LABORATORIO DE REDES Y SISTEMAS OPERATIVOS

TEMÁTICA:

Tutorial de instalación, configuración y uso de “SEAFILE”.

INTEGRANTES:

David Ignacio Duarte
Antonella Pezzola
Sofía Justiniano

PROFESOR:

José Luis Di Biase

Tabla de Contenidos

1. Introducción

Seafile se posiciona como una alternativa ligera y eficiente a servicios comerciales populares como Dropbox y Google Drive. Su principal ventaja radica en la capacidad de ser alojado en servidores propios, lo que otorga a las organizaciones un control total sobre sus datos y garantiza el cumplimiento de políticas de privacidad y seguridad específicas. Esta característica lo convierte en una opción ideal para instituciones, equipos de trabajo o entornos educativos que buscan independencia de plataformas de terceros.

El objetivo de este trabajo es explorar una herramienta de software libre orientada al manejo de archivos distribuidos, integrando conceptos fundamentales de redes, servicios y administración de sistemas, tal como se abordan en la cursada de Laboratorio de Redes y Sistemas Operativos.

2. ¿Qué es Seafile?

Seafile es un software de código abierto diseñado para la sincronización, almacenamiento y colaboración de archivos, destacándose por ser rápido, eficiente y seguro.

Su arquitectura se basa en un sistema de **bibliotecas** ("libraries") que pueden ser sincronizadas entre distintos dispositivos y usuarios. Esto facilita el intercambio de información, permite compartir y versionar archivos de forma controlada, y ofrece una robusta capacidad de recuperación ante errores de edición o eliminaciones accidentales.

A diferencia de soluciones comerciales, Seafile puede instalarse en un servidor propio, brindando a los administradores un control granular sobre la infraestructura de almacenamiento, la gestión de usuarios y la personalización del entorno según las necesidades específicas de la organización.

Es una solución multiplataforma, ofreciendo clientes nativos para sistemas operativos de escritorio como Windows, macOS y Linux, así como aplicaciones móviles para Android e iOS. Además, cuenta con una interfaz web intuitiva que simplifica la gestión de archivos, usuarios y configuraciones, haciendo que sea accesible tanto para usuarios finales como para administradores.

3. Requisitos del Sistema

Según la documentación oficial de Seafile, los requisitos mínimos recomendados para la instalación de Seafile Server son los siguientes:

Componente	Requisito Mínimo
CPU	2 núcleos (se recomienda una velocidad superior a 2 GHz)
RAM	2 GB
Almacenamiento	Al menos 10 GB + espacio para archivos (se recomiendan 50 GB)
Sistema Operativo	Ubuntu 24.04, Ubuntu 22.04, Debian 12, Debian 11
Base de Datos	MySQL, MariaDB (recomendada), SQLite (para pruebas)
Dependencias	Python 3.6+, pip, Django, librerías específicas (se instalarán durante la guía)

Para este tutorial, se usó una máquina virtual con Ubuntu 22.04, 8 núcleos de CPU con 2.8GHz, 8GB de ram y 50GB de almacenamiento.

4. Instalación Paso a Paso

Nota importante: A partir de la versión 13.0, Seafile Community Edition ya no ofrece soporte para instalación mediante binarios precompilados. Por lo tanto, para este tutorial se utiliza la **versión 12.0.14**, la última versión que permite una instalación simple sin la necesidad de Docker o compilación de código fuente.

4.1. Instalación de Base de Datos, Python y Librerías Necesarias

Para comenzar, es fundamental asegurarse de que el sistema cuente con las herramientas y librerías básicas para la instalación de Seafile. Ejecute los siguientes comandos en la terminal:

```
sudo apt-get update
sudo apt-get install -y mysql-server \
      python3 python3-dev python3-setuptools python3-pip \
      libmysqlclient-dev ldap-utils libldap2-dev
default-libmysqlclient-dev \
      build-essential pkg-config libmemcached-dev memcached
```

4.2. Creación de Directorio Estándar de Instalación

Por convención, Seafile se instala en el directorio /opt/seafile. Aunque puede modificarse, se recomienda seguir esta ruta. Cree el directorio y navegue hacia él:

sudo mkdir /opt/seafile
cd /opt/seafile

4.3. Instalación de Dependencias de Python

Desde el directorio de instalación (/opt/seafile), instale las dependencias de Python específicas que Seafile requiere. Es crucial especificar las versiones exactas para evitar conflictos:

```
sudo pip3 install --timeout=3600 \
      django==4.2.* future==1.0.* mysqlclient==2.1.* \
      pymysql pillow==10.4.* pylibmc captcha==0.6.*
markupsafe==2.0.1 \
      jinja2 sqlalchemy==2.0.* psd-tools django-pylibmc \
      django_simple_captcha==0.6.* djangosaml2==1.9.* \
```

```
pysaml2==7.2.* pycryptodome==3.16.* cffi==1.15.1 \
python-ldap==3.4.3 lxml gevent==24.2.*
```

Se instalan diversas librerías como Django (framework web), mysqlclient (conector MySQL), pillow (procesamiento de imágenes), pylibmc (cliente Memcached), entre otras, todas en versiones compatibles con Seafile 12.0.14.

4.4. Creación de Usuario "seafile"

No es una buena práctica ejecutar aplicaciones de servidor como root debido a los riesgos de seguridad. Por ello, se recomienda crear un usuario dedicado para Seafile:

```
sudo adduser seafile
```

Se le solicitará una contraseña para el nuevo usuario. Para este tutorial, se utilizará "filesea123", pero en un entorno de producción, se debe elegir una contraseña robusta y segura. Puede presionar Enter para omitir los campos opcionales como nombre completo, dirección, etc.

Una vez creado el usuario, cambie la propiedad del directorio de instalación de Seafile para que el nuevo usuario tenga permisos sobre él:

```
sudo chown -R seafile: /opt/seafile
```

Finalmente, cambie al usuario seafile para continuar con la instalación:

```
su seafile
```

Se le pedirá la contraseña que estableció para el usuario seafile.

4.5. Descarga de Seafile Server

Desde el directorio estándar de instalación (/opt/seafile), descargue el paquete de instalación de Seafile Server y descomprimirlo:

```
wget
https://s3.eu-central-1.amazonaws.com/download.seadrive.org/seafile-s
erver_12.0.14_x86-64.tar.gz
```

Este comando descarga la versión 12.0.14 del servidor Seafile:

```
tar -xzf seafile-server_12.0.14_x86-64.tar.gz
```

Este comando descomprime el archivo .tar.gz, creando un directorio llamado seafile-server-12.0.14 que contiene todos los scripts y binarios necesarios para instalar y ejecutar el servidor.

4.6. Configuración de Base de Datos

Antes de ejecutar el script de instalación de Seafile con MySQL, es necesario crear manualmente las bases de datos y el usuario correspondiente en MySQL. Esto se realiza desde un usuario con privilegios sudo (el usuario principal), utilizando el cliente de línea de comandos de MySQL.

Primero, cambie al usuario principal que tiene acceso a sudo:

```
su USER  # Reemplace USER con su nombre de usuario principal
```

Luego, acceda a la línea de comandos de MySQL como root:

```
sudo mysql -u root -p # Se le pedirá la contraseña de root de MySQL
si la tiene configurada
```

Dentro del cliente MySQL, ejecute los siguientes comandos para crear las bases de datos y el usuario seafile, asignando los permisos adecuados:

```
CREATE DATABASE `ccnet_db` CHARACTER SET = 'utf8';
CREATE DATABASE `seafile_db` CHARACTER SET = 'utf8';
CREATE DATABASE `seahub_db` CHARACTER SET = 'utf8';

CREATE USER 'seafile'@'localhost' IDENTIFIED WITH
mysql_native_password BY 'seafile';

GRANT ALL PRIVILEGES ON `ccnet_db`.* TO `seafile`@localhost;
GRANT ALL PRIVILEGES ON `seafile_db`.* TO `seafile`@localhost;
```

```
GRANT ALL PRIVILEGES ON `seahub_db`.* TO `seafile`@localhost;

FLUSH PRIVILEGES;
EXIT;
```

Se crean tres bases de datos: *ccnet_db* (para la gestión de usuarios y grupos), *seafile_db* (para los metadatos de los archivos) y *seahub_db* (para la interfaz web).

Se crea un usuario seafile con contraseña seafile (para el tutorial; en producción, use una contraseña fuerte).

Se otorgan todos los privilegios al usuario seafile sobre las tres bases de datos.

Finalmente, cambie nuevamente al usuario seafile:

```
su seafile
```

Asegúrese de estar en el directorio correcto donde se encuentra el script de instalación de Seafile:

```
cd /opt/seafile/seafile-server-12.0.14
```

Y ejecute el script de configuración de Seafile para MySQL:

```
./setup-seafile-mysql.sh
```

Este script interactivo le guiará a través de la configuración. A continuación, se detallan las respuestas típicas para este tutorial:

Nombre del servidor: seafile-labo (ejemplo)

IP del servidor: Para obtener su IP local, ejecute hostname -I en otra terminal. Ingrese la IP obtenida (formato xxx.yyy.z.ww).

Puerto: 8082 (por defecto)

Inicialización de bases de datos de Seafile: Elija la opción 2 (usar bases de datos existentes).

Host MYSQL: localhost (por defecto)

Puerto MYSQL: 3306 (por defecto)

Usuario MYSQL: seafile

Contraseña MySQL: seafile

Base de datos existente para ccnet: ccnet_db

Base de datos existente para seafile: seafile_db

Base de datos existente para seahub: seahub_db

Después de ingresar la información, se mostrará un resumen de la configuración. Presione ENTER para continuar. El script creará las tablas necesarias en las bases de datos. Deberá ver un mensaje de éxito similar a:

```
-----------------------------------------------------------------
Your seafile server configuration has been finished successfully.
-----------------------------------------------------------------

run seafile server:       ./seafile.sh { start | stop | restart }
run seahub  server:       ./seahub.sh  { start <port> | stop | restart <port> }

-----------------------------------------------------------------
If you are behind a firewall, remember to allow input/output of these tcp
ports:
-----------------------------------------------------------------

port of seafile fileserver:   8082
port of seahub:                 8000

When problems occur, Refer to

       https://download.seafile.com/published/seafile-manual/home.md

for information.
```

4.7. Creación de .env y Configuración de Variables de Entorno

Seafile requiere un archivo de variables de entorno para su correcta ejecución. Este archivo, llamado .env, debe crearse en el directorio base de instalación (/opt/seafile).

Navegue al directorio base:

```
cd /opt/seafile
```

Cree y edite el archivo .env utilizando un editor de texto como nano:

```
nano .env
```

Pegue el siguiente contenido, asegurándose de reemplazar `SEAFILE_SERVER_HOSTNAME` con la IP local que definió en la configuración anterior:

```
JWT_PRIVATE_KEY=123
SEAFILE_SERVER_PROTOCOL=http
SEAFILE_SERVER_HOSTNAME=192.168.1.44 # REEMPLAZAR CON SU IP LOCAL
SEAFILE_MYSQL_DB_HOST=localhost
SEAFILE_MYSQL_DB_PORT=3306
SEAFILE_MYSQL_DB_USER=seafile
SEAFILE_MYSQL_DB_PASSWORD=seafile
SEAFILE_MYSQL_DB_CCNET_DB_NAME=ccnet_db
SEAFILE_MYSQL_DB_SEAFILE_DB_NAME=seafile_db
SEAFILE_MYSQL_DB_SEAHUB_DB_NAME=seahub_db
```

`JWT_PRIVATE_KEY`: Clave interna para firmar tokens de sesión. Para el tutorial, se usa 123. En producción, genere una clave compleja.

`SEAFILE_SERVER_HOSTNAME`: La dirección IP o nombre de dominio donde Seafile será accesible.

Guarde el archivo (Ctrl+O, Enter) y salga del editor (Ctrl+X).

4.8. Ejecución del Script del Servidor

Finalmente, una vez que las variables de entorno están declaradas, regrese al directorio donde se encuentran los scripts de Seafile:

```
cd /opt/seafile/seafile-server-12.0.14
```

Inicie el servidor Seafile:

```
./seafile.sh start
```

Debería ver un mensaje similar a:

```
Cannot find JWT_PRIVATE_KEY value from environment, try to read .env
file.
Starting seafile server, please wait ...
Seafile server started
```

Luego, inicie el frontend de Seafile, conocido como Seahub:

```
./seahub.sh start
```

Este script le solicitará un email para el usuario **ADMINISTRADOR** y una contraseña. Para este tutorial, usaremos:

Email: labo-tutorial-admin@gmail.com

Contraseña: labo-tutorial-admin

¡Y listo! Seafile y Seahub deberían estar funcionando. Puede acceder a la interfaz web desde su navegador:

Desde el mismo servidor: http://localhost:8000

y debe loguearse con las credenciales del usuario ADMIN. Podrá ver la interfaz y realizar configuraciones, agregar nuevos usuarios, etc.

4.9. Acceso externo y configuración de red

Por defecto, Seafile sólo se expone a la IP *127.0.0.1*, es decir, sólo accesible desde el mismo equipo. Para permitir el acceso desde otros dispositivos de la red (como su celular, otra PC, etc..), es necesario que el servidor escuche en **todas las interfaces de red**.

Para lograr esto, debemos cambiar “*127.0.0.1” por “0.0.0.0*”.

Antes, debemos detener nuestros servicios:

```
/opt/seafile/seafile-server-12.0.14/./seafile.sh stop
/opt/seafile/seafile-server-12.0.14/./seahub.sh stop
```

Debe editar el archivo de configuración de Seahub:

```
nano /opt/seafile/conf/gunicorn.conf.py
```

cambiando la siguiente línea

```
bind = “127.0.0.1:8000”
```

por:

```
bind = “0.0.0.0:8000”
```

Esta configuración nos permitirá conectarnos desde otros dispositivos de la misma red. No obstante, podremos seguir teniendo problemas de red y de **CORS** por ejemplo, al querer subir y compartir archivos. ¿Por qué sucede esto? **CORS** (Cross-Origin Resource Sharing) es un mecanismo de seguridad de los navegadores modernos que impide que una aplicación web (en origen A) haga peticiones a otro dominio u origen (B) sin permisos explícitos.

Al ejecutar Seahub, Seafile lanza su interfaz web utilizando **GUNICORN**, un servidor web minimalista de alto rendimiento escrito en Python. Por defecto, este servicio escucha sólo en la dirección *12.0.0.1:8000*, lo que impide el acceso desde otros dispositivos de la red local. Sin embargo, aún cambiando esto, no es suficiente cuando se intenta acceder desde una aplicación web que consuma su API o interactúe con recursos compartidos, ya que inmediatamente se topará con un problema de CORS.

La documentación oficial de Seafile asume siempre el uso de **Nginx** o **Apache** como servidor web frontal y concluímos que por esta razón no incluye ninguna sección específica sobre CORS, ni sobre exponer Gunicorn directamente. La arquitectura recomendada por los desarrolladores

de Seafile está pensada para entornos productivos, donde el uso de un **proxy inverso** como Nginx, es una práctica estándar.

Un proxy inverso es un servidor intermedio que recibe las solicitudes del cliente y las reenvía al servidor real (en este caso Gunicorn, corriendo Seahub). Usar Nginx como proxy inverso ofrece además múltiples beneficios como el manejo adecuado de headers HTTP, mejor rendimiento, mayor seguridad y escalabilidad.

Por esta razón, debemos instalar Nginx, **desde root** (cambiar el usuario con `su USER`), con el comando:

```
sudo apt install nginx -y
```

Luego de la instalación, tiene que iniciar el servidor y habilitarlo:

```
sudo systemctl start nginx
sudo systemctl enable nginx
```

y finalmente, crear un archivo de configuración para Seafile:

```
touch /etc/nginx/sites-available/seafile.conf
```

eliminando también los archivos por defecto:

```
rm /etc/nginx/sites-enabled/default
rm /etc/nginx/sites-available/default
```

Ahora, crearemos un link simbólico para la nueva configuración, con la idea de separar las configuraciones disponibles de las activas, para tener un mayor control al activar y desactivar sitios fácilmente, sin borrarlos y mantener organizada la administración:

```
ln -s /etc/nginx/sites-available/seafile.conf
/etc/nginx/sites-enabled/seafile.conf
```

Ahora, debemos configurar Nginx en el nuevo archivo, editándolo:

```
nano /etc/nginx/sites-available/seafile.conf
```

y pegando el siguiente archivo (VER COMENTARIO):

```
log_format seafileformat '$http_x_forwarded_for $remote_addr
[$time_local] "$request" $status $body_bytes_sent "$http_referer"
"$http_user_agent" $upstream_response_time';

server {
    listen 80;
    server_name 192.168.1.44; #REEMPLAZAR POR SU IP

    proxy_set_header X-Forwarded-For $remote_addr;

    location / {
         proxy_pass         http://127.0.0.1:8000/;
         proxy_set_header   Host $http_host;
         proxy_set_header   X-Real-IP $remote_addr;
         proxy_set_header   X-Forwarded-For
$proxy_add_x_forwarded_for;
         proxy_set_header   X-Forwarded-Host $server_name;
         proxy_read_timeout  1200s;

         # used for view/edit office file via Office Online Server
         client_max_body_size 0;

         access_log      /var/log/nginx/seahub.access.log
seafileformat;
         error_log       /var/log/nginx/seahub.error.log;
    }

    location /seafhttp {
```

```
        rewrite ^/seafhttp(.*)$ $1 break;
        proxy_pass http://127.0.0.1:8082/;
        client_max_body_size 0;
        proxy_set_header   X-Forwarded-For
$proxy_add_x_forwarded_for;

        proxy_read_timeout  36000s;
        proxy_send_timeout  36000s;

        send_timeout  36000s;

        access_log      /var/log/nginx/seafhttp.access.log
seafileformat;
        error_log       /var/log/nginx/seafhttp.error.log;
    }
    location /media {
        root /opt/seafile/seafile-server-latest/seahub;
    }
}
```

Finalmente, nos aseguraremos de que el archivo quede correctamente guardado y reiniciamos Nginx para que la configuración tome efecto:

```
nginx -t
nginx -s reload
```

Ahora sí, ¡tenemos todo configurado para usar perfectamente Seafile y Seahub! Podemos iniciar los servicios:

```
/opt/seafile/seafile-server-12.0.14/./seafile.sh start
/opt/seafile/seafile-server-12.0.14/./seahub.sh start
```

y debemos ser capaces de subir archivos y compartirlos en una misma red.

5. Documentación de Problemas y Soluciones

Durante la instalación y configuración de Seafile, es común encontrarse con ciertos desafíos. A continuación, se documentan los problemas más recurrentes y sus posibles soluciones, lo que puede servir como referencia para futuras implementaciones .

5.1. Seafile sólo accesible desde localhost

La interfaz web de Seafile funcionaba en localhost, y no era accesible desde otros dispositivos por IP o dominio local. Lo solucionamos configurando el inicio de Seahub para que escuche en todas las interfaces de red (paso **4.9** del tutorial).

5.2. Falta de conocimiento del rol de Nginx

Fue confuso si era necesario instalar Apache o qué rol cumplía Nginx en el ecosistema de Seafile. La documentación oficial menciona a Nginx como proxy inverso, pero no queda claro que Seahub **ya es un servidor web Django** y que Nginx es sólo opcional como proxy.

Conceptualmente aprendimos que Seahub ya corre en un servidor web, y Nginx nos resuelve los problemas de HTTPs, CORS, redirecciones, etc.

5.3. CORS

Seafile expone internamente dos servicios -> Seahub en el puerto 8000, y seaf-server en el puerto 8082. Si accedìamos desde otra máquina o dispositivo que no era el mismo servidor (otro origen), el browser bloqueada las peticiones por política de CORS. Modificamos varios archivos de configuración sin éxito en absoluto, sin existir en Seafile una documentación directa y documentada para habilitar CORS. La solución fue usar Nginx como proxy inverso para enrutar peticiones tanto a Seahub (8000) como a la API (8082) y así controlar los headers de respuesta.

No nos fue posible solucionar el problema de CORS sin Nginx.

5.4. Problemas con Dependencias Python al Iniciar seahub

Problema: La ejecución de seahub.sh puede fallar si las dependencias de Python no están correctamente instaladas o si existen conflictos de versiones. La documentación oficial no especifica claramente qué paquetes o versiones exactas se deben instalar, o cómo resolver dependencias complejas.

Solución:

Verificar la lista de dependencias: Asegúrese de que todas las librerías listadas en el paso 4.3 estén instaladas con las versiones especificadas. Un error común es que pip3 no instale la versión correcta o que alguna dependencia secundaria tenga un conflicto.

Entornos Virtuales: Para entornos de producción, es altamente recomendable utilizar un entorno virtual de Python (venv o conda). Esto aísla las dependencias de Seafile de otras instalaciones de Python en el sistema, previniendo conflictos.

```
python3 -m venv seafile_env
source seafile_env/bin/activate
pip install --timeout=3600 ... # Instalar dependencias dentro del
entorno virtual
```

Luego, los scripts de Seafile se ejecutarán dentro de este entorno activado.

Registros de Errores: Revise los archivos de registro de Seafile (ubicados típicamente en seafile-server-latest/logs/) para obtener mensajes de error más detallados sobre las dependencias faltantes o conflictivas.

6. Conclusiones

Al finalizar este trabajo práctico, sacamos conclusiones importantes sobre la instalación, configuración y operación de Seafile como una solución robusta para la gestión de archivos distribuidos.

6.1. Ventajas de Seafile

Control Total y Privacidad Garantizada: Seafile nos permite alojar los datos en servidores propios, lo que asegura un control absoluto sobre la información y un cumplimiento estricto de las políticas de privacidad y seguridad de la organización. Esta autonomía respecto a servicios de terceros es una ventaja clave, especialmente en entornos donde la confidencialidad es primordial.

Arquitectura Sólida para la Colaboración Eficaz: Su diseño, basado en el concepto de **bibliotecas** y un avanzado sistema de **versionado de archivos**, facilita muchísimo la colaboración entre usuarios. Esto posibilita un seguimiento detallado de los cambios, la recuperación de versiones anteriores y la restauración ante eventuales errores de edición o eliminaciones accidentales, fortaleciendo significativamente la resiliencia de los datos.

Multiplataforma y Usabilidad Intuitiva: La disponibilidad de clientes nativos para los principales sistemas operativos (Windows, macOS, Linux, Android, iOS) y una interfaz web súper intuitiva y amigable, garantiza que Seafile sea accesible para los usuarios. A su vez, ofrece herramientas de

gestión eficientes para los administradores, simplificando las tareas operativas.

6.2. Desafíos de la Implementación

La implementación de Seafile presentó algunos desafíos que consideramos fundamental documentar para futuras implementaciones:

Gestión de Dependencias Python: Un Punto Crítico: La documentación oficial no siempre es lo suficientemente clara en cuanto a las versiones exactas de las dependencias de Python. Esto puede generar conflictos entre librerías o requerir la instalación de versiones muy específicas para el correcto funcionamiento de `seahub.sh`.

Variables de Entorno en `.env`: La Necesidad de Exportación Explícita: Constatamos que los scripts de inicio `seafile.sh` y `seahub.sh` no cargan automáticamente las variables de entorno definidas en el archivo `.env`. Fue indispensable exportar manualmente estas variables (por ejemplo, `export JWT_PRIVATE_KEY=…`) antes de iniciar los servicios.

6.3. Desventajas de Seafile

Configuración Inicial: A diferencia de soluciones en la nube ya armadas (como Google Drive o Dropbox), Seafile requiere una instalación y configuración inicial que puede ser un poco más compleja. Hay que tener conocimientos de servidores, bases de datos y Python para dejarlo andando bien.

Mantenimiento y Actualizaciones: Todo el mantenimiento, las actualizaciones de software, la gestión de la base de datos y la seguridad recaen en el equipo o la persona que lo administra. Esto implica dedicar tiempo y recursos constantemente para asegurar que funcione correctamente y esté protegido.

Escalabilidad y Hardware: Si bien es escalable, el rendimiento y la capacidad de Seafile van a depender directamente del hardware del servidor donde lo tengas instalado. Si la cantidad de usuarios o el volumen de archivos crece mucho, vas a necesitar invertir en más recursos (CPU, RAM, disco) para que no se ponga lento.

Soporte Comunitario vs. Comercial: La versión Community Edition (la que usamos) se basa en el soporte de la comunidad. Si bien hay foros y documentación, a veces encontrar soluciones a problemas muy específicos puede llevar más tiempo que si tuvieras un soporte técnico pago de una solución comercial.

Acceso Remoto y Configuración de Red: Para que Seafile sea accesible desde fuera de la red local, hay que configurar correctamente el firewall, el router y, posiblemente, un dominio y certificados SSL. Esto agrega una capa extra de complejidad que no existe en los servicios en la nube.

En resumen, si bien Seafile te da un control total, esa misma autonomía implica una mayor responsabilidad y dedicación en la gestión y el mantenimiento.

6.4. Lecciones Aprendidas

Este trabajo práctico nos llevamos de aprendizaje :

La Documentación: La necesidad de documentar **cada uno** de los pasos de instalación y configuración, incluyendo las soluciones a los errores más comunes, es fundamental. Una documentación clara y detallada no solo optimiza el tiempo y el esfuerzo en futuras implementaciones, sino que también facilita la resolución de problemas.

Verificación Rigurosa de Compatibilidad: Resulta crucial verificar siempre la compatibilidad de versiones entre Python, sus dependencias y el sistema operativo. Las incompatibilidades pueden convertirse en una fuente significativa de errores, a menudo difíciles de diagnosticar y resolver.

La clave para solucionar problemas: saber dónde buscar. Poder analizar bien los mensajes de error en la consola del navegador y en los logs del servidor es fundamental para resolver las problemáticas de forma rápida y eficiente.