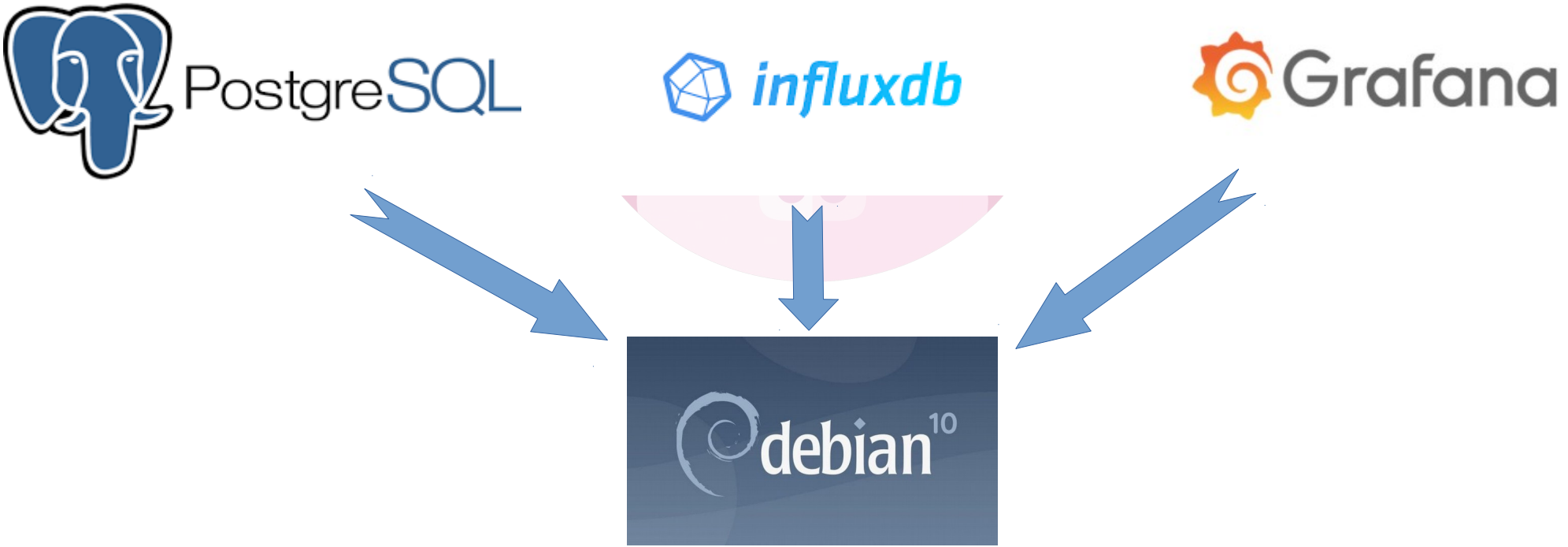
# PIG

## PostgreSQL, InfluxDB and Grafana

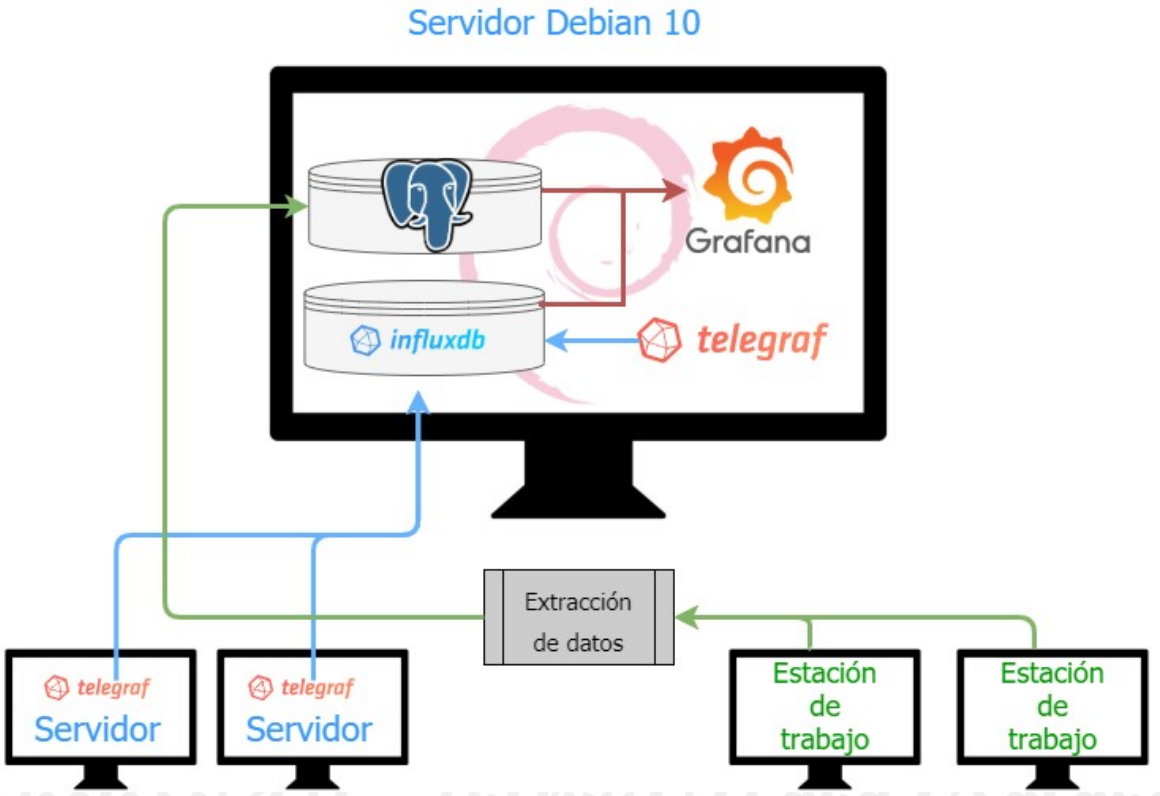
SANTIAGO PREGO LÓPEZ – ASIR 2020 – IES SAN CLEMENTE

# ¿Qué es PIG?




PostgreSQL, InfluxDB and Grafana

# Diagrama de componentes



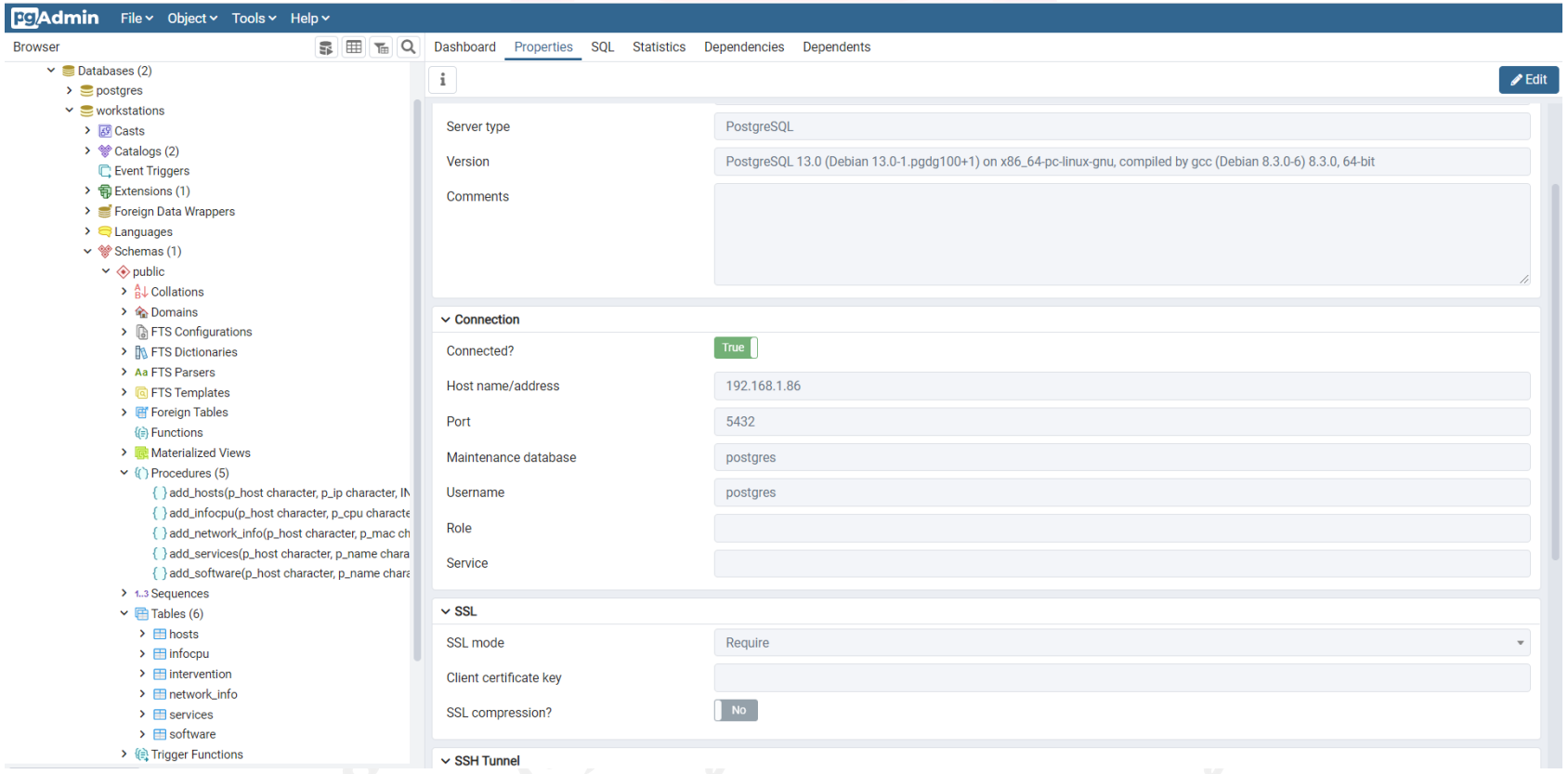
# Trabajo Realizado

Instalación Debian
Instalación PostgreSQL
Instalación Influxdb
Instalación Telegraf
Instalación Grafana
Escaneo

# Instalación Debian 10

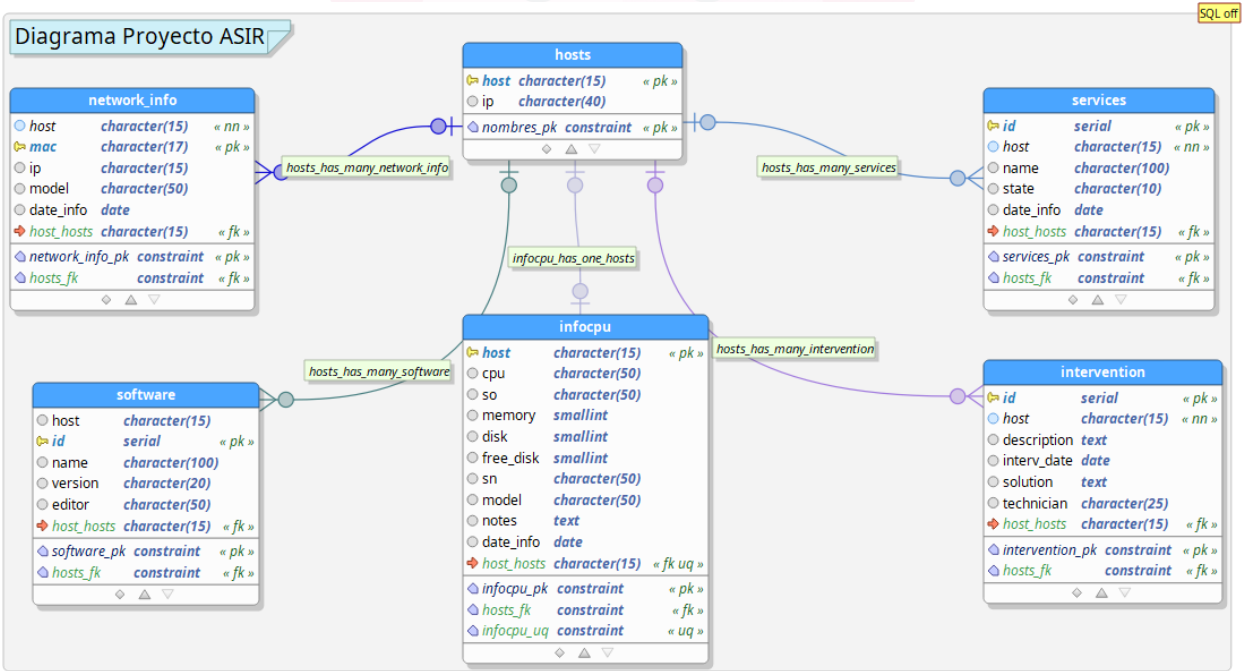
Realicé una instalación mínima sin entorno de escritorio y fui agregando lo paquetes necesarios para reducir el tamaño final.

# Instalación PostgreSQL 13



# Base de datos

Se creó una base de datos de muestra para almacenar los datos recopilados por el escaneo.



# Base de datos

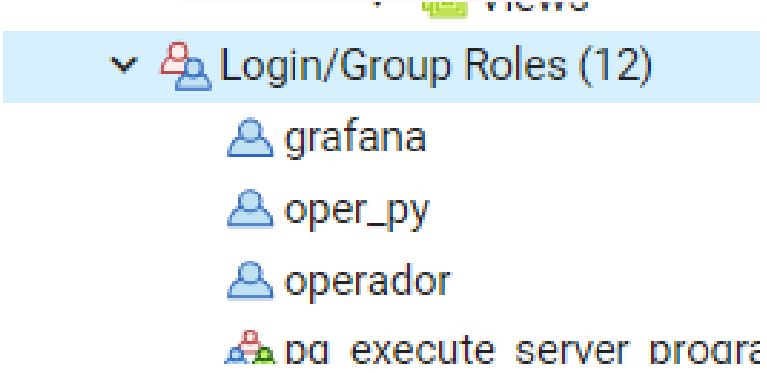
## Cree procedimientos almacenados para la inserción de los datos.

Code

```
BEGIN
    INSERT INTO infocpu(host, cpu, so, memory, disk, free_disk, sn, model)
    VALUES (P_host, P_cpu, P_so, P_memory, P_disk, P_free_disk, P_sn, P_model)
        ON CONFLICT (host)
    DO
        UPDATE SET cpu = P_cpu, so = P_so, memory = P_memory, disk = P_disk,
        free_disk = P_free_disk, sn = P_sn, model = P_model, date_info = NOW();

EXCEPTION WHEN OTHERS THEN
        P_msg_error := SQLERRM;
END;

```

# Base de datos

Se crearon diferentes usuarios para limitar el acceso a lo necesario.



# Certificado auto firmado

Se crea el certificado y la clave privada para configurar posteriormente https en las aplicaciones.

```
openssl req -x509 -out localhost.crt -keyout localhost.key \
-newkey rsa:2048 -nodes -sha256 \
-subj '/CN=localhost' -extensions EXT -config <( \
printf "[dn]\nCN=localhost\n[req]\ndistinguished_name = dn\n[EXT]\
nsubjectAltName=DNS:localhost\nkeyUsage=digitalSignature\
nextendedKeyUsage=serverAuth")
```

# Influxdb 1.8

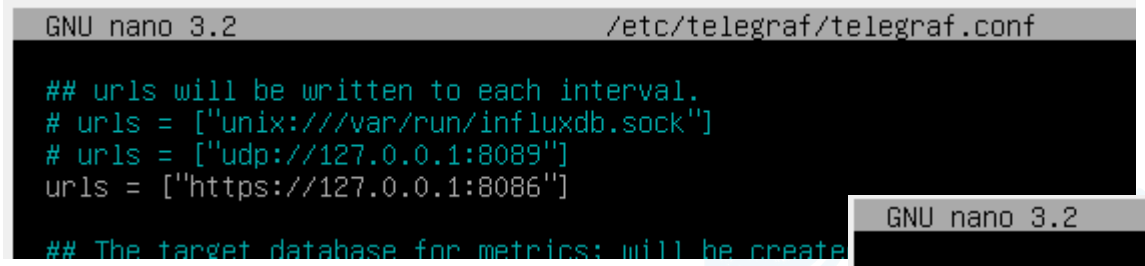
Se instaló y configuró Influxdb habilitando https y creando un usuario de acceso.

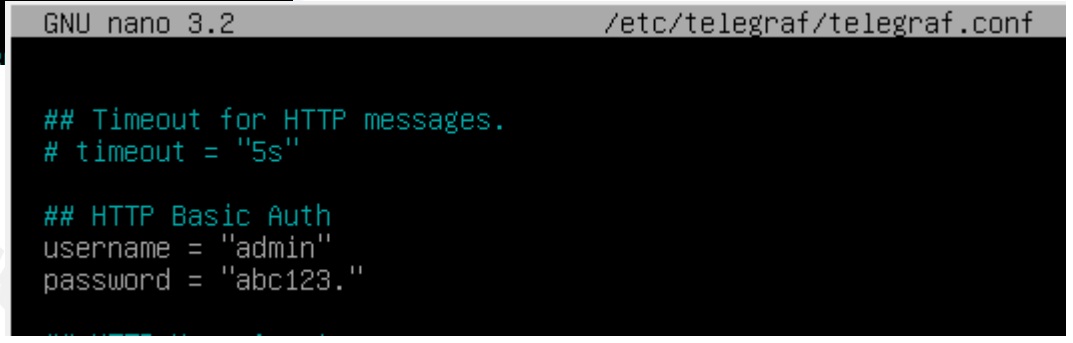
```
GNU nano 3.2                        /etc/influxdb/influxdb.conf

# ping-auth-enabled = false

# Determines whether HTTPS is enabled.
https-enabled = true

# The SSL certificate to use when HTTPS is enable
https-certificate = "/etc/influxdb/localhost.crt"

# Use a separate private key location.
https-private-key = "/etc/influxdb/localhost.key"
```

```
root@pig-server:~# influx -ssl -unsafeSsl -host localhost -username admin -password abc123.
Connected to https://localhost:8086 version 1.8.3
InfluxDB shell version: 1.8.3
> show users
user    admin
----    -----
admin   true
grafana false
tg_user false
> show databases
name: databases
name
----
_internal
telegraf
>
```

# Instalación Telegraf

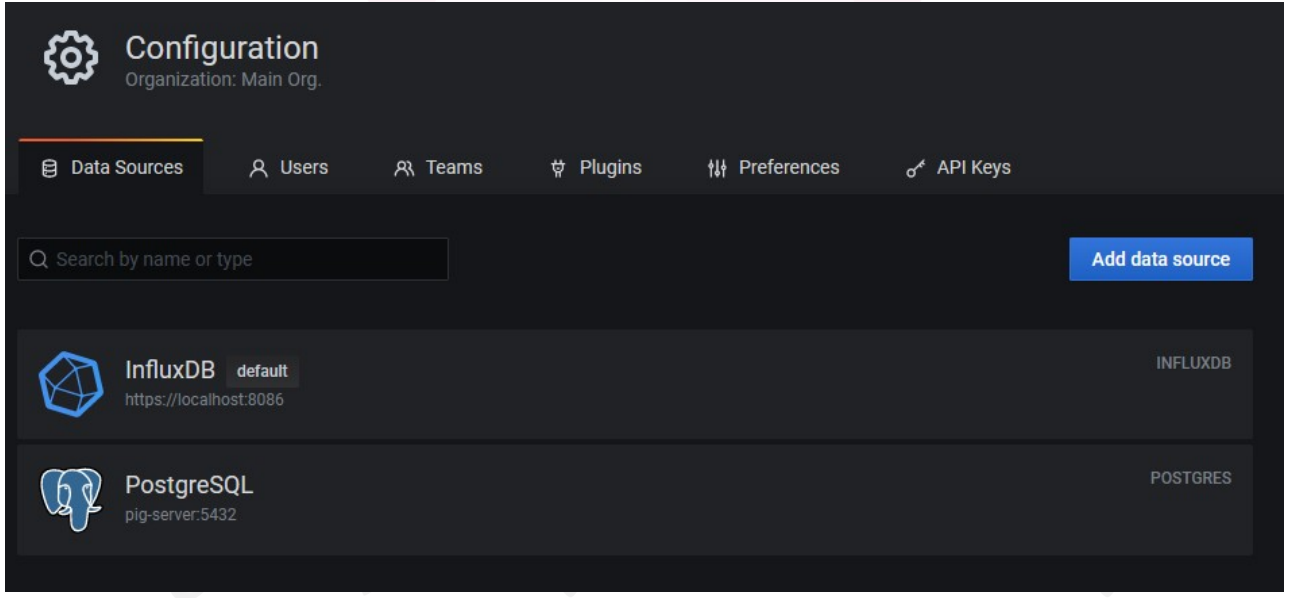
Se realizó una instalación básica de Telegraf con la base de datos por defecto, y se configuró el acceso a Influxdb.



# Instalación de Grafana 7.2

## Se instala y configuran las fuentes de datos en Grafana



# Instalación de Grafana 7.2

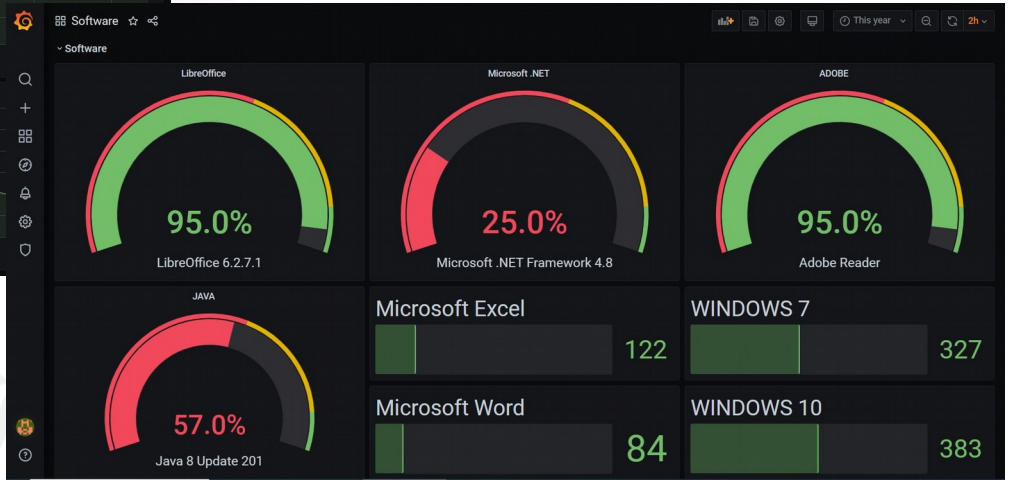
## Se habilita la conexión a través de https

```
 GNU nano 3.2                                  /etc/grafana/grafana.ini

#################################### Server ####################
[server]
# Protocol (http, https, h2, socket)
protocol = https
```

```
 GNU nano 3.2                                  /etc/grafana/grafana.ini

# https certs & key file
cert_file = /etc/grafana/localhost.crt
cert_key = /etc/grafana/localhost.key
```

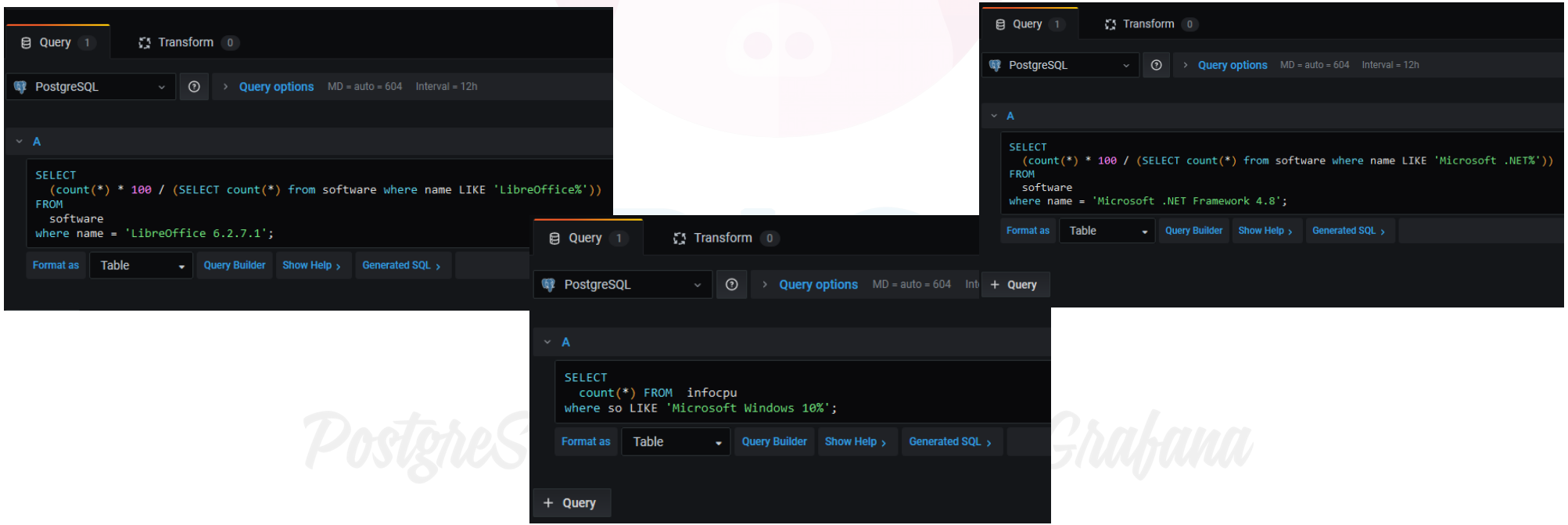
# Instalación de Grafana 7.2

Se crean paneles de muestra para el servidor y los datos de las estaciones de trabajo.

# Instalación de Grafana 7.2

Para el panel de software se usaron consultas de ejemplo sobre la base de datos PostgreSQL.



# Escaneo de equipos

Se creó una muestra de software para escaneo de equipos usando Python y WMI.

Se realizó un escaneo de prueba obteniendo datos de casi 800 equipos.

# Escaneo de equipos

Se puede escanear mediante un rango de IP o usando un listado de equipos.

```python
import data_module as dataw
import dbpg_module as dbpg
import socket

red_ini = [192,168,1,86] # Primera IP donde comenzará el escaneo
red_fin = [192,168,1,142] # Ultima IP que escaneará


# Comprobamos si el host admite la conexion al puerto 135 (RPC) us
def scan(IP):
    s = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
    socket.setdefaulttimeout(0.50)
```

```python
    try:
        #En exta conexión se especifica un usuario concreto con el que conectarse a los equipos remotos, este debe de tener permisos para realiza
        #c = wmi.WMI(computer=computer, impersonation_level='impersonate', authentication_level='pktPrivacy', user="sprego", password="abc123.")
        c = wmi.WMI(computer=computer, impersonation_level='impersonate', authentication_level='pktPrivacy') # Con esta conexión el usuario que e
        for interface in c.Win32_NetworkAdapterConfiguration(IPEnabled=1): # IPEnabled=1 indica la tarjeta activa
            model = interface.description
            mac = interface.MACAddress
            ip = interface.IPAddress[0]
            dbpg.SaveData().network_info(computer,mac,ip,model)

    except wmi.x_wmi as e:
        print(e.com_error.strerror,e.com_error.excepinfo[2])
        return e.com_error.strerror,e.com_error.excepinfo[2]
```

```python
try:
    with open("PCS.txt", "r", encoding='utf-8') as PCS: # Abrimos el fichero donde almacenamos los equipos a escanear.
        hosts = PCS.readlines()
    i = 0
    for host in hosts:
        i=i+1
        host=host.split("\n")
        scan(host[0]) # Escaneamos cada host para ver si nos podemos conectar.

except IOError as e:
    print("I/O error({0}): {1}".format(e.errno, e.strerror))
```

# Tiempos de escaneo

Se realizaron varias pruebas para comprobar el tiempo que llevaba el escaneo.

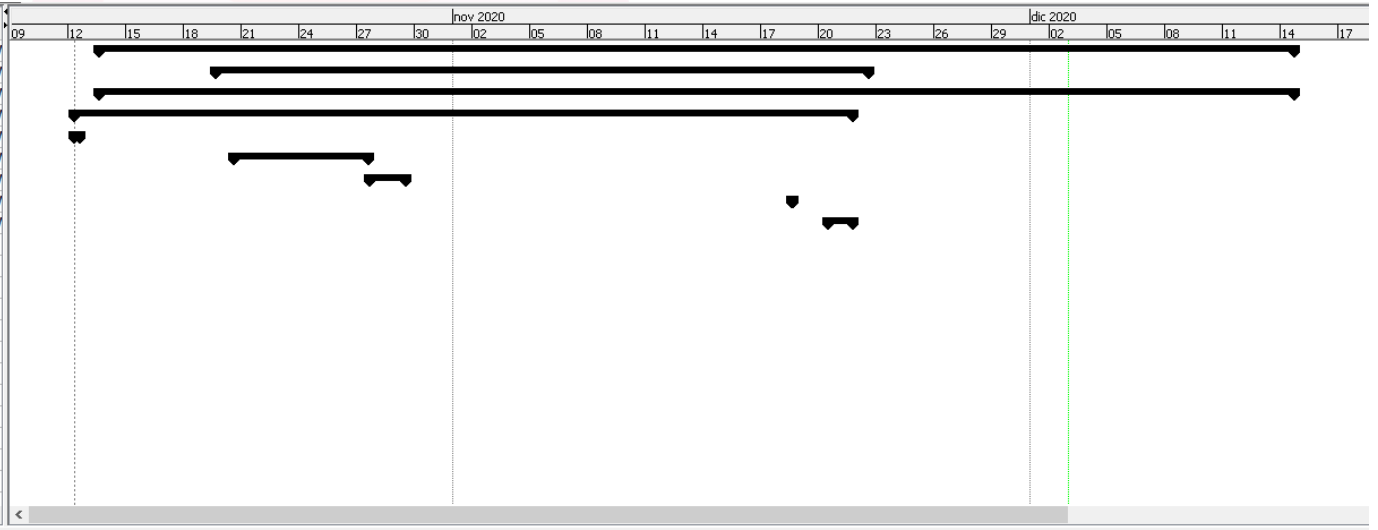
Escaneo de lista de 800 pcs
Detectados 733 equipos
Tiempo empleado entre 18 y 24 horas

Escaneo de 102 ips
Detectados 25 equipos
Tiempo empleado 23 Minutos

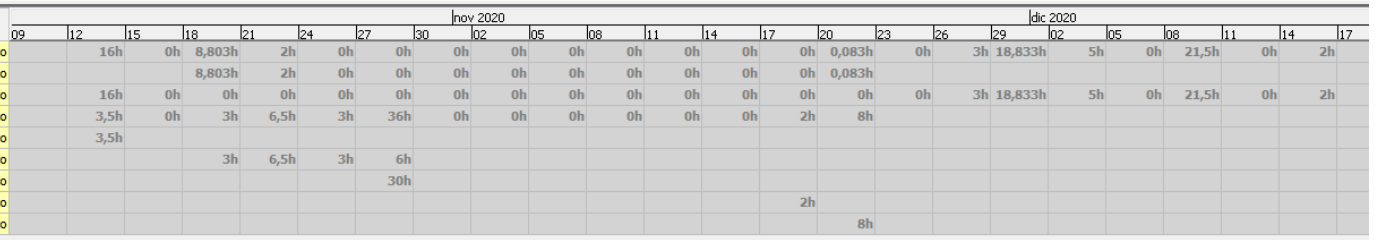
Escaneo de 253 ips
Detectados 63 equipos
Tiempo Empleado 198 minutos

# Diagrama de Gantt

| | | Nombre | Duracion | Inicio | |
|---|---|---|---|---|---|
| 1 | | Fase 1 - Entregable y Documentación | 186,375 days? | 13/10/20 15:00 | 14/ |
| 2 | ✔ | Creacion del entregable | 101,906 days? | 19/10/20 16:30 | 22/ |
| 14 | | Documentación | 186,375 days? | 13/10/20 15:00 | 14/ |
| 29 | ✔ | Fase 2 - Software Escaneo | 121,5 days? | 12/10/20 8:00 | 21/ |
| 30 | ✔ | Modulo de escaneo de red. | 1 day? | 12/10/20 8:00 | 12/ |
| 33 | ✔ | Modulo de almacenamiento de datos. | 21 days? | 20/10/20 15:00 | 27/ |
| 38 | ✔ | Modulo de extraccion de datos. | 5,688 days? | 27/10/20 16:30 | 29/ |
| 42 | ✔ | Modulo de escaneo de lista | 0,25 days? | 18/11/20 15:00 | 18/ |
| 44 | ✔ | Pruebas de escaneo | 3,875 days? | 20/11/20 13:00 | 21/ |




| | Nombre | Trabajo | Duracion | | |
|---|---|---|---|---|---|
| 1 | Fase 1 - Entregable y Documentación | 77,22 horas | 186,375 days? | 13/10/ | Trabajo |
| 2 | Creacion del entregable | 10,887 horas | 101,906 days? | 19/10/ | Trabajo |
| 14 | Documentación | 66,333 horas | 186,375 days? | 13/10/ | Trabajo |
| 29 | Fase 2 - Software Escaneo | 62 horas | 121,5 days? | 12/10/ | Trabajo |
| 30 | Modulo de escaneo de red. | 3,5 horas | 1 day? | 12/10/ | Trabajo |
| 33 | Modulo de almacenamiento de datos. | 18,5 horas | 21 days? | 20/10/ | Trabajo |
| 38 | Modulo de extraccion de datos. | 30 horas | 5,688 days? | 27/10/ | Trabajo |
| 42 | Modulo de escaneo de lista | 2 horas | 0,25 days? | 18/11/ | Trabajo |
| 44 | Pruebas de escaneo | 8 horas | 3,875 days? | 20/11/ | Trabajo |



# Distribución de tiempos