Servidor Ubuntu

Documentación

XXXX

25 de agosto del 2025

Contenido

1. Servidor Ubuntu

El objetivo de implementar un servidor casero es eliminar la dependencia de la **PC-3**. En este servidor se ejecutarán dos servicios web: el **Sistema Documental Interno** y una aplicación web destinada a la **predicción de inmuebles**.

1.1. Preparación del hardware

- CPU: Intel i3-4130
- RAM: 2 GB
- Disco duro/SSD: 500 GB
- Antena WiFi
- ISO de Ubuntu Server 24.04.3 LTS
- Software Rufus

1.2. Conexión a red

La siguiente configuración permite establecer la conexión WiFi en el servidor mediante **netplan**. Primero se edita el archivo de configuración, se ajustan los permisos para mayor seguridad y luego se aplican los cambios. En el archivo **wifi.yaml** se define la interfaz de red inalámbrica, el uso de **DHCP** y las credenciales del punto de acceso.

```
nano /etc/netplan/wifi.yaml
sudo chmod 600 /etc/netplan/wifi.yaml
sudo netplan apply
```

```
network:
  version: 2
  renderer: networkd
  wifis:
    wlp3s0:
      dhcp4: true
      optional: true
      access-points:
        "RESET BS":
          password: "061804159"
```

Código 1: Contenido del archivo wifi.yaml

1.3. Conexión remota

La conexión al servidor se realizará mediante `SSH` a través del puerto 22. En Windows, se debe abrir la terminal (`cmd`) y ejecutar el siguiente comando. El usuario es **noroot** y la contraseña es **noroot**:

```
ssh noroot@192.168.18.51
```

2. Despliegue Sistema Documental Interno (SDI)

Esta aplicación fue desarrollada con el objetivo de optimizar la gestión documental de la empresa, permitiendo un control más ordenado y eficiente de expedientes, trámites, partidas, planos y otros documentos administrativos. Su implementación busca reducir la duplicidad de información, agilizar los procesos internos y garantizar un acceso centralizado y seguro a la documentación

2.1. Diagrama de arquitectura

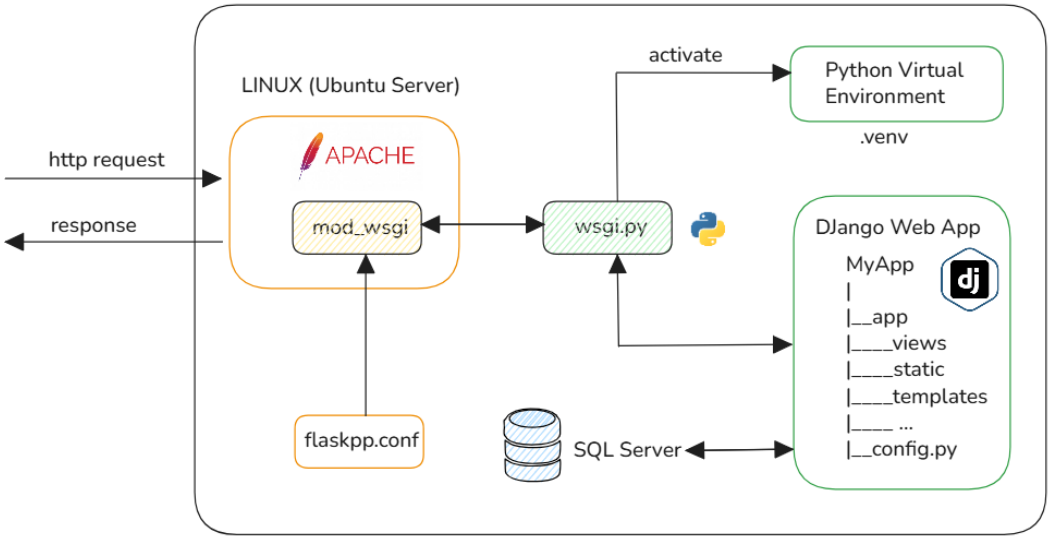

Figura 1: Despliegue de la aplicación

Dirección URL

https://github.com/santa-clara-sac/SDI

2.2. Instalación de paquetes

Los siguientes comandos son componentes principales del entorno de despliegue. Incluye la instalación de **Apache** con **mod_wsgi** para ejecutar aplicaciones **Python** y herramientas de desarrollo, así como Docker para la gestión de contenedores de bases de datos.

```
# Instalaci n Apache y mod_wsgi
sudo apt install apache2 libapache2-mod-wsgi-py3 -y
# Instalaci n Python y dependencias
sudo apt install python3 python3-pip python3-venv -y
# Instalaci n docker para el contenedor de Base de Datos
sudo apt install pkg-config default-libmysqlclient-dev build-essential
sudo apt install docker-ce docker-ce-cli containerd.io -y
```

Código 2: Paquete apache python3 y docker

Lo siguiente es configurar el firewall **UFW** (Uncomplicated Firewall) para permitir el tráfico **HTTP** y **HTTPS**, habilitando el acceso completo a los servicios de Apache desde el exterior.

```
sudo ufw allow 'Apache Full'
```

El siguiente bloque corresponde a la configuración de un VirtualHost en Apache para atender solicitudes en el **puerto 80**. El archivo a modificar es **/etc/apache2/sites-available/miapp.conf**. En él se definen los parámetros principales del servidor (nombre, administrador), la integración de la aplicación Django mediante **WSGI**, así como las rutas de acceso para archivos estáticos y archivos multimedia. Además, se especifican los registros de errores y accesos asociados al sitio.

```
<VirtualHost *:80>
    ServerName server
    ServerAdmin admin@server
    # Configuraci n de WSGI para Django
   WSGIProcessGroup sdi
    WSGIScriptAlias / /var/www/SDI/principal/wsgi.py

    <Directory /var/www/SDI/principal>
        <Files wsgi.py>
            Require all granted
        </Files>
    </Directory>

    # Archivos estaticos
    Alias /static /var/www/SDI/staticfiles
    <Directory /var/www/SDI/staticfiles>
        Require all granted
    </Directory>

    # Archivos multimedia (subidos por usuarios)
    Alias /media /opt/temp
    <Directory /opt/temp>
        Require all granted
    </Directory>

    ErrorLog ${APACHE_LOG_DIR}/miapp_error.log
    CustomLog ${APACHE_LOG_DIR}/miapp_access.log combined
</VirtualHost>
```

2.3. Base de datos SQL Server

La base de datos se implementará como un contenedor de SQL Server 2022, desplegado mediante Docker a partir de la imagen oficial provista por Microsoft.

```
docker run -e "ACCEPT_EULA=Y"
   -e "SA_PASSWORD=password123A"
   -p 1433:1433
   -v /opt/mssql/data:/var/opt/mssql/data
   --name SQLServer
   --restart unless-stopped
   -d mcr.microsoft.com/mssql/server:2022-latest
```

Se abre el puerto en el firewall de Ubuntu para permitir conexiones al servidor de base de datos desde el exterior.

```
sudo ufw allow 1433/tcp
sudo ufw reload
```

2.4. Entorno virtual y preparación de la aplicación

Para configurar el entorno y preparar la aplicación de Django para producción, se deben seguir los siguientes pasos:

1. Crear y activar un entorno virtual:

```
python -m venv .venv
source .venv/bin/activate
```

2. Generar y aplicar las migraciones en la base de datos:

```
python manage.py makemigrations
python manage.py migrate
```

3. Recopilar los archivos estáticos en una carpeta central lista para producción:

```
python manage.py collectstatic
```

4. Configurar y habilitar el sitio en Apache con WSGI:

```
sudo a2ensite miapp.conf
sudo a2enmod wsgi
sudo systemctl restart apache2
```

Tras ejecutar los comandos, será posible acceder a la aplicación mediante su dirección IP **192.168.18.51**

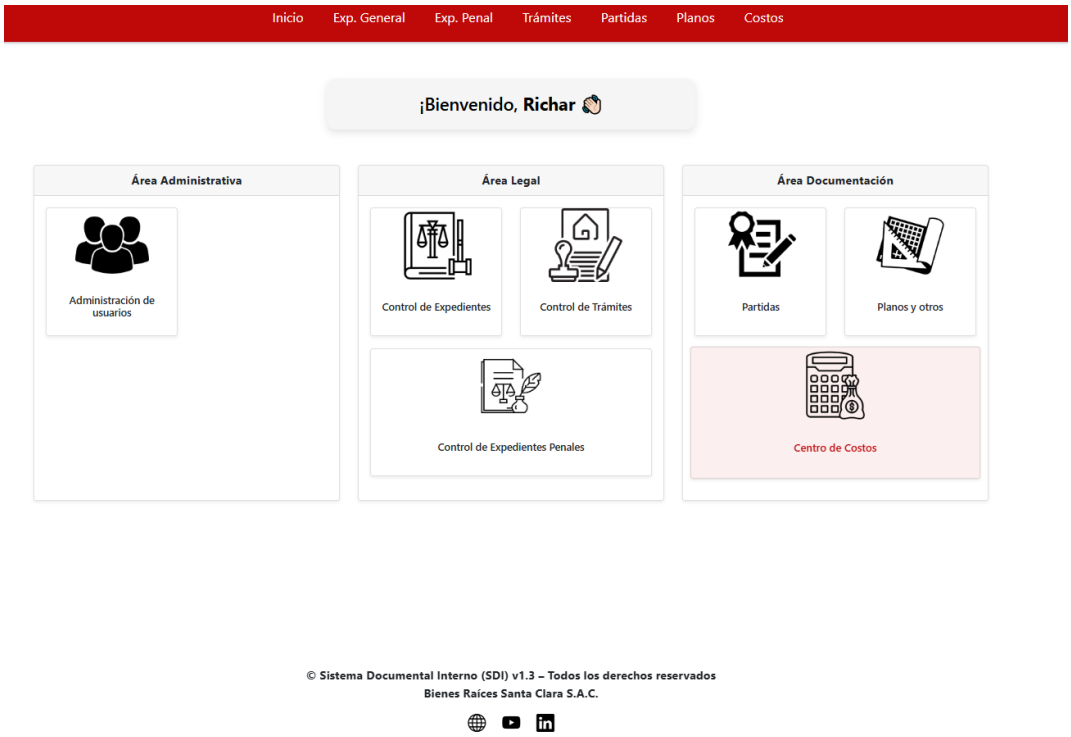

Figura 2: Página principal de la aplicación

3. Despliegue modelo Machine Learning

3.1. Diagrama de arquitectura

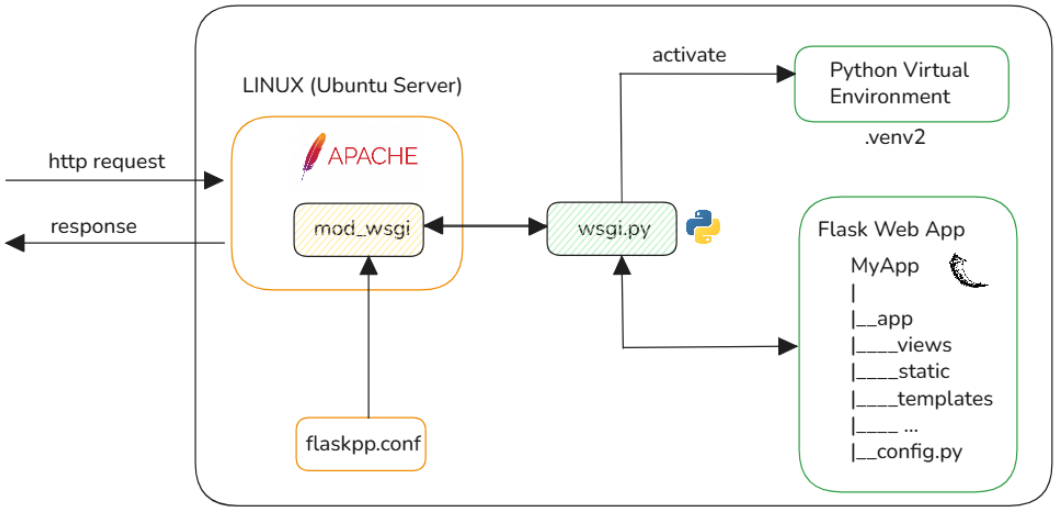

Figura 3: Página principal de la aplicación

Dirección URL

https://github.com/surcodev/precio-inmueble-ml

Para la puesta en producción del modelo de Machine Learning mediante un servidor web, se utilizó **Apache2** junto con el módulo **mod_wsgi**, el cual permite ejecutar aplicaciones desarrolladas en *Python* de forma eficiente. A continuación, se describen los pasos principales para su implementación.

3.2. Entorno virtual

En primer lugar, se debe crear un entorno virtual dentro del directorio donde reside la aplicación Flask, con el fin de aislar las dependencias necesarias para la ejecución del modelo:

```
cd /var/www/flaskapp
python3 -m venv .venv2
source .venv2/bin/activate
pip install -r requirements.txt
```

Posteriormente, se crea el archivo **WSGI** en la ruta **/var/www/flaskapp/flaskapp.wsgi**, el cual actúa como punto de entrada entre Apache y la aplicación Flask:

```
import sys
import logging
logging.basicConfig(stream=sys.stderr)
sys.path.insert(0, "/var/www/flaskapp")
from app import app as application
```

Código 3: Contenido del archivo flaskapp.wsgi

3.3. Configuración de Apache

Una vez configurado el archivo WSGI, es necesario definir la configuración de Apache. Para ello, se genera el archivo **/etc/apache2/sites-available/flaskapp.conf**, en el cual se especifican parámetros como el puerto de escucha, la dirección del servidor y la ubicación del archivo WSGI. En este caso, se utilizó el **puerto 7070** para el despliegue:

```
<VirtualHost *:7070>
    ServerName 192.168.18.51
    ServerAdmin admin@localhost

    WSGIDaemonProcess flaskapp python-path=/var/www/flaskapp:/var/www/flaskapp
   /.venv2/lib/python3.12/site-packages
    WSGIProcessGroup flaskapp
    WSGIScriptAlias / /var/www/flaskapp/flaskapp.wsgi

    <Directory /var/www/flaskapp>
        Require all granted
    </Directory>

    ErrorLog ${APACHE_LOG_DIR}/flaskapp_error.log
    CustomLog ${APACHE_LOG_DIR}/flaskapp_access.log combined
</VirtualHost>
```

Edita el archivo **/etc/apache2/ports.conf** y agrega la siguiente línea:

```
Listen 7070
```

Finalmente, se habilita el sitio configurado, se abre el puerto correspondiente en el firewall y se reinicia el servicio de Apache para aplicar los cambios:

```
sudo a2ensite flaskapp
sudo ufw allow 7070/tcp
sudo systemctl restart apache2
sudo systemctl status apache2
```

Figura 4: Página principal del modelo de despliegue de ML

4. Síntesis

En la Tabla 1 se presentan los puertos abiertos en el servidor junto con los servicios asociados que se encuentran en ejecución. Esta información permite identificar los accesos habilitados y los procesos que se mantienen activos en el entorno.

Puerto	Servicio
22	SSH
80	HTTP
139	NetBIOS-SSN
445	Microsoft-DS
1433	MS-SQL-S
7070	RealServer

Tabla 1: Puertos abiertos y servicios activos en el servidor

Adicionalmente, se configuró una carpeta compartida mediante **Samba**, con el propósito de facilitar la transferencia de archivos hacia el sistema de forma ágil. Esta carpeta es accesible desde equipos con sistema operativo Windows utilizando el atajo de teclado `Windows + R` e ingresando la dirección de red:

```
192.168.18.51\temp
```

De esta manera, los usuarios pueden gestionar documentos directamente desde su estación de trabajo sin necesidad de acceder al servidor por línea de comandos.

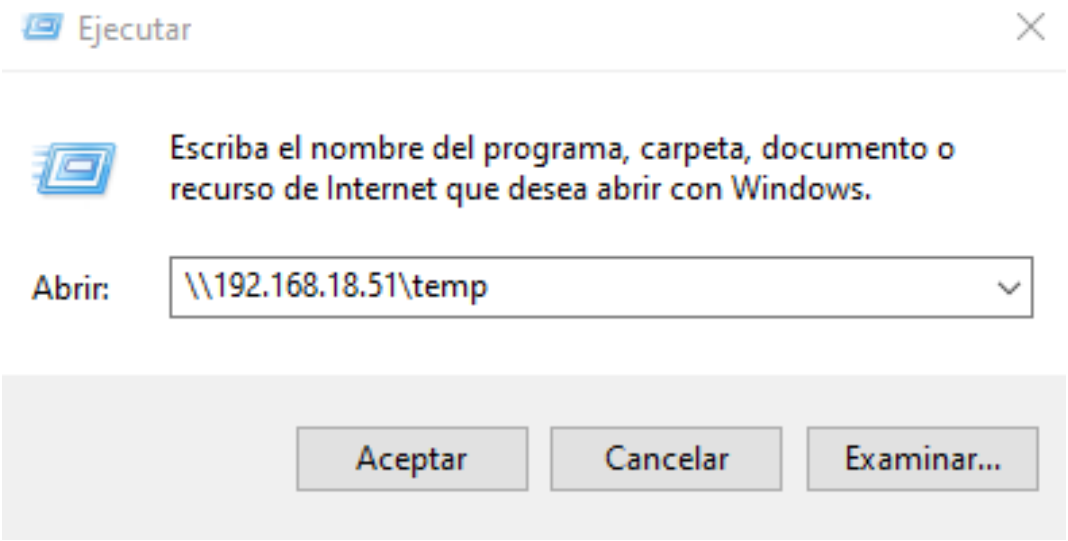

Figura 5: Acceso a la carpeta compartida mediante Samba desde Windows