



ANEXO TFC
Guía técnica de implementación

Nombre del alumno: David Moreno Bolivar
Desarrollo de Aplicaciones Multiplataforma
Memoria del Proyecto de DAM
IES Abastos.
Curso 2024/25. Grupo 7U.
25 de mayo de 2025
Tutor individual: Asier Agra Monte

Indice

1. Servidor VPS.....	4
2. Instalacion de software.....	5
2.1 Ubuntu.....	5
2.2 Gestion de memoria (Swap).....	5
2.3 Personalizacion de terminal remota.....	6
2.4 Docker.....	8
2.5 Node.....	9
2.6 Node-RED.....	9
2.6.1 Securizando Node-RED.....	11
2.7 MySQL » MariaDB.....	12
2.8 Crear BD.....	13
2.9 Primera comunicación entre node-red y MariaDB.....	14
3. Creacion de la BD panpaceli.....	17
3.1 Modificaciones de la BD tras creacion.....	22
3.2 Creacion de proyecto panpaceli en flutter.....	22
4. Conexion segura para la API.....	23
5. Comandos utiles.....	25

1. Servidor VPS

El servidor utilizado para el desarrollo del proyecto es una máquina virtual (VPS) proporcionada por Ionos y ubicada en España. Este entorno presenta una configuración de hardware limitada, compuesta por una CPU de un vCore, 500 MB de memoria RAM y 10 GB de almacenamiento en disco NVMe SSD. El sistema operativo instalado es Ubuntu 24.04.1 LTS (noble), y sobre él se ejecuta un entorno de contenedores basado en Docker

Estas restricciones han condicionado ciertas decisiones durante la fase de diseño y desarrollo, siendo necesario realizar ajustes en la arquitectura inicial del proyecto para adaptarse a las capacidades del entorno.

Las especificaciones técnicas son las siguientes:

RED	
IP	82.223.50.169
SOFTWARE	
OS	Ubuntu 24.04.1 LTS (noble)
DOCKER	26.1.3, build 26.1.3-0ubuntu1~24.04.1
NODE (docker)	20.19.0
MARIADB (docker)	11.7.2-MariaDB-ubu2404
VISUAL STUDIO CODE	1.99.3
SCRCPY	1.21
VPS INFO.	
PROVEEDOR	Ionos
TIPO	Máquina virtual
CREACION	45397,85
UBICACION	España
VPS SPEC.	
CPU	1 vCore
RAM	500Mb
DISCO	10Gb NVMe SSD

2. Instalacion de software

2.1 Ubuntu

La instalación del sistema operativo Ubuntu 24.04 se realiza a partir de la imagen oficial que ofrece IONOS desde su panel de control. Este proceso es completamente automatizado y tarda entre 5 y 10 minutos. Una vez completada la instalación, el panel muestra la contraseña del usuario **root**, necesaria para acceder inicialmente al servidor.

Tras la instalación, el VPS permite el acceso remoto mediante conexión SSH utilizando el usuario **root** pero como práctica de seguridad, no daremos acceso ssh a este usuario, sino a otro autorizado.

Una vez establecida la primera conexión como root, se procede a la creación de un nuevo usuario con permisos administrativos para utilizarlo como acceso principal al servidor.

```
adduser [REDACTED]
```

Le daremos permisos de sudo

```
usermod -aG sudo [REDACTED]
```

También debemos actualizar el sistema

```
sudo apt update && sudo apt-get upgrade
```

2.2 Gestion de memoria (Swap)

Esta VPS cuenta con una cantidad muy limitada de memoria RAM lo que provoca que ciertos servicios, como MySQL, se cierren de forma inesperada al alcanzar el límite de memoria disponible.

Para mitigar este problema y asegurar una mayor estabilidad del sistema, se ha optado por la creación de una partición de intercambio (*swap*) en el disco. Esta técnica permite al sistema operativo utilizar una parte del almacenamiento como memoria virtual, lo cual proporciona un pequeño margen adicional que puede resultar decisivo en situaciones de alta carga de trabajo o consumo de recursos.

```
sudo fallocate -l 1G /swapfile
sudo chmod 600 /swapfile
sudo mkswap /swapfile
sudo swapon /swapfile
```

Para que siga funcionando tras un reinicio agregamos a `/etc/fstab` :

```
/swapfile none swap sw 0 0
```

2.3 Personalizacion de terminal remota

Con el objetivo de mejorar la experiencia de administración desde la terminal, se realizaron algunos ajustes en el archivo de configuración `.bashrc` del usuario administrador. Estos cambios incluyen la personalización del prompt, la ejecución automática de herramientas de monitorización y la definición de alias útiles.

El archivo `.bashrc` es un script de configuración que se ejecuta automáticamente cada vez que se inicia una nueva sesión interactiva de terminal con el shell Bash (Bourne Again SHell)

El siguiente fragmento fue añadido al final del archivo `.bashrc`

```
# Mensaje personalizado
neofetch
# Estado del sistema
bttop

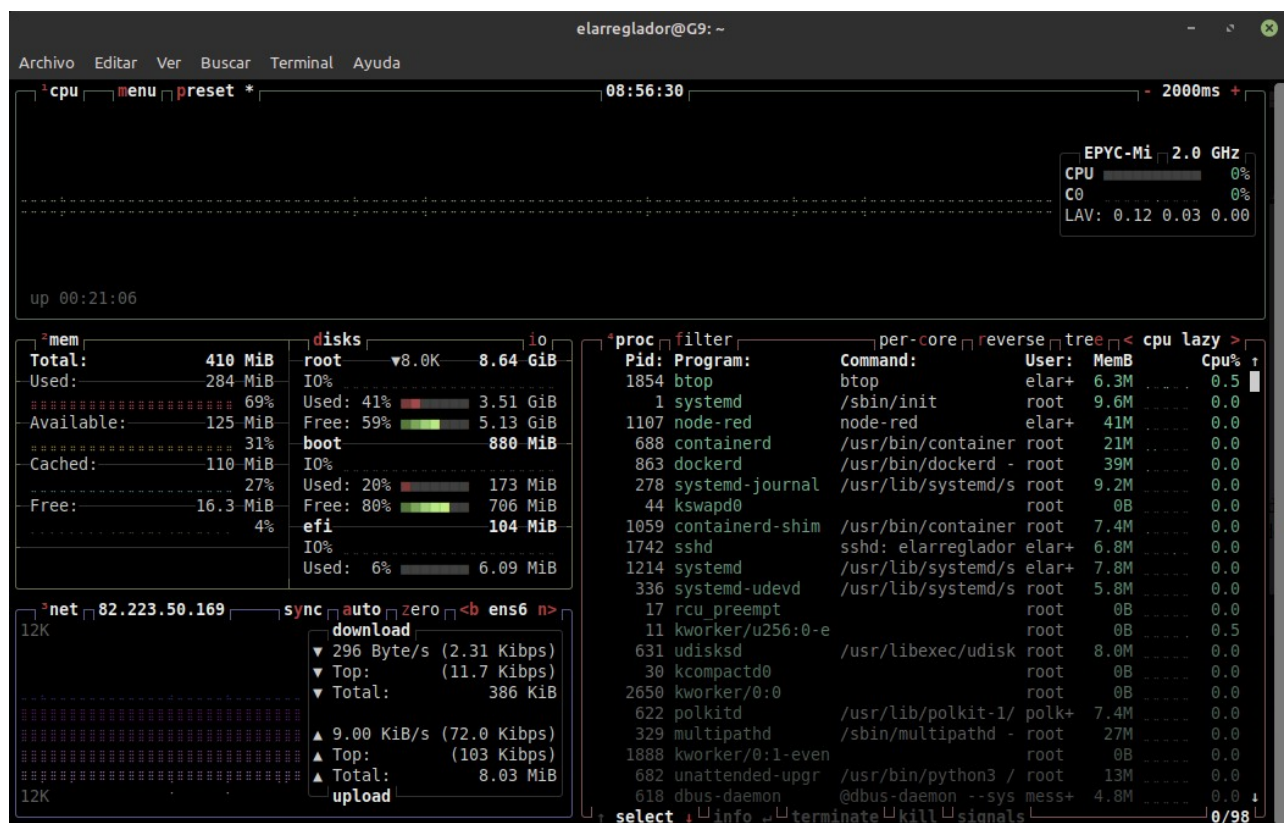
# Alias
alias top='bttop'

# Senyalizacion de terminal para clientes ssh
export PS1="\[\e[41m\]\u@\h:\w \${\[\e[0m\]} "
echo -ne "\033]0;\001 SSH REMOTO \001\007"
```

Por otro lado instalaremos `bttop`:

```
sudo apt-get install bttop
```

De este modo al entrar nos mostrara el estado actual del servidor



Gracias a esta configuración , cada vez que se accede al servidor por SSH, se muestra automáticamente información útil sobre el sistema y el estado actual de los recursos. Además, el entorno de terminal queda claramente diferenciado del local por dos razones:

- En la **parte superior de la ventana del terminal** aparece una advertencia visual que indica que se trata de una sesión remota: **SSH REMOTO**.
- El **prompt del sistema** se presenta con un **fondo rojo**, lo cual ayuda a distinguir rápidamente que se está trabajando en un entorno de servidor y no en el equipo personal.

Esta personalización no solo mejora la usabilidad, sino que también reduce el riesgo de errores al dejar clara la distinción entre entornos locales y remotos.

```
SSH REMOTO

just raised the bar for easy, resilient and secure K8s cluster deployment.
https://ubuntu.com/engage/secure-kubernetes-at-the-edge
Expanded Security Maintenance for Applications is not enabled.
10 updates can be applied immediately.
To see these additional updates run: apt list --upgradable
1 additional security update can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm

Last login: Fri Apr 18 08:34:42 2025 from 79.117.197.167
elarreplador@ubuntu:~$
.:+oossssoo+/-.:
.:+ssssssssssssssssss+:`
-+ssssssssssssssssssyyssss+-
.ossssssssssssssssssdMMMMNyssssso.
/ssssssssssshdmmNNmyMMMMhssssss/
+ssssssssshmydMMMMMMNdddyssssssss+
/ssssssssshMMMMyhhyyyhmMMMMhssssssss/
.ssssssssdMMMMNhsssssssshMMMMdssssssss.
+ssssshhhyNMMNyssssssssssyNMMMyssssssss+
ossyNMMNyMMhssssssssssshmmhssssssso
ossyNMMNyMMhssssssssssshmmhssssssso
+ssssshhhyNMMNyssssssssssyNMMMyssssssss+
.ssssssssdMMMMhssssssssshMMMMdssssssss.
/ssssssshNMMMyhhyyyhdMMMMhssssssss/
+sssssssssdmydMMMMMMNdddyssssssss+
/ssssssssshdmmNNmyMMMMhssssss/
.ossssssssssssssssssdMMMMNyssssso.
-+ssssssssssssssssssyyssss+-
.:+ssssssssssssssssss+:`
.:+oossssoo+/-.:

elarreplador@ubuntu:~$
```

UBICACION	España
VPS SPEC.	
CPU	1 vCore
RAM	500Mb
DISCO	10Gb NVMe S

```
OS: Ubuntu 24.04.2 LTS x86_64
Host: KVM/QEMU (Standard PC (i440FX + PIIX, 1996) pc-i440fx-6.1)
Kernel: 6.8.0-57-generic
Uptime: 1 min
Packages: 779 (dpkg)
Shell: bash 5.2.21
Terminal: /dev/pts/0
CPU: AMD EPYC-Milan (1) @ 1.996GHz
GPU: 00:02.0 Red Hat, Inc. QXL paravirtual graphic card
Memory: 252MiB / 410MiB
```

Instalacion de software

2.4 Docker

Para el despliegue y gestión de servicios en contenedores, se instala Docker junto con Docker Compose mediante el gestor de paquetes apt, a continuación, se habilita y se inicia el servicio de Docker para que esté disponible al arrancar el sistema.

Los comandos utilizados son los siguientes:

```
sudo apt update
sudo apt install docker.io docker-compose -y
sudo systemctl enable docker
sudo systemctl start docker
```

Tras la instalación, se configura el sistema para que el usuario actual pueda ejecutar comandos de Docker sin necesidad de anteponer sudo ,esto se consigue añadiendo al usuario al grupo docker:

```
sudo usermod -aG docker $USER
```

Nota: Para que los cambios en los grupos surtan efecto, es necesario cerrar la sesión del usuario y volver a iniciarla.

2.5 Node

Para el correcto funcionamiento de node-RED se requiere tener instalado Node.js. Para ello, se utiliza el gestor de paquetes de Ubuntu para instalar las dependencias necesarias, y posteriormente se añade el repositorio oficial de NodeSource para disponer de la versión más reciente compatible.

En primer lugar, se actualiza la lista de paquetes y se instalan herramientas básicas requeridas para la configuración del repositorio:

```
sudo apt update
sudo apt install -y curl gnupg
```

A continuación, se agrega el repositorio oficial de Node.js (versión 20) desde NodeSource:

```
curl -fsSL https://deb.nodesource.com/setup_20.x | sudo -E bash -
```

Una vez añadido el repositorio, se procede con la instalación de Node.js y npm (gestor de paquetes de Node):

```
sudo apt install -y nodejs
```

Por último, se instala una dependencia global que será necesaria más adelante en el desarrollo del proyecto, concretamente para funciones de cifrado:

```
sudo npm install -g bcryptjs
```

2.6 Node-RED

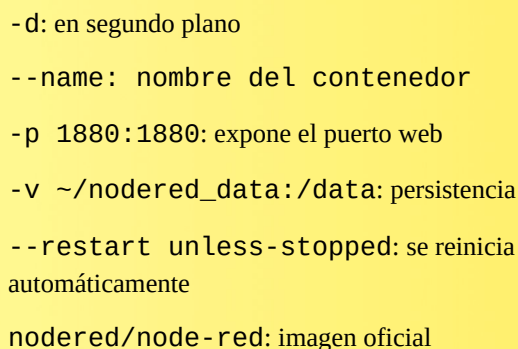
Node-RED correrá dentro de un contenedor Docker, aprovechando su facilidad de despliegue y aislamiento. Además, se configura un volumen persistente para que, en caso de reinicio del servicio o del propio sistema, no se pierdan los flujos ni la configuración de la herramienta.

Node-RED almacena por defecto sus flujos, configuraciones y credenciales en el directorio `/data` dentro del contenedor. Para garantizar la persistencia, se enlaza dicho directorio con una carpeta del sistema anfitrión (host). Esto se logra mediante la opción `-v` de Docker.

```
mkdir -p ~/nodered_data
```

Ahora lanzamos el contenedor:

```
docker run -d \
  --name nodered \
  -p 1880:1880 \
  -v ~/nodered_data:/data \
  --restart unless-stopped \
  nodered/node-red
```















- d: en segundo plano
- name: nombre del contenedor
- p 1880:1880: expone el puerto web
- v ~/nodered_data:/data: persistencia
- restart unless-stopped: se reinicia automáticamente
- nodered/node-red: imagen oficial


Una vez que Node-RED se encuentra en funcionamiento dentro del contenedor, es necesario permitir el acceso externo a su interfaz web, que por defecto opera en el **puerto 1880**.

Para ello, es imprescindible acceder al **panel de control de IONOS**

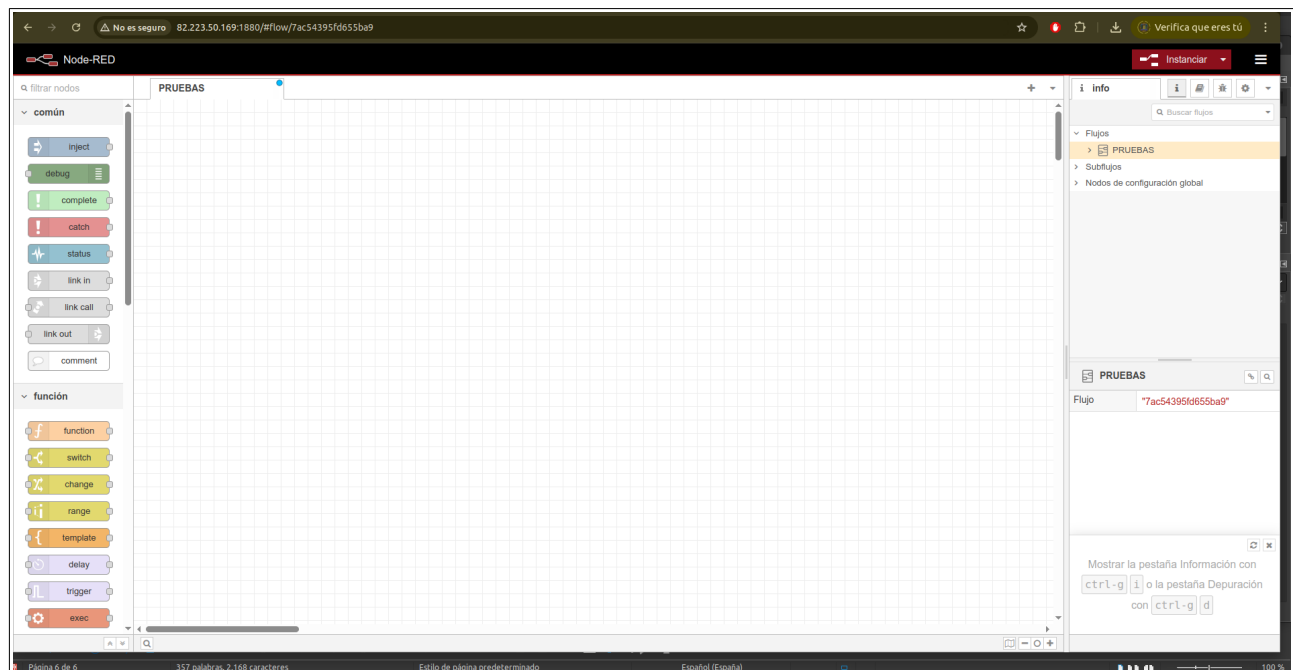
Configuración

Entrada

Acción	IP permitida	Protocolo	Puerto(s)	Descripción	
Permitir	Todas	TCP	22	ssh	 
Permitir	Todas	TCP	80	http	 
Permitir	Todas	TCP	443	https	 
Permitir	Todas	TCP	8118	privoxy	 
Permitir	Todas	TCP	1880	node-red	 
Permitir	<input type="text" value="todas"/>	<input type="text" value="TCP"/>	<input type="text"/>	<input type="text"/>	 

 Insertar configuraciones predefinidas

Una vez añadida la regla en el firewall, podemos abrir el navegador y verificar que la aplicación funciona correctamente:



Cuando reiniciemos la maquina virtual no sera necesario lanzar el contenedor de node-red ya que hemos indicado la opcion -d

2.6.1 Securizando Node-RED

Actualmente, el acceso a Node-RED no requiere ningún tipo de autenticación, lo que representa un riesgo de seguridad al dejar la interfaz expuesta sin control. Por tanto, es necesario implementar una capa de protección para restringir el acceso a usuarios autorizados.

Node-RED no almacena las contraseñas en texto plano, sino que utiliza un hash seguro para protegerlas. Para generar el hash de la contraseña que usaremos en la configuración de acceso, ejecutaremos un comando en el servidor, **fuera del contenedor** de Node-RED.

Este comando permitirá obtener el hash cifrado necesario para definir la contraseña de acceso en el archivo de configuración de Node-RED, asegurando que la contraseña real nunca se almacene directamente.

```
mkdir -p ~/bcrypt-hasher && cd ~/bcrypt-hasher
npm init -y
npm install bcryptjs
node -e "console.log(require('bcryptjs').hashSync('██████████', 8))"
```

Esto nos retornara un hash, accederemos al interior del docker de node-red:

```
docker exec -it nodered bash
```

Buscamos el archivo de configuracion y lo editamos con nano:

```
find /data -name settings.js
nano /data/settings.js
```

Dejamos la configuracion de este modo:

```
/** To password protect the Node-RED editor and admin API, the following
 * property can be used. See https://nodered.org/docs/security.html for details.
 */

adminAuth: {
  type: "credentials",
  users: [{
    username: " ",
    password: " ",
    permissions: "*"
  }]
},
```

Salimos del contenedor y lo reiniciamos:

```
exit
docker restart nodered
```

A partir de este momento node-red nos consultara usuario y clave.



2.7 MySQL » MariaDB

Durante la configuración inicial del servidor, se detectó que la memoria RAM disponible es insuficiente para ejecutar MySQL de manera estable, como alternativa más ligera y compatible, se optó por instalar **MariaDB**, un sistema de gestión de bases de datos relacional, aunque es un fork de MySQL tiene un menor consumo de recursos y ofrece un rendimiento adecuado para entornos con limitaciones de hardware.

Esta elección garantiza la continuidad del proyecto sin comprometer la estabilidad del servidor, manteniendo compatibilidad con las herramientas y consultas desarrolladas para MySQL

Para garantizar la persistencia de la base de datos y facilitar la gestión de MariaDB, se crea primero una carpeta local en el servidor que servirá como volumen para almacenar los datos de forma permanente:

```
mkdir mariadb_data
```

A continuación, se lanza un contenedor Docker de MariaDB utilizando esta carpeta para la persistencia, configurando las credenciales necesarias y exponiendo el puerto 3306 para conexiones externas:

```
docker run \
  --name mariadb \
  -e MYSQL_ROOT_PASSWORD=[REDACTED] \
  -e MYSQL_USER=[REDACTED] \
  -e MYSQL_PASSWORD=[REDACTED] \
  -v $HOME/mariadb_data:/var/lib/mysql \
  -p 3306:3306 \
  -d mariadb:latest
```

Para verificar que el contenedor se ha iniciado correctamente y está en ejecución, se utiliza el siguiente comando:

```
docker ps
```

Confirmamos que aunque escaso, queda espacio en disco y memoria:

```
$ df -h
```

Filesystem	Size	Used	Avail	Use%	Mounted on
tmpfs	42M	1.2M	40M	3%	/run
/dev/vda1	8.7G	5.4G	3.3G	62%	/
tmpfs	206M	0	206M	0%	/dev/shm
tmpfs	5.0M	0	5.0M	0%	/run/lock
/dev/vda16	881M	112M	707M	14%	/boot
/dev/vda15	105M	6.1M	99M	6%	/boot/efi
tmpfs	42M	12K	42M	1%	/run/user/1000

```
$ free -h
```

	total	used	free	shared	buff/cache	available
Mem:	410Mi	255Mi	15Mi	56Ki	159Mi	154Mi
Swap:	1.0Gi	258Mi	765Mi			

2.8 Crear BD

Para administrar la base de datos desde el servidor, se instala el cliente de MariaDB/MySQL, que permite conectarse al contenedor de MariaDB desde fuera del entorno Docker.

Los comandos utilizados para instalar el cliente son:

```
sudo apt-get update
sudo apt install mysql-client-core-8.0
```

Una vez instalado, se establece la conexión al servidor MariaDB ejecutándose en el contenedor, usando la dirección local 127.0.0.1, el puerto 3306 y el usuario root:

```
mysql -h 127.0.0.1 -P 3306 -u root -p
```

Al acceder, se configuran los permisos para el usuario creado previamente, otorgándole todos los privilegios necesarios para operar desde cualquier host:

```
GRANT ALL PRIVILEGES ON *.* TO '██████████'@'%' WITH GRANT OPTION;
FLUSH PRIVILEGES;
exit
```

Después de configurar los permisos, se accede a MariaDB utilizando el usuario creado para realizar operaciones básicas, como la creación de una base de datos de prueba y una tabla con datos de ejemplo.

La conexión se realiza con el siguiente comando, proporcionando la contraseña cuando se solicite:

```
mysql -h 127.0.0.1 -P 3306 -u ██████████ -p
```

Dentro del cliente MariaDB, se crean la base de datos y la tabla de prueba:

```
CREATE DATABASE testdb;

USE testdb;

CREATE TABLE usuarios (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(100),
    correo VARCHAR(100)
);

INSERT INTO usuarios (nombre, correo) VALUES
('Juan Perez', 'juan@example.com'),
('Maria Lopez', 'maria@example.com');
```

Para comprobar que los datos se han insertado correctamente en la tabla `usuarios`, se ejecuta la siguiente consulta:

```
SELECT * FROM usuarios;
```

2.9 Primera comunicación entre node-red y MariaDB

Para permitir que Node-RED se comuniquen con bases de datos SQL, se instala el plugin oficial `node-red-node-mysql` que permite la ejecución de consultas SQL desde los flujos de Node-RED.

La instalación se realiza desde la interfaz web de Node-RED siguiendo estos pasos:

1. Acceder al menú principal (icono de hamburguesa).
2. Seleccionar **Administrar paleta**.
3. Ir a la pestaña **Instalar**.
4. Buscar `node-red-node-mysql`.
5. Pulsar **Instalar** y confirmar la instalación.
6. Cerrar la ventana al finalizar.

Una vez instalado, se importa el siguiente flujo de ejemplo, que realiza una consulta para obtener todos los registros de la tabla `usuarios` en la base de datos `testdb` de MariaDB:

```
[
  {
    "id": "b836cafbafeb1a8a7",
    "type": "mysql",
    "z": "7ac54395fd655ba9",
    "mydb": "f7026e5a57dbf0d1",
    "name": "MariaDB",
    "x": 640,
    "y": 240,
    "wires": [
      [
        "12df76e1ef734e04"
      ]
    ]
  },
  {
    "id": "12df76e1ef734e04",
    "type": "debug",
    "z": "7ac54395fd655ba9",
    "name": "Debug Result",
    "active": true,
    "tosidebar": true,
    "console": false,
    "tostatus": false,
    "complete": "true",
    "targetType": "full",
    "statusVal": "",
    "statusType": "auto",
    "x": 900,
    "y": 240,
    "wires": []
  },
  {
    "id": "9f3bb56c6a366fbb",
    "type": "function",
    "z": "7ac54395fd655ba9",
    "name": "function 1",
    "func": "msg.topic = \"SELECT * FROM usuarios;\";\nreturn msg;",
    "outputs": 1,
    "timeout": 0,
    "noerr": 0,
    "initialize": "",
    "finalize": "",
    "libs": [],
    "x": 420,
    "y": 240,
    "wires": [
      [
        "b836cafbafeb1a8a7"
      ]
    ]
  },
  {
    "id": "4b5a79a8c4cef235",
    "type": "inject",
    "z": "7ac54395fd655ba9",
    "name": "",
    "props": [
      {
        "p": "payload"
      },
      {
        "p": "topic",

```

```

        "vt": "str"
      }
    ],
    "repeat": "",
    "crontab": "",
    "once": false,
    "onceDelay": 0.1,
    "topic": "",
    "payload": "",
    "payloadType": "date",
    "x": 130,
    "y": 240,
    "wires": [
      [
        "9f3bb56c6a366fbb"
      ]
    ]
  },
  {
    "id": "f7026e5a57dbf0d1",
    "type": "MySQLdatabase",
    "name": "MariaDB",
    "host": "82.223.50.169",
    "port": "3306",
    "db": "testdb",
    "tz": "GMT",
    "charset": "UTF8MB4"
  }
]

```

Deberíamos poder ver el resultado en la columna de depuración del lateral derecho de node-db

The screenshot shows the Node-RED web interface. The workflow in the center consists of four nodes: 'marca tiempo' (inject), 'function 1' (function), 'MariaDB' (MySQL database), and 'Debug Result' (debug console). The 'MariaDB' node is currently showing a green 'OK' status. On the right side, the 'depuración' (debug) panel is open, displaying the following JSON output:

```

19/4/2025, 9:35:15  nodo: Debug Result
SELECT * FROM usuarios; : msg : Object
  ▼ object
    _msgid: "f0603a36206e89ac"
    ▼ payload: array[2]
      ▼ 0: object
        id: 1
        nombre: "Juan Perez"
        correo: "juan@example.com"
      ▼ 1: object
        id: 2
        nombre: "Maria Lopez"
        correo: "maria@example.com"
    topic: "SELECT * FROM usuarios;"

```


3. Creacion de la BD panpaceli

Accedemos a MariaDB y creamos la nueva BD

```
mysql -h 127.0.0.1 -P 3306 -u [REDACTED] -p
CREATE DATABASE panpaceli;
USE panpaceli;
```

Ahora crearemos las tablas necesarias para nuestra app

```
CREATE TABLE usuarios (
  id INT PRIMARY KEY AUTO_INCREMENT,
  mail VARCHAR(100) NOT NULL,
  hash TEXT NOT NULL,
  salt TEXT NOT NULL,
  vendedor BOOLEAN NOT NULL,
  alta TIMESTAMP NOT NULL,
  last_login TIMESTAMP,
  verificado_mail BOOLEAN NOT NULL,
  expulsado BOOLEAN NOT NULL,
  suspendido BOOLEAN NOT NULL,
  problema_impago BOOLEAN NOT NULL
);
```

```
CREATE TABLE vendedores (
  id INT PRIMARY KEY,
  nombre_empresa VARCHAR(100) NOT NULL,
  nombre_comercial VARCHAR(100) NOT NULL,
  persona_contacto VARCHAR(100) NOT NULL,
  url_logotipo VARCHAR(200),
  cif VARCHAR(15) NOT NULL,
  telefono VARCHAR(15),
  mail VARCHAR(35) NOT NULL,
  dir_calle VARCHAR(50) NOT NULL,
  dir_num VARCHAR(25) NOT NULL,
  dir_cp VARCHAR(10) NOT NULL,
  dir_provincia VARCHAR(50) NOT NULL,
  dir_poblacion VARCHAR(50) NOT NULL,
  min_allowed_rate TINYINT NOT NULL,
  latitud DECIMAL(9,6),
  longitud DECIMAL(9,6),
  notas TEXT,
  CONSTRAINT fk_vendedor_usuario FOREIGN KEY (id) REFERENCES usuarios(id) ON
DELETE CASCADE
);
```

```
CREATE TABLE clientes (
  id INT PRIMARY KEY,
  nombre VARCHAR(100) NOT NULL,
  dni VARCHAR(15),
  telefono VARCHAR(15),
  mail VARCHAR(35) NOT NULL,
  dir_calle VARCHAR(50),
  dir_num VARCHAR(25),
  dir_cp VARCHAR(10),
  dir_provincia VARCHAR(50),
  dir_poblacion VARCHAR(50),
```

```
        CONSTRAINT fk_cliente_usuario FOREIGN KEY (id) REFERENCES usuarios(id) ON
DELETE CASCADE
);
```

```
CREATE TABLE productos (
    id INT PRIMARY KEY AUTO_INCREMENT,
    id_vendedor INT NOT NULL,
    nombre VARCHAR(30) NOT NULL,
    activo BOOL NOT NULL,
    precio DECIMAL(10,2) NOT NULL,
    url_imagen VARCHAR(200),
    descripcion VARCHAR(1000),
    ingredientes VARCHAR(300),
    peso_gramos INT,
    cantidad VARCHAR(15),
    plazo_entrega VARCHAR(50),
    CONSTRAINT fk_producto_vendedor FOREIGN KEY (id_vendedor) REFERENCES
vendedores(id) ON DELETE CASCADE
);
```

```
CREATE TABLE pedidos (
    id INT PRIMARY KEY AUTO_INCREMENT,
    time_stamp DATETIME NOT NULL,
    id_vendedor INT NOT NULL,
    id_cliente INT NOT NULL,
    id_producto INT NOT NULL,
    unidades INT NOT NULL,
    estado_encargado BOOL,
    estado_elaboracion BOOL,
    estado_pdte_entrega BOOL,
    estado_entregado BOOL,
    cancelado_cliente BOOL,
    cancelado_vendedor BOOL,
    vendedor_valorado BOOL,
    cliente_valorado BOOL,
    CONSTRAINT fk_pedidos_vendedor FOREIGN KEY (id_vendedor) REFERENCES
vendedores(id) ON DELETE CASCADE,
    CONSTRAINT fk_pedidos_cliente FOREIGN KEY (id_cliente) REFERENCES
clientes(id) ON DELETE CASCADE,
    CONSTRAINT fk_pedidos_producto FOREIGN KEY (id_producto) REFERENCES
productos(id) ON DELETE CASCADE
);
```

```
CREATE TABLE rate_vendedores (
    id INT PRIMARY KEY AUTO_INCREMENT,
    time_stamp DATETIME NOT NULL,
    id_pedido INT NOT NULL,
    valoracion TINYINT NOT NULL,
    id_vendedor INT NOT NULL,
    id_cliente INT NOT NULL,
    comentario VARCHAR(500),
    CONSTRAINT fk_rate_pedido FOREIGN KEY (id_pedido) REFERENCES pedidos(id) ON
DELETE CASCADE,
    CONSTRAINT fk_rate_vendedor FOREIGN KEY (id_vendedor) REFERENCES
vendedores(id) ON DELETE CASCADE,
    CONSTRAINT fk_rate_cliente FOREIGN KEY (id_cliente) REFERENCES clientes(id)
ON DELETE CASCADE
);
```

```
CREATE TABLE rate_clientes (
    id INT PRIMARY KEY AUTO_INCREMENT,
    time_stamp DATETIME NOT NULL,
    id_pedido INT NOT NULL,
    valoracion TINYINT NOT NULL,
```

```

        id_vendedor INT NOT NULL,
        id_cliente INT NOT NULL,
        comentario VARCHAR(500),
        CONSTRAINT fk_ratecli_pedido FOREIGN KEY (id_pedido) REFERENCES pedidos(id)
ON DELETE CASCADE,
        CONSTRAINT fk_ratecli_vendedor FOREIGN KEY (id_vendedor) REFERENCES
vendedores(id) ON DELETE CASCADE,
        CONSTRAINT fk_ratecli_cliente FOREIGN KEY (id_cliente) REFERENCES
clientes(id) ON DELETE CASCADE
);

```

```

CREATE TABLE a_gluten (
    id INT PRIMARY KEY AUTO_INCREMENT,
    id_producto INT NOT NULL,
    libre BOOL NOT NULL,
    trazas BOOL NOT NULL,
    contiene BOOL NOT NULL,
    CONSTRAINT fk_gluten_producto FOREIGN KEY (id_producto) REFERENCES
productos(id) ON DELETE CASCADE
);

```

```

CREATE TABLE a_crustaceo (
    id INT PRIMARY KEY AUTO_INCREMENT,
    id_producto INT NOT NULL,
    libre BOOL NOT NULL,
    trazas BOOL NOT NULL,
    contiene BOOL NOT NULL,
    CONSTRAINT fk_crustaceo_producto FOREIGN KEY (id_producto) REFERENCES
productos(id) ON DELETE CASCADE
);

```

```

CREATE TABLE a_huevo (
    id INT PRIMARY KEY AUTO_INCREMENT,
    id_producto INT NOT NULL,
    libre BOOL NOT NULL,
    trazas BOOL NOT NULL,
    contiene BOOL NOT NULL,
    CONSTRAINT fk_huevo_producto FOREIGN KEY (id_producto) REFERENCES
productos(id) ON DELETE CASCADE
);

```

```

CREATE TABLE a_pescado (
    id INT PRIMARY KEY AUTO_INCREMENT,
    id_producto INT NOT NULL,
    libre BOOL NOT NULL,
    trazas BOOL NOT NULL,
    contiene BOOL NOT NULL,
    CONSTRAINT fk_pescado_producto FOREIGN KEY (id_producto) REFERENCES
productos(id) ON DELETE CASCADE
);

```

```

CREATE TABLE a_cacahuete (
    id INT PRIMARY KEY AUTO_INCREMENT,
    id_producto INT NOT NULL,
    libre BOOL NOT NULL,
    trazas BOOL NOT NULL,
    contiene BOOL NOT NULL,
    CONSTRAINT fk_cacahuete_producto FOREIGN KEY (id_producto) REFERENCES
productos(id) ON DELETE CASCADE
);

```

```

CREATE TABLE a_soja (
    id INT PRIMARY KEY AUTO_INCREMENT,
    id_producto INT NOT NULL,

```

```

        libre BOOL NOT NULL,
        trazas BOOL NOT NULL,
        contiene BOOL NOT NULL,
        CONSTRAINT fk_soja_producto FOREIGN KEY (id_producto) REFERENCES
productos(id) ON DELETE CASCADE
);

```

```

CREATE TABLE a_leche (
    id INT PRIMARY KEY AUTO_INCREMENT,
    id_producto INT NOT NULL,
    libre BOOL NOT NULL,
    trazas BOOL NOT NULL,
    contiene BOOL NOT NULL,
    CONSTRAINT fk_leche_producto FOREIGN KEY (id_producto) REFERENCES
productos(id) ON DELETE CASCADE
);

```

```

CREATE TABLE a_frutos_cascara (
    id INT PRIMARY KEY AUTO_INCREMENT,
    id_producto INT NOT NULL,
    libre BOOL NOT NULL,
    trazas BOOL NOT NULL,
    contiene BOOL NOT NULL,
    CONSTRAINT fk_frutos_cascara_producto FOREIGN KEY (id_producto) REFERENCES
productos(id) ON DELETE CASCADE
);

```

```

CREATE TABLE a_apio (
    id INT PRIMARY KEY AUTO_INCREMENT,
    id_producto INT NOT NULL,
    libre BOOL NOT NULL,
    trazas BOOL NOT NULL,
    contiene BOOL NOT NULL,
    CONSTRAINT fk_apio_producto FOREIGN KEY (id_producto) REFERENCES
productos(id) ON DELETE CASCADE
);

```

```

CREATE TABLE a_mostaza (
    id INT PRIMARY KEY AUTO_INCREMENT,
    id_producto INT NOT NULL,
    libre BOOL NOT NULL,
    trazas BOOL NOT NULL,
    contiene BOOL NOT NULL,
    CONSTRAINT fk_mostaza_producto FOREIGN KEY (id_producto) REFERENCES
productos(id) ON DELETE CASCADE
);

```

```

CREATE TABLE a_sesamo (
    id INT PRIMARY KEY AUTO_INCREMENT,
    id_producto INT NOT NULL,
    libre BOOL NOT NULL,
    trazas BOOL NOT NULL,
    contiene BOOL NOT NULL,
    CONSTRAINT fk_sesamo_producto FOREIGN KEY (id_producto) REFERENCES
productos(id) ON DELETE CASCADE
);

```

```

CREATE TABLE a_so2 (
    id INT PRIMARY KEY AUTO_INCREMENT,
    id_producto INT NOT NULL,
    libre BOOL NOT NULL,
    trazas BOOL NOT NULL,
    contiene BOOL NOT NULL,

```

```

        CONSTRAINT fk_so2_producto FOREIGN KEY (id_producto) REFERENCES
productos(id) ON DELETE CASCADE
);

CREATE TABLE a_altramuz (
    id INT PRIMARY KEY AUTO_INCREMENT,
    id_producto INT NOT NULL,
    libre BOOL NOT NULL,
    trazas BOOL NOT NULL,
    contiene BOOL NOT NULL,
    CONSTRAINT fk_altramuz_producto FOREIGN KEY (id_producto) REFERENCES
productos(id) ON DELETE CASCADE
);

CREATE TABLE a_molusco (
    id INT PRIMARY KEY AUTO_INCREMENT,
    id_producto INT NOT NULL,
    libre BOOL NOT NULL,
    trazas BOOL NOT NULL,
    contiene BOOL NOT NULL,
    CONSTRAINT fk_molusco_producto FOREIGN KEY (id_producto) REFERENCES
productos(id) ON DELETE CASCADE
);

```

Para confirmar que las tablas se han creado correctamente dentro de la base de datos testdb, se ejecuta la siguiente consulta desde el cliente MariaDB:

```

mysql> SHOW TABLES;
+-----+
| Tables_in_panpaceli |
+-----+
| a_altramuz          |
| a_apio              |
| a_cacahuete         |
| a_crustaceo         |
| a_frutos_cascara    |
| a_gluten            |
| a_huevo             |
| a_leche             |
| a_molusco           |
| a_mostaza           |
| a_pescado           |
| a_sesamo            |
| a_so2               |
| a_soja              |
| clientes            |
| pedidos            |
| productos           |
| rate_clientes       |
| rate_vendedores     |
| usuarios            |
| vendedores          |
+-----+
21 rows in set (0.00 sec)

```

Para realizar una copia de seguridad (backup) de la base de datos panpaceli desde la terminal del servidor, se utiliza mysqldump:

```
mysqldump -h 127.0.0.1 -u [REDACTED] -p --skip-column-statistics panpaceli > backup_panpaceli.sql
```

3.1 Modificaciones de la BD tras creacion

Durante la creacion del usuario solo disponemos de su mail, salt y hash, no sabemos si es un vendedor o un cliente, por lo que necesitamos que acepte nulos

```
ALTER TABLE usuarios MODIFY vendedor TINYINT(1) NULL;
```

No deberiamos admitir que se cree un segundo usuario con un correo ya existente

```
ALTER TABLE usuarios ADD UNIQUE (mail);
```

3.2 Creacion de proyecto panpaceli en flutter

Como primer paso, se verifica que Flutter está correctamente instalado y disponible en el sistema ejecutando el comando:

```
flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[✓] Flutter (Channel stable, 3.29.3, on Linux Mint 21.2 6.8.0-57-generic, locale es_ES.UTF-8)
[✓] Android toolchain - develop for Android devices (Android SDK version 35.0.0)
[✓] Chrome - develop for the web
[✓] Linux toolchain - develop for Linux desktop
[✓] Android Studio (version 2023.3)
[✓] Android Studio (version 2024.1)
[✓] Android Studio (version 2024.2)
[✓] IntelliJ IDEA Community Edition (version 2023.3)
[✓] VS Code (version 1.99.3)
[✓] Connected device (2 available)
[✓] Network resources

• No issues found!
```

A continuacion generamos el proyecto panpaceli:

```
flutter create panpaceli
Creating project panpaceli...
Resolving dependencies in `panpaceli`... (1.5s)
Downloading packages...
Got dependencies in `panpaceli`.
Wrote 130 files.
```

All done!

You can find general documentation for Flutter at: <https://docs.flutter.dev/>
Detailed API documentation is available at: <https://api.flutter.dev/>
If you prefer video documentation, consider:
<https://www.youtube.com/c/flutterdev>

In order to run your application, type:

```
$ cd panpaceli
$ flutter run
```

Your application code is in panpaceli/lib/main.dart.

Accedemos al directorio y ejecutamos visual studio code para iniciar la creacion de la aplicación

```
cd panpaceli/
code .
```

4. Conexion segura para la API

Para garantizar la seguridad y confidencialidad en la comunicación entre el cliente y el servidor, es fundamental cifrar el tráfico utilizando HTTPS. Esto se consigue mediante la instalación y configuración de un certificado SSL/TLS.

En entornos de desarrollo o para pruebas internas, es común utilizar un certificado autofirmado, que permite habilitar HTTPS sin necesidad de adquirir un certificado emitido por una autoridad certificadora externa. Aunque un certificado autofirmado no es reconocido como totalmente fiable por los navegadores, es suficiente para cifrar el tráfico y prevenir la interceptación de datos en tránsito.

Para implementar esta solución, se genera un certificado autofirmado en el servidor y se configura el servidor web o el proxy inverso para que utilice este certificado en las conexiones HTTPS.

Se genera un certificado autofirmado utilizando la herramienta OpenSSL con el siguiente comando:

```
openssl req -x509 -nodes -days 365 \  
-newkey rsa:2048 \  
-keyout key.pem \  
-out cert.pem \  
-subj "/C=ES/ST=Valencia/L=Valencia/O=Panpaceli/OU=Dev/CN=82.223.50.169"
```

Parte	Explicación
openssl req	Inicia una solicitud de certificado.
-x509	Genera un certificado autofirmado (en vez de una CSR que enviarías a una autoridad certificadora).
-nodes	No cifra la clave privada, para que no te pida contraseña al usarla (útil para servidores).
-days 365	El certificado será válido durante 365 días .
-newkey rsa:2048	Crea una nueva clave privada con algoritmo RSA y longitud de 2048 bits .
-keyout key.pem	Guarda la clave privada en el archivo <code>key.pem</code> .
-out cert.pem	Guarda el certificado en el archivo <code>cert.pem</code> .
-subj "/..."	Define directamente los datos del certificado para no tener que responder al asistente interactivo.

-subj desglosado:

```
java  
CopiarEditar  
/C=ES      → País: España  
/ST=Valencia → Estado/Provincia  
/L=Valencia → Localidad  
/O=Panpaceli → Organización  
/OU=Dev     → Unidad organizativa  
/CN=localhost → Common Name → dominio o IP
```

Para habilitar HTTPS en Node-RED, es necesario modificar su archivo de configuración `settings.js` ubicado en el directorio de datos persistentes (`~/nodered_data/settings.js`). Se deben descomentar y adaptar las siguientes líneas para que Node-RED utilice el certificado y la clave privada generados previamente:

```
https: {  
  key: require("fs").readFileSync('/data/certs/key.pem'),  
  cert: require("fs").readFileSync('/data/certs/cert.pem')  
},  
requireHttps: true,
```


Ahora sera necesario reiniciar el docker para que tenga acceso a las credenciales:

```
docker run -d \
  --name nodered \
  -p 1880:1880 \
  -v ~/nodered_data:/data \
  -v ~/certs-node-red:/data/certs \
  --restart unless-stopped \
  nodered/node-red
```

5. Comandos utiles

DOCKER	docker ps	Ver los contenedores docker en ejecución
	docker ps -a	listar todos los contenedores de Docker en tu sistema, tanto los que están en ejecución como los que están detenidos.
	docker exec -it mariadb mariadb -u root -p	Acceder al bash del docker de mariadb
SQL	mysql -h 127.0.0.1 -P 3306 -u ████████ -p	Acceder a la consola SQL
	USE nombre_base_datos;	Selecciona la BD
	SHOW TABLES;	Mostrar tablas de la base de datos actual
	DESCRIBE nombre_tabla;	Ver la estructura de una tabla
	SELECT * FROM nombre_tabla;	Consultar todos los datos de una tabla
	UPDATE nombre_tabla SET correo = 'nuevo@example.com' WHERE id = 1;	Actualizar datos en una tabla
	DELETE FROM nombre_tabla WHERE id = 1;	Eliminar datos de una tabla
	EXIT;	Salir de MariaDB