
ITES

Práctico Integrador

Docker

**Base: MQTT + n8n + MySQL + phpMyAdmin +
Flask (read-only) + Nginx (solo Flask) +
Portainer**

Extra: APP (Movil) + Control de ESP8266

Integrantes:

Raiburn, Jesus

Litterini, Juan

Milanese, Jonatan

Materia:

Software Factory III

Año:

3er Año – 2025

INDICE

INTRODUCION	3
OBJETIVOS	4
SERVICIOS Y COMPONENTES	5
1. Infraestructura – Docker Compose	
2. Mosquitto (MQTT broker)	
3. MySQL	
4. phpMyAdmin	
5. n8n (orquestación de flujos)	
Workflow MQTT	
Workflow Webhook	
6. Flask (Python)	
7. Nginx	
8. Portainer	
9. App Móvil (MAUI Blazor Hybrid)	
10. ESP8266 con LED físico	
11. Flujo de trabajo completo	
COMANDOS BASICOS	8
GUIA PASO A PASO	10
1. Crear carpeta con las siguientes carpetas, archivos y servicios, con sus respectivas configuraciones y códigos.	
2. Descargar las Imágenes y levantar los Contenedores.	
3. Verificar el estado, comprobar los servicios y los endpoints.	
4. Control del ESP8266 - Led físico, mediante MQTT, y carga en base de datos.	
5. APP-Control del ESP8266 - Led físico, mediante APP, y carga en base de datos.	
CONCLUSION	27

INTRODUCION

En el marco de este trabajo práctico se desarrolló una mini-plataforma integrada con Docker Compose, cuyo objetivo principal fue demostrar la interacción entre distintos servicios de software, una aplicación móvil y un dispositivo IoT real (ESP8266).

La propuesta combina tecnologías de contenedores, bases de datos, automatización de flujos, desarrollo web y mobile, junto con hardware IoT, mostrando cómo se puede diseñar un ecosistema completo, modular y escalable.

La plataforma implementa dos flujos end-to-end:

Workflow MQTT: un usuario interactúa con una aplicación móvil (encendido/apagado de un LED), pasando por la comunicación mediante MQTT y el almacenamiento en MySQL, hasta la visualización en una interfaz web y la ejecución física en un microcontrolador.

Workflow Webhook: expone un endpoint HTTP que recibe peticiones, procesa los datos y los inserta en la base de datos.

En ambos casos, la información termina almacenada en MySQL, se puede consultar por medio de Flask/Nginx o phpMyAdmin.

En síntesis, la mini-plataforma crea un ecosistema IoT + Backend + Base de datos + Frontend + Gestión, todo integrado con Docker.

OBJETIVOS

- a. Diseñar y desplegar una plataforma distribuida utilizando Docker Compose para simplificar la orquestación y la gestión de servicios.
- b. Integrar un broker MQTT (Mosquitto) para la comunicación en tiempo real entre la aplicación móvil, los flujos de automatización (n8n) y el dispositivo IoT (ESP8266).
- c. Implementar una base de datos MySQL como repositorio central de la información generada por la plataforma.
- d. Exponer la base de datos a través de phpMyAdmin para facilitar la administración y validación de registros.
- e. Configurar n8n con dos workflows complementarios:
Workflow MQTT para validar e insertar mensajes recibidos por tópicos.
Workflow Webhook para procesar peticiones HTTP provenientes de la web.
- f. Desarrollar una aplicación Flask que consulte la base de datos y muestre los resultados en una tabla HTML servida mediante Nginx.
- g. Incorporar Portainer como interfaz gráfica de administración de contenedores para monitorear y gestionar la infraestructura.
- h. Desarrollar una aplicación móvil en .NET MAUI Blazor Hybrid que permita enviar comandos de encendido y apagado de un LED y registre dichos eventos en la base de datos.
- i. Programar un ESP8266 con código en Arduino IDE para recibir mensajes MQTT y controlar físicamente un LED, mostrando la integración con el hardware.
- j. Realizar pruebas de funcionamiento end-to-end, en ambos flujos (MQTT y Webhook), verificando inserciones en la base de datos (MySQL) y respuesta en el hardware (ESP8266 - LED físico).

SERVICIOS Y COMPONENTES

Breve resumen de los servicios y componentes utilizados en la mini-plataforma:

1. Infraestructura – Docker Compose

Archivos principales:

docker-compose.yml

Archivo principal que define todos los servicios que forman la plataforma (qué imágenes usar, puertos, redes, volúmenes).

Es el “plano” de la mini-plataforma. Sería como “el encargado de encender y conectar todo”. Con un comando levanta esta mini-plataforma.

.env

Archivo con variables (usuarios, contraseñas, puertos).

Permite configurar sin modificar el docker-compose.yml.

2. Mosquitto (MQTT broker)

Servicio de mensajería liviana orientada a dispositivos IoT. Funciona con modelo publish/subscribe (pub/sub).

Sería como un “cartero” de mensajitos. Los dispositivos publican en un tópico y otros se suscriben.

Permite que el ESP8266, la app móvil y n8n se comuniquen mediante tópicos (ej: lab/g1/req/orders.create).

Ejemplo:

La app publica “LED=ON”.

El ESP8266 está suscrito y enciende el LED físico.

Otro flujo (n8n) inserta ese dato en MySQL.

3. MySQL

Base de datos relacional que guarda los registros. Es el centro de persistencia de datos.

En este caso, se usa la tabla ledactions para almacenar las acciones (ej: ON/OFF).

4. phpMyAdmin

Interfaz web para administrar MySQL. Permite ver y editar la base de datos desde el navegador. Útil para mostrar fácilmente los registros.

Sería una web para mirar/editar MySQL sin usar consola.

5. n8n (orquestación de flujos)

Plataforma de automatización similar a Node-RED pero más completa, que permite conectar sistemas y procesar datos sin programar todo desde cero.

Sería un “orquestador”. Escucha MQTT, mete datos en MySQL, expone webhooks HTTP y publica respuestas a MQTT.

En este proyecto se usan dos Workflow:

Workflow MQTT:

Se suscribe al broker Mosquitto.

Valida los mensajes entrantes.

Si es válido, entonces inserta en MySQL y publica respuesta ok: true.

Si es inválido, entonces publica respuesta ok: false.

Workflow Webhook:

Expone un endpoint HTTP (/webhook-test/led_control).

Recibe peticiones desde la Web (ej.: ON/OFF).

Inserta el registro en MySQL.

Devuelve una respuesta en JSON con el resultado de la operación.

Con esta doble integración, n8n actúa como puente entre el mundo IoT (MQTT) y el mundo web/app (HTTP), centralizando la lógica de negocio.

6. Flask (Python)

Microframework web en Python. Permite mostrar datos almacenados de forma simple y rápida (solo lectura).

En este proyecto expone una ruta /app/tabla que consulta MySQL y devuelve una tabla HTML con los registros.

7. Nginx

Servidor web y proxy inverso. Sirve la aplicación Flask al público en el puerto 8080.

Hace de “frente” de la plataforma web, separando la lógica interna (Flask) de la exposición externa.

8. Portainer

Interfaz gráfica de administración de Docker. Permite ver contenedores, redes y volúmenes sin usar la consola.

Útil para para mostrar la infraestructura en tiempo real.

Seria como una “ventana” donde ver/administrar contenedores.

9. App Móvil (MAUI Blazor Hybrid)

Aplicación desarrollada en .NET MAUI con Blazor Hybrid.

Simula un panel de control para el LED:

Botón ON/OFF que manda la orden vía MQTT/n8n a la base de datos.

Permite la interacción entre móvil, backend y hardware real.

10. ESP8266 con LED físico

Microcontrolador Wi-Fi programado desde Arduino IDE.

Se conecta al broker Mosquitto y escucha las órdenes MQTT. Según el mensaje recibido (ON/OFF), enciende o apaga un LED físico conectado al pin.

Ejemplo de integración IoT en tiempo real.

11. Flujo de trabajo completo

- Usuario en la App móvil: pulsa ON.
- Mensaje MQTT: llega al broker Mosquitto.
- n8n recibe, valida, inserta en MySQL y responde “OK”.
- ESP8266 recibe la orden: enciende el LED físico.
- En Flask (vía Nginx) o en phpMyAdmin se puede ver el registro guardado.
- Portainer permite mostrar todo corriendo en contenedores.

COMANDOS BASICOS

Comandos basicos en la CLI (Command-Line Interface) POWERSHEL de WINDOWS para la implementación y gestion de la mini-plataforma. (También se pueden puede utilizar la terminal integrada en Docker o la CLI CMD):

- **Descargar las Imágenes y Levantar los Contenedores:**

docker compose up -d

- **Verificar el Estado:**

docker compose ps

- **Detener los Contenedores:**

docker compose down

- **Detener y Eliminar Contenedores, Redes y Volúmenes (eliminando datos):**

docker compose down -v

- **Ver los Logs (para depuración):**

docker compose logs -f

- **Liberar Espacio con Prune (Para limpiar el sistema y eliminar todos los recursos de Docker, incluidos contenedores detenidos, redes sin uso, imágenes sin referencia y caché de compilación):**

docker system prune -a

- **Reiniciar Contenedores, ej.: el contenedor de Portainer**

docker compose restart portainer

- **Ver configuración de la red (Para mostrar la configuración de red TCP/IP de la computadora, incluyendo la dirección IP, máscara de subred, puerta de enlace predeterminada y servidor DNS)**

Ipconfig

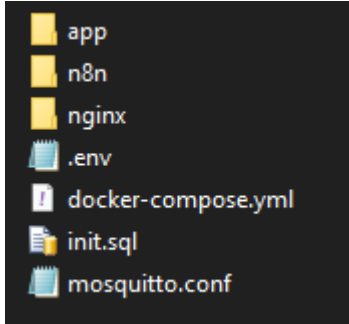
- **Verificar la conectividad y el estado de un dispositivo remoto en una red (como una página web o un servidor), ejemplo: ver la conectividad del dominio google.com o su dirección de IP (DNS) 8.8.8.8**

ping google.com

ping 8.8.8.8

GUIA PASO A PASO

1. Crear carpeta con las siguientes carpetas, archivos y servicios, con sus respectivas configuraciones y códigos:



Ejemplos de carpetas y archivos:

docker-compose.yml

```
docker-compose.yml X
C: > Users > Jhony > Desktop > Proyectos > practico-docker-a > docker-compose.yml > ...
1  name: lab-iot
2
   >Run All Services
3  services:
   >Run Service
4      mysql:
5          image: mysql:8.0
6          environment:
7              MYSQL_DATABASE: ${MYSQL_DATABASE}
8              MYSQL_USER: ${MYSQL_USER}
9              MYSQL_PASSWORD: ${MYSQL_PASSWORD}
10             MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}
11             TZ: ${TZ}
12         volumes:
13             - mysql_data:/var/lib/mysql
14             - ./init.sql:/docker-entrypoint-initdb.d/init.sql:ro
15         networks: [labnet]
16
   >Run Service
17     phpmyadmin:
18         image: phpmyadmin:5-apache
19         environment:
20             PMA_HOST: mysql
21         ports:
22             - "8081:80"
23         depends_on: [mysql]
24         networks: [labnet]
25
   >Run Service
26     mosquitto:
27         image: eclipse-mosquitto:2
```

.env:

```
TZ=America/Argentina/Buenos_Aires
MYSQL_DATABASE=ledactions
MYSQL_USER=leduser
MYSQL_PASSWORD=ledjxz14pass
MYSQL_ROOT_PASSWORD=rootjxz14pass
FLASK_SECRET=supersecret
MYSQL_HOST=mysql
N8N_BASIC_AUTH_USER=Jexzus14
N8N_BASIC_AUTH_PASSWORD=Jexzus091218
```

mosquito.conf

```
listener 1883 0.0.0.0
allow_anonymous true
|
```

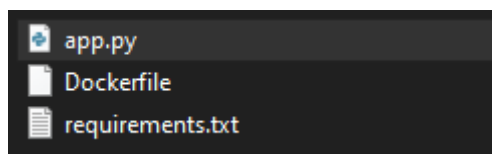
init.sql

```
-- Crea las tablas si no existen
CREATE TABLE IF NOT EXISTS orders (
  id INT AUTO_INCREMENT PRIMARY KEY,
  customer VARCHAR(100) NOT NULL,
  item VARCHAR(100) NOT NULL,
  qty INT NOT NULL,
  correlation_id VARCHAR(64),
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE IF NOT EXISTS led_events (
  id INT AUTO_INCREMENT PRIMARY KEY,
  device_id VARCHAR(64) NOT NULL,
  action VARCHAR(8) NOT NULL,
  source VARCHAR(16) NOT NULL,
  correlation_id VARCHAR(64),
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- esto otorga todos los privilegios al usuario 'leduser' en la base de datos 'ledactions'
GRANT ALL PRIVILEGES ON ledactions.* TO 'leduser'@'%';
-- Aplica los cambios de privilegios
FLUSH PRIVILEGES;
```

En carpeta app (flask):



app.py

```
app.py X
C: > Users > Jhony > Desktop > Proyectos > practico-docker-a > app > app.py > ...
1  import os
2  import mysql.connector
3  from flask import Flask, render_template_string
4
5  app = Flask(__name__)
6
7  DB_CFG = {
8      "host": os.getenv("MYSQL_HOST", "mysql"),
9      "user": os.getenv("MYSQL_USER", "leduser"),
10     "password": os.getenv("MYSQL_PASSWORD", "ledjxz14pass"),
11     "database": os.getenv("MYSQL_DATABASE", "ledactions"),
12 }
13
14 ROW_TMPL = """
15 <!doctype html>
16 <html>
17     <head><meta charset="utf-8"><title>led_events</title></head>
18     <body>
19         <h2>Tabla: led_events</h2>
20         <table border="1" cellpadding="6">
21             <thead>
22                 <tr>
23                     <th>id</th><th>device_id</th><th>action</th><th>source</th>
24                     <th>correlation_id</th><th>created_at</th>
25                 </tr>
26             </thead>
```

Dockerfile

```
FROM python:3.11-slim
WORKDIR /app

# librerías necesarias para mysqlclient
RUN apt-get update && apt-get install -y build-essential default-libmysqlclient-dev && r

COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

COPY . .


EXPOSE 8000

# gunicorn: 2 workers, bind en 0.0.0.0:8000, objeto WSGI "app" dentro de app.py
CMD ["gunicorn", "-w", "2", "-b", "0.0.0.0:8000", "app:app"]
```

requirements.txt

```
Flask
mysql-connector-python
gunicorn
```


En carpeta n8n/workflows:

 complete_workflows.json

complete_workflows.json

```
C:\Users\Jhony\Desktop\Proyectos\practico-docker-u> n8n > workflows > {} complete_workflows.json > ...
1  {}
2  "name": "Complete LED Control",
3  "nodes": [
4    {
5      "parameters": {
6        "path": "led_control",
7        "options": {}
8      },
9      "name": "Control Webhook",
10     "type": "n8n-nodes-base.webhook",
11     "typeVersion": 1,
12     "position": [
13       250,
14       200
15     ],
16     "id": "e2f0a1c7-c752-4416-b188-4e08b1a8f94a"
17   },
18   {
19     "parameters": {
20       "functionCode": "let data = {};\\nlet body = $json.body;\\ndata.action = body.command;\\ndata.d",
21       "options": {}
22     },
23     "name": "Validar y Preparar",
24     "type": "n8n-nodes-base.function",
25     "typeVersion": 1,
26     "position": [
27       450,
28       200
29     ],
30     "id": "a933f7f8-51f7-4186-b4d0-4e4b525d81f2"
31   }
32 ]
```

En carpeta nginx:

 default.conf

default.conf

```
server { listen 80; location / { proxy_pass http://flask:8000; } }
```

2. Descargar las Imágenes y levantar los Contenedores

Ubicarse en la carpeta donde está el compose, y abrir la CLI: *docker compose up -d*

```
PS C:\Users\Jhony\Desktop\Proyectos\practico-docker-a> docker compose up -d
[+] Running 8/8
  Network lab-iot_labnet          Created
  Container lab-iot-mysql-1       Started
  Container lab-iot-mosquitto-1   Started
  Container lab-iot-n8n-1         Started
  Container lab-iot-portainer-1   Started
  Container lab-iot-flask-1       Started
  Container lab-iot-phpmyadmin-1  Started
  Container lab-iot-nginx-1       Started
PS C:\Users\Jhony\Desktop\Proyectos\practico-docker-a> _
```

ITES – SOFTWARE FACTORY III

Practico Integrador Docker

3. Verificar el estado, comprobar los servicios y los endpoints:

docker compose ps

```
PS C:\Users\Jhony\Desktop\Proyectos\practico-docker-> docker compose ps
```

NAME	IMAGE	COMMAND	SERVICE	CREATED	STATUS
lab-iot-flask-1	lab-iot-flask	"unicorn -w 2 -b 0..."	flask	5 minutes ago	Up 5 minute
lab-iot-mosquitto-1	eclipse-mosquitto:2	"/docker-entrypoint..."	mosquitto	5 minutes ago	Up 5 minute
lab-iot-mysql-1	mysql:8.0	"docker-entrypoint.s..."	mysql	5 minutes ago	Up 5 minute
lab-iot-n8n-1	n8nio/n8n:latest	"tini -- /docker-ent..."	n8n	5 minutes ago	Up 5 minute
lab-iot-nginx-1	nginx:alpine	"/docker-entrypoint..."	nginx	5 minutes ago	Up 5 minute
lab-iot-phpmyadmin-1	phpmyadmin:5-apache	"/docker-entrypoint..."	phpmyadmin	5 minutes ago	Up 5 minute
lab-iot-portainer-1	portainer/portainer-ce:latest	"/portainer"	portainer	5 minutes ago	Up 5 minute

```
PS C:\Users\Jhony\Desktop\Proyectos\practico-docker->
```

En docker se puede controlar las imágenes descargadas y los contenedores creados

Imágenes

Local						
2.37 GB / 2.68 GB in use 8 images						
Last refresh: 23 hours ago						
	Name	Tag	Image ID	Created	Size	Actions
<input type="checkbox"/>	nginx	alpine	4a86014ec699	2 months ago	52.54 MB	
<input type="checkbox"/>	eclipse-mosquitto	2	42292b8c6592	3 months ago	10.34 MB	
<input type="checkbox"/>	portainer/portainer-ce	latest	e6b0d4bc3234	5 days ago	186.29 MB	
<input type="checkbox"/>	phpmyadmin	5	2d9a07a01a58	8 months ago	568.21 MB	
<input type="checkbox"/>	phpmyadmin	5-apache	2d9a07a01a58	8 months ago	568.21 MB	
<input type="checkbox"/>	mysql	8.0	94753e67a0a9	6 days ago	780.48 MB	
<input type="checkbox"/>	n8nio/n8n	latest	81d52fcea9c2	3 days ago	976.53 MB	
<input type="checkbox"/>	practico-docker-n-flask	latest	478890c63655	21 hours ago	205.27 MB	

Showing 9 items

Contenedores

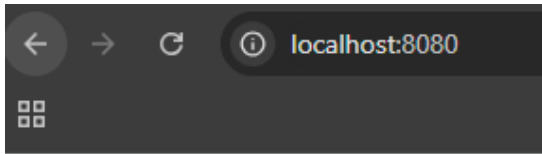
Containers						
Container CPU usage 1.15% / 400% (4 CPUs available)				Container memory usage 694.97MB / 2.73GB		
Show charts						
	Name	Container ID	Image	Port(s)	CPU (%)	Actions
<input type="checkbox"/>	lab-iot	-	-	-	1.15%	
<input type="checkbox"/>	n8n-1	26c0df41862a	n8nio/n8n 5678:5678		0%	
<input type="checkbox"/>	mysql-1	102c31a48a32	mysql:8.0		1.05%	
<input type="checkbox"/>	portainer-1	07c351e49db2	portainer/p 9443:9443		0%	
<input type="checkbox"/>	mosquitto-1	e19f95f00e74	eclipse-mo 5004:1883		0.06%	
<input type="checkbox"/>	flask-1	e432378754a2	lab-iot-flas 5000:5000		0.02%	
<input type="checkbox"/>	phpmyadmin-1	d8d7c02fe29	phpmyadm 8081:80		0.02%	
<input type="checkbox"/>	nginx-1	5e74966593c6	nginx:alpin 8080:80		0%	

Showing 8 items

Chequear servicios y endpoints.

IMPORTANTE: Todos los Usuarios, Claves, Host, y Puertos, surgen del .env que creamos, y que usaremos para la configuración de los distintos servicios.

Flask vía Nginx: <http://localhost:8080>



OK

Usá [/led-events](#) para ver registros.

Flask vía Nginx: Tabla: <http://localhost:8080/app/tabla>)

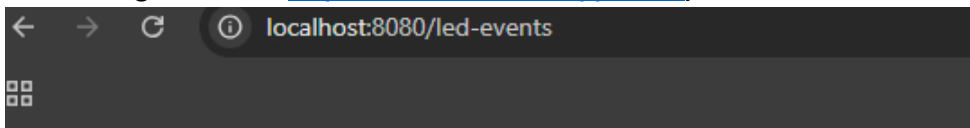


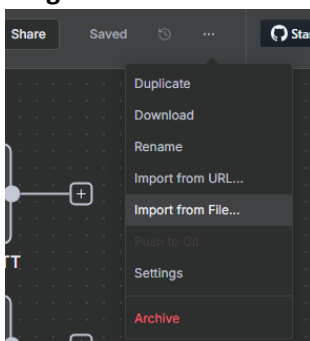
Tabla: led_events

id	device_id	action	source	correlation_id	created_at
27	jesus	OFF	device	1759106580947	2025-09-29 00:43:00
26	jesus	ON	device	1759106510644	2025-09-29 00:41:50
25	jesus	ON	device	1759106170328	2025-09-29 00:36:10
24	jesus	ON	device	1759104970409	2025-09-29 00:16:10
23	jesus	OFF	device	1759104956708	2025-09-29 00:15:56
22	jesus	ON	device	1759104932829	2025-09-29 00:15:32
21	jesus	ON	device	1759104632339	2025-09-29 00:10:32

n8n: <http://localhost:5678>

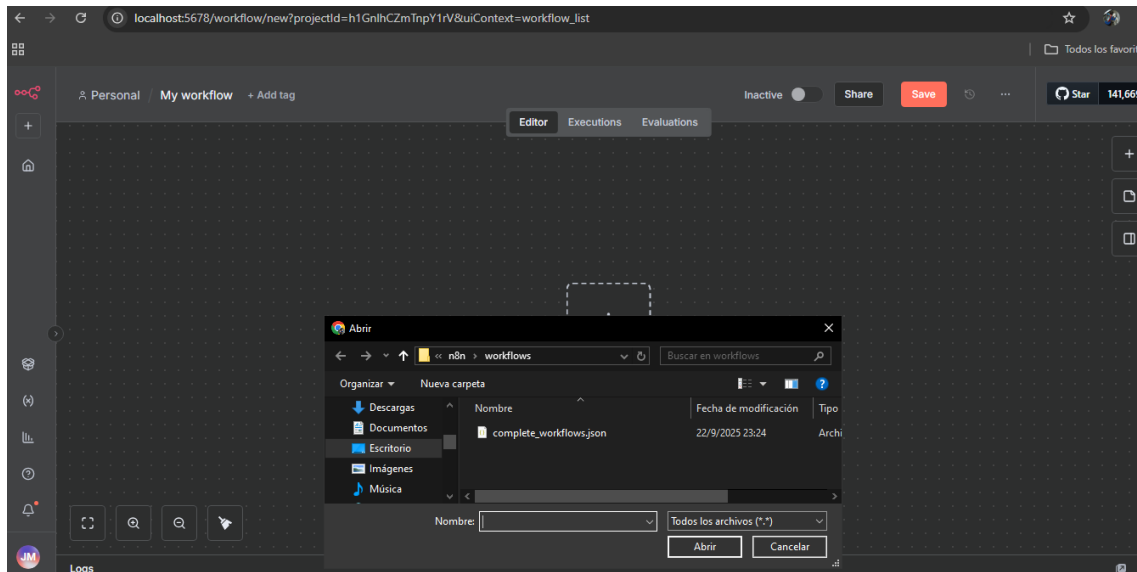
(La primera vez nos pedirá nuestros datos)

Cargamos el workflows: Desde Import From File

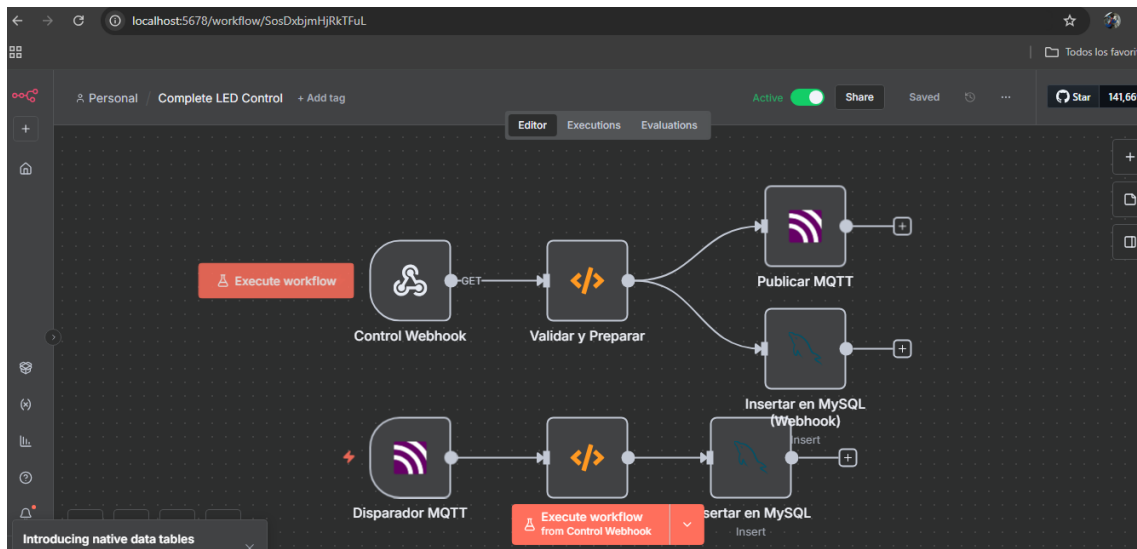


ITES – SOFTWARE FACTORY III

Practico Integrador Docker



Unimos los nodos

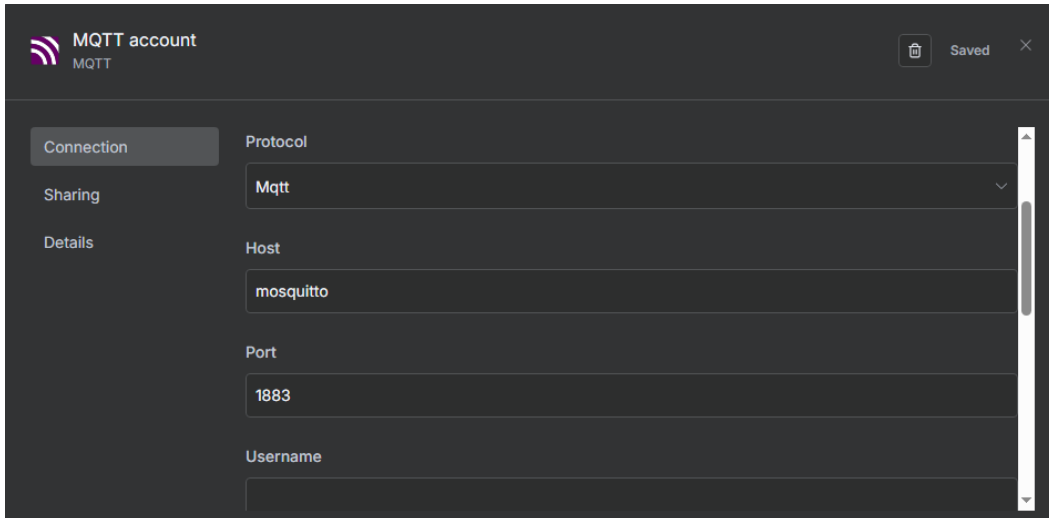


Creamos las Credenciales MSQL y MQTT

A screenshot of the MySQL account configuration form in n8n. The form is titled 'MySQL account' and has a 'Save' button. It contains the following fields: 'Host' (mysql), 'Database' (ledactions), 'User' (leduser), and 'Password' (masked with dots). The form is organized into sections: 'Connection', 'Sharing', and 'Details'.

ITES – SOFTWARE FACTORY III

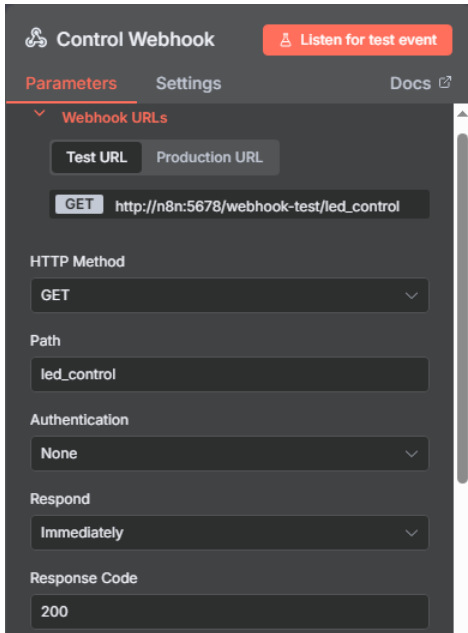
Practico Integrador Docker



The image shows the MQTT account configuration interface. It has a dark theme with a sidebar on the left containing 'Connection', 'Sharing', and 'Details'. The main area is titled 'MQTT account' and has a 'Saved' button. The 'Connection' tab is active, showing fields for 'Protocol' (set to 'Mqtt'), 'Host' (set to 'mosquitto'), 'Port' (set to '1883'), and 'Username' (empty).

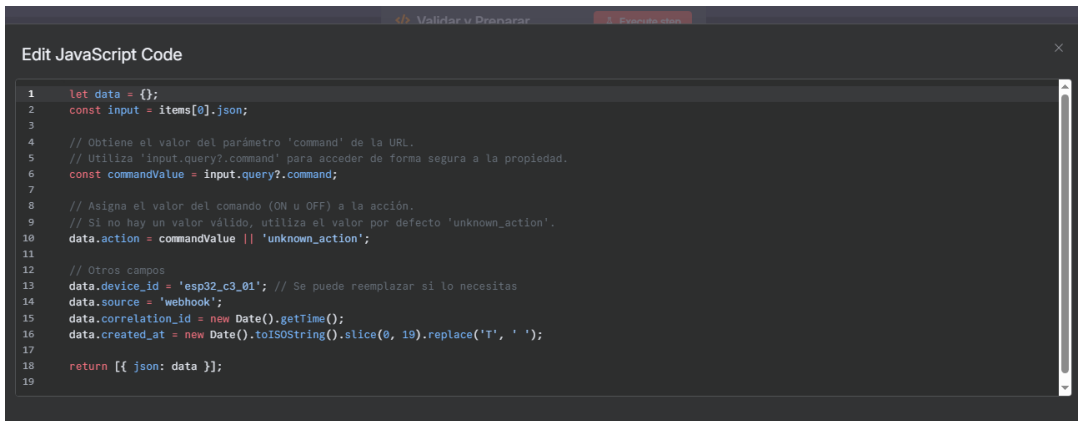
Configuramos los nodos (Gadget): Workflows de Webhook y de MQTT

Control Webhook



The image shows the 'Control Webhook' configuration interface. It has a dark theme with a sidebar on the left containing 'Parameters', 'Settings', and 'Docs'. The 'Parameters' tab is active, showing fields for 'Webhook URLs' (with 'Test URL' and 'Production URL' buttons), 'HTTP Method' (set to 'GET'), 'Path' (set to 'led_control'), 'Authentication' (set to 'None'), 'Respond' (set to 'Immediately'), and 'Response Code' (set to '200').

Validar y Preparar



```
1 let data = {};  
2 const input = items[0].json;  
3  
4 // Obtiene el valor del parámetro 'command' de la URL.  
5 // Utiliza 'input.query?.command' para acceder de forma segura a la propiedad.  
6 const commandValue = input.query?.command;  
7  
8 // Asigna el valor del comando (ON u OFF) a la acción.  
9 // Si no hay un valor válido, utiliza el valor por defecto 'unknown_action'.  
10 data.action = commandValue || 'unknown_action';  
11  
12 // Otros campos  
13 data.device_id = 'esp32_c3_01'; // Se puede reemplazar si lo necesitas  
14 data.source = 'webhook';  
15 data.correlation_id = new Date().getTime();  
16 data.created_at = new Date().toISOString().slice(0, 19).replace('T', ' ');  
17  
18 return [{ json: data }];  
19
```

Publicar MQTT

The 'Publicar MQTT' configuration interface features a dark theme. At the top, there is a title bar with the MQTT icon, the title 'Publicar MQTT', and an 'Execute step' button. Below the title bar are three tabs: 'Parameters' (selected), 'Settings', and 'Docs'. The 'Parameters' section includes a 'Credential to connect with' dropdown menu set to 'MQTT account', a 'Topic' text field containing 'led_control', a 'Send Input Data' toggle switch that is turned on, and an 'Options' section with 'No properties' and an 'Add option' button.

Insertar en MySQL (Webhook)

The 'Insertar en MySQL (Webhook)' configuration interface has a dark theme. The top bar includes the title 'Insertar en MySQL (Webhook)' and an 'Execute step' button. Below are 'Parameters', 'Settings', and 'Docs' tabs. The 'Parameters' section contains a 'Credential to connect with' dropdown set to 'MySQL account', a yellow notification banner about a new node version, an 'Operation' dropdown set to 'Insert', a 'Table' section with a 'From list' dropdown and a text field containing 'led_events', a 'Columns' text field containing 'device_id,action,source,correlation_id,created_at', and an 'Options' section with 'No properties' and an 'Add modifiers' button.

Disparador MQTT

The 'Disparador MQTT' configuration interface features a dark theme. The top bar shows the title 'Disparador MQTT' and an 'Execute step' button. Below are 'Parameters', 'Settings', and 'Docs' tabs. The 'Parameters' section includes a 'Credential to connect with' dropdown set to 'MQTT account', a 'Topics' text field containing 'lab/g1/req/led', and an 'Options' section with 'No properties' and an 'Add option' button.

ITES – SOFTWARE FACTORY III

Practico Integrador Docker

Preparar Datos (MQTT)

```
Back to canvas

Edit JavaScript Code

1 let data = {};
2 const input = items[0].json;
3
4 // Extract the value from the MQTT message payload
5 const actionValue = input.message;
6
7 // Get the current date and format it for MySQL
8 const now = new Date();
9 const mysqlDatetime = now.toISOString().slice(0, 19).replace('T', ' ');
10
11 // Assign the values to the data object
12 data.action = actionValue;
13 data.device_id = 'jesus';
14 data.source = 'device';
15 data.correlation_id = now.getTime();
16 data.created_at = mysqlDatetime;
17
18 return [{ json: data }];
```

Insertar en MySQL

Insertar en MySQL Execute step

Parameters Settings Docs

Credential to connect with

MySQL account

New node version available: get the latest version with added features from the nodes panel.

Operation

Insert

Table

From list led_events

Columns

device_id,action,source,correlation_id,created_at

Options

Dar Save y Active

Personal Complete LED Control + Add tag

Active Share Save

Editor Executions Evaluations

phpMyAdmin: <http://localhost:8081>
(Nos pedirá nuestros datos, según el .env)

phpMyAdmin

Bienvenido a phpMyAdmin

Idioma (Language)

Español - Spanish

Iniciar sesión

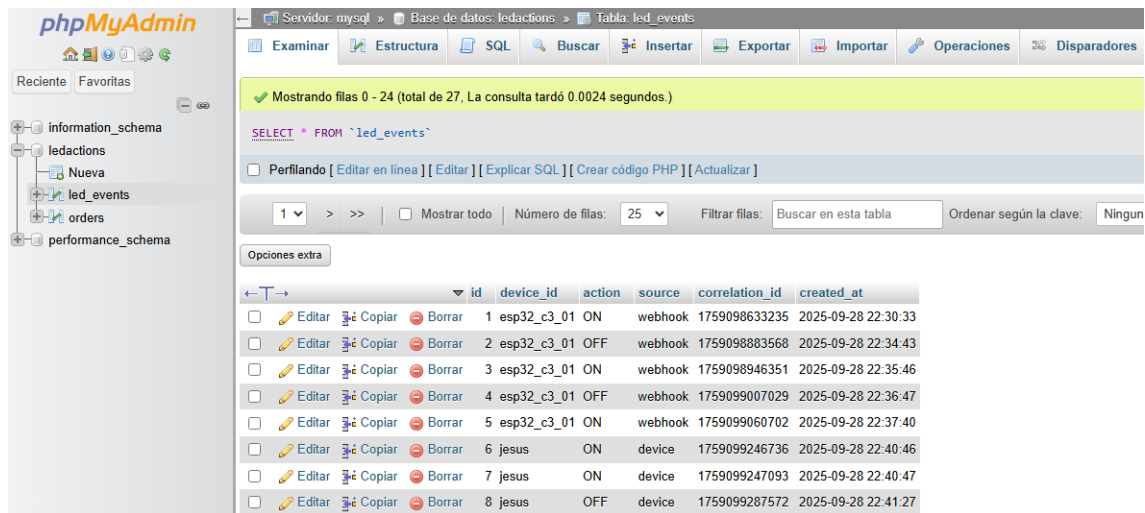
Usuario: leduser

Contraseña:

Iniciar sesión

ITES – SOFTWARE FACTORY III

Practico Integrador Docker



Mostrando filas 0 - 24 (total de 27, La consulta tardó 0.0024 segundos.)

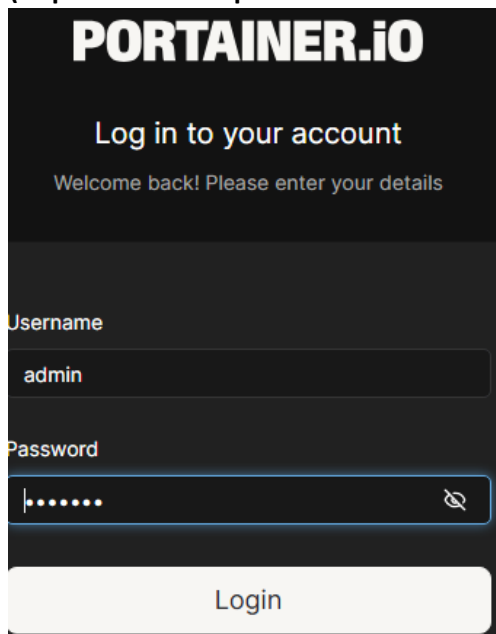
`SELECT * FROM `led_events``

☐ Perfilando [[Editar en línea](#)] [[Editar](#)] [[Explicar SQL](#)] [[Crear código PHP](#)] [[Actualizar](#)]

Número de filas: 25 Filtrar filas: Ordenar según la clave: Ninguno

	id	device_id	action	source	correlation_id	created_at
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	1	esp32_c3_01	ON	webhook	1759098633235	2025-09-28 22:30:33
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	2	esp32_c3_01	OFF	webhook	1759098833568	2025-09-28 22:34:43
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	3	esp32_c3_01	ON	webhook	1759098946351	2025-09-28 22:35:46
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	4	esp32_c3_01	OFF	webhook	1759099007029	2025-09-28 22:36:47
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	5	esp32_c3_01	ON	webhook	1759099060702	2025-09-28 22:37:40
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	6	jesus	ON	device	1759099246736	2025-09-28 22:40:46
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	7	jesus	ON	device	1759099247093	2025-09-28 22:40:47
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	8	jesus	OFF	device	1759099287572	2025-09-28 22:41:27

Portainer: <https://localhost:9443>
(La primera vez te pide crear usuario admin)



PORTAINER.IO

Log in to your account

Welcome back! Please enter your details

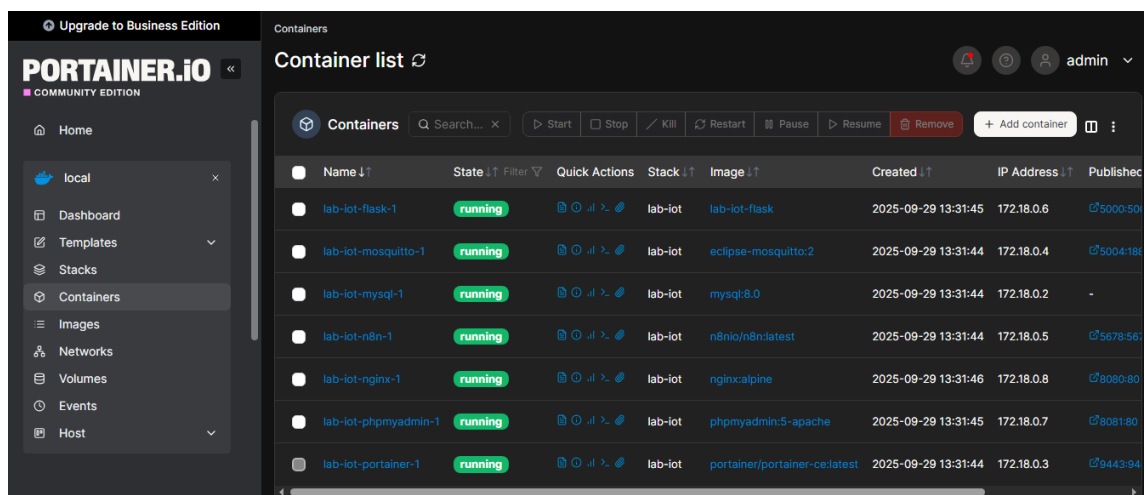
Username

admin

Password

.....

Login



Containers

Container list

Name	State	Quick Actions	Stack	Image	Created	IP Address	Publisher
lab-iot-flask-1	running	Stop Restart Kill Pause Resume Remove	lab-iot	lab-iot-flask	2025-09-29 13:31:45	172.18.0.6	5000:5000
lab-iot-mosquitto-1	running	Stop Restart Kill Pause Resume Remove	lab-iot	eclipse-mosquitto:2	2025-09-29 13:31:44	172.18.0.4	5004:1800
lab-iot-mysql-1	running	Stop Restart Kill Pause Resume Remove	lab-iot	mysql:8.0	2025-09-29 13:31:44	172.18.0.2	-
lab-iot-n8n-1	running	Stop Restart Kill Pause Resume Remove	lab-iot	n8nio/n8n:latest	2025-09-29 13:31:44	172.18.0.5	5678:5678
lab-iot-nginx-1	running	Stop Restart Kill Pause Resume Remove	lab-iot	nginx:alpine	2025-09-29 13:31:46	172.18.0.8	8080:80
lab-iot-phpmyadmin-1	running	Stop Restart Kill Pause Resume Remove	lab-iot	phpmyadmin:5-apache	2025-09-29 13:31:45	172.18.0.7	8081:80
lab-iot-portainer-1	running	Stop Restart Kill Pause Resume Remove	lab-iot	portainer/portainer-ce:latest	2025-09-29 13:31:44	172.18.0.3	9443:9443

ITES – SOFTWARE FACTORY III

Practico Integrador Docker

Prueba en MQTT: localhost:1883

Edit project

Project name
LAB-IOT-CASA

MQTT settings

MQTT Broker
192.168.0.103

Port
5004

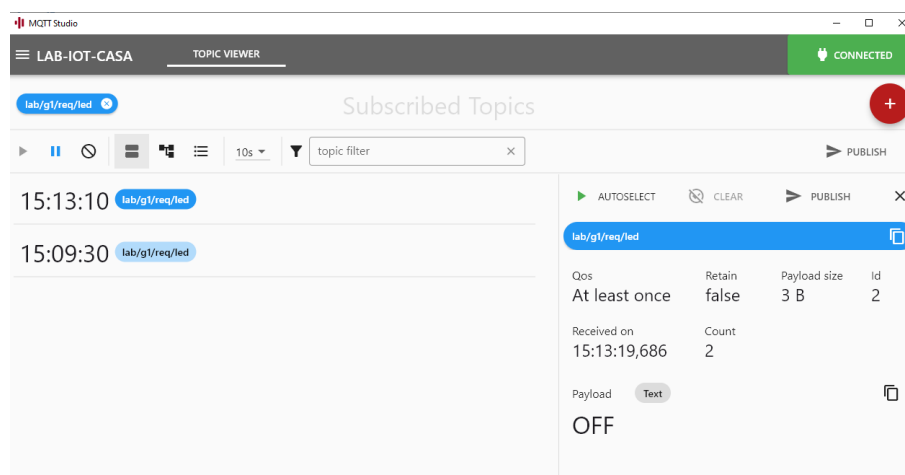
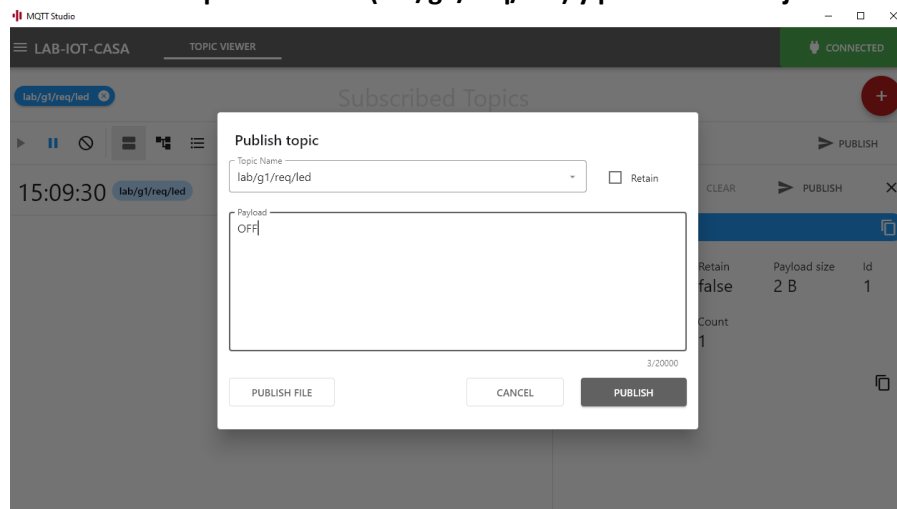
Client Id
616029

Username
Password

☐ Use SSL ☐ Use Websockets

CANCEL OK

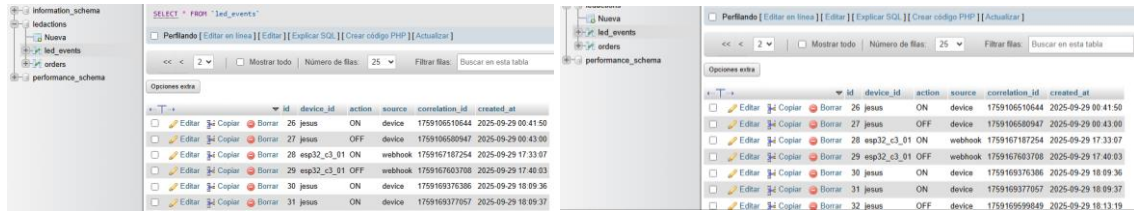
Suscribirse al tópic correcto (lab/g1/req/led) y publicar mensaje



ITES – SOFTWARE FACTORY III

Practico Integrador Docker

Antes y Despues (PhpMyAdmin)



Antes y después (Flask)



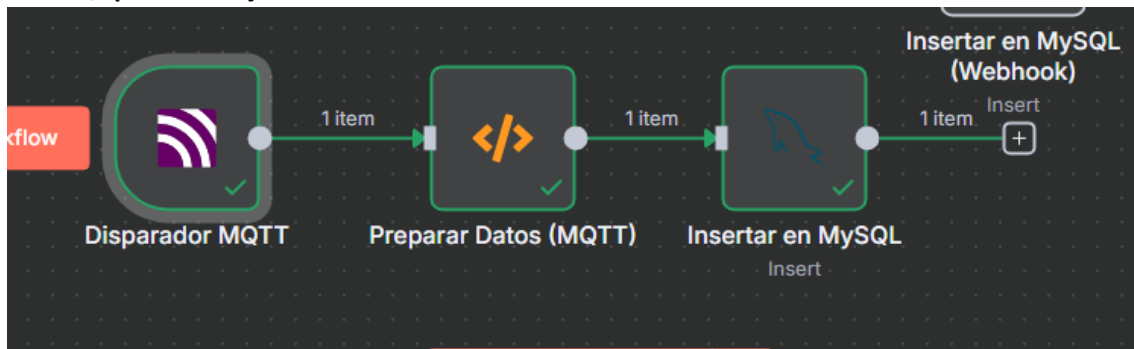
Tabla: led_events

id	device_id	action	source	correlation_id	created_at
34	Webhook	ON	webhook	1759170535417	2025-09-29 18:28:55
33	MQTTstudio	ON	device	1759170485357	2025-09-29 18:28:05
32	jesus	OFF	device	1759169599849	2025-09-29 18:13:19

Tabla: led_events

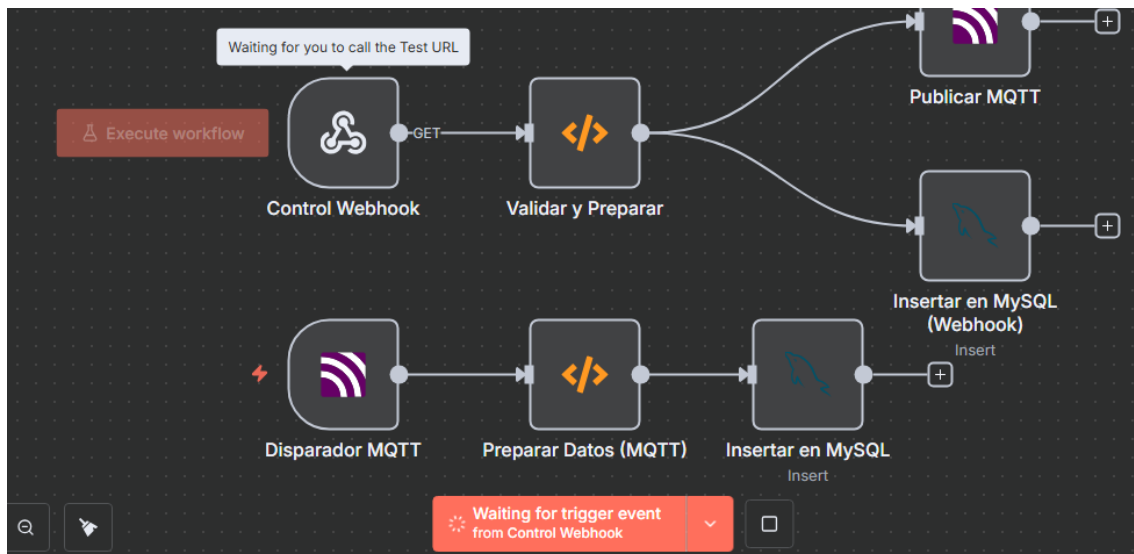
id	device_id	action	source	correlation_id	created_at
34	Webhook	ON	webhook	1759170535417	2025-09-29 18:28:55
33	MQTTstudio	ON	device	1759170485357	2025-09-29 18:28:05

En n8n, queda el flujo en verde con tilde.



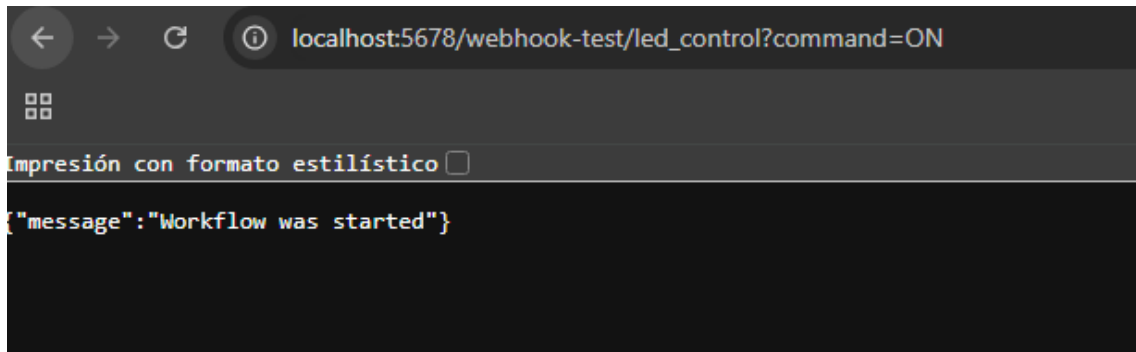
Pruebas Webhook

Poner a correr el Workflow de Webhook



En una ventana de un navegador:

ITES – SOFTWARE FACTORY III
Practico Integrador Docker



Ver antes y despues en la tabla por Flask y por PhpMyAdmin

Antes (Flask)

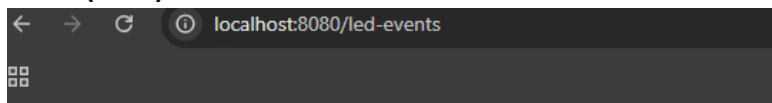


Tabla: led_events

id	device_id	action	source	correlation_id	created_at
27	jesus	OFF	device	1759106580947	2025-09-29 00:43:00
26	jesus	ON	device	1759106510644	2025-09-29 00:41:50

Despues (Flask)

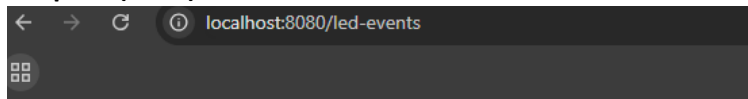
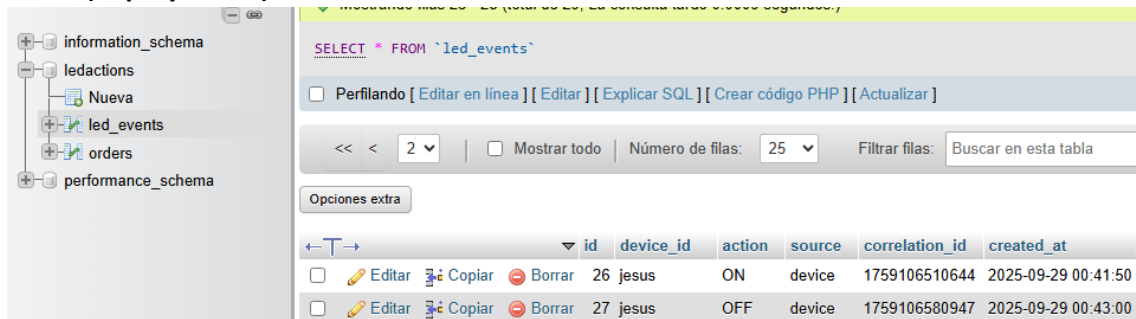


Tabla: led_events

id	device_id	action	source	correlation_id	created_at
28	esp32_c3_01	ON	webhook	1759167187254	2025-09-29 17:33:07
27	jesus	OFF	device	1759106580947	2025-09-29 00:43:00

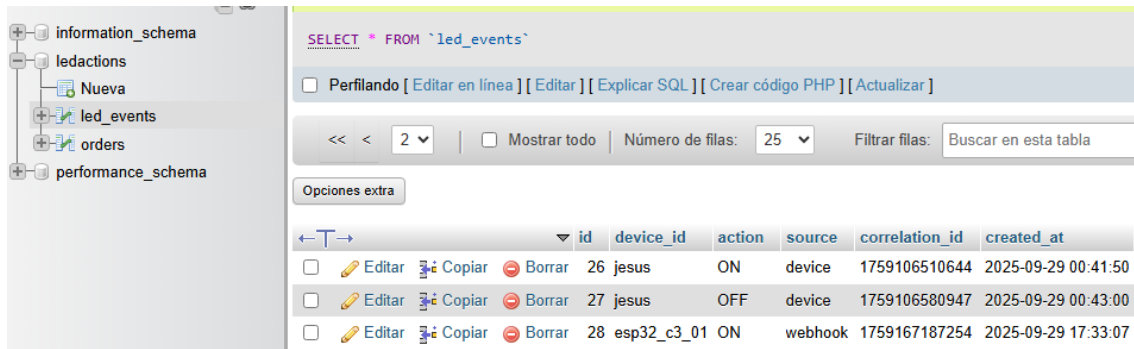
Antes (PhpMyAdmin)



Despues (PhpMyAdmin)

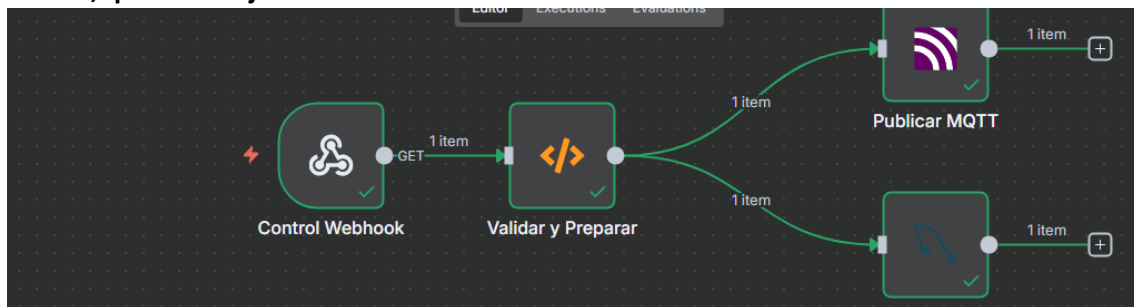
ITES – SOFTWARE FACTORY III

Practico Integrador Docker



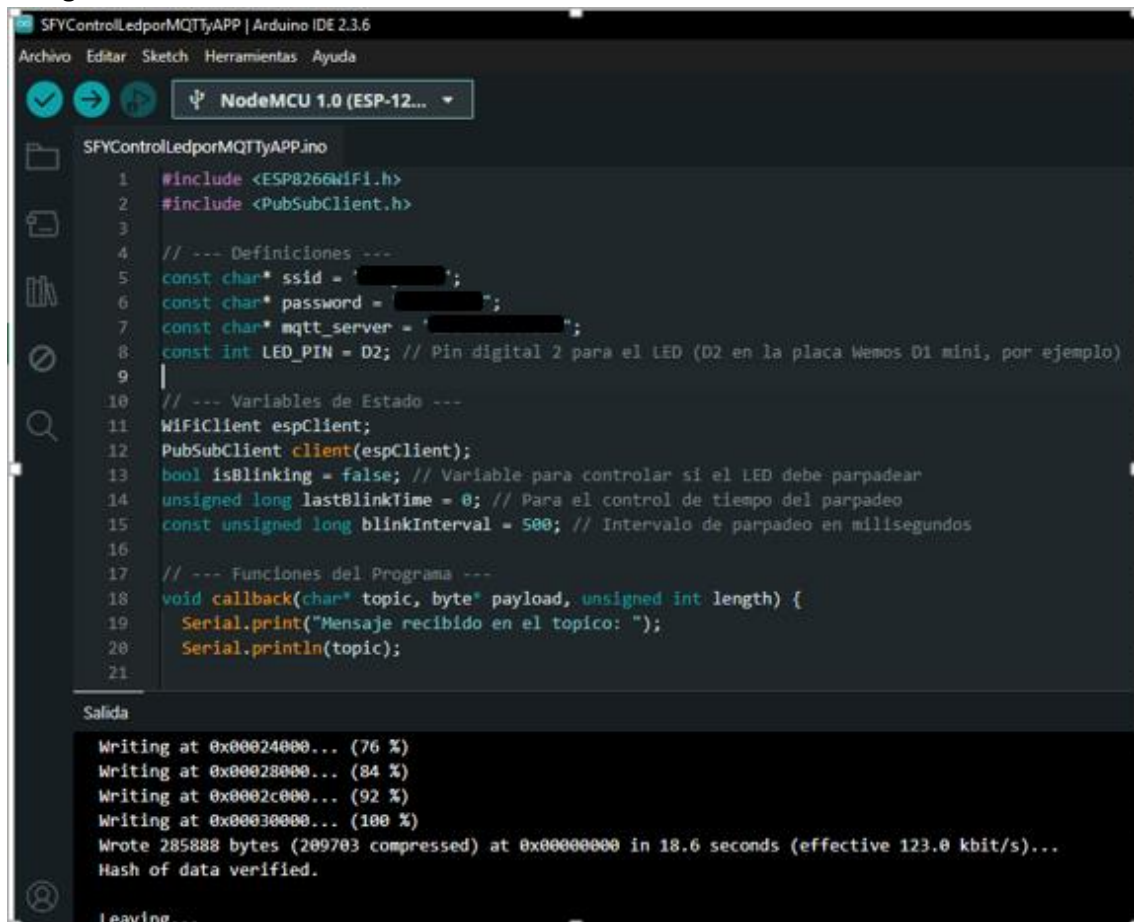
	id	device_id	action	source	correlation_id	created_at
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	26	jesus	ON	device	1759106510644	2025-09-29 00:41:50
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	27	jesus	OFF	device	1759106580947	2025-09-29 00:43:00
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	28	esp32_c3_01	ON	webhook	1759167187254	2025-09-29 17:33:07

En n8n, queda el flujo en verde con tilde.



4. Control del ESP8266 - Led físico, mediante MQTT, y carga en base de datos:

Codigo ArduinoID

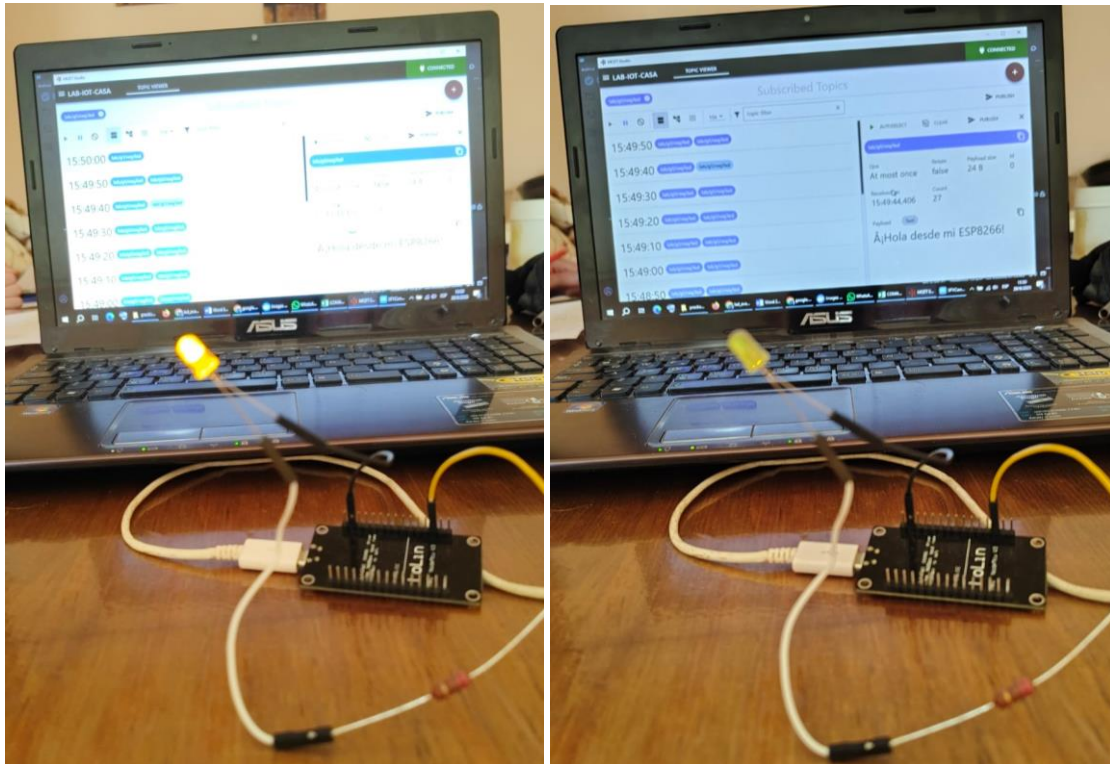


```
1 #include <ESP8266WiFi.h>
2 #include <PubSubClient.h>
3
4 // --- Definiciones ---
5 const char* ssid = " ";
6 const char* password = " ";
7 const char* mqtt_server = " ";
8 const int LED_PIN = D2; // Pin digital 2 para el LED (D2 en la placa Wemos D1 mini, por ejemplo)
9
10 // --- Variables de Estado ---
11 WiFiClient espClient;
12 PubSubClient client(espClient);
13 bool isBlinking = false; // Variable para controlar si el LED debe parpadear
14 unsigned long lastBlinkTime = 0; // Para el control de tiempo del parpadeo
15 const unsigned long blinkInterval = 500; // Intervalo de parpadeo en milisegundos
16
17 // --- Funciones del Programa ---
18 void callback(char* topic, byte* payload, unsigned int length) {
19     Serial.print("Mensaje recibido en el topico: ");
20     Serial.println(topic);
21 }
```

Salida

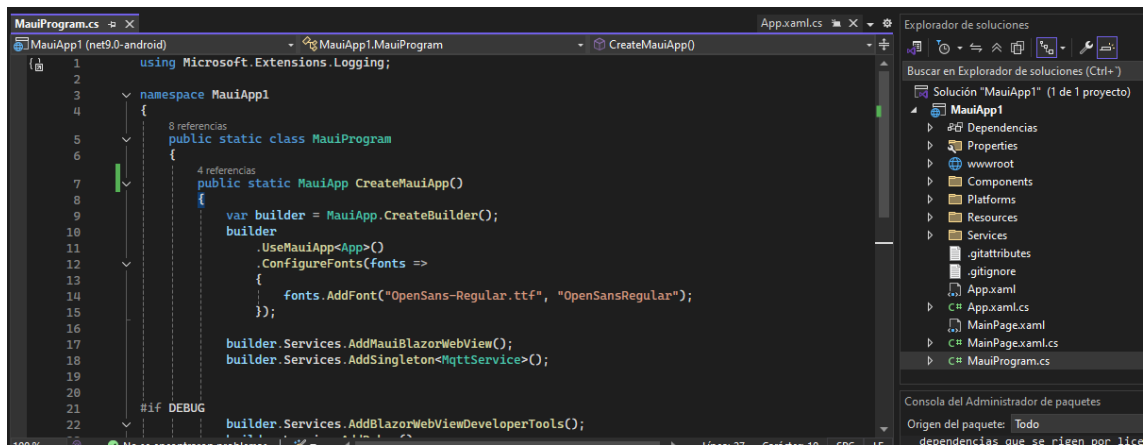
```
Writing at 0x00024000... (76 %)
Writing at 0x00028000... (84 %)
Writing at 0x0002c000... (92 %)
Writing at 0x00030000... (100 %)
Wrote 285888 bytes (209703 compressed) at 0x00000000 in 18.6 seconds (effective 123.0 kbit/s)...
Hash of data verified.
Leaving...
```


Control del ESP8266-LED



5. Control del ESP8266 - Led físico, mediante APP, y carga en base de datos:

Codigo en Visual Studio Community – APP Blazor Hibrid

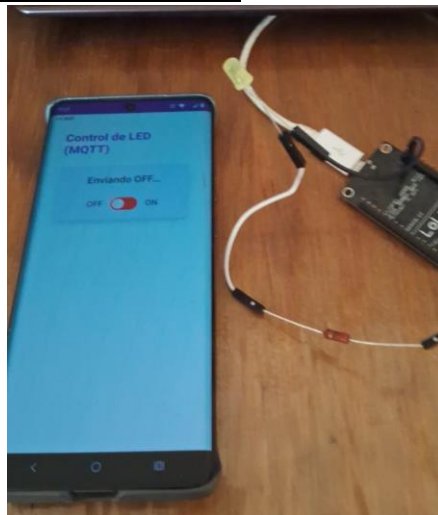
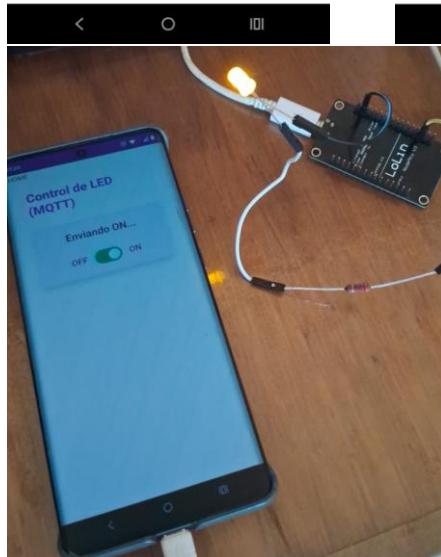
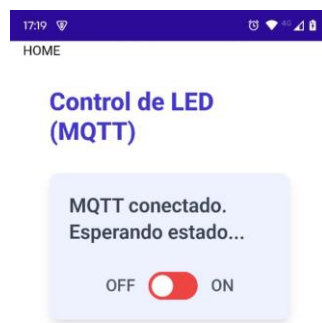
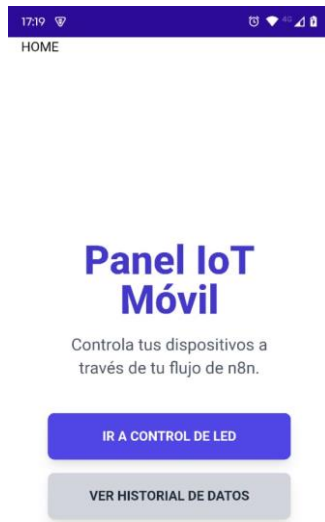


Inicio de APP

Luego de Cargar la APP como Release en el Movil

ITES – SOFTWARE FACTORY III

Practico Integrador Docker



CONCLUSION

El desarrollo de esta mini-plataforma integradora permitió comprobar cómo, a partir de contenedores orquestados con Docker Compose, es posible combinar diferentes tecnologías para lograr una solución completa que abarca desde la capa de infraestructura hasta el hardware IoT.

La integración de Mosquitto, MySQL, phpMyAdmin, n8n, Flask, Nginx y Portainer demostró la potencia de los entornos basados en contenedores para simplificar la gestión, el despliegue y la interoperabilidad entre servicios. Al mismo tiempo, el agregado de una aplicación móvil en .NET MAUI Blazor Hybrid y un dispositivo ESP8266 permitió validar la interacción en escenarios reales: el envío de comandos desde la app, su procesamiento en flujos de n8n (tanto vía MQTT como vía Webhook), el registro en la base de datos y la ejecución física en el LED.

Este trabajo no solo consolidó conceptos de virtualización, bases de datos y desarrollo de aplicaciones, sino que también mostró cómo articular software y hardware en una solución práctica, escalable y replicable. En definitiva, se logró un ejemplo claro de ecosistema IoT-Web-Mobile donde la automatización, la observabilidad y la persistencia de datos trabajan de forma integrada y coordinada.