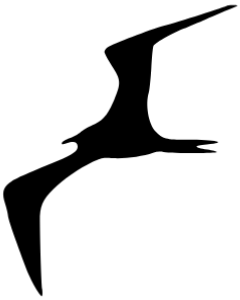
Laboratorio de Sistemas Operativos y Redes

Tecnicatura Universitaria en Programación Informática



FRIGATE

Materia:	**Laboratorio de Sistemas Operativos y Redes**
Profesor:	**Jose Luis Dibiase**
Integrantes:	**Bechtholdt Cristopher** **Correa Sebastian** **Ferreyra Valentín**
Fecha:	**11 de diciembre de 2025**

1. Introducción

Frigate es un sistema de grabación de video en red (NVR - Network Video Recorder) de código abierto diseñado específicamente para vigilancia doméstica y automatización del hogar. A diferencia de los sistemas de monitoreo tradicionales que se basan únicamente en detección de movimiento, *Frigate* utiliza inteligencia artificial (IA) para realizar detección de objetos en tiempo real sobre transmisiones de cámaras IP.

Algunas de las características principales de *Frigate* son:

- Utiliza hardware especializado como el Google Coral TPU (Tensor Processing Unit) para realizar **detecciones rápidas** y **precisas** de objetos directamente en el dispositivo, garantizando que todo el procesamiento se realice localmente sin depender de servicios en la nube.
- Se integra de manera nativa con **Home Assistant**, permitiendo automatizaciones complejas y notificaciones basadas en eventos de detección.
- Ofrece una **interfaz web** completa para configuración y visualización en vivo de las cámaras, con soporte para tecnologías como WebRTC y MSE (Media Source Extensions), proporcionando visualización en tiempo real con baja latencia.

Frigate está construido utilizando tecnologías modernas y se ejecuta típicamente en contenedores **Docker**, lo que facilita su despliegue y mantenimiento. Su arquitectura permite, entre otras cosas, procesamiento distribuido de múltiples cámaras y streaming en tiempo real.

Este informe tiene como objetivo analizar el proceso de instalación, configuración y funcionamiento de *Frigate*, así como su integración en un entorno de sistemas operativos y redes, evaluando sus capacidades como solución de monitoreo de video.

2. Hardware utilizado

El equipo utilizado para la instalación y utilización de Frigate cuenta con los siguiente requisitos:

- Sistema operativo: (VM) Windows 11, ejecutando Oracle VirtualBox con Ubuntu 24.04
- CPU: Intel i7-1165G7
- RAM: 16,0gb ram DDR4
- Almacenamiento: SSD 477gb
- Dispositivos capaces de soportar RTSP (Real-Time Streaming Protocol), con su correspondiente cámara, buena conectividad y capacidades de procesamiento y memoria.
    - Celular Poco X7 PRO

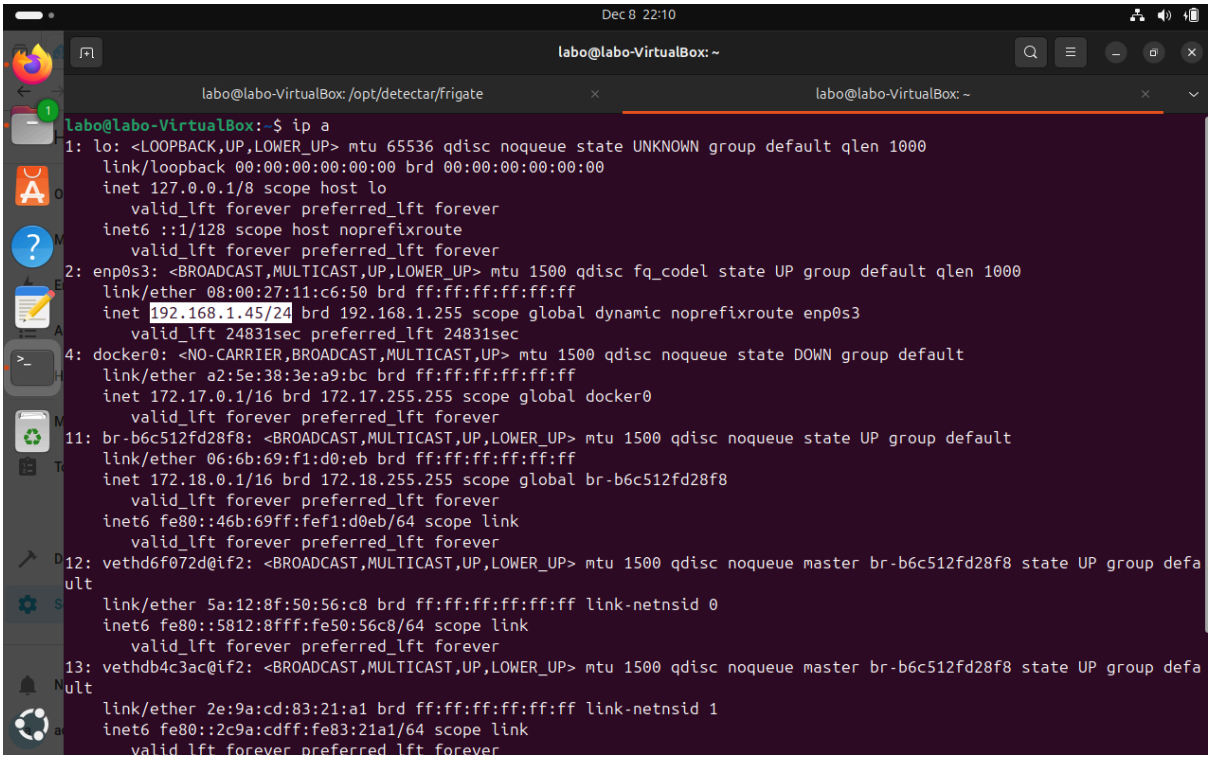
3. Proceso de instalación

Variables utilizadas:

```
<ip_camara>
<direccion_ip_host>
```

La primera variable se seteará en el paso 0.4
Para la segunda, necesitamos correr el siguiente comando en la computadora en la que realizamos la instalación.

```
ip a
```



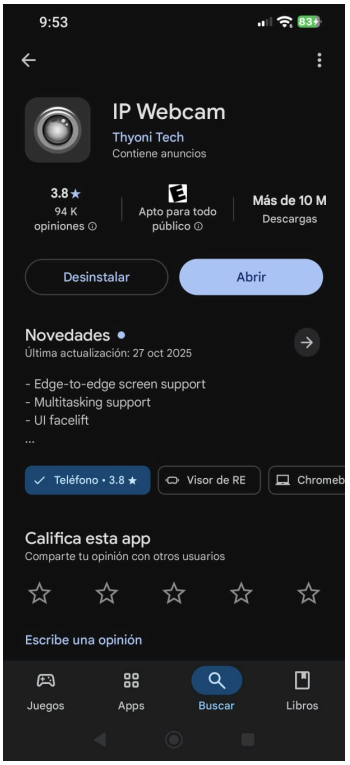
En nuestro caso, la variable de dirección ip host, quedará de la siguiente manera:

```
<direccion_ip_host> = 192.168.1.45
```

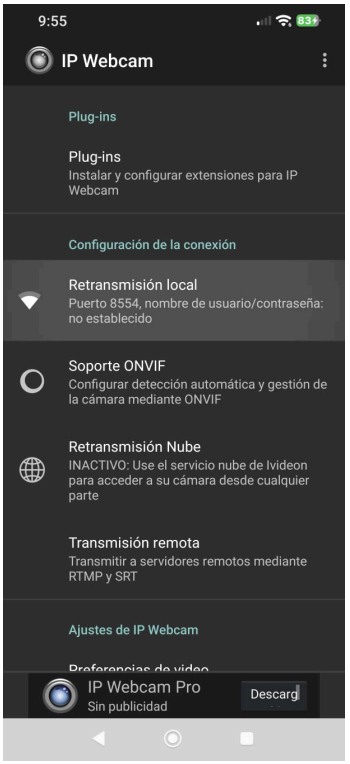
PD: Tenés que estar en la misma red en la que estará la cámara, ya que transmiten información localmente.

Paso 0: Pre-requisitos.

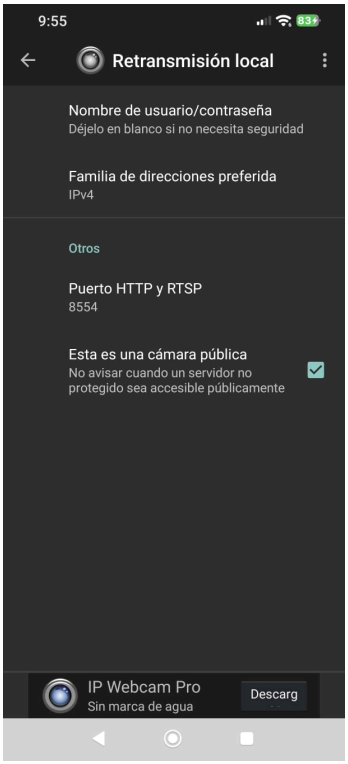
0.1: Instalar la aplicación “IP Webcam” en el dispositivo que transmitirá la imagen



0.2: Una vez abierta la aplicación, vamos a “Retransmisión local”

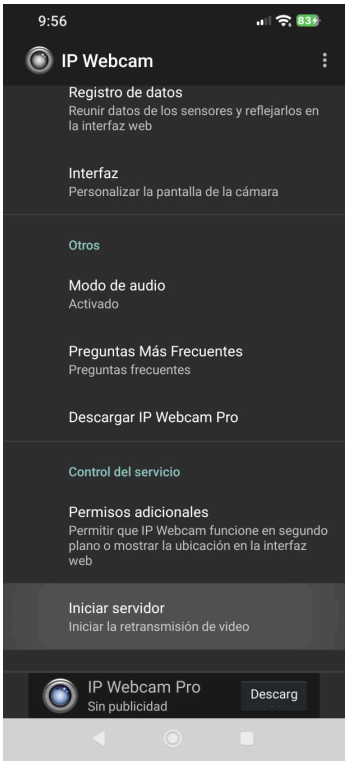


Al hacerle click, nos va a llevar a la siguiente pantalla. Le damos click a “Esta es una cámara pública” y luego cambiamos la opción de **“puerto HTTP y RTSP”** por **“8554”**

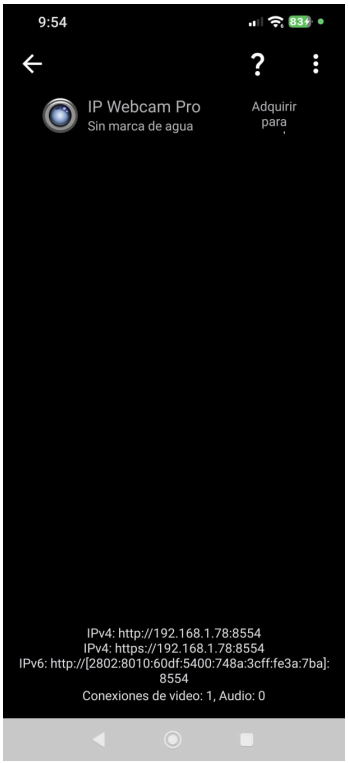


0.3: Desde la pantalla principal, ir a “Preferencias de video”, Elegir “Resolución de video” y seleccionar la opción de **“1280 x 720”**

0.4: Como último paso para ejecutar nuestra cámara, tenemos que ir al final del dashboard de la aplicación, y apretar **“Iniciar servidor”**.



Esto nos mostrará los siguientes datos:



Debajo podemos ver la dirección iPv4 junto al puerto, que usaremos eventualmente en la instalación.
En nuestro caso, la variable <**ip_camara**> será a partir de ahora:

```
<ip_camara> = 192.168.1.78:8554
```

Paso 1: Docker.

Se deberán ejecutar los siguientes comandos en la terminal:

1.1: Dependencias y GPG Key

```
sudo apt update
sudo apt install ca-certificates curl
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o
/etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc
```

1.2: Agregar repositorio Docker

```
sudo tee /etc/apt/sources.list.d/docker.sources <<EOF
```

```
Types: deb
URIs: https://download.docker.com/linux/ubuntu
Suites: $(. /etc/os-release && echo
"${UBUNTU_CODENAME:-$VERSION_CODENAME}")
Components: stable
Signed-By: /etc/apt/keyrings/docker.asc
EOF
```

1.3: Habilitar el repositorio universe

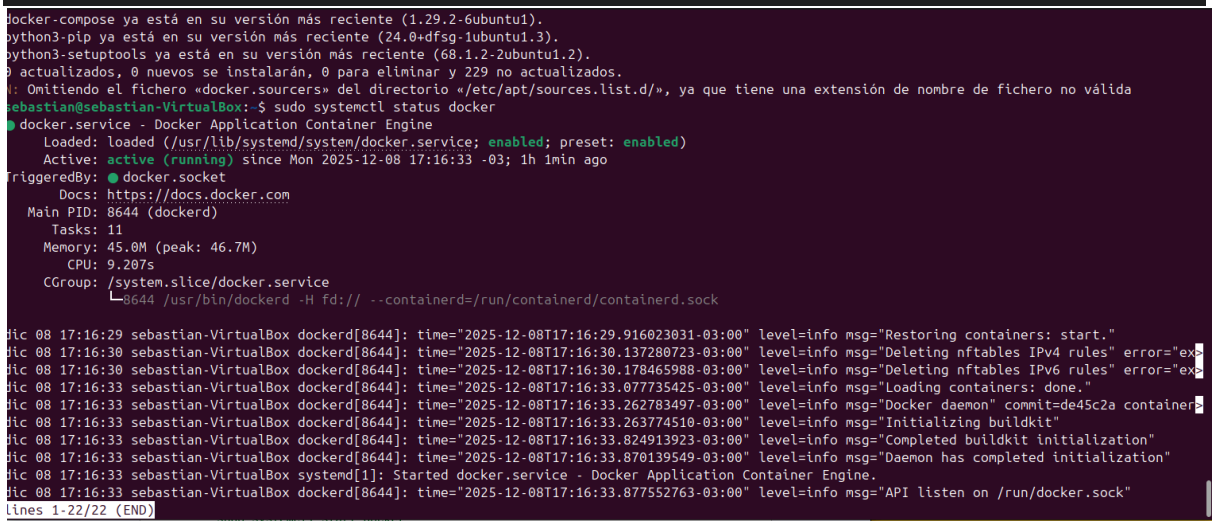
```
sudo add-apt-repository universe
sudo apt update
```

1.4: Instalar Docker + Docker Compose

```
sudo apt install docker-ce docker-ce-cli containerd.io
docker-buildx-plugin docker-compose-plugin docker-compose python3-pip
python3-setuptools -y
```

1.5: Verificación del servicio funcionando correctamente

```
sudo systemctl status docker
```



Si no está corriendo:

```
sudo systemctl start docker
```

Paso 2: Creación de la estructura del proyecto

```
sudo mkdir -p /opt/detectar
sudo mkdir -p /opt/detectar/homeassistant/config
sudo mkdir -p /opt/detectar/frigate
sudo mkdir -p /opt/detectar/frigate/media
sudo mkdir -p /opt/detectar/mosquitto/config
sudo mkdir -p /opt/detectar/mosquitto/data
sudo mkdir -p /opt/detectar/mosquitto/log

sudo chown -R $USER:$USER /opt/detectar
```

Paso 3: Crear el archivo docker-compose.yml

```
nano /opt/detectar/docker-compose.yml
```

En ese archivo, pegar lo siguiente:

```
version: '3.9'

services:

  homeassistant:
    container_name: home-assistant
    image: homeassistant/home-assistant:stable
    restart: unless-stopped
    environment:
      - TZ=America/Argentina/Buenos_Aires
    volumes:
      - ./homeassistant/config:/config
    ports:
      - "8123:8123"
    networks:
      - home-assistant-network
    depends_on:
      - mosquitto
      - frigate

  frigate:
    container_name: frigate
    image: ghcr.io/blakeblackshear/frigate:stable
    privileged: true
    restart: unless-stopped
    shm_size: "512mb"
    volumes:
      - /etc/localtime:/etc/localtime:ro
      - ./frigate/config.yml:/config/config.yml:ro
```

```
      - ./frigate/media:/media/frigate
      - type: tmpfs
        target: /tmp/cache
        tmpfs:
          size: 256000000
    ports:
      - "5000:5000"
      - "8554:8554"
      - "8555:8555/tcp"
      - "8555:8555/udp"
    networks:
      - home-assistant-network
    environment:
      - TZ=America/Argentina/Buenos_Aires

  mosquitto:
    image: eclipse-mosquitto:latest
    container_name: mosquitto
    restart: unless-stopped
    volumes:
      -
./mosquitto/config/mosquitto.conf:/mosquitto/config/mosquitto.conf:ro
      -
./mosquitto/config/passwordfile:/mosquitto/config/passwordfile:ro
      - ./mosquitto/data:/mosquitto/data
      - ./mosquitto/log:/mosquitto/log
    ports:
      - "1883:1883"
      - "9001:9001"
    networks:
      - home-assistant-network

networks:
  home-assistant-network:
    driver: bridge
```

Paso 4: Configurar Mosquitto (MQTT)

Mosquitto nos permite tener un canal de difusión de eventos, para que Home Assistant y Frigate se comuniquen entre ellos, y puedan generar las respectivas alertas.

```
nano /opt/detectar/mosquitto/config/mosquitto.conf
```

En el archivo creado, pegar lo siguiente:

```
listener 1883
allow_anonymous true
persistence true
persistence_location /mosquitto/data/
log_dest file /mosquitto/log/mosquitto.log
```

Paso 5: Configurar Frigate

```
nano /opt/detectar/frigate/config.yml
```

En este archivo, pegar lo siguiente:

```
mqtt:
  host: mosquitto

cameras:
  camera_1:
    ffmpeg:
      inputs:
        - path: rtsp://<ip_camara>/h264.sdp
          roles:
            - detect
            - record
            - stream
    face_recognition:
      enabled: true
```

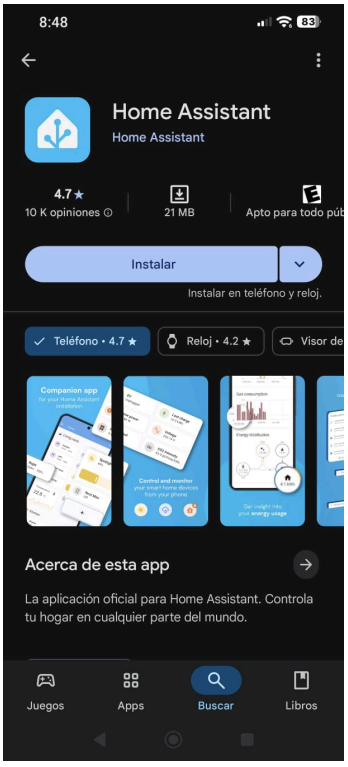
Paso 6: Levantar todo el stack

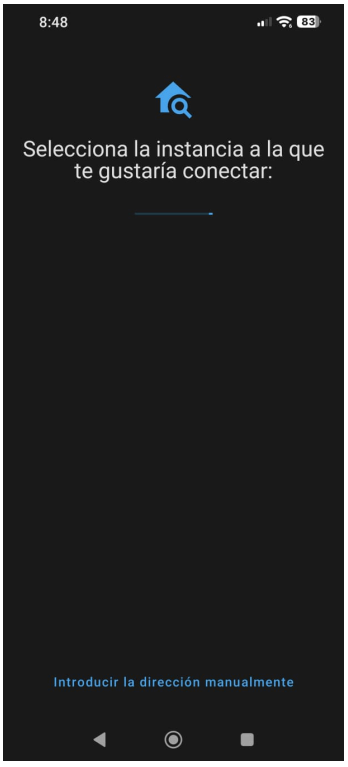
```
cd /opt/detectar
sudo docker-compose up -d
```

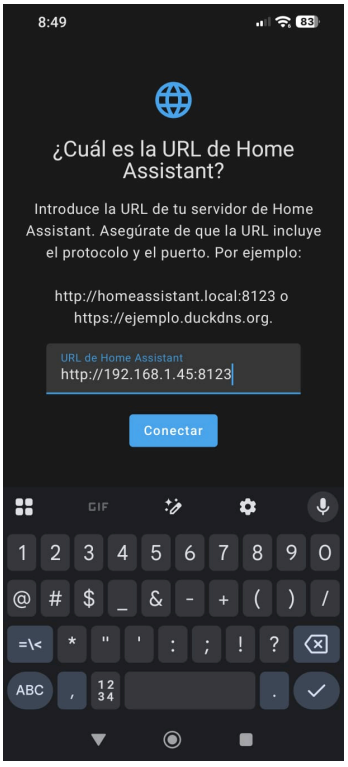
```
5f0656422b7e: Pull complete
37a5db328c40: Pull complete
819bd17e811a: Pull complete
9bb2e9cfed75: Pull complete
0bbd24ed0d23: Pull complete
046f009e3bcf: Pull complete
4f4fb700ef54: Pull complete
7387866dc8c1: Pull complete
bad103feec64: Pull complete
ceb355ee3a5a: Pull complete
886d6a41fd6c: Pull complete
9a6cb407b136: Pull complete
b3e60fefb562: Pull complete
4112b93cf829: Pull complete
31e7cc0d6ae2: Pull complete
13bbc883d32a: Pull complete
98bd069202e1: Pull complete
1afead586dc7: Pull complete
2d475d60a8df: Pull complete
5b98492f02fa: Pull complete
eec82d8ba23d: Pull complete
be8d9f06dee9: Pull complete
61716060355b: Download complete
Digest: sha256:60dd3a329324b92017b7c8665648d65a9568a5a4205e93cad67ab9dec432f2d1
Status: Downloaded newer image for homeassistant/home-assistant:stable
Creating frigate   ... done
Creating mosquitto ... done
Creating home-assistant ... done
sebastian@sebastian-VirtualBox:/opt/detectar$
```

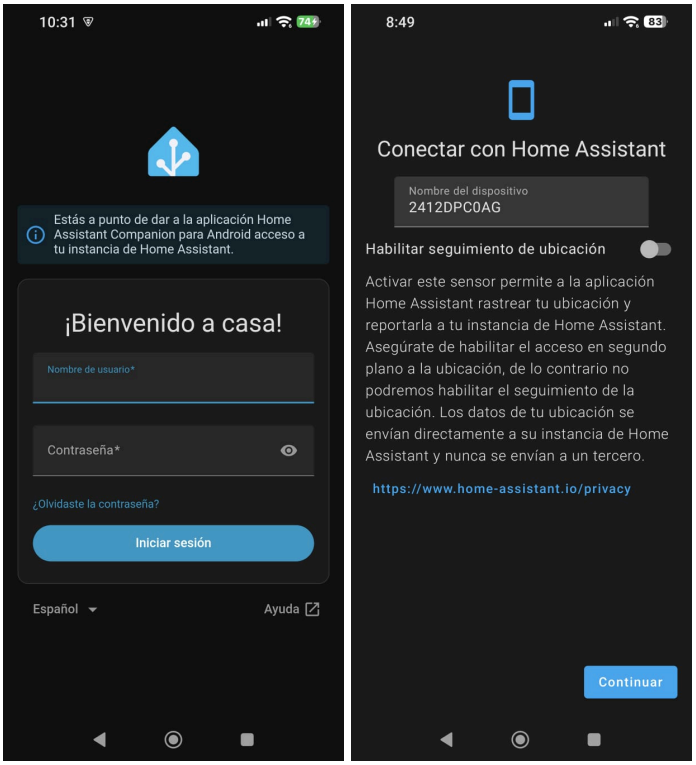
Paso 7: Vincular un celular con Home Assistant

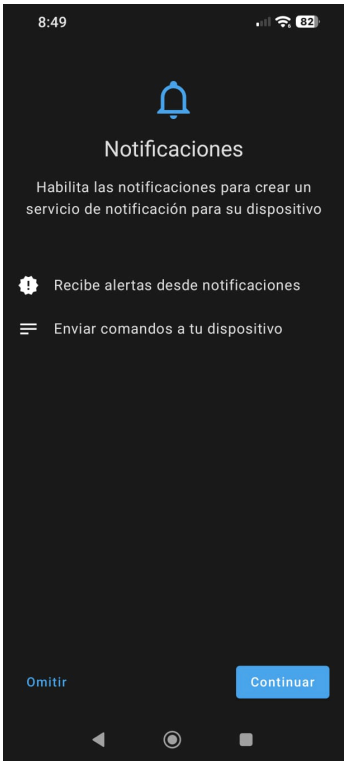
Una vez levantado el stack, es necesario instalar la aplicación de Home Assistant en un celular, para poder recibir los eventos:

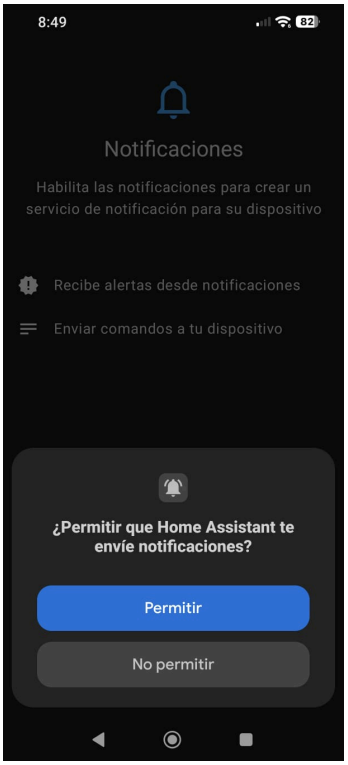












La misma se puede instalar en la Play Store de Android. Si no encuentra la instancia automáticamente, seleccionar “Introducir la dirección manualmente”, e ingresar en el campo de texto:

```
http://<direccion_ip_host>:8123
```

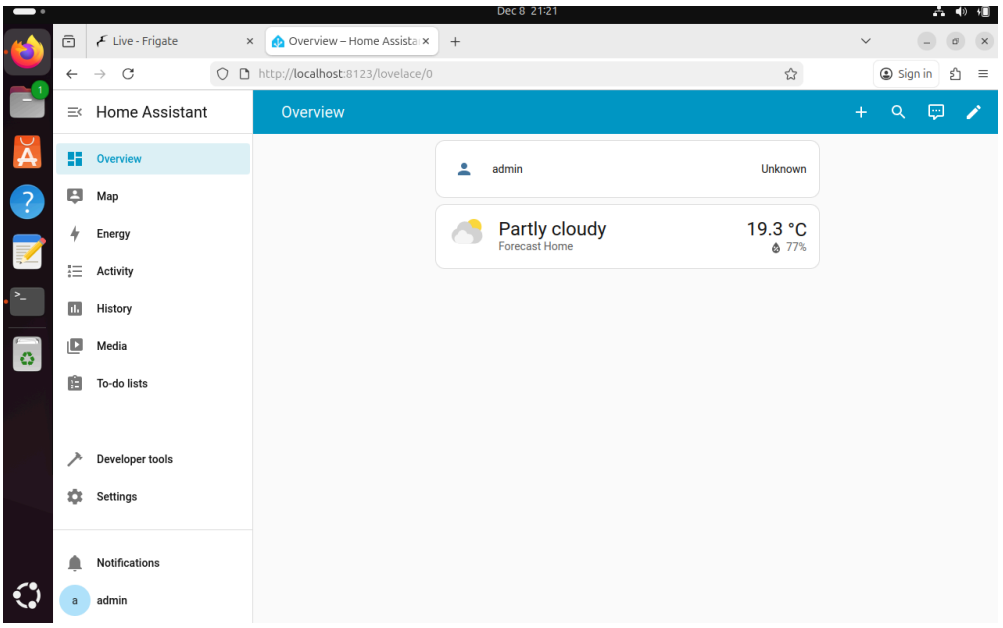
Luego, deberemos conectarnos con el mismo usuario y contraseña que registramos a continuación en el siguiente paso:

Paso 8: Consumir eventos y generar notificaciones

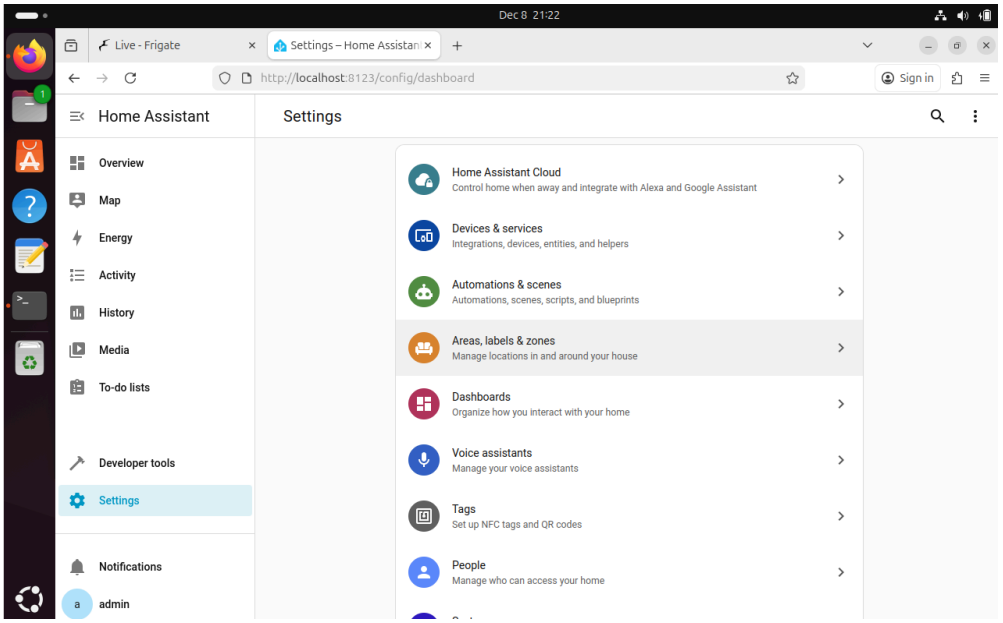
El próximo paso requiere ir a la página web de Home Assistant:

```
http://<direccion_ip_host>:8123
```

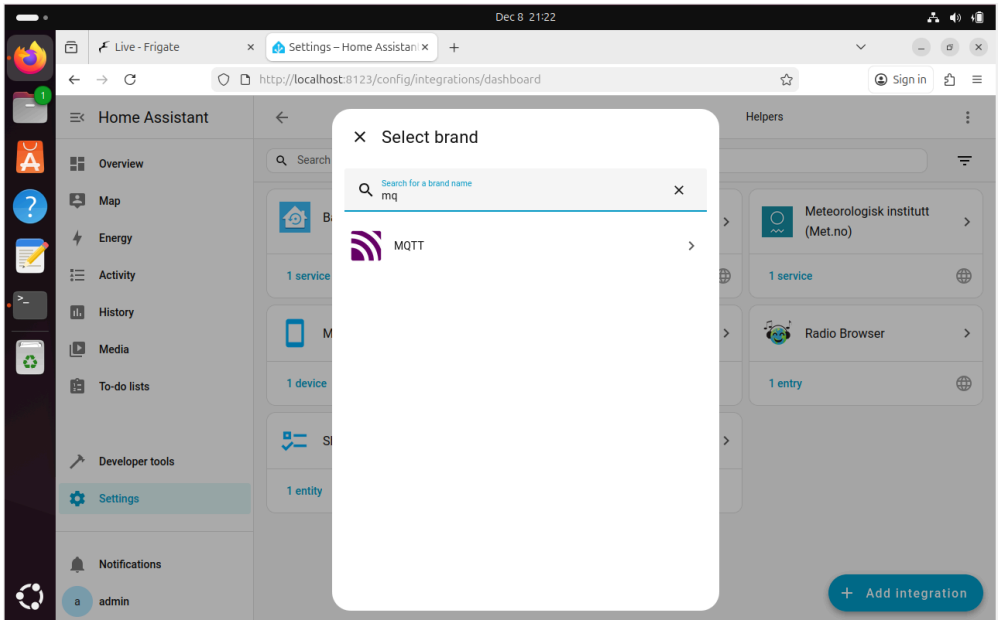
Al ser la primera vez, nos pedirá que nos registremos. Hay que ingresar el nombre de usuario y la contraseña. La misma será con la que iniciemos sesión en el dispositivo móvil para recibir las notificaciones.



Presionar “Settings”, y luego ir a la sección “Devices & services”



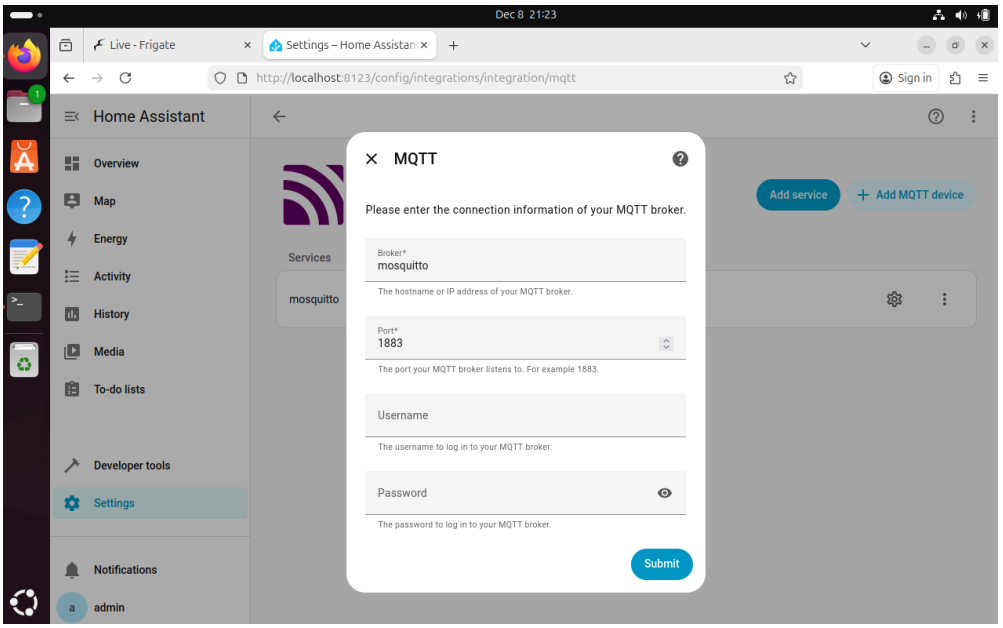
Presionar el botón “Add integrations” debajo a la derecha, y buscar y elegir la opción **“MQTT”**



Rellenar con la siguiente información:

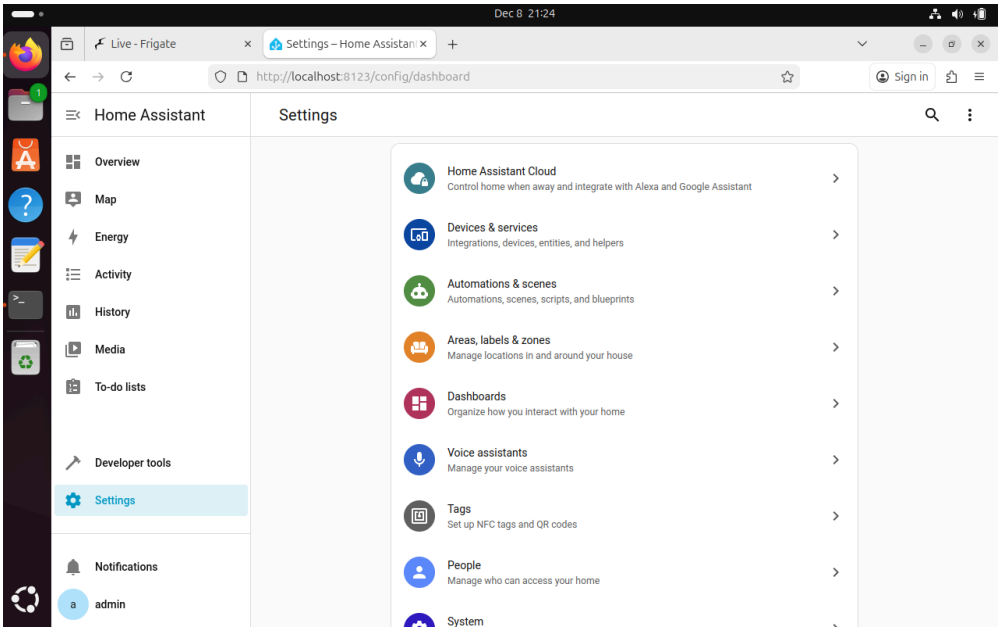
Broker: **mosquitto**

Port: **1883**

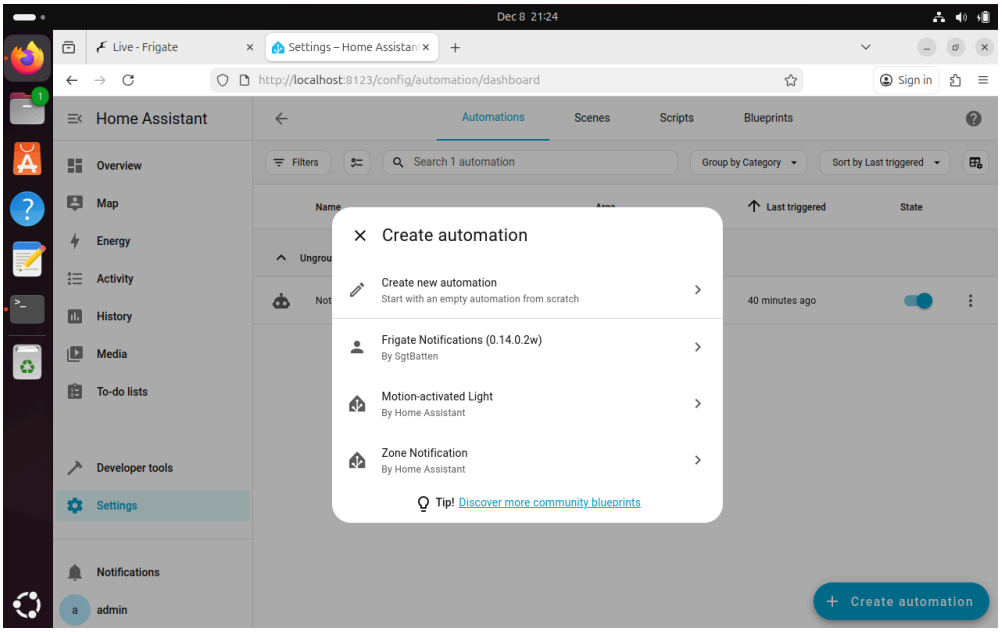


Apretar el botón “Submit” y volver atrás.

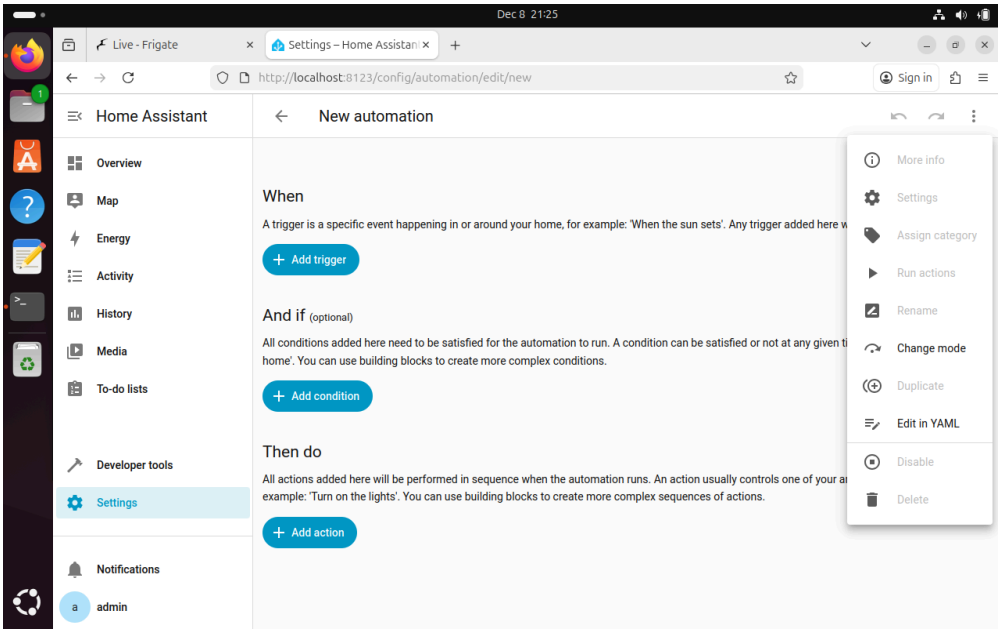
Desde la pantalla inicial de “Settings”, ir a “Automations & scenes”.



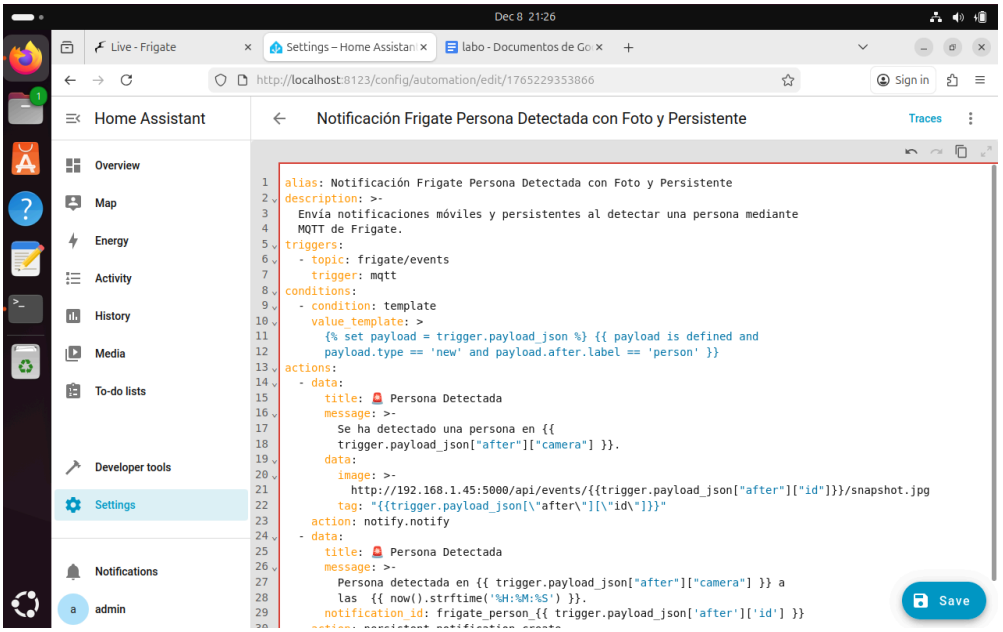
Apretar el botón “Create automation” y elegir “Create new automation”.



Ir al botón de 3 puntos, de arriba a la derecha, y elegir “Edit in YAML”



En la entrada de texto, pegar lo siguiente y apretar “**Save**”:




```
alias: Notificación Frigate Persona Detectada con Foto y
Persistente
description: >-
  Envía notificaciones móviles y persistentes al detectar una
persona mediante
  MQTT de Frigate.
triggers:
  - topic: frigate/events
```

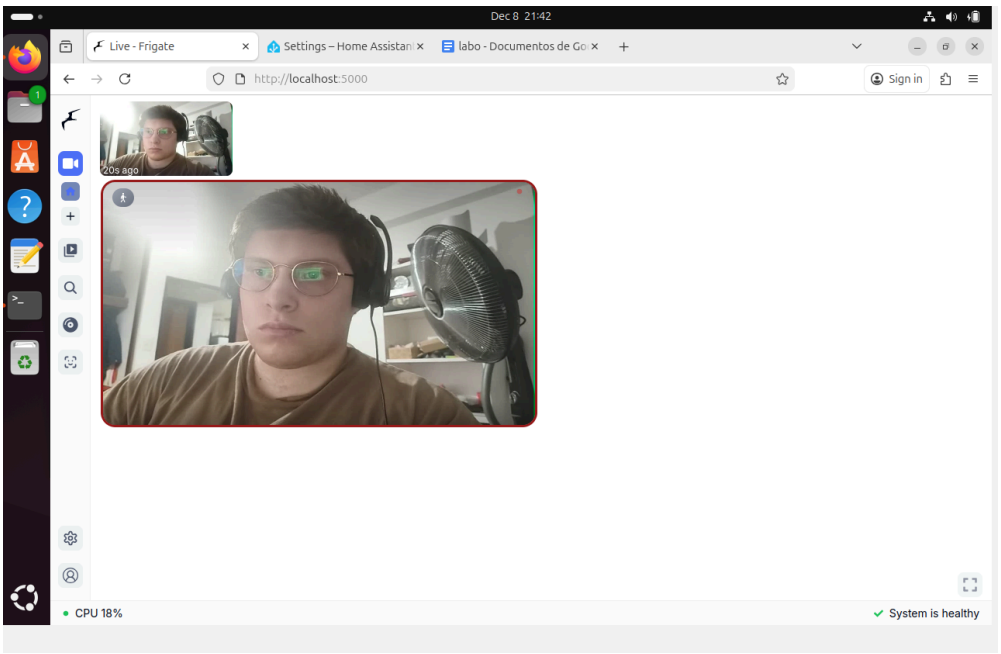
```
    trigger: mqtt
conditions:
  - condition: template
    value_template: >
      {% set payload = trigger.payload_json %} {{ payload is
defined and
      payload.type == 'new' and payload.after.label == 'person' }}
actions:
  - data:
      title: 🚨 Persona Detectada
      message: >-
        Se ha detectado una persona en {{
        trigger.payload_json["after"]["camera"] }}.
      data:
        image: >-

http://<direccion_ip_host>:5000/api/events/{{trigger.payload_json[
"after"]["id"]}}/snapshot.jpg
        tag: "{{trigger.payload_json[\"after\"][\"id\"]}}"
    action: notify.notify
  - data:
      title: 🚨 Persona Detectada
      message: >-
        Persona detectada en {{
trigger.payload_json["after"]["camera"] }} a
        las  {{ now().strftime('%H:%M:%S') }}.
      notification_id: frigate_person_{{
trigger.payload_json['after']['id'] }}
    action: persistent_notification.create
mode: single
```

Paso 9: Generar un evento de prueba

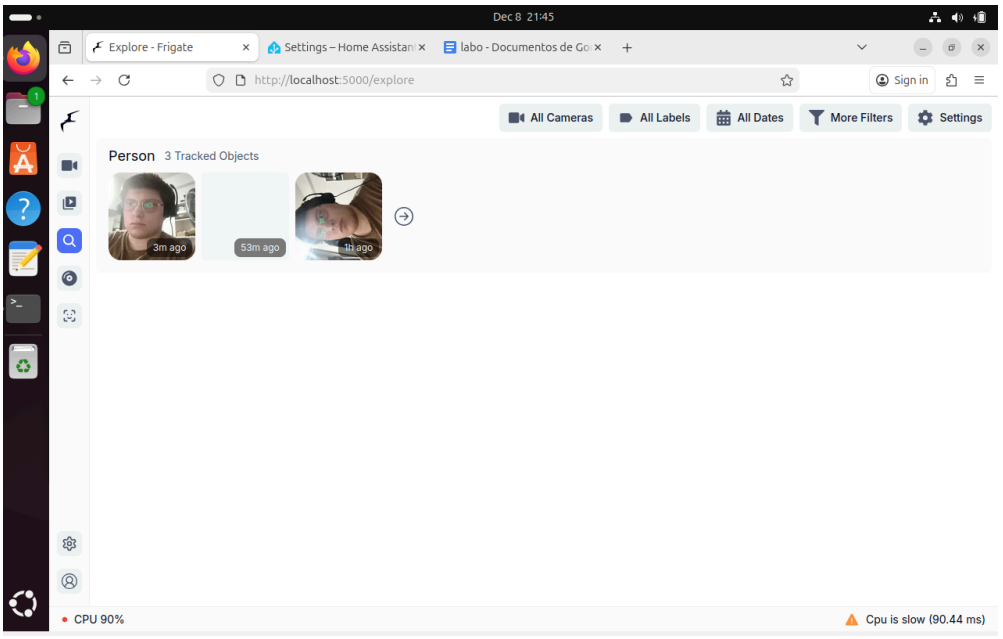
Para comenzar, debemos de ir a la página web de **Frigate**

```
http://<direccion_ip_host>:5000
```

Una vez dentro, si tenemos la cámara conectada correctamente, y una persona se acerca, vamos a ver un ícono de persona, arriba a la izquierda. Eso significa que está detectando a una persona, y va a generar un evento:

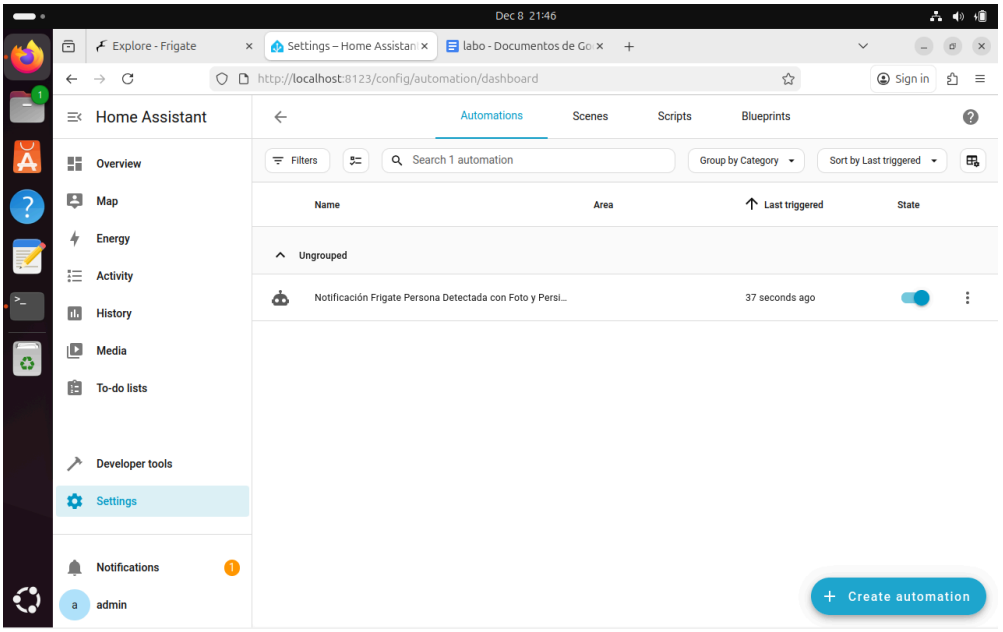


Si vamos a la opción de “Explorar”, podemos ver que tenemos la lista de personas detectadas:

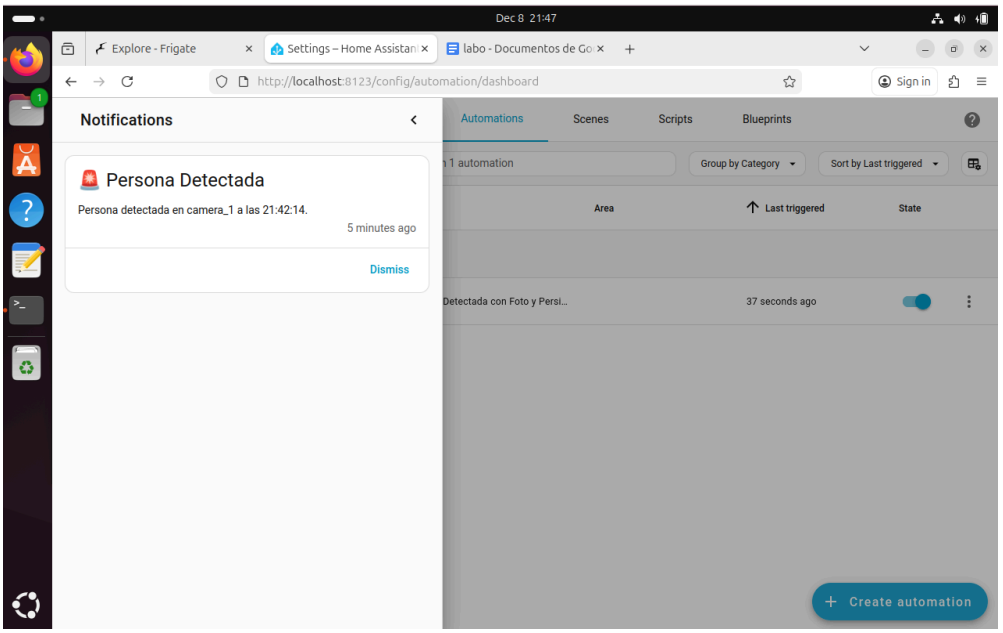


Esto implica que se generó el evento correctamente.

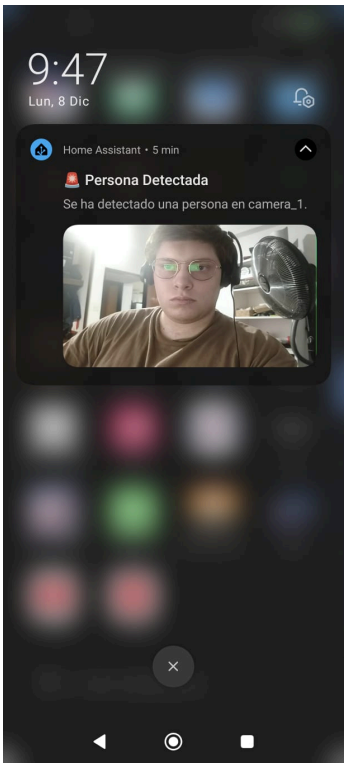
Si vamos a la página de Home Assistant, podremos ver que ahora aparece un badge en las notificaciones:



Al apretarla, nos mostrará la notificación generada:



Y en el dispositivo celular, nos debería de haber llegado la siguiente notificación:



4. Problemas encontrados

- En otro dispositivo utilizado como backup, al momento de levantar todo el stack, se tardaba mucho tiempo y ocurrió que se cortó la luz, por lo tanto se desconectó del internet.

  Pero no hay que paniquear ya que esto se resuelve volviendo a poner el comando `”sudo docker-compose up -d”.` Verificando lo que descargo, y terminando los faltantes. Hacerlo las veces necesarias hasta que los 3 docker estén “done”.

- En el mismo dispositivo, al momento de ver la cámara en el Frigate, se distorsionaba, y mostraba una imagen muy retardada, sin siquiera poder reconocer si una persona pasaba por esta.

  La forma de resolver este problema, fue modificando el archivo en `/opt/detectar/frigate/config.yml` de la siguiente manera:

```
ymqtt:
```

```
  host: mosquitto

cameras:
  camera_1:
    ffmpeg:
      inputs:
        - path: rtsp://<ip_camara>/h264.sdp
          roles:
            - detect
            - record
            - stream
          inputs_args:
            - -avoid-negative_ts
            - make_zero
    face_recognition:
      enabled: true
```

Lo que hace el -avoid-negative_ts es activar la lógica para manejar timestamps negativos y gracias a esto el make_zero, indica a FFMPEG que, si encuentra un timestamp negativo, lo ajuste a cero, es decir convertirlo en el punto de inicio.