CAPÍTULO 9. Node-RED and more

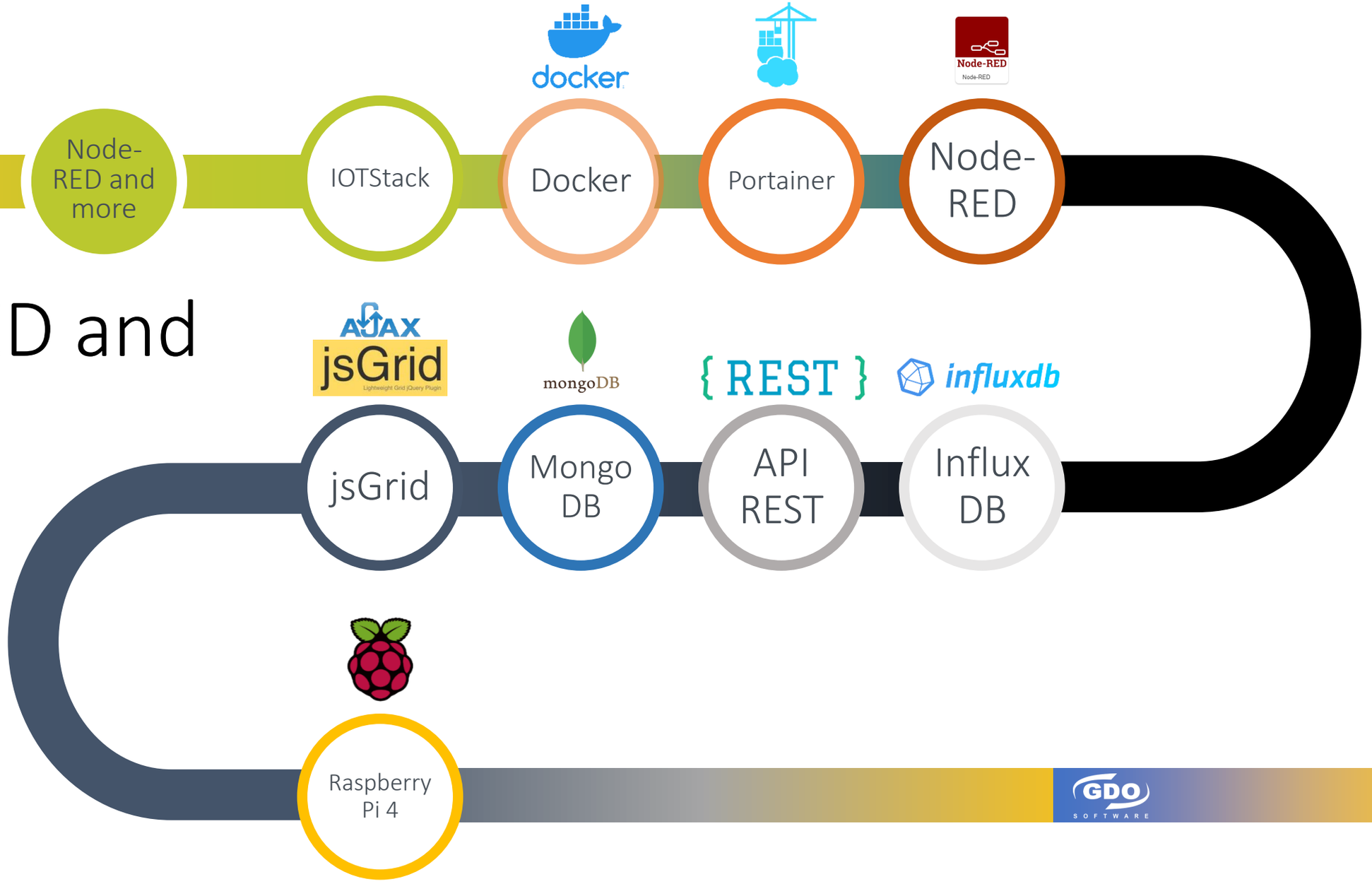
v.1.4 SEPTIEMBRE 2024

Ricardo Moraleda Gareta

[Director departamento de software de GDO Software]

Node-RED and more

v.1.4 SEPTIEMBRE 2024



IOT Stack

Entorno

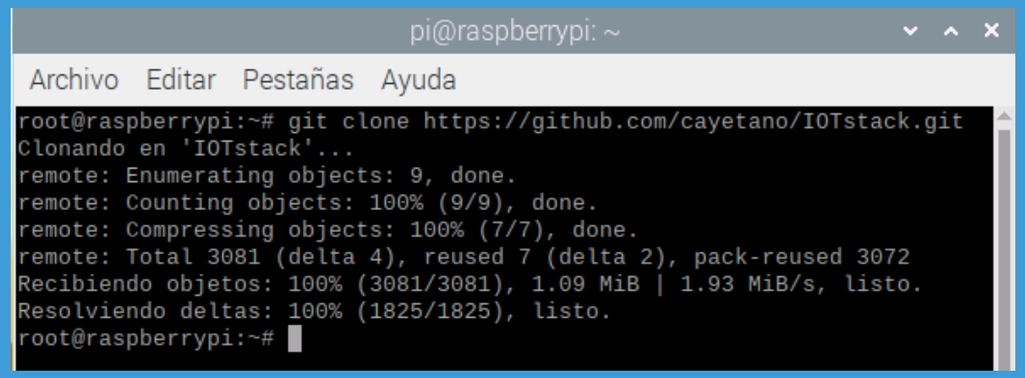
Muchas veces me ha pasado tener que volver a instalar de 0 todas las aplicaciones en la Raspberry Pi 4 porque no tenía una copia de seguridad. Volver a instalar cada aplicación lleva su tiempo por este motivo presento cómo instalar Docker y un menú para la gestión de las aplicaciones típicas de IOT.

El proyecto está en github y es conocido por IOTStack. Partiré del proyecto de un youtuber (@Cayetano Gómez).

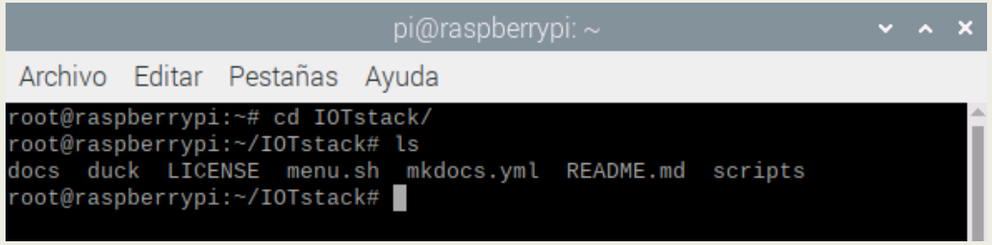
https://youtu.be/kv3fqcbAtns < -- > https://github.com/cayetano/IOTstack

Es un fork de Sensorslot y a su vez del original de GCGarner.

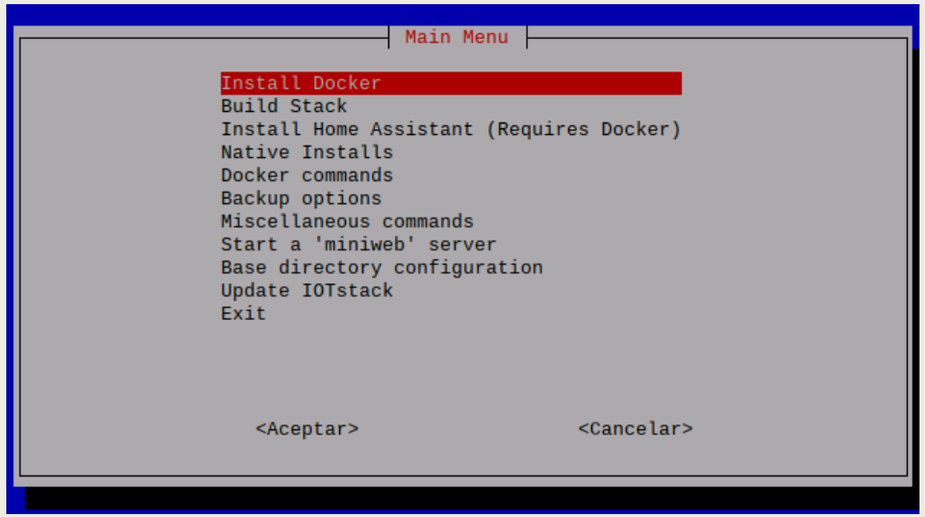
sudo -i



Se clona el proyecto en local y crea una carpeta llamada "IOTStack"



Arrancamos el menú con el comando "./menu.sh"

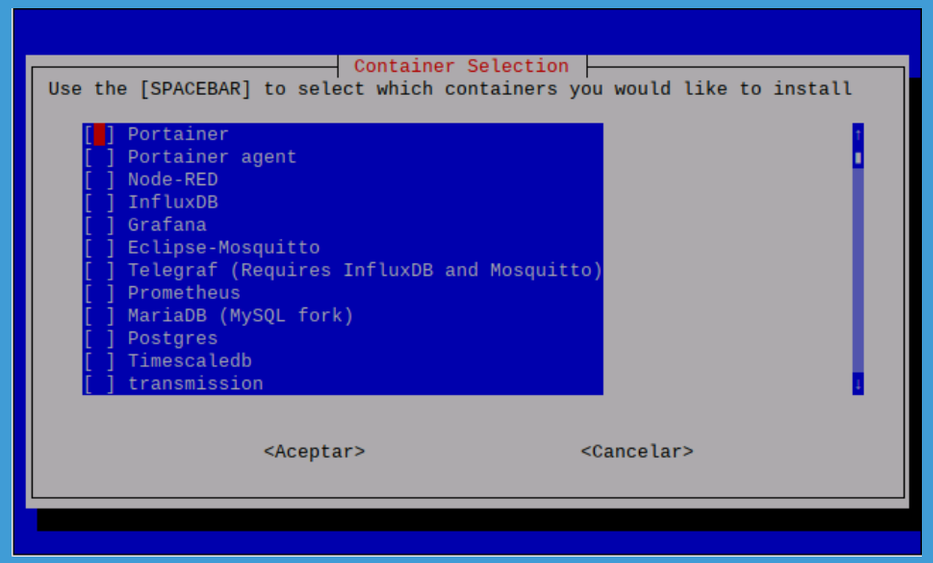


Docker

Docker

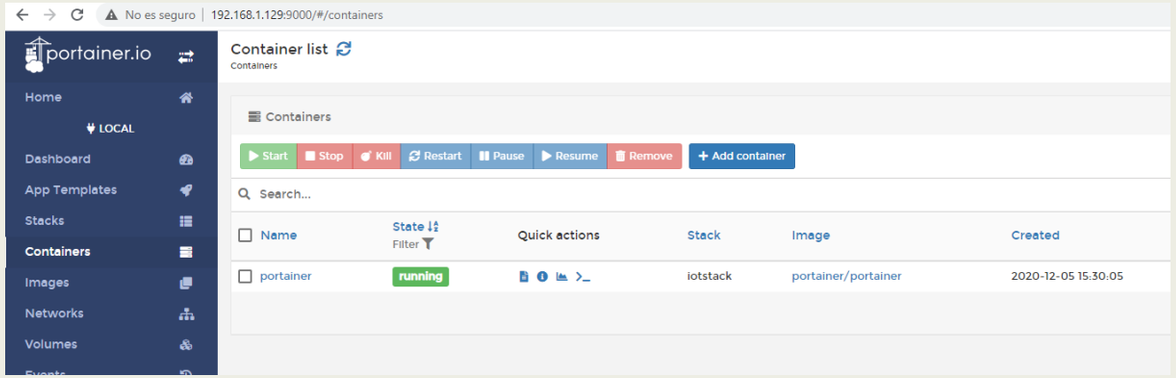
Seleccionaremos la primera opción del menú "Install Docker" y cuando acabe reiniciamos la RPI4. Volvemos a abrir el menú.

Luego seleccionamos la segunda opción del menú "Build Stack". Aparecerá una lista de aplicaciones instaladas o por instalar. Son aplicaciones típicas de IOT (34 aplicaciones). En docker se les llama contenedores.



Lo primero de todo, instalaré **Portainer** v.2.0.0 (Gestor web de las contenedores instalados) en el puerto 9000.

Para entrar poner la IP de la RPI4 y el puerto 9000 en el navegador web. Se gestionará un entorno local.



Podemos ver los contenedores que tenemos instalados. En este caso, de momento, **Portainer**.

Desde aquí se podrán hacer imágenes de estos contenedores para moverlos a otro sitio o como Backup para reponerlos en caso de fallo o desastre.

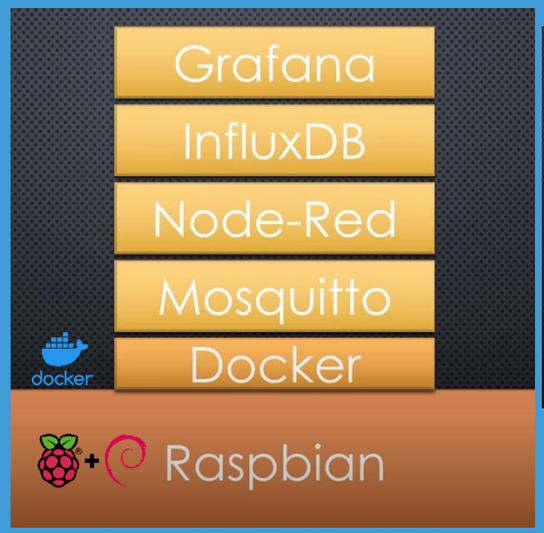
Docker vs Virtualización

Docker vs Virtualización

Basado en https://youtu.be/a6mjt8tWUws

(vídeo #295 de Andreas Spiess)

Como pincelada, los contenedores contienen lo estrictamente necesario (aplicación) para funcionar en el sistema operativo pero no contiene parte del sistema operativo. Los contenedores son más flexibles y más ligeros. Es la principal diferencia entre ambos mundos.



Microsoft Hyper-V

vmware

Los datos de los contenedores se pueden separar o sacar del contenedor al sistema operativo de manera que independicemos la parte de aplicación (instalación) de los datos. En el caso de borrar el contenedor los datos no se pierden.

Es más, no sería necesario hacer copia de seguridad del contenedor entero ya que reponerlo es relativamente sencillo y rápido. Únicamente se haría Backup de los datos (alojados en el S.O.) – volúmenes.

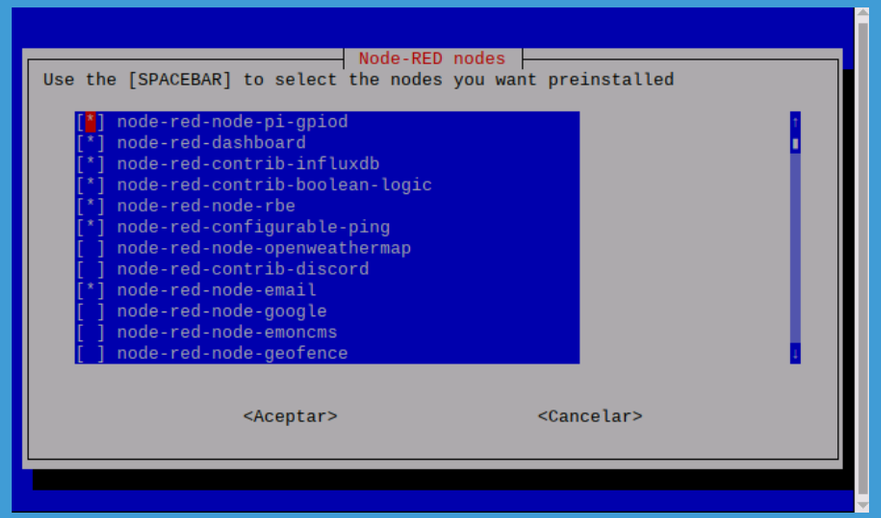
Volúmenes de datos

IOT Stack / Node-RED

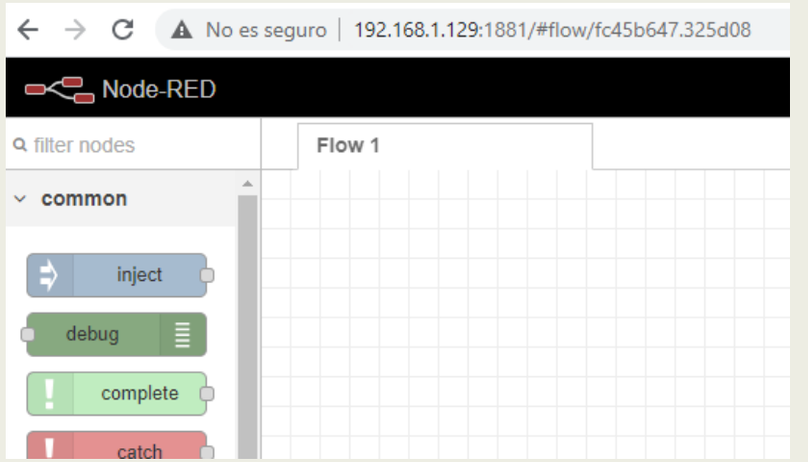
Instalar Node-RED

Volvemos al menú “Build Stack” y seleccionamos Node-RED. Lo instalaré en el puerto 1881 porque ya tenía anteriormente esta aplicación en el puerto 1880.

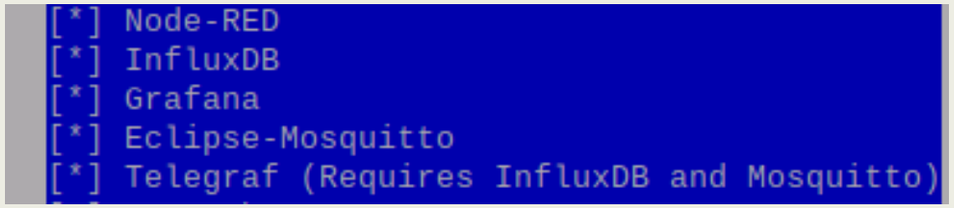
Como podéis ver te da una lista de nodos de entrada y puedes marcar muchos otros y no tener que instalarlos uno a uno desde el propio Node-RED.



Puerto 1881



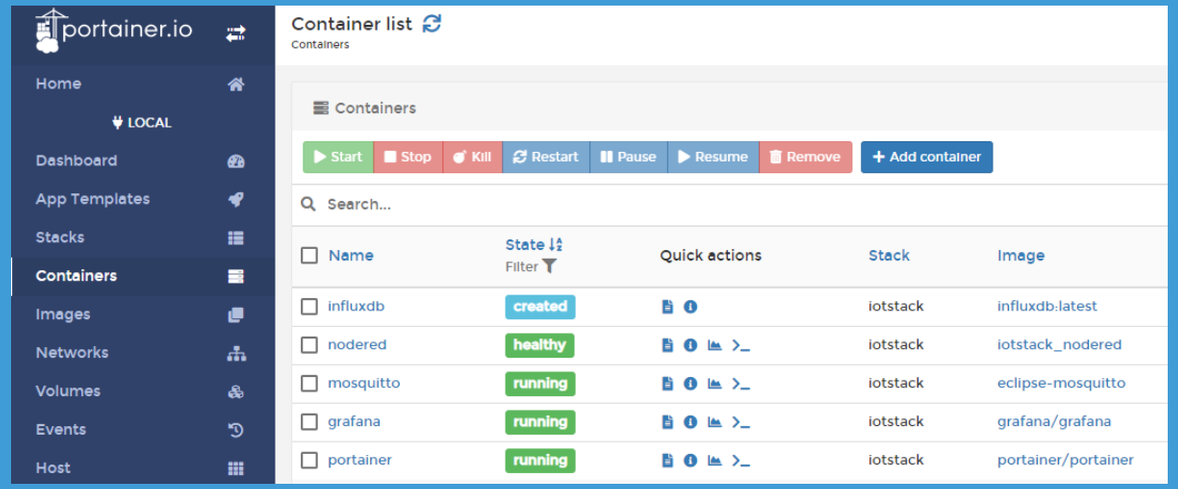
También instalaré InfluxDB, Grafana, Eclipse-Mosquitto y Telegraf.

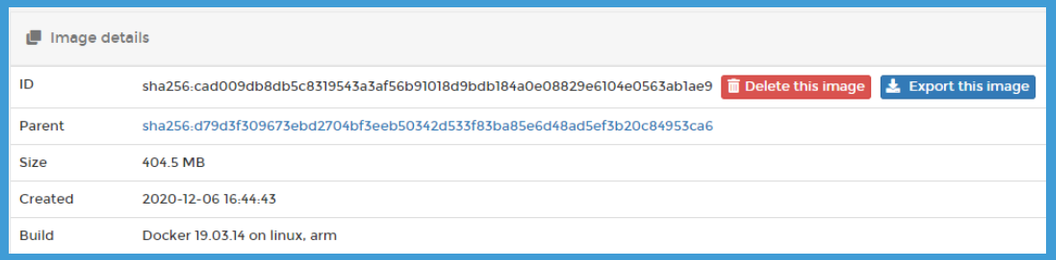


Portainer

Gestión de contenedores

Podemos ver los 5 contenedores instalados en Docker con la aplicación Portainer.

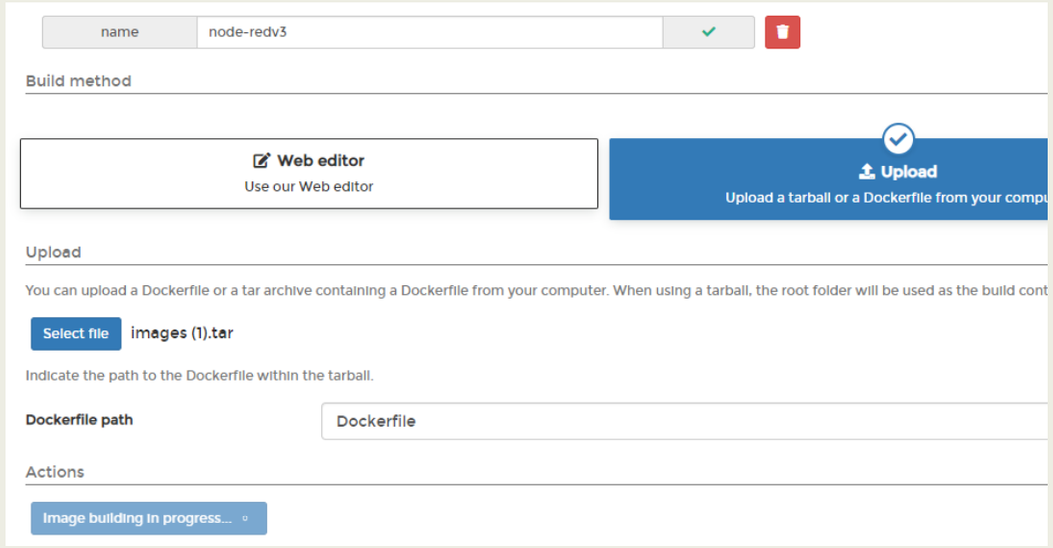




Gestión de imágenes/volúmenes

También puedes hacer imágenes de los contenedores y en algún momento reponerlas en caso de Recovery o moverlas a otra plataforma que tenga Docker.

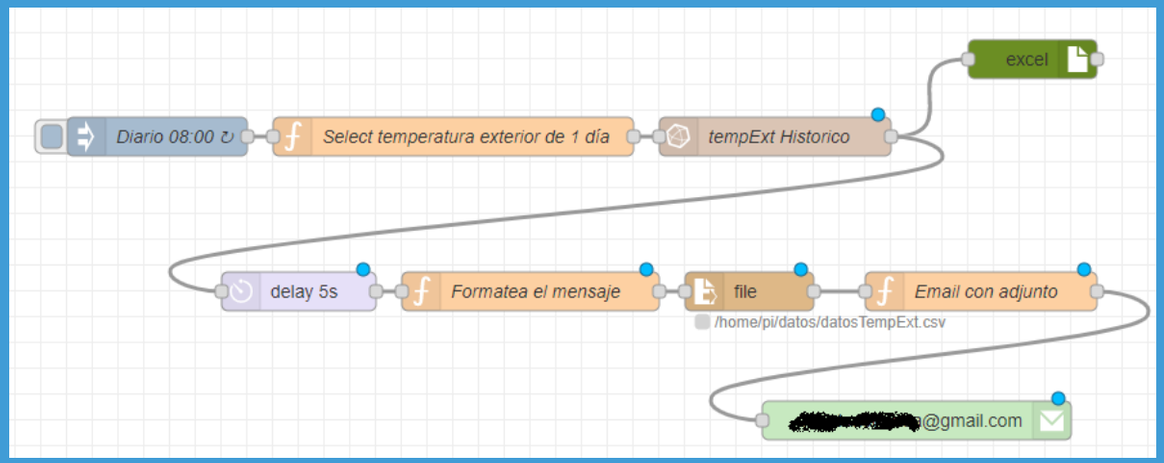
Me he exportado un fichero "images(1).tar" (con la imagen de node-red) de unos 400 Mbytes y la estoy subiendo de nuevo.



Node-RED

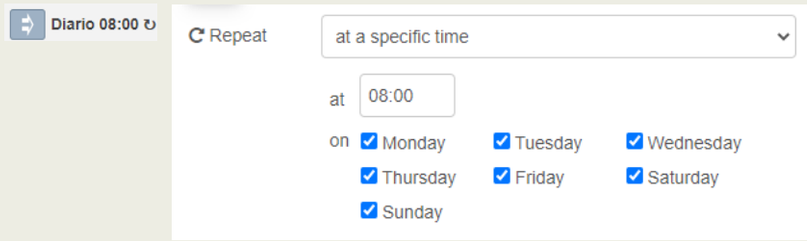
#1-BBDD-CSV-Email

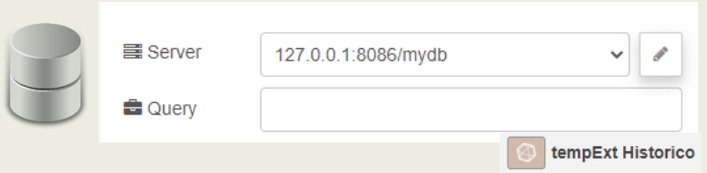
Clásico ejemplo de extracción de datos de una base de datos, generación de fichero CSV y envío como adjunto por e-mail. El cómo recolecto estos datos está en el "**Capítulo8. Domótica IoT**"



BBDD

Cada día a las 08:00h ejecuta una query para obtener los datos de temperatura del día anterior. Los guarda en un fichero CSV y lo adjunta por e-mail.





influxdb

Node-RED

CSV

excel

File * /home/pi/datos/datosTempExt.csv

Name Name

CSV

Formatea el mensaje

```
1 msg.filename = "/home/pi/datos/datosTempExt.csv";
2 msg.topic = 'Datos temperatura exterior';
3 return msg;
```

file

Filename Filename

Output a single utf8 string

Encoding default

Name Name

Email con adjunto

```
1
2 msg.payload = "";
3
4 msg.attachments = [{
5     filename: msg.filename ,
6     path: msg.filename ,
7     content: msg.payload
8 }];
9
10 return msg;
```

E-mail

To [redacted]@gmail.com

Server smtp.gmail.com

Port 465 Use secure connection.

Userid [redacted]@gmail.com

Password ••••••••

Use TLS?

Name Name

Entrando en el correo GMAIL, veo el adjunto.

Datos temperatura exterior Recibidos x

[redacted]@gmail.com
para mí

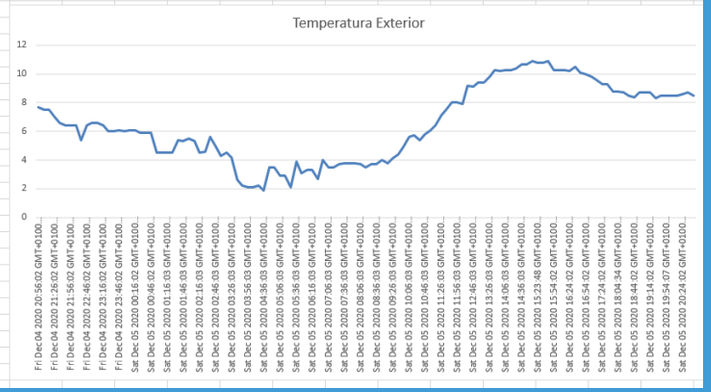
/home/pi/datos/datosTempExt.csv
8.4 KB

Node-RED

Datos CSV y gráfica MS Excel

Me he bajado el CSV y he generado con MS Excel la gráfica con la fuente de datos enviada en el fichero adjunto del e-mail.

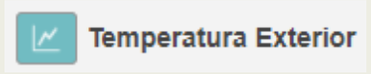
	A	B
1	time	value
2	Fri Dec 04 2020 20:56:02 GMT+0100 (hora estándar de Europa central)	7,7
3	Fri Dec 04 2020 21:06:18 GMT+0100 (hora estándar de Europa central)	7,5
4	Fri Dec 04 2020 21:16:02 GMT+0100 (hora estándar de Europa central)	7,5
5	Fri Dec 04 2020 21:26:02 GMT+0100 (hora estándar de Europa central)	7
6	Fri Dec 04 2020 21:36:02 GMT+0100 (hora estándar de Europa central)	6,6
7	Fri Dec 04 2020 21:46:02 GMT+0100 (hora estándar de Europa central)	6,4
8	Fri Dec 04 2020 21:56:02 GMT+0100 (hora estándar de Europa central)	6,4
9	Fri Dec 04 2020 22:06:02 GMT+0100 (hora estándar de Europa central)	6,4
10	Fri Dec 04 2020 22:26:02 GMT+0100 (hora estándar de Europa central)	5,4
11	Fri Dec 04 2020 22:46:02 GMT+0100 (hora estándar de Europa central)	6,4
12	Fri Dec 04 2020 22:56:02 GMT+0100 (hora estándar de Europa central)	6,6
13	Fri Dec 04 2020 23:06:03 GMT+0100 (hora estándar de Europa central)	6,6
14	Fri Dec 04 2020 23:16:02 GMT+0100 (hora estándar de Europa central)	6,4
15	Fri Dec 04 2020 23:26:18 GMT+0100 (hora estándar de Europa central)	6
16	Fri Dec 04 2020 23:36:02 GMT+0100 (hora estándar de Europa central)	6
17	Fri Dec 04 2020 23:46:02 GMT+0100 (hora estándar de Europa central)	6,1
18	Fri Dec 04 2020 23:56:02 GMT+0100 (hora estándar de Europa central)	6
19	Sat Dec 05 2020 00:06:02 GMT+0100 (hora estándar de Europa central)	6,1
20	Sat Dec 05 2020 00:16:02 GMT+0100 (hora estándar de Europa central)	6,1
21	Sat Dec 05 2020 00:26:02 GMT+0100 (hora estándar de Europa central)	5,9
22	Sat Dec 05 2020 00:36:02 GMT+0100 (hora estándar de Europa central)	5,9
23	Sat Dec 05 2020 00:46:02 GMT+0100 (hora estándar de Europa central)	5,9
24	Sat Dec 05 2020 00:56:04 GMT+0100 (hora estándar de Europa central)	4,5
25	Sat Dec 05 2020 01:06:03 GMT+0100 (hora estándar de Europa central)	4,5
26	Sat Dec 05 2020 01:16:03 GMT+0100 (hora estándar de Europa central)	4,5
27	Sat Dec 05 2020 01:26:03 GMT+0100 (hora estándar de Europa central)	4,5

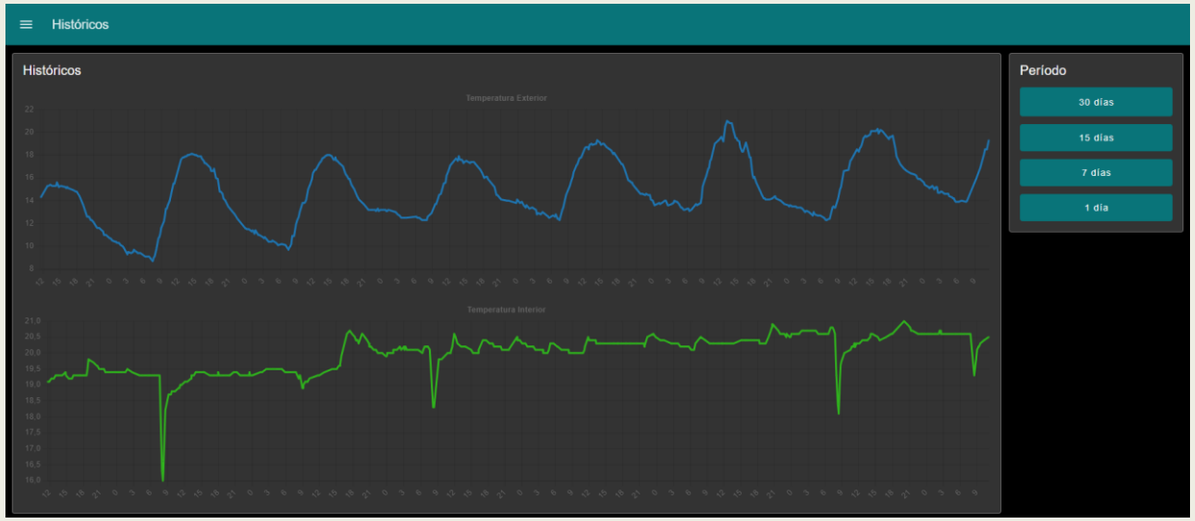


Gráfica con Node-RED

Gráfica generada con el nodo Chart del Dashboard 2.0 de Node-RED sacados directamente de la base de datos.

Los detalles están en el "**Capítulo8. Home Automation IoT**"



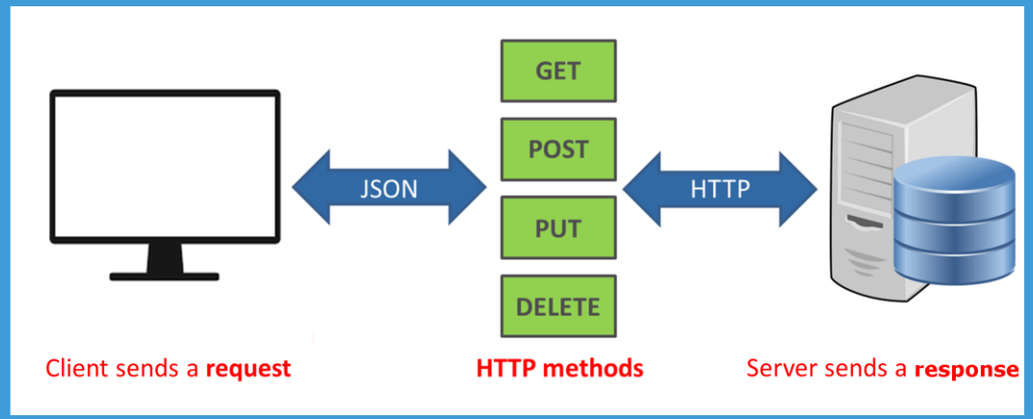


API REST { REST }

#2-API REST

API: Application Programming Interface

REST: REpresentational State Transfer



REST es cualquier interfaz entre sistemas que use HTTP para obtener datos o generar operaciones sobre esos datos en todos los formatos posibles, como XML y JSON. Un concepto importante en REST es la existencia de **recursos** (elementos de información).

Ejemplo TMB

Por ejemplo API REST pública de TMB (Transports Metropolitans de Barcelona)

Para la parada de bus 318 (Padilla-Gran Vía) de la línea 62.

https://api.tmb.cat/v1/ibus/lines/**62**/stops/**318**?app_id=119fcb80&app_key=195af9834ad91535a8c75d89bb1103d7

Nos devuelve un JSON con la información de tiempo restante para llegar el siguiente autobús.


```
{"status":"success","data":{"ibus":[{"routeId":"0620","text-ca":"10 min","t-in-s":642,"destination":"Ciutat Meridiana","t-in-min":10}]}}
```


Status: Success y t-in-min = 10 minutos y en segundos t-in-s = 642

Si lo miro en la propia web para comprobar los datos: https://www.tmb.cat/es/barcelona/tmb-ibus/proximos-buses/-/lineabus/parada-ibus/318

API REST { REST }

Ejemplo TMB

Para ver las paradas de la línea 62 de bus:

https://api.tmb.cat/v1/transit/linies/bus/**62**/parades?app_id=119fcb80&app_key=195af9834ad91535a8c75d89bb1103d7

Por cada parada de la línea me da la siguiente información. El recorte es de la parada 318 (Padilla-Gran Vía).

Se puede ver sus coordenadas geográficas, nombre parada, dirección, descripción línea y un largo etcétera.


```
"id": "PAR_REC.fid-26499ae6_1764e093daf_-7709",
"geometry": {
  "type": "Point",
  "coordinates": [
    2.182958,
    41.401357
  ]
},
"geometry_name": "GEOMETRY",
"properties": {
  "ID_RECORREGUT": 2751388,
  "CODI_PARADA": 318,
  "NOM_PARADA": "Padilla - Gran Via",
  "DESC_PARADA": "Padilla/Diputació",
  "CODI_INTERC": 0,
  "NOM_INTERC": null,
  "ADRECA": "Padilla, 162",
  "NOM_POBLACIO": "Barcelona",
  "NOM_DISTRICTE": "Eixample",
  "NOM_VIA": "Carrer de Padilla",
  "ORDRE": 8,
  "DISTANCIA_PAR_ANTERIOR": 253.49017,
  "ES_ORIGEN": 0,
  "ES_DESTI": 0,
  "ES_MAXINODE": 0,
  "ID_LINIA": 2862,
  "CODI_LINIA": 62,
  "NOM_LINIA": "62",
  "DESC_LINIA": "Pl. Catalunya / Ciutat Meridiana",
  "ORIGEN_SENTIT": "Pl. Catalunya",
  "DESTI_SENTIT": "Ciutat Meridiana",
  "ID_OPERADOR": 2,
  "NOM_OPERADOR": "TB",
  "CODI_FAMILIA": 1,
  "NOM_FAMILIA": "Convencionals",
  "ORDRE_FAMILIA": 13,
  "ORDRE_LINIA": 62,
  "ID_TIPUS_SERVEI": 1,
  "ID_TIPUS_SUB_SERVEI": 101,
  "NOM_TIPUS_SUB_SERVEI": "Sentit Comercial Anada",
  "ID_SENTIT": 1,
  "SENTIT": "A",
  "DESC_SENTIT": "Anada",
  "ID_TIPUS_RECORREGUT": 2,
  "ID_TIPUS_DIA": 1,
  "CALENDARI_EMULAT": 0,
  "DATA": "2020-12-06Z",
  "DATA_INICI": "2020-12-06Z",
  "DATA_FI": null,
  "COLOR_REC": "DC241F",
  "PUNTS_PARADA": 1
```


Ejemplo TMB

Con esta información podríamos pintar en un mapa las paradas de la línea.

Node-RED

mongoDB

#3-Ejemplo CRUD { REST }

Haremos un ejemplo WEB de lista, alta, baja y modificación típico. Esta es la correspondencia de los verbos con los métodos REST.

C	Create	POST
R	Read	GET
U	Update	PUT
D	Delete	DELETE

Es una adaptación de este ejemplo:

https://flows.nodered.org/flow/e32f2b942bce77ef6079c0642b93c036

Mongo DB

mongoDB

Los datos se persisten en una base de datos No SQL (Mongo DB). En una colección.

Lo primero es instalar Mongo DB en la Raspberry Pi, añadirla como servicio e iniciarla. En la Shell crear la bbdd "db".

```
pi@raspberrypi:~ $ sudo apt install mongodb
pi@raspberrypi:~ $ sudo systemctl enable mongodb
pi@raspberrypi:~ $ sudo systemctl start mongodb
pi@raspberrypi:~ $ mongo
```

```
> show databases
local   0.03125GB
> use db
switched to db db
```

```
pi@raspberrypi:/ $ mongod --version
db version v2.4.14
```

O bien utilizar Docker para generar un contenedor con la base de datos.

Diccionario NO SQL-SQL: **Colección** es como una Tabla, **Documento** es un objeto que podría ser una row de la tabla y **Elemento** es una propiedad del Documento, es decir, podría ser una column de la row.

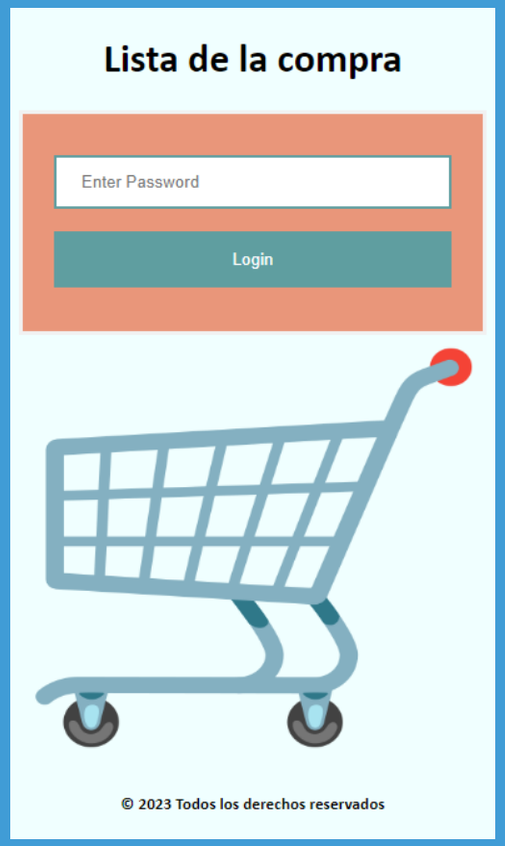
Node-RED

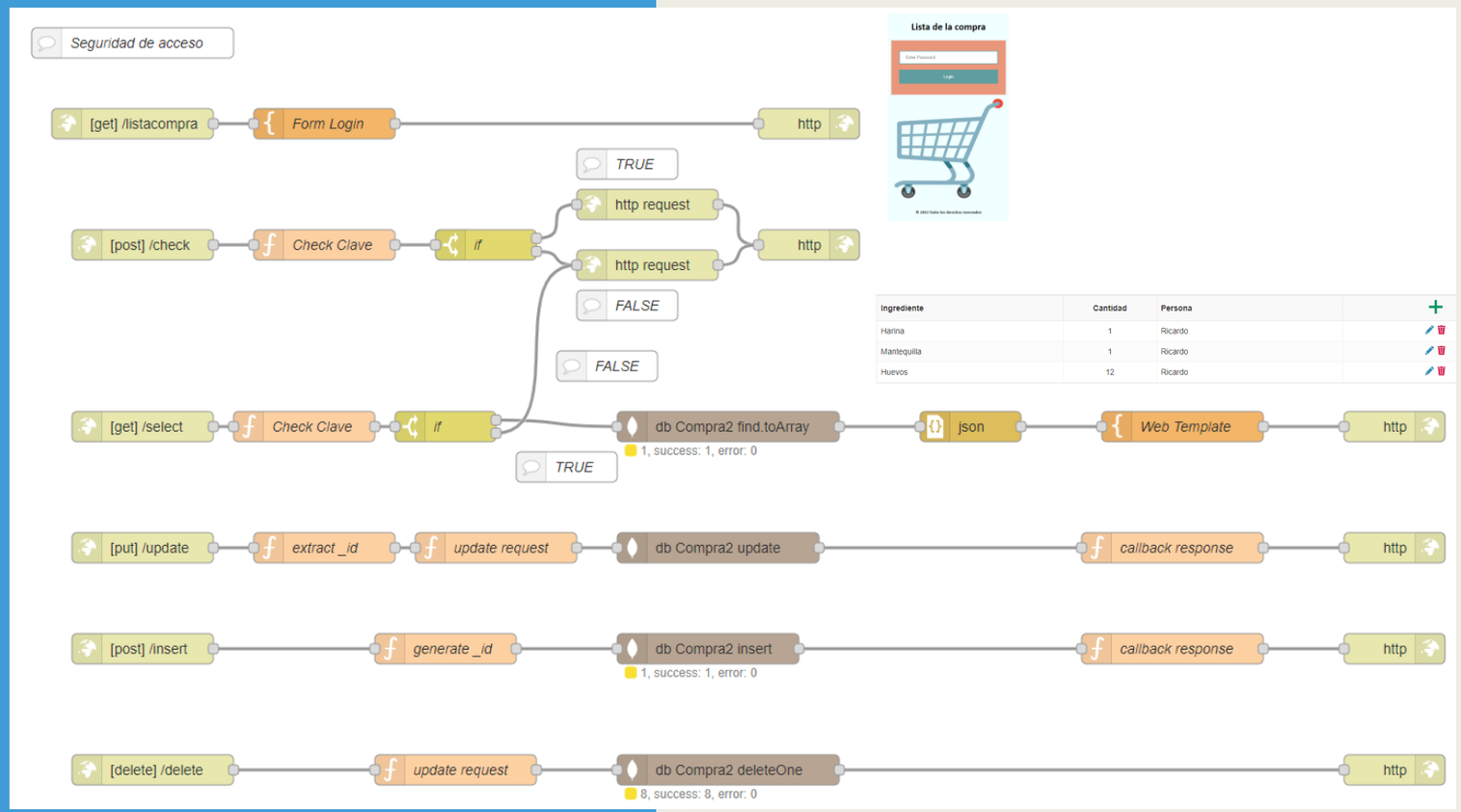
mongoDB

Flujo Node-RED

Mongo DB

mongoDB





Los nodos MongoDB son en versión “node-red-contrib-mongodb2”

La colección en MongoDB será “**Compra2**”

Node-RED

mongoDB

GET

Poner en el navegador web: http://192.168.1.129:1880/**listacompra**

[get] /listacompra

Ingrediente	Cantidad	Persona
Harina	1	Ricardo
Mantequilla	1	Ricardo
Huevos	12	Ricardo

Web Template

```
<!DOCTYPE html>
<html lang="es">
<head>
    <title>Lista de la compra</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/bootstrap-timepicker/0.5.2/css/bootstrap-timepicker.min.css" />
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
    <link type="text/css" rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/jsgrid/1.5.3/jsgrid.min.css" />
    <link type="text/css" rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/jsgrid/1.5.3/jsgrid-theme.min.css" />
    <script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/jsgrid/1.5.3/jsgrid.min.js"></script>
```

Web Template { REST }

```
<script type="text/javascript">
    $(function () {
        var db = {{#payload}}{{{.}}}{{/payload}};

            $("#jsgrid").jsGrid({
                width: "100%",
                inserting: true,
                editing: true,
                sorting: true,
                paging: true,

                data: db,

                fields: [
                    { title:"Ingrediente", name: "ing", type: "text", width: 50 },
                    { title:"Cantidad", align:"left",  name: "cant", type:"number" , width: 25},
                    { title:"Persona", name: "nom", type: "text", width: 50 },
                    { type: "control" }
                ],

                controller: {
                    insertItem: function(item) {
                        return $.ajax({
                            type: "POST",
                            url: "/insert",
                            data: item
                        });
                    },
                    updateItem: function(item) {
                      return $.ajax({
                            type: "PUT",
                            url: "/update",
                            data: item
                        });
                    },
                    deleteItem: function(item) {
                        return $.ajax({
                            type: "DELETE",
                            url: "/delete",
                            data: item
                        });
                    }
                }
            });
        });
</script>
```

Utilizaremos jsGrid

http://js-grid.com/demos/

jsGrid
Lightweight Grid jQuery Plugin

AJAX

```
<body class="container">
    <section class="row">

        <div class="col-md-6"></div>
        <div class="col-md-6" id="jsgrid">
        </div>
    </section>
</body>
</html>
```

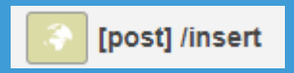
Node-RED

POST

Al hacer + salen las casillas para insertar el nuevo documento o registro.

Ingrediente	Cantidad	Persona	+
Sal	1	Ricardo	+
Harina	1	Ricardo	
Mantequilla	1	Ricardo	
Huevos	12	Ricardo	

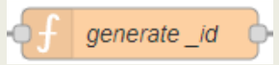
Una vez escrito pulsar + (se llama a la URL /insert)



Ingrediente	Cantidad	Persona	+
Harina	1	Ricardo	
Mantequilla	1	Ricardo	
Huevos	12	Ricardo	
Sal	1	Ricardo	

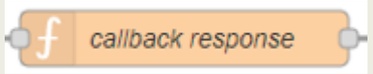
INSERT

Antes del nodo insertar de MongoDB genero el _id propio con el timestamp de inserción. Así es String y no ObjectID(_id) que me da problemas.




```
1  msg.payload=msg.req.body;
2  msg.payload._id = ""+new Date().getTime();//timestamp
3  return msg;
```

Una vez insertado en la BBDD con la operación "Insert" se le devuelve al usuario el listado de nuevo. Es necesario reiniciar la página.




```
1
2  msg.payload=msg.req.body;
3  return msg;
```

Para verlo desde la Shell de MongoDB → db.Compra2.find()

```
pi@raspberrypi:/ $ mongo
MongoDB shell version: 2.4.14
connecting to: test
Server has startup warnings:
Sat Dec 12 18:02:16.803 [initandlisten]
Sat Dec 12 18:02:16.803 [initandlisten] ** NOTE: This is a 32 bit MongoDB binary.
Sat Dec 12 18:02:16.803 [initandlisten] **       32 bit builds are limited to less than 2GB of data (or less with --journal).
Sat Dec 12 18:02:16.803 [initandlisten] **       See http://dochub.mongodb.org/core/32bit
Sat Dec 12 18:02:16.803 [initandlisten]
> use db
switched to db db
> db.Compra2.find()
{ "_id" : "1607892568228", "ing" : "Harina", "cant" : "1", "nom" : "Ricardo" }
{ "ing" : "Mantequilla", "cant" : "1", "nom" : "Ricardo", "_id" : "1607944391944" }
{ "ing" : "Huevos", "cant" : "12", "nom" : "Ricardo", "_id" : "1607944405117" }
{ "ing" : "Sal", "cant" : "1", "nom" : "Ricardo", "_id" : "1607945248887" }
>
```

Node-RED

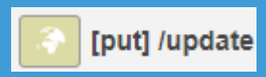
mongoDB

PUT

Al hacer click en el lápiz o bien click en el propio registro.

Ingrediente	Cantidad	Persona	+
Harina	1	Ricardo	
Mantequilla	1	Ricardo	
Huevos	12	Ricardo	
Sal	1	Ricardo	✓ ✗

Una vez escrito pulsar el check (se llama a la URL /update)



Ingrediente	Cantidad	Persona	+
Harina	1	Ricardo	
Mantequilla	1	Ricardo	
Huevos	12	Ricardo	
Sal	2	Ricardo	

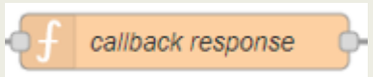
UPDATE

Antes del nodo actualizar de MongoDB extraigo el _id y lo paso como filter (query) y los datos (sin ._id).


```
1 msg._id=msg.payload._id;
2 return msg;
```


```
1 msg.result=msg.req.body;
2 delete msg.req.body._id;
3 msg.payload=[{"_id":msg._id},   msg.req.body];
4 return msg;
```

Una vez actualizado en la BBDD con la operación "update" se le devuelve al usuario el listado de nuevo actualizado.




```
1
2 msg.payload=msg.result;
3 return msg;
```

Para verlo desde la Shell de MongoDB → db.Compra2.find()

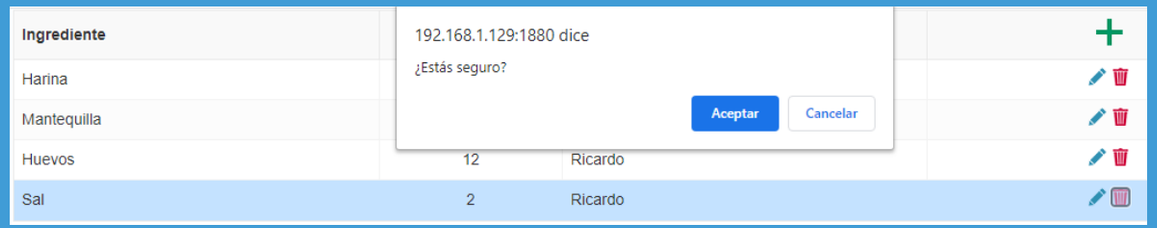
```
> db.Compra2.find()
{ "_id" : "1607892568228", "ing" : "Harina", "cant" : "1", "nom" : "Ricardo" }
{ "ing" : "Mantequilla", "cant" : "1", "nom" : "Ricardo", "_id" : "1607944391944" }
{ "ing" : "Huevos", "cant" : "12", "nom" : "Ricardo", "_id" : "1607944405117" }
{ "_id" : "1607945248887", "ing" : "Sal", "cant" : "2", "nom" : "Ricardo" }
>
```

Node-RED

mongoDB

DELETE

Al hacer click en la papelera. Pregunta si estás seguro.



Opciones del jsGrid:

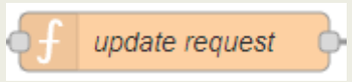
```
confirmDeleting: true,
deleteConfirm: "¿Estás seguro?",
```

Una vez pulsar Aceptar (se llama a la URL /delete)

Ingrediente	Cantidad	Persona
Harina	1	Ricardo
Mantequilla	1	Ricardo
Huevos	12	Ricardo

DELETE

Antes del nodo borrar de MongoDB, quito el ._id de los datos (msg.req.body)




```
1
2  delete msg.req.body._id;
3  msg.payload=[
4     msg.req.body
5     ];
6  return msg;
```

Una vez borrado de la BBDD con la operación "deleteOne" se le devuelve al usuario el listado de nuevo actualizado. Es necesario reiniciar la página.

Para verlo desde la Shell de MongoDB → db.Compra2.find()

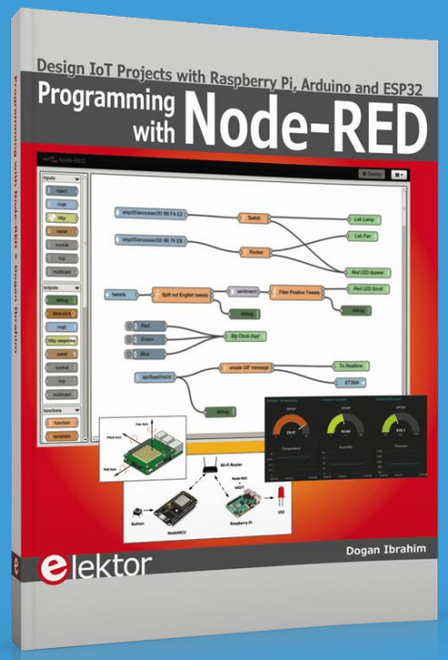
```
> db.Compra2.find()
{ "_id" : "1607892568228", "ing" : "Harina", "cant" : "1", "nom" : "Ricardo" }
{ "ing" : "Mantequilla", "cant" : "1", "nom" : "Ricardo", "_id" : "1607944391944" }
{ "ing" : "Huevos", "cant" : "12", "nom" : "Ricardo", "_id" : "1607944405117" }
>
```

Lecturas y recursos

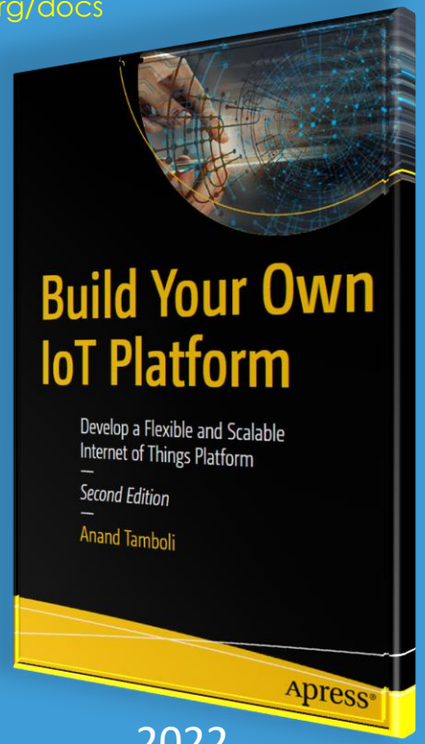
Lecturas recomendadas

Página de documentación de Node-RED:

https://nodered.org/docs



2020



2022

Recursos

- Cookbook

https://cookbook.nodered.org/

- New Dashboard 2.0 → 2023

https://dashboard.flowfuse.com/

- DevOps for Node-RED

https://flowfuse.com/

FlowFuse

Node-RED and more
v.1.4 SEPTIEMBRE 2024

https://www.linkedin.com/in/ricardo-moraleda-gareta-9421099

https://www.linkedin.com/company/gdo-electric1996/

RICARDO MORALEDA GARETA

GDO SOFTWARE