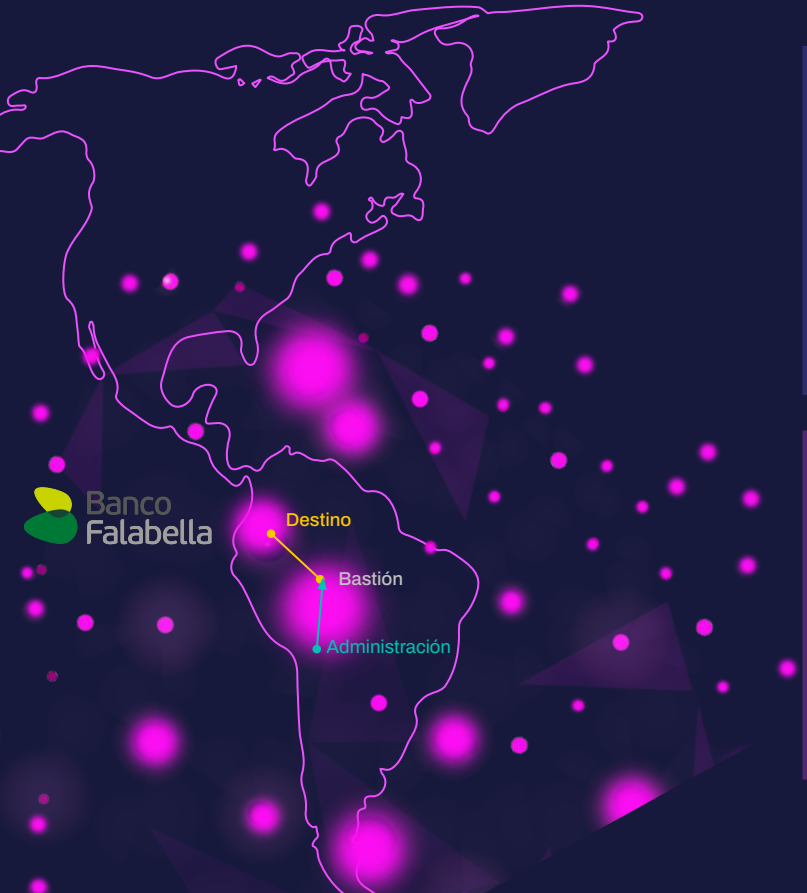
# Automatización del acceso vía SSH a la administración de infraestructura Linux a través de un servidor de gestión

Administración de Sistemas Informáticos y Redes
IES Zaidín-Vergeles
Laura Ramos Granados

Granada, junio 2025

# Contexto y problemática: Entorno segmentado y acceso crítico




- Conocer o indagar manualmente las rutas de acceso
- Ejecutar múltiples conexiones para llegar a un destino
- Verificar la conectividad uno a uno

- Consumo de tiempo
- Mayor carga operativa
- Riesgo de error humano

# Solución propuesta y objetivo: Automatización de accesos


Script de automatización en Python

| | | |
|---|---|---|
| 1 | Consulta de BD | Centralizar la gestión de accesos desde un único servidor |
| 2 | Encuentra la ruta | Minimizar la carga operativa y la dependencia del conocimiento individual |
| 3 | Ejecuta saltos SSH | Establecer una conexión remota eficiente, segura y estandarizada |
| 4 | Logra sesión en destino | Administrar a los servidores finales sin intervención manual directa |

# Entorno de pruebas: Red segmentada y aprovisionamiento

## Stack de aprovisionamiento y virtualización

**Alma Linux**
Sistema operativo base

**Vagrant**
Orquestación de máquinas

**VirtualBox**
Proveedor de virtualización

## Nodos del entorno de pruebas

**admin-server**
Orquesta el salto y contiene la BD

**bastion**
Nodo intermedio

**destination**
Máquina objetivo

## Script de configuración inicial

**1** Establece confianza **SSH**

**2** Instala **PostgreSQL** y configura la BD

**3** Crea el entorno virtual de Python

# Script de automatización:
## Arquitectura modular

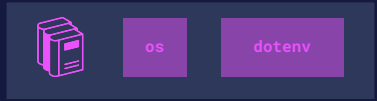
```
entorno_pruebas/
├── Vagrantfile
├── data.sql
├── .env
├── scripts/
│   ├── common_functions.sh
│   ├── provision_admin.sh
│   ├── provision_bastion.sh
│   └── provision_destination.sh
├── remote_access_tool/
│   ├── __init__.py
│   ├── app.py                  # Interfaz web con Flask (Dashboard)
│   ├── config.py               # Carga .env y configuración de logging
│   ├── db.py                   # Módulo de acceso a PostgreSQL
│   ├── interactive.py          # Sesión SSH interactiva
│   ├── main.py                 # Punto de entrada del script CLI
│   ├── ssh.py                  # Funciones para conexión SSH
│   ├── utils.py                # Script completo
│   ├── requirements.txt
│   ├── static/
│   │   └── css/
│   │       └── style.css
│   └── templates/
│       ├── base.html
│       ├── dashboard.html
│       └── index.html
├── tests/
│   ├── integration/
│   │   ├── test_parametro_invalido.py
│   │   └── test_salto_completo.py
│   └── unit/
│       ├── __init__.py
│       ├── test_db.py
│       └── test_ssh.py
├── .gitignore
└── README.md
```

# Script de automatización: Configuración global del entorno

1

### Carga segura del entorno

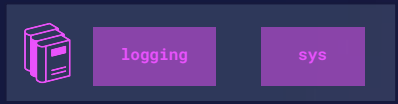
Desde .env para base de datos y conexión SSH




2

### Registro de eventos

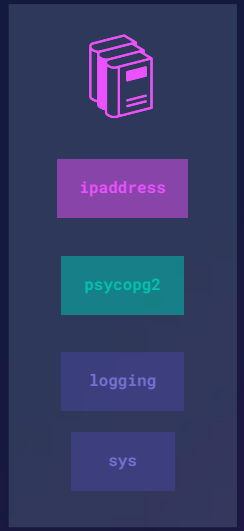
Registra la ejecución del script en archivo persistente y consola




```python
import os
from dotenv import load_dotenv
import logging

# Cargar variables desde .env
load_dotenv()

# Configurar logging
logging.basicConfig(
    level=logging.INFO,
    format="%(asctime)s - %(levelname)s - %(message)s",
    handlers=[
        logging.FileHandler("app.log"),
        logging.StreamHandler()
    ]
)

# Variables de entorno
DB_USER = os.getenv("DB_USER")
DB_PASS = os.getenv("DB_PASS")
DB_HOST = os.getenv("DB_HOST")
DB_NAME = os.getenv("DB_NAME")
DB_PORT = os.getenv("DB_PORT", "5432")

SSH_USER = os.getenv("SSH_USER")
SSH_KEY_PATH = os.getenv("SSH_KEY_PATH", "~/.ssh/id_rsa")
```

# Script de automatización: Conexión y consulta a BD

1. Carga las variables de entorno
2. Valida el parámetro recibido
3. Establece conexión con la BD
4. Ejecuta la consulta SQL




```
def validar_parametro(nombre_o_ip):
    try:
        ipaddress.ip_address(nombre_o_ip)
        logging.info(f"El parámetro proporcionado es una
dirección IP: {nombre_o_ip}")
        return "ip"
    except ValueError:
        logging.info(f"El parámetro proporcionado es un nombre:
{nombre_o_ip}")
        return "nombre"
```

```
def connect_to_db():
    try:
        connection = psycopg2.connect(
            user=DB_USER,
            …
        return connection
    except psycopg2.Error as e:
        logging.error(f"Error al conectar a la base de datos: {e}")
        sys.exit(1)
```

```
def obtener_maquina_y_bastion(nombre_o_ip):
  …
    try:
…
                query = f"""
                    SELECT m.name, m.ip, b.bastion_ip
                    FROM machines AS m
                    INNER JOIN bastions AS b
                    ON m.country = b.country AND m.environment = b.environment
                    WHERE m.{"ip" if tipo_parametro == "ip" else "name"} = %s;
                """
…
```

# Script de automatización: Conexión SSH y salto

```
def conectar_ssh_con_claves(hostname, username, clave_privada, bastion=None)
```

**1** Instancia el cliente SSH

```
ssh_client  = paramiko.SSHClient()
```

**2** Valida y carga la clave SSH

```
    private_key_path = os.path.expanduser(clave_privada)
        if not os.path.isfile(private_key_path):
            logging.error(...)
            return None
        logging.info(...)
        clave = paramiko.RSAKey.from_private_key_file(private_key_path)
```

**3** Lógica de conexion SSH

```
bastion_transport = bastion.get_transport()
tunnel = bastion_transport.open_channel("direct-tcpip", (hostname, 22),
("127.0.0.1", 0))
ssh_client.connect(hostname, username=username, pkey=clave, sock=tunnel)
```

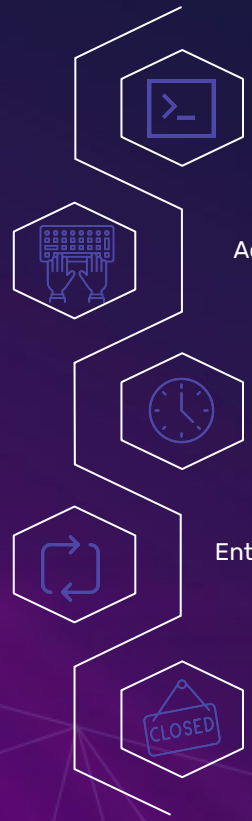
**4** Manejo de errores

```
except Exception as e:
        logging.error(f"Error conectando a {hostname}
mediante SSH: {e}")
        return None
```

**5** Si todo sale bien…

```
return ssh_client
```

# Script de automatización: Sesión remota interactiva




Guarda la configuración actual de la terminal

`termios.tcgetattr()`

Activa modo raw, para leer carácter a carácter

`tty.setraw()`

Espera datos desde el usuario o desde el servidor, sin bloquear

`select.select()`

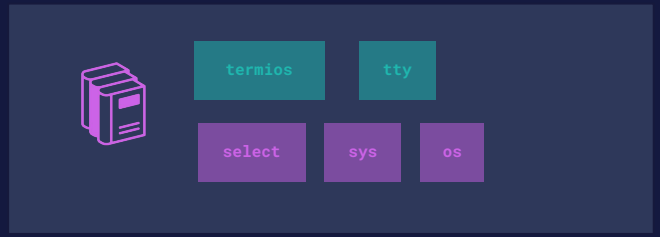
Entra en un bucle continuo

`os.read()`
`os.write()`

Lee lo que escribes y lo envía a la sesión SSH del servidor destino

Lee lo que responde el servidor y lo imprime en pantalla

Cuando se cierra la sesión, restaura el modo normal del terminal.



# Script de automatización: Interfaz web (dashboard)

Inicialización de la aplicación Flask

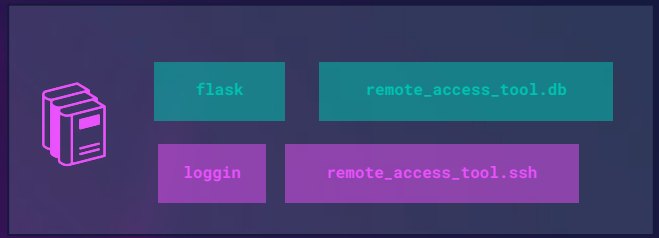
Formulario de entrada

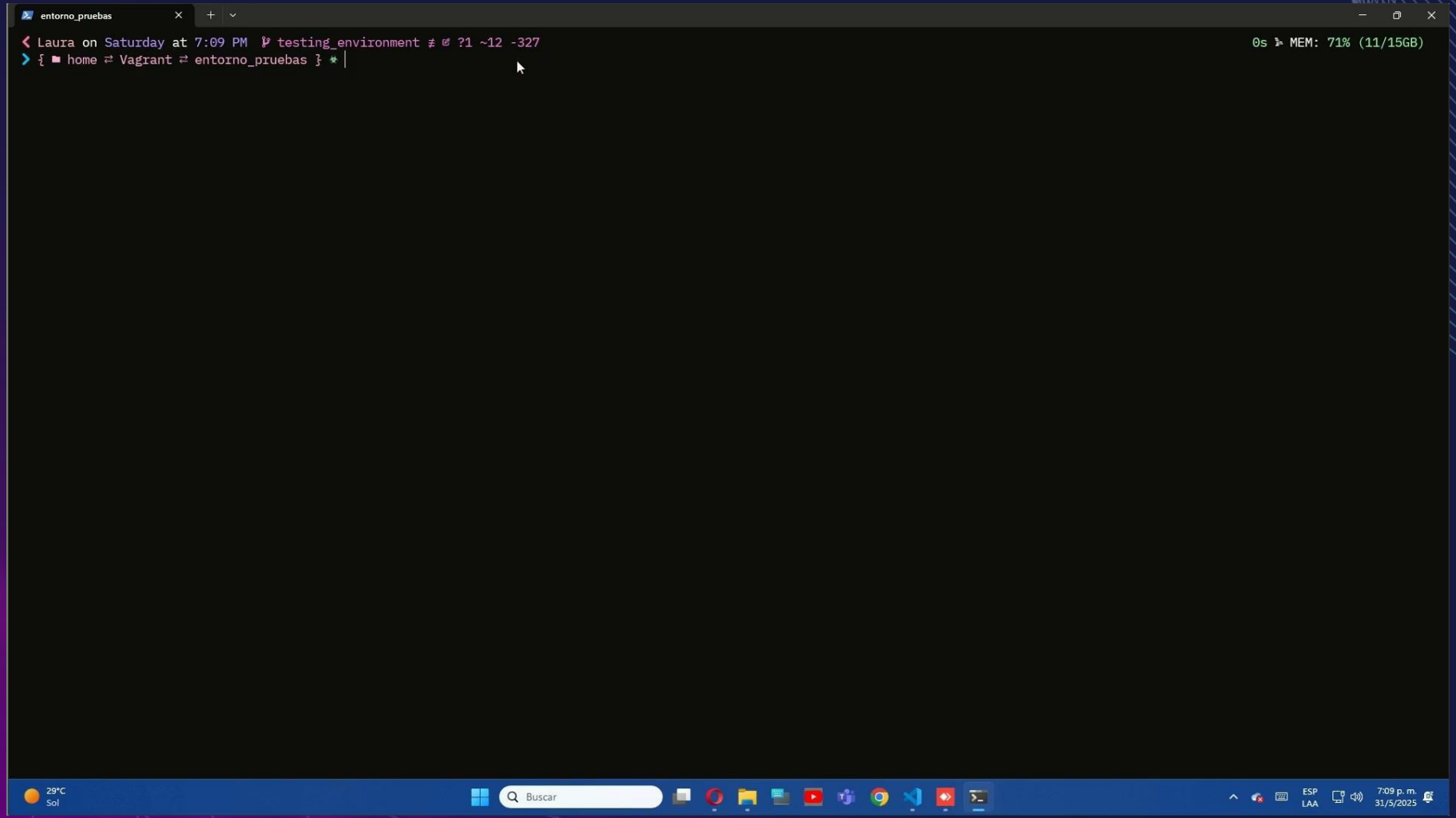
Validación de entrada

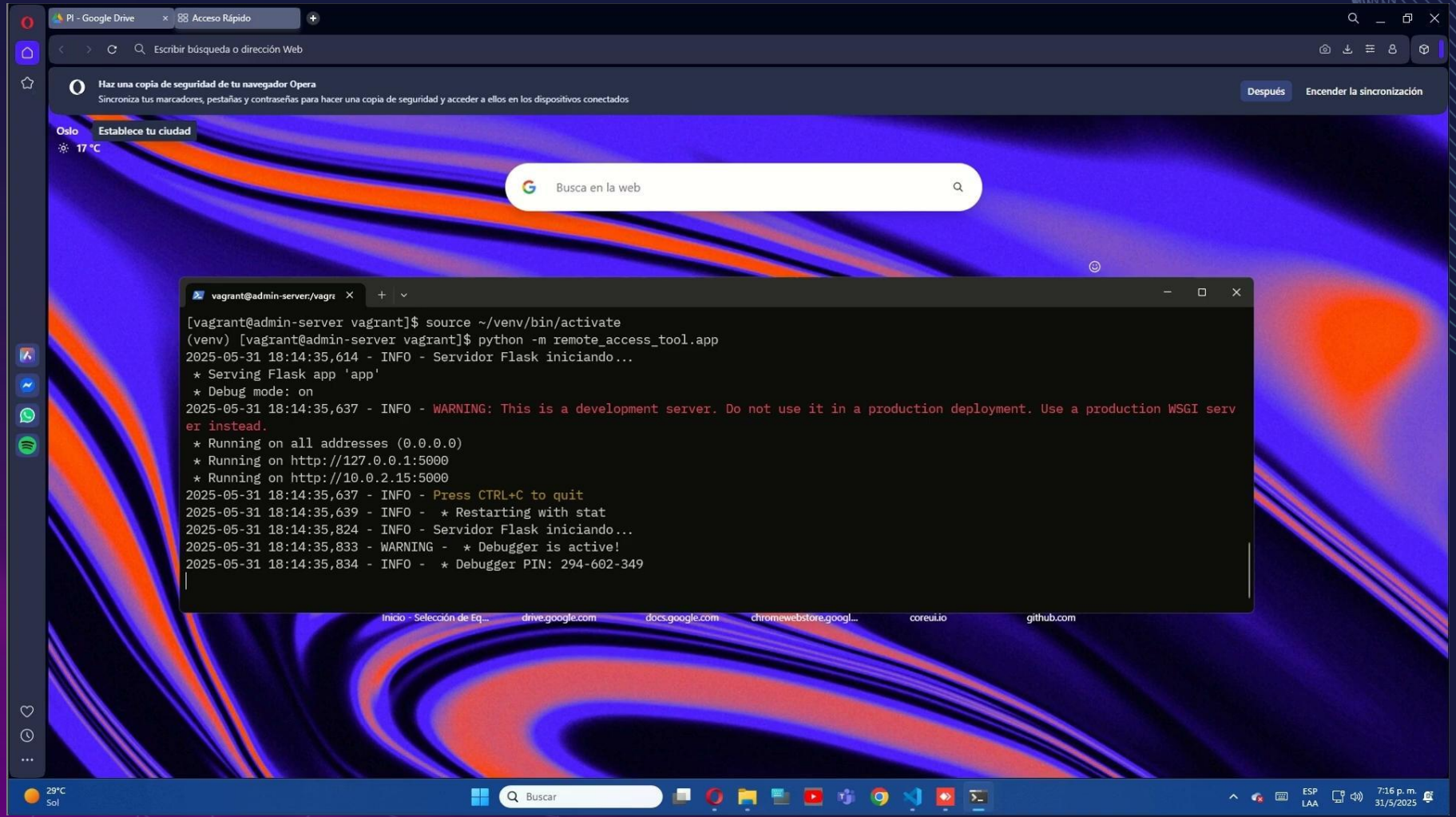
Obtiene los datos y conexión

Salto SSH, ejecución de comandos y procesamiento

Renderizado del dashboard (Ninja2)




entorno_pruebas
Laura on Saturday at 7:09 PM testing_environment ≠ ?1 ~12 -327
0s MEM: 71% (11/15GB)
{ home ⇄ Vagrant ⇄ entorno_pruebas }
29°C
Sol
Buscar
ESP
LAA
7:09 p. m.
31/5/2025


PI - Google Drive
Acceso Rápido
Escribir búsqueda o dirección Web
Haz una copia de seguridad de tu navegador Opera
Sincroniza tus marcadores, pestañas y contraseñas para hacer una copia de seguridad y acceder a ellos en los dispositivos conectados
Después
Encender la sincronización
Oslo
Establece tu ciudad
17 °C
Busca en la web
vagrant@admin-server:/vagra
[vagrant@admin-server vagrant]$ source ~/venv/bin/activate
(venv) [vagrant@admin-server vagrant]$ python -m remote_access_tool.app
2025-05-31 18:14:35,614 - INFO - Servidor Flask iniciando...
* Serving Flask app 'app'
* Debug mode: on
2025-05-31 18:14:35,637 - INFO - WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://10.0.2.15:5000
2025-05-31 18:14:35,637 - INFO - Press CTRL+C to quit
2025-05-31 18:14:35,639 - INFO - * Restarting with stat
2025-05-31 18:14:35,824 - INFO - Servidor Flask iniciando...
2025-05-31 18:14:35,833 - WARNING - * Debugger is active!
2025-05-31 18:14:35,834 - INFO - * Debugger PIN: 294-602-349
Inicio - Selección de Eq...
drive.google.com
docs.google.com
chromewebstore.googl...
coreui.io
github.com
29°C
Sol
Buscar
ESP
LAA
7:16 p. m.
31/5/2025

# Gracias por su atención

Gracias a todo el profesorado el ciclo ASIR del IES Zaidín Vergeles por su dedicación.

En especial a la profesora Aurora (Administración de Sistemas Operativos) y al profesor Miguel Ángel Moreno Garrido (Planificación y Administración de Redes), por enseñarme tanto, contagiarme su pasión por estas asignaturas y motivarme con su forma de enseñar.

Y gracias también a mi tutor Jaime Fernández Ortega por su acompañamiento y apoyo constante durante todo el proyecto.

Laura Ramos Granados

CREDITS: This presentation template was created by **Slidesgo**, including icons by **Flaticon**, and infographics & images by **Freepik**

Please keep this slide for attribution