

Proyecto Final Capstone

Año: 2025

## Integrantes:

* Scipioni Julian
* Tapuerca Thiago

**Tecnicatura Universitaria en Programación**

**Cátedra “Arquitectura y Sistemas Operativos”**

**Profesor:** Mateo Menvielle

## 

## Resumen

Este proyecto consiste en la aplicación de los conocimientos adquiridos en la cátedra de Arquitectura y Sistemas Operativos. Con el fin de utilizar tecnologías como Docker, Node-Red y agregarles conceptos previamente estudiados.

# Introducción

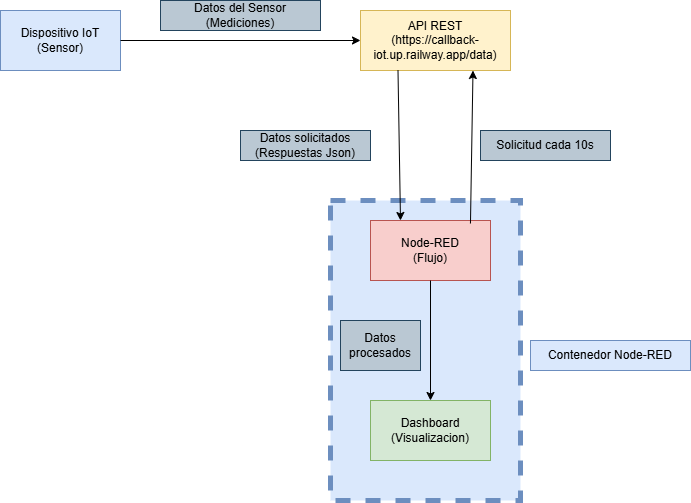
Este proyecto consiste en la extracción de datos de una API REST, utilizando Node-Red para poder procesar y mostrar esos datos. Conteniendo toda la solución del proyecto en un contenedor de Docker, para que dicha solución sea aislada y no afecte a terceros.

## Objetivo General

El proyecto busca sintetizar los conocimientos adquiridos en el curso mediante el diseño de un sistema contenerizado que:

* Consume datos en tiempo real desde un endpoint IoT público (https://callback-iot.up.railway.app/data).
* Procesa y visualiza los dos registros más recientes usando Node-RED.
* Empaqueta la solución en contenedores Docker para garantizar portabilidad y escalabilidad.

## Diagrama de Bloques



## Fundamentación Teórica

### ¿Qué es Docker y para qué se utiliza?

Docker es una plataforma de código abierto que simplifica el desarrollo, el despliegue y la ejecución de aplicaciones al utilizar la tecnología de contenedores. Esta nos permite empaquetar una aplicación y todas sus dependencias (como bibliotecas, configuraciones y variables de entorno) en un entorno estándar llamado contenedor.

Un contenedor es una unidad ligera, ejecutable e independiente que incluye todo lo necesario para ejecutar el software.

Principalmente se utiliza para resolver el problema de la inconsistencia en los entornos de desarrollo, prueba y producción, y para facilitar la gestión y el despliegue de aplicaciones.

### ¿Qué es una API REST y cómo se consulta?

Una API REST, es una API web que sigue las reglas del estilo arquitectónico REST (Representational State Transfer), y se comunica por HTTP usando operaciones como GET, POST, PUT, DELETE. Estas operaciones pueden ser ejecutadas tanto por un navegador o a través de líneas de comandos.

### ¿Qué características generales tiene un sistema IoT y qué tipo de datos genera?

Un sistema IoT, es una red de dispositivos conectados por medios de internet, que interactúan entre ellos y pueden recopilar información y actuar en base a esa información recopilada.

**Características Generales**:

* Sensores y Actuadores

Los sensores recopilan información del entorno (temperatura, humedad, movimiento, etc.).Y los actuadores ejecutan acciones en respuesta (abrir una válvula, encender una luz, etc.).

* Conectividad

Utiliza redes como Wi-Fi, Bluetooth, ZigBee, LoRa, NB-IoT, 4G/5G para enviar datos a servidores o nubes.

* Procesamiento de Datos

Puede haber procesamiento en el borde (edge computing) o en la nube (cloud computing).

* Automatización e Inteligencia
* Interfaz de Usuario
* Escalabilidad

Capacidad para crecer con más dispositivos o usuarios sin perder funcionalidad.

* Seguridad

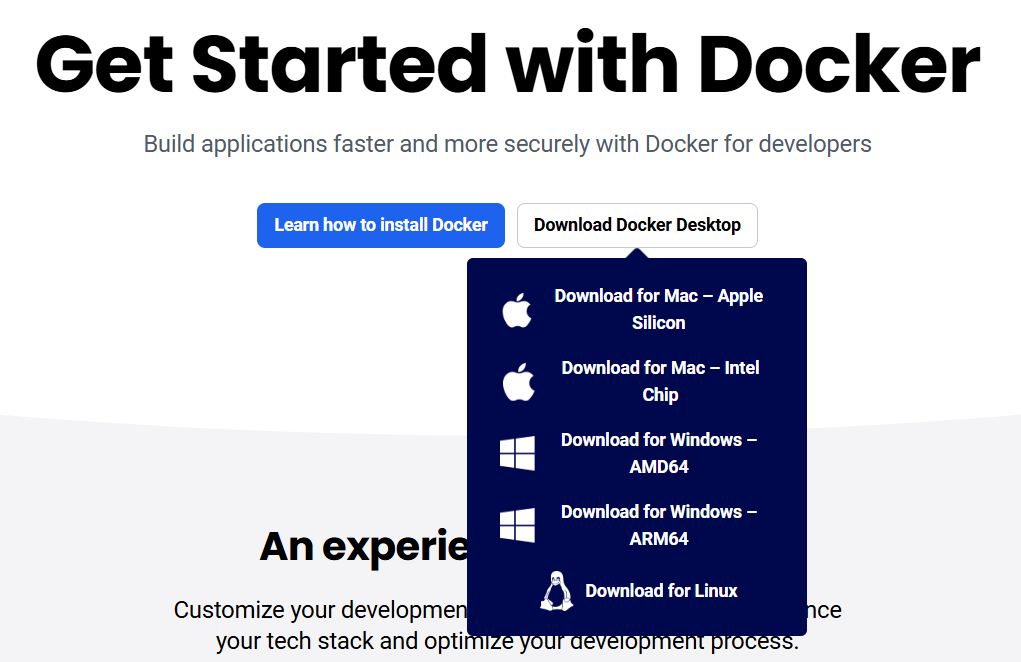
### ¿Qué papel cumple una plataforma como Node‑RED en este tipo de arquitectura?

Primero que nada debemos saber que Node-RED es una plataforma de programación visual basada en flujos, desarrollada por IBM, que permite conectar dispositivos, servicios y APIs de forma sencilla. Se usa especialmente en proyectos IoT, ya que en el contexto de la Arquitectura IoT, tiene una gran participación en la capa de procesamiento de datos que aquí es donde se produce la integración, el procesamiento y el control de los datos.

# Desarrollo del Proyecto

Para empezar el desarrollo del proyecto, previamente se hizo un análisis de lo que se pedía como solución y se busco la manera de ejecutarla lo mejor posible. A continuación explicaremos el proceso del desarrollo de la solución.

## Instalacion y Utilizacion de Docker

Para comenzar el proyecto, primero instalamos Docker Desktop desde su página oficial.(<https://www.docker.com/get-started/>)

Una vez que clickeamos “Download Docker Desktop”, seleccionamos la opción que más nos convenga. En nuestro caso, seleccionamos Download for Windows - AMD64.

Una vez que tengamos instalado Docker Desktop, podremos comenzar con el proyecto.

## Instalación y Contenerización de Node-RED

Con docker ya instalado, el siguiente paso es instalar y guardar dentro de un contenedor de Docker la app Node-RED. Para ello haremos lo siguiente:

En la terminal de PowerShell de Windows ejecutamos estos comandos.

1. Creamos la carpeta que contendrá el archivo docker-compose.yml



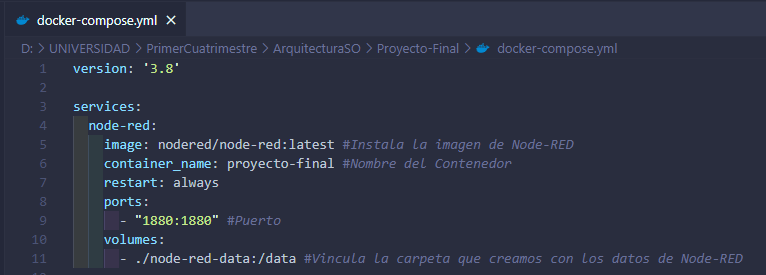
1. Dentro de esta carpeta, crearemos un subdirectorio para los datos de Node-RED.



1. Luego de esto volvemos a el directorio raíz del proyecto y creamos el archivo docker-compose.yml



1. Una vez que ejecutemos el comando se nos abrira visual studio code, donde debemos pegar el siguiente código.



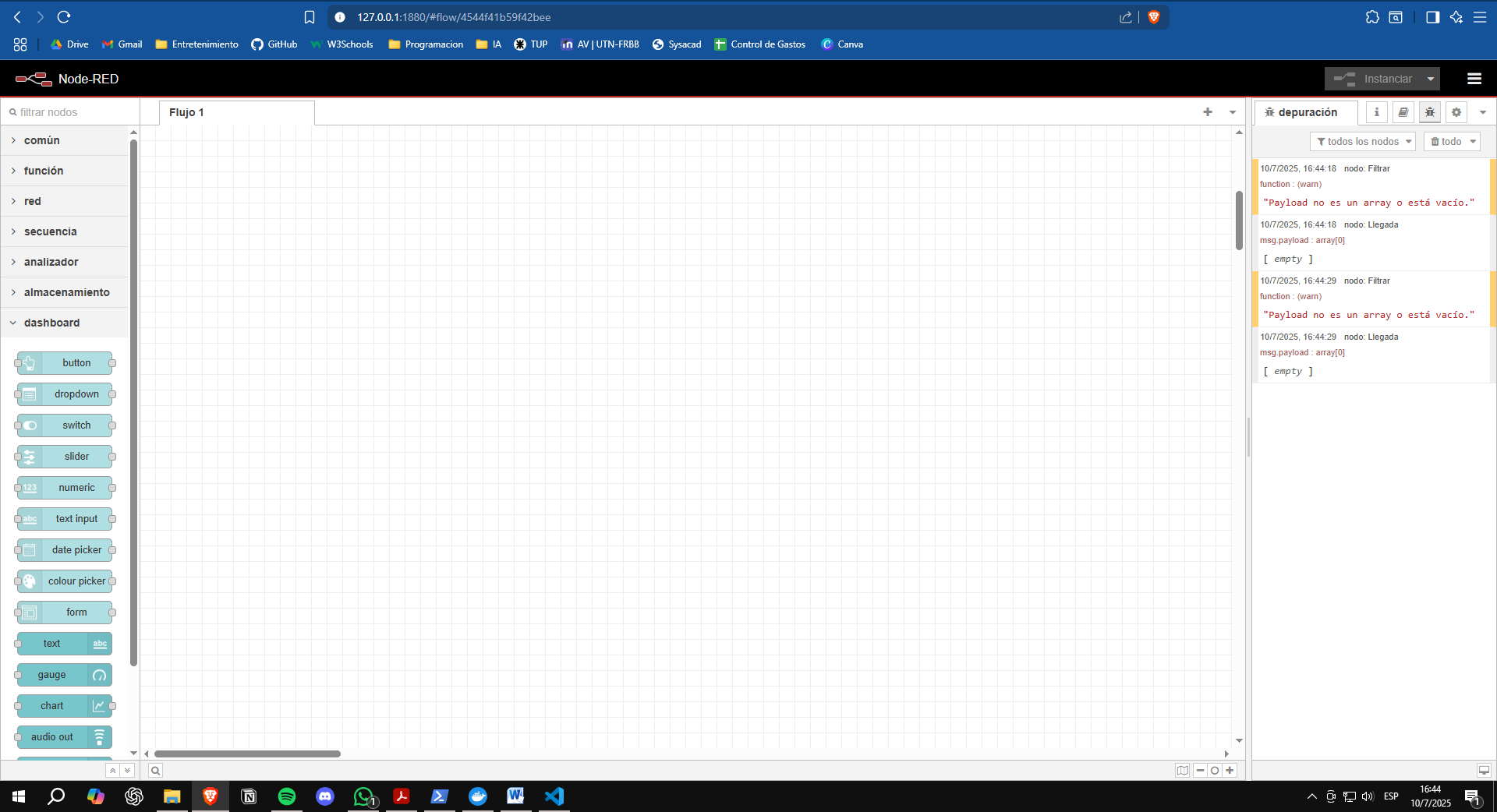
1. Con estos pasos ya instalamos Node-RED y lo guardamos en el Contenedor. Ahora para abrir Node-RED, debemos en nuestra terminal de comandos levantar el contenedor, con el siguiente comando:



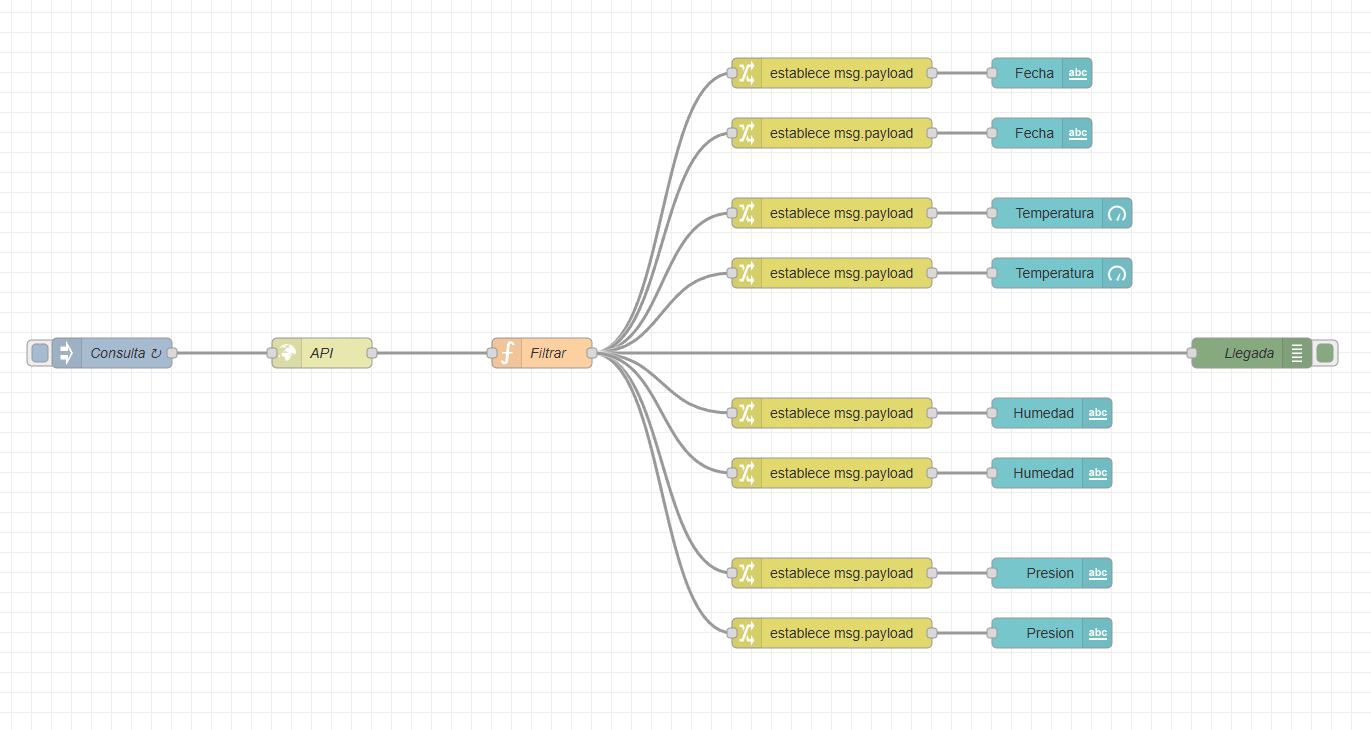
1. Por último, abrimos el navegador y ponemos la IP, con el puerto.



1. Finalmente deberíamos estar en un espacio de trabajo en Node-RED.



## Creación del Flujo de Datos con Node-RED

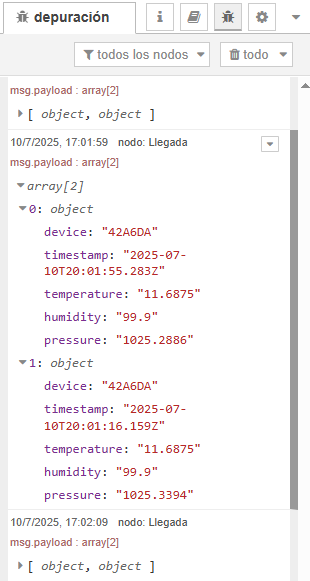


El funcionamiento del flujo consta de 5 bloques principales, programados con el finde lograr el funcionamiento esperado.

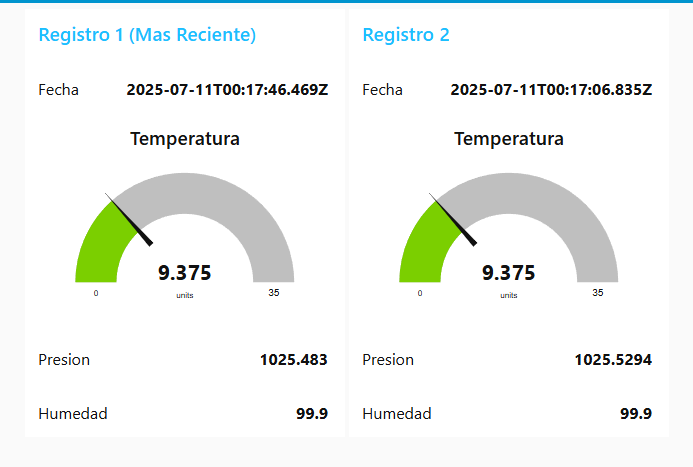
1. Primero se utiliza un bloque inject (Consulta), este se encarga de hacer la consulta a la API, en el intervalo de tiempo que queramos y activar el flujo.
2. El bloque llamado API, bloque de tipo HTTP Request, es el que solicita los datos a la API dada, utilizando el método GET.
3. El siguiente nodo, el cual tiene conectado la API, es un nodo función programado para ordenar el Json recibido y retornar los 2 valores con la fecha más reciente.
4. Los nodos de color amarillo, se llaman change y ayudan a cambiar el dato que reciben los widgets encargados de mostrar dicho datos.
5. Por último los bloques celestes son 2 tipos de widgets, text, muestra los datos en formato texto y gauge genera un gráfico de visualización.

## Pruebas Funcionales

Para comprobar el funcionamiento del flujo, podemos ver los mensajes de depuración de Node-RED.



O otra forma, ya que se utilizaron widgets es ver la pestaña dashboard, donde los datos estan representados por los widgets.



# Conclusiones

## Resultados obtenidos

El sistema consume datos cada 10 segundos, filtra los dos registros más recientes y los visualiza en un dashboard accesible vía web. La contenerización con Docker permite replicar la solución en cualquier entorno compatible.

## Reflexión final

Este proyecto demuestra la viabilidad de integrar tecnologías modernas (contenedores, IoT, flujos de datos) para resolver problemas del mundo real. La experiencia consolida competencias en arquitectura de sistemas, evidenciando cómo la virtualización optimiza el ciclo de vida del desarrollo software.

Personalmente, ayudó a poner en práctica lo aprendido en la cátedra y dar como un cierre más práctico.