = "cn=admin,dc=carlete,dc=sl";
    $adminPass = "admin270404";
    ldap_bind($this->ldapconn, $adminDn, $adminPass);
}

// =====
// FUNCIONES DE AUTENTICACIÓN
// =====

// Función para verificar si las credenciales de un usuario son correctas
public function authenticate($user, $password) {
    // Construimos el DN (Distinguished Name) completo del usuario
    $dn = "cn=$user,ou=Usuarios,dc=carlete,dc=sl";

    // Intentamos autenticar con el usuario y contraseña proporcionados
    // El @ suprime los warnings de PHP si falla la autenticación
    return @ldap_bind($this->ldapconn, $dn, $password);
}
```

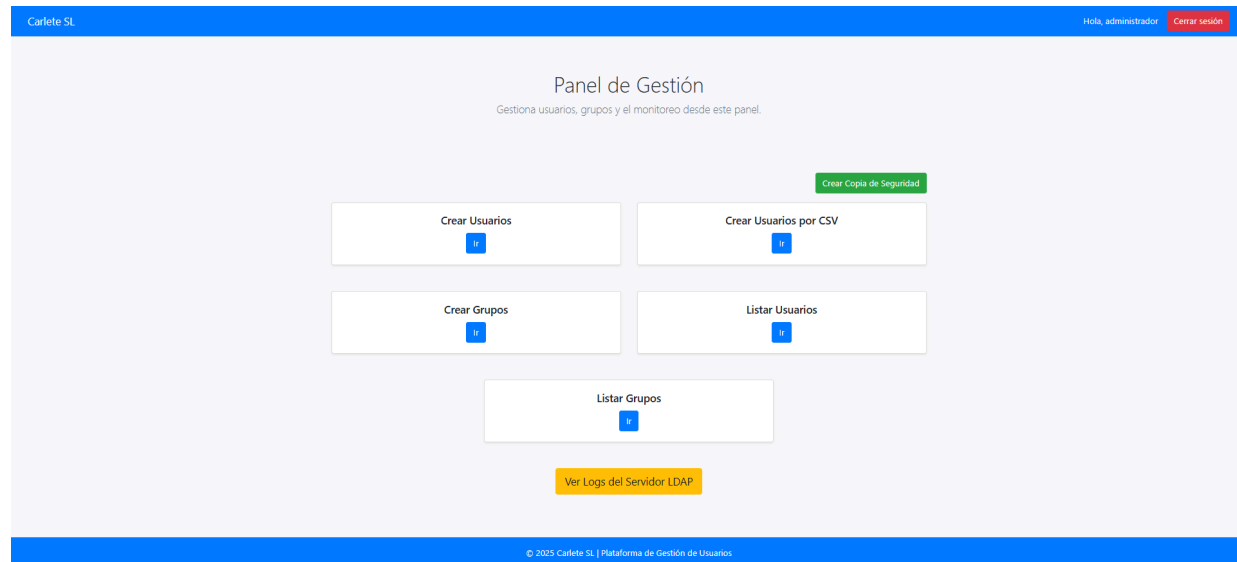
- **controllers:** Dentro de este directorio encontraremos una serie de ficheros que contendrán todos los controladores para efectuar las acciones de manera correcta dentro de la app web, como lo son por ejemplo:

UsuariosController.php: Dentro de este fichero se listan todos los controladores que nos permiten, listar usuarios, crear usuarios (con csv también), mostrar las vistas de formularios y de csv, eliminar usuarios y además cerrar la sesión en la app web.

```
public > controllers > UsuarioController.php

1  <?php
2  /**
3   * UsuarioController.php
4   * Controlador principal para la gestión de usuarios en el sistema LDAP
5   * Incluye funcionalidades para listar, crear, eliminar usuarios y cargar desde CSV
6   */
7
8  // Iniciamos una Sesión para mantener estado entre peticiones
9  session_start();
10
11 require_once "models/ModeloLDAP.php";
12 require_once 'views/view.php';
13
14 class UsuarioController {
15
16     /**
17      * Método para listar todos los usuarios del sistema
18      * Obtiene todos los usuarios desde LDAP y los muestra en una vista
19      */
20     public function listar() {
21         $ldap = new ModeloLDAP();
22         $usuarios = $ldap->getAllUsers();
23
24         // Pasar directamente el array de usuarios a la vista
25         View::show("views/lista_usuarios.php", $usuarios);
26     }
27 }
```

- views: Dentro de este directorio se alojarán todas las vistas que se le mostrarán al usuario final sobre la app web de gestión, como lo es por ejemplo la vista de gestion.php:



2. Explicación de código a destacar:

En este punto explicaré la funcionalidad detallada de la opción en la vista de gestión llamada “Hacer copia de Seguridad”, a nivel de código.

En primer lugar esta opción está desarrollada en un controlador llamado BackupController.php el cual se vería así:

```
public > controllers > BackupController.php
1  <?php
2  /**
3   * BackupController.php
4   * Controlador para gestionar las copias de seguridad del sistema LDAP
5   */
6  require_once 'models/ModeloLDAP.php';
7
8  class BackupController {
9      private $modelo;
10
11      /**
12       * Constructor de la clase
13       * Inicializa el modelo LDAP necesario para las operaciones de backup
14       */
15      public function __construct() {
16          $this->modelo = new ModeloLDAP();
17      }
18
19      /**
20       * Método para generar una copia de seguridad
21       * Llama al modelo para crear el backup y redirige al index
22       */
23      public function generar() {
24          // Genera la copia de seguridad a través del modelo LDAP
25          $this->modelo->generarCopiaSeguridad();
26
27          // Redirige al usuario de vuelta al index principal
28          header("Location: index.php"); // o "gestion.php" si va aparte
29          exit();
30      }
31  }
32  ?>
```

Este controlador cuando es invocado a la hora de pulsar el respectivo botón en la vista de gestión.php, invoca el modeloldap proveniente del fichero ModeloLDAP.php, almacenando en una variable privada llamada \$modelo.

Posteriormente, crea una función pública en la cual mediante la variable privada \$modelo, invoca el método llamado generarCopiaSeguridad();

Ahora una vez lo llama, paso a explicar qué hace este método. Este método se encarga de generar una variable llamada \$timestamp, la cual guarda la fecha en la que se ejecuta la acción en ese momento, posteriormente genera una variable llamada ruta, en la cual almacena la ruta en la cual se guardará el archivo backup dentro del contenedor openldap, después genera otra variable en la cual almacena el comando a ejecutar dentro del contenedor apache-php y mediante el uso de la dependencia shell_exec, ejecuta el comando en el contenedor apache-php, devolviendo la ruta completa y almacenando un nuevo fichero en el volumen compartido que aparece en la variable \$ruta:

```
// =====  
// FUNCIONES DE BACKUP Y SEGURIDAD  
// =====  
  
// Función para generar una copia de seguridad del directorio LDAP  
public function generarCopiaSeguridad() {  
    $timestamp = date('Ymd_His'); // Timestamp para el nombre del archivo  
    $ruta = "/ldap/Seguridad/backup-$timestamp.ldif"; // Ruta donde se guardará el backup  
    $comando = "docker exec openldap slapcat -v -l $ruta"; // Comando para crear el backup  
    shell_exec($comando); // Ejecutamos el comando  
    return $ruta; // Devolvemos la ruta del archivo creado  
}
```

```
▼ ldap  
  ▼ Seguridad  
    backup-20250604_092300.ldif  
    backup-20250604_092301.ldif  
    backup-20250604_101801.ldif  
    backup-20250604_102754.ldif  
    backup-20250604_110959.ldif  
    backup-20250606_170130.ldif  
    backup-20250607_151510.ldif  
    backup-20250609_082402.ldif  
    backup-20250609_082530.ldif  
    backup-20250609_093256.ldif  
    backup-20250609_095718.ldif  
    backup-20250609_110122.ldif
```

8. Conclusión:

Este proyecto destaca por la facilidad de configurar un servidor LDAP a través de una app web desarrollada en PHP, con la finalidad de facilitar a los desarrolladores el uso de este tipo de servidores de gestión de usuarios internos.

9. Mejoras a Futuro:

1. Uso de firewall para maximizar la seguridad:

Mediante el uso de S.O como lo es Pfsense se puede crear un router/firewall que mediante una serie de redes y una serie de reglas de direccionamiento de red pueda maximizar la seguridad sobre la app web, segmentando la red en varias vlans, separando a los administradores que la controlan en una red virtual y a los usuarios normales en otra.

2. Creación de usuarios POSIX:

Si bien se crean usuarios y grupos en el servidor LDAP, estos no tiene la estructura necesaria para poder más tarde iniciar sesión en una máquina física o virtual ya que no tendrían ni UID y GID como si lo tiene los usuarios POSIX.

3. Mejora de interfaz gráfica:

La interfaz gráfica se puede ajustar mucho más para que sea lo más amigable y agradable posible de cara al usuario final (administradores) facilitando su uso.

10. WebGrafía y enlaces de Interés:

- Herramientas de Desarrollo:
 1. [Visual Studio \(descargar\)](#)
 2. [Manual PHP - LDAP](#)
 3. IAs (Depuración de Errores y Estructuración de Código):
 - Chat Gpt: [Chat GPT](#)
 - Claude: [Claude](#)
- Herramientas de Implementación:
 1. [Docker-Desktop](#)
 2. [Manual Docker-Compose](#)
- Repositorio Github Con app web de Gestión DE Usuarios LDAP By Carlos Márquez:
 1. [Repositorio GitHub Propio](#)

