PROGRAMMATION LOW-CODE

DESCRIPTION DU THÈME

Propriétés	Description
Intitulé long	Initiation à la programmation "Low code" (Node-Red)
Formation(s) concernée(s)	☐ Classes de première Sciences et technologies du management et de la gestion (STMG) ☐ Terminale STMG Système d'information de gestion (SIG) ☒ BTS Services Informatiques aux Organisations
Matière(s)	☐ Sciences de gestion ☐ SIG ☒ Bloc 1 – Support et mise à disposition de services informatiques ☒ Bloc 2 – Spécialisation SISR/SLAM ☐ Bloc 3 – Cybersécurité des services informatiques
Présentation	Découvrir la programmation graphique type No-Code ou Low-Code avec une implémentation de concepts essentiels
Savoirs	B1.6.4 Mettre en oeuvre des outils et des stratégies de veille informationnelle B2.1.3 Utiliser des composants d'accès aux données
Compétences	Pour certains types de ressources : labo, exolab
Transversalité	Veille informationnelle (Bloc 1)
Prérequis	Eléments de langage JavaScript, fonctionnement de Node.JS (serveur) et NPM, et disposer d'une BDD MYSQL
Outils	
Mots-clés	node-red, npm, javascript, low-code, programmation
Durée	2h
Auteur·e·s	ROUMANET David
Version	v 32
Date de publication	22 Mars 2023

DERNIÈRES RÉVISIONS

Ce tableau contient les modifications apportées au document après sa publication uniquement.

Date	Auteur·e	Description

SOMMAIRE

Ce document est une initiation simple à Node-Red : l'objectif pédagogique est une ouverture vers de nouvelles méthodes de programmation. Le document est conçu pour être utilisé en fin de deuxième année de BTS SIO, lorsque la plupart des concepts importants sont acquis.

A INTRODUCTION

Régulièrement, de nouvelles rumeurs apparaissent sur différentes méthodes de programmation. Je ne citerais ici que les principales :

- L'IA remplacera les codeurs
- La nouvelle manière de coder, est sans code
- Go[1] remplacera la plupart des langages
- Web Assembly est l'avenir de la programmation dans les navigateurs
- …

La plupart de ces déclarations sont liées à un engouement pour de nouvelles technologies. Dans cette découverte, nous allons nous intéresser au codage sans code ou presque (No Code ou Low Code).

1 NO CODE / LOW CODE

Depuis plusieurs années, on rencontre cette nouvelle méthode de programmation, dite graphique.

L'objectif de ces outils *No Code* est de permettre de transformer la phase de codage en phase de gestion de processus, pour simplifier le travail du développeur, mais surtout, diminuer les coûts de développement.

Pour cela, il faut utiliser des outils graphiques permettant de relier des cellules spécialisées dans lesquelles on saisit le minimum d'informations.

Toutefois, ce genre de plateforme a également ses inconvénients : le code devient moins accessible, le développeur est plus dépendant de l'outil.

Par exemple, en JavaScript, vous pouvez utiliser l'éditeur de votre choix, comme VSCode, Notepad++, PHPStorm, NetBeans… ce n'est plus le cas en *No Code*.

Nous allons donc aborder de manière très légère ce concept, avec l'environnement Node-Red qui est Low code (il implique encore de la programmation traditionnelle), afin d'avoir une culture et une compréhension globale du fonctionnement.

2 PRÉREQUIS

Pour cette approche "découverte", nous utiliserons les concepts suivants :

- NodeJS et installation de framework avec NPM
- les API RESTFul
- JavaScript

En SISR, il s'agit surtout de découvrir une technologie innovante de programmation.

En SLAM, l'objectif est d'aborder les concepts de la programmation dite "sans code".

1 *Remplacez par votre langage préféré : Rust, ADA, PHP, etc.*

B NODE-RED

1 PRÉSENTATION

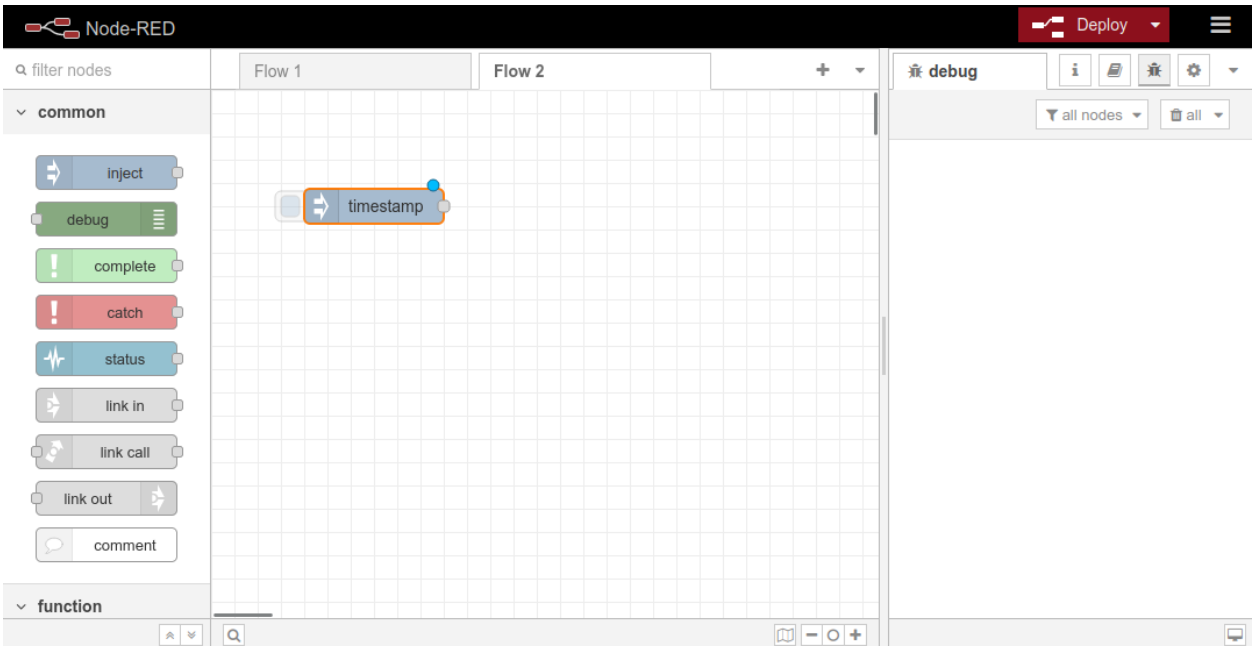
Node-Red est un langage de développement graphique créé par IBM, au début de l'année 2013.

Son objectif est de relier des équipements matériels, des API et des services en ligne pour faciliter l'intégration des IoT (Internet of Things).

Il fait désormais partie de la fondation JS depuis 2016 et est diffusé en open-source.

2 FONCTIONNEMENT

L'éditeur graphique est un service disponible dans un navigateur Internet, mais l'ensemble s'appuie sur NodeJS pour l'exécution (côté serveur).



La documentation[2] est facilement accessible et claire. L'importation et l'exportation de programmes Node-Red se fait au format JSON.

Node-Red utilise plusieurs types de blocs, dans la barre latérale gauche :

- Les blocs **communs** sont les plus fréquemment rencontrés
- Les blocs **fonctions** sont programmables
- Les blocs **réseaux** permettent la communication avec le réseau
- Les blocs **séquences** sont des blocs qui facilitent le traitement de données (joindre, séparer, trier…)
- Les blocs d'**analyses** travaillent sur la conversion des formats de données (CSV, JSON, XML…)
- Les blocs de **stockage** servent pour la gestion des fichiers

2 https://nodered.org/docs/

3 INSTALLATION ET EXÉCUTION

L'environnement Node-Red s'appuie sur le gestionnaire de module NPM et Node.JS pour fonctionner :

- L'installation de Node-Red se fait avec l'instruction `npm install -g --unsafe-perm node-red`.
- L'exécution de la commande `node-red` lance un serveur sur le port local 1880 : http://127.0.0.1:1880
- L'arrêt de Node-Red est commandé par la séquence de touches `CTRL+C`.
- Les activités (fichiers de flux ou flow files) s'enregistrent dans le répertoire /home/*user*/.Node-red (Linux) ou AppData (Windows)

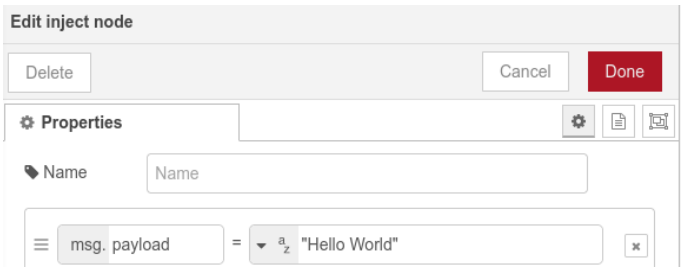
```
node-red
 ✔
16 Oct 22:41:53 - [info]

Welcome to Node-RED
===================

16 Oct 22:41:53 - [info] Node-RED version: v3.0.2
16 Oct 22:41:53 - [info] Node.js  version: v16.16.0
16 Oct 22:41:53 - [info] Linux 5.19.14-1-MANJARO x64 LE
16 Oct 22:41:53 - [info] Loading palette nodes
16 Oct 22:41:53 - [info] Settings file  : /home/david/.node-red/settings.js
16 Oct 22:41:53 - [info] Context store  : 'default' [module=memory]
16 Oct 22:41:53 - [info] User directory : /home/david/.node-red
16 Oct 22:41:53 - [warn] Projects disabled : editorTheme.projects.enabled=false
16 Oct 22:41:53 - [info] Flows file     : /home/david/.node-red/flows.json
16 Oct 22:41:53 - [info] Server now running at http://127.0.0.1:1880/
16 Oct 22:41:53 - [warn]

---------------------------------------------------------------------
Your flow credentials file is encrypted using a system-generated key.

If the system-generated key is lost for any reason, your credentials
file will not be recoverable, you will have to delete it and re-enter
your credentials.

You should set your own key using the 'credentialSecret' option in
your settings file. Node-RED will then re-encrypt your credentials
file using your chosen key the next time you deploy a change.
---------------------------------------------------------------------

16 Oct 22:41:53 - [info] Starting flows
16 Oct 22:41:53 - [info] Started flows
```

Un rappel de sécurité indique que Node-Red n'est pas sécurisé (section orange ci-dessus). Cette initiation n'ira pas jusqu'à la sécurisation d'une application Node-Red, cependant, vous pouvez vous reporter à la documentation officielle pour le faire plus tard : https://nodered.org/docs/user-guide/runtime/securing-node-red

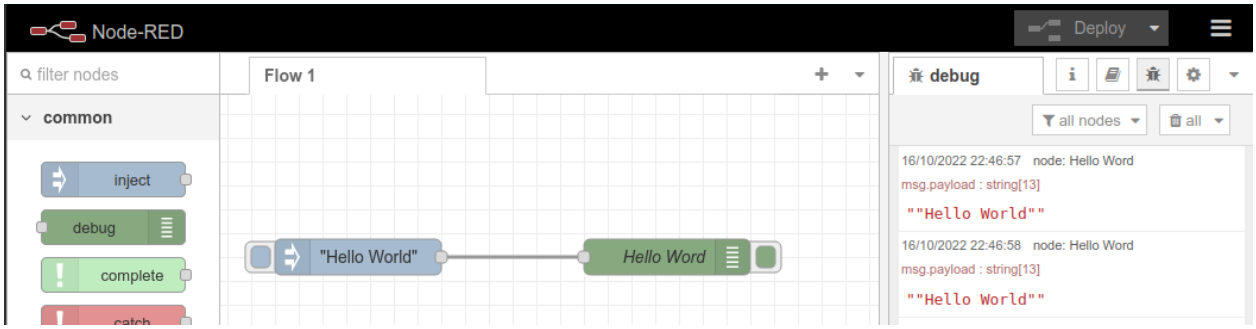
C EXEMPLES D'APPLICATIONS

1 APPLICATION N°1 : HELLO WORD

La première application simple est l'affichage d'un "Hello World" dans la console de débogage. Pour cela, vous devez créer 5 étapes :



- Faire un glisser/déposer du symbole `common → inject` sur la feuille centrale [Flow 1]
- Un double-clic sur ce symbole pour créer la chaîne de caractères (string) "Hello World"
- Faire un glisser/déposer du symbole `common → debug` à droite du symbole inject "Hello World"
- Un double-clic pour renommer le débogueur en "Hello World"
- Créer un lien entre les deux points de ces deux symboles, comme ci-dessous.

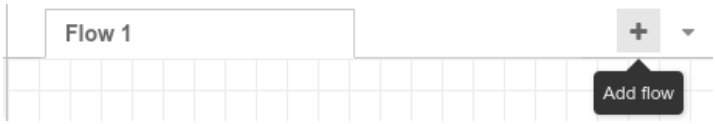


Activez votre programme en cliquant sur le bouton rouge `Deploy` en haut à droite, puis cliquez sur l'onglet [Debug] 

Enfin, cliquez sur la zone bleue à gauche du symbole injecteur "Hello World" : la fenêtre de débogage doit afficher un message à chaque clic.

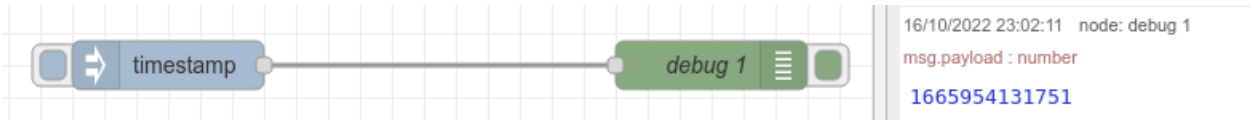
2 APPLICATION N°2 : PASSAGE DE VALEUR

Créez un nouveau projet en ajoutant un onglet dans l'interface Node-Red :



Pour supprimer un "flow", allez dans le `menu hamburger → Flows → Delete`.

Dans ce projet, il s'agit de relier à nouveau un `common → inject` (pas de configuration, on garde TimeStamp par défaut) et un `common → debug` par défaut.



Le nombre qui s'affiche à chaque clic est un estampillage horaire (en secondes). Nous allons insérer une fonction qui va convertir le timestamp en format date avant de l'envoyer dans la console de débogage.

Faites glisser un objet `function → function` entre les deux objets précédents (avec un peu de chance, le lien change en pointillé, lâchez l'objet). Si cela ne fonctionne pas, débranchez le lien et connectez-le à la nouvelle fonction, puis créez un nouveau lien vers l'autre objet.

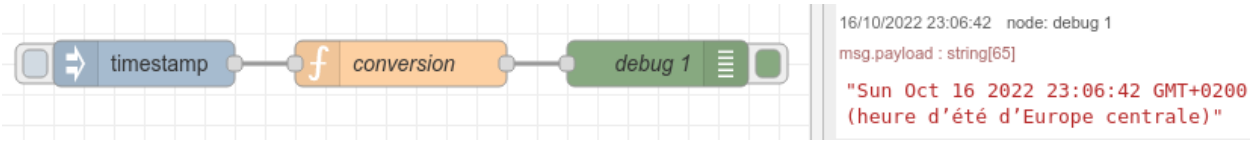
Programmez ensuite l'objet fonction en double-cliquant dessus, comme suit :

name : conversion

Onglet [onMessage]

```
let date = new Date(msg.payload);
msg.payload = date.toString();
return msg;
```

Validez puis relancez le déploiement de la solution (bouton `Deploy`). Désormais, l'affichage dans la console doit changer à chaque nouveau clic sur l'objet 'timestamp'.



Par défaut, les objets génèrent des messages dans un objet au format JSON, nommé msg.

Msg.payload contient donc le message injecté par l'objet `common → inject`.

Essayez d'utiliser la fonction .split(" ") pour n'afficher que l'heure au format HH:MM:SS

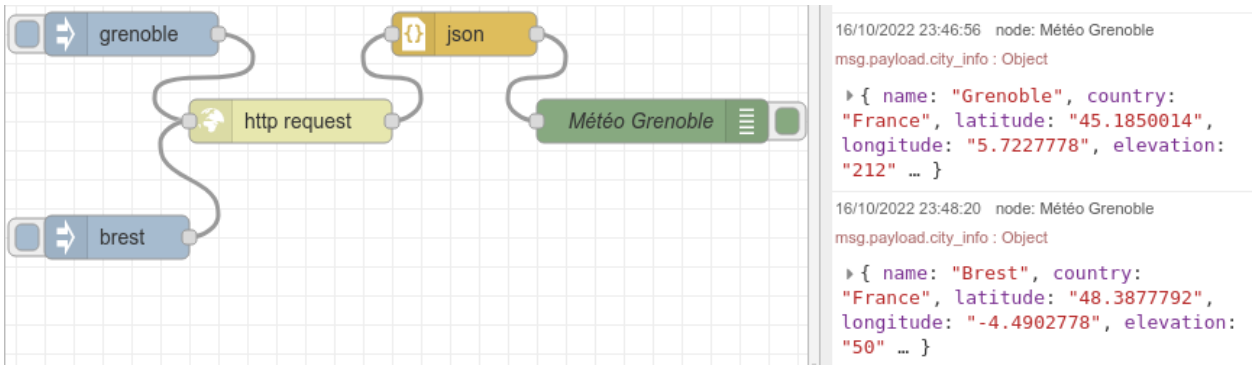
3 APPLICATION N°3 : INTERROGATION API RESTFUL

Node-Red est évidemment bien plus puissant et nous allons interroger la météo du site https://www.prevision-meteo.ch/services/json/

Pour fonctionner, notre application va transmettre une chaîne de caractère (la ville dont nous voulons avoir la météo) à un nouvel objet : `network → http request`. La réponse étant au format JSON, nous utiliserons également un objet `parser → json`.

- Dans l'objet `common → inject`, modifiez la valeur du msg.payload par une chaîne "grenoble"
- Dans l'objet `network → http request` ajoutez {{payload}} après l'url du service API (URL = https://www.prevision-meteo.ch/services/json/{{payload}})
- Dans l'objet `parser → json`, choisissez l'action "Convert between JSON String and Object"
- Dans l'objet `common → debug`, modifiez l'affichage pour une partie de l'objet : msg.payload.city_info

Reliez les éléments entre eux comme ci-dessous et déployez pour valider le fonctionnement.



Dans l'exemple ci-dessus, vous pouvez constater que nous pouvons utiliser plusieurs injecteurs sur un même objet.

Sauriez-vous afficher la ville et sa température courante (current_condition) avec deux debug ?

---

4 APPLICATION N°4 : INTERACTION MYSQL

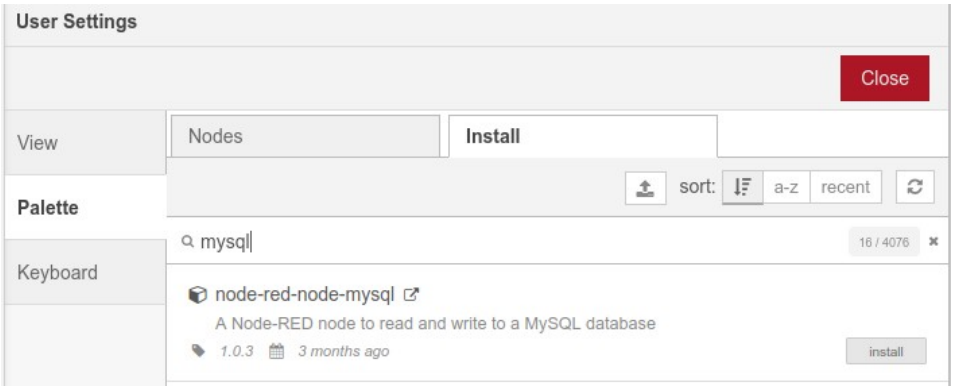
Node-Red n'est pas fait pour créer des sites web (et notamment ceux avec un frond-end important) mais bien pour faciliter l'intégration des IoT : ces sondes ou équipements utilisent le protocole MQTT[3] et sont limités à des messages courts. Node-Red permet donc de réaliser des microservices pour ces appareils.

Cependant, Node-Red étant très orienté API RESTful, on retrouve la réception de données, via un formulaire web. Dans la réalité, il faut plutôt imaginer *un composant transmettant ses données et Node-Red les stockant en base de données*.

Préparation

Pour importer la bibliothèque MySQL, il faut lancer Node-Red et dans l'interface graphique, allez dans le menu Manage Palettes (touches `ALT+P`)

Dans l'onglet [Install] recherchez 'mysql' puis cliquez sur le bouton `install`.



Dans le menu des objets, il y a normalement un objet `Storage → MySQL`.

Créez un nouvel onglet Node-Red et nommez-le "Livre d'Or" pour commencer un nouveau projet proprement.

3 *Message Queuing Telemetry Transport (https://fr.wikipedia.org/wiki/MQTT)*

Voici la configuration de la base :

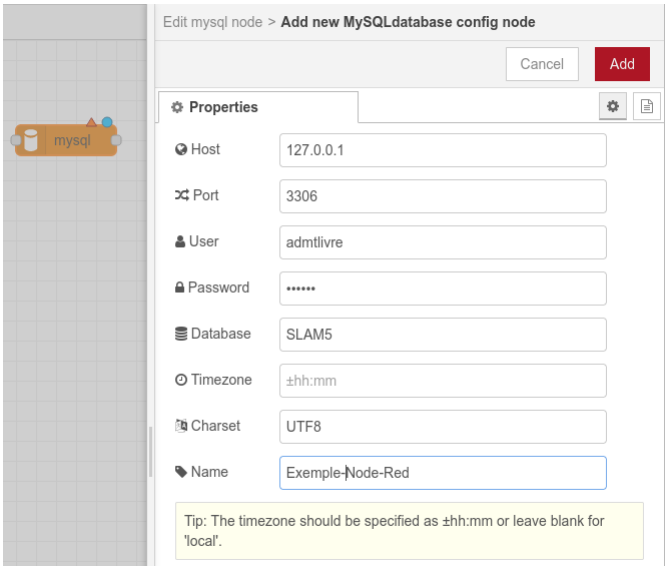
```
CREATE DATABASE livredor CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;
USE livredor;

CREATE TABLE `tlivre` (
  `id` int(11) NOT NULL,
  `name` varchar(50) NOT NULL,
  `message` varchar(300) NOT NULL,
  `evaluation` int(1)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--
-- Contenu de la table `matable`
--

INSERT INTO `tlivre` (`id`, `name`, `message`, `evaluation`) VALUES
(1, 'NORRIS Chuck', 'Best web site David ;)', 5),
(2, 'KENT Clark', 'Need informations about Batman please', 4),
(3, 'PARR Hélène', 'Hey, nothing about girl in computers science!!!', 0),
(4, 'DE LA VEGA Don Diego', 'sometime Zorro? Your website is zero', 0),
(5, 'M. BEANS', 'Are you serious? Do not play with security', 2);

CREATE USER 'admtlivre'@'localhost' IDENTIFIED BY 'adm123';
GRANT ALL PRIVILEGES ON * . * TO 'admtlivre'@'localhost';
```

4.1 Réalisation

Dans la barre latérale gauche de Node-Red, nous trouvons maintenant un objet `mysql` dans la section **storage**.

Glissez-déposez le symbole `storage → mysql` dans l'espace principal de programmation.

Double-cliquez sur celui-ci pour afficher les propriétés et cliquez sur l'icône d'édition (crayon) pour saisir les informations sur la base de données.



User : **admtlivre**
Password : **adm123**
Database : **livredor**
Name : **Exemple-Node-Red**

Cliquez sur le bouton Add puis validez la création en cliquant sur le bouton Done.

Le symbole mysql n'a plus de triangle rouge (▲), indiquant que la configuration est active.

Insérez maintenant un objet `function → function` puis double-cliquez dessus pour y insérer une requête SQL. Le format de la fonction est le suivant :

- msg.topic : contient la requête SQL
- msg.payload : contient les données à insérer dans la table ou la base
- return msg : contient le résultat de la requête

4.1.1 Requête de lecture

Voici le contenu de la fonction :

```
msg.topic = "SELECT * FROM tlivre";
return msg;
```

Ajoutez un bouton `common → Inject` (videz le contenu et changez le symbole si vous le souhaitez) ;

Ajoutez un débogage `common → Debug` et assurez-vous qu'il affiche msg.payload. Renommez-le "Result SQL".

Enfin, cliquez sur le bouton Deploy de l'éditeur.

Le résultat est le suivant :

On note qu'un point vert avec la mention Connected apparaît (indiquant que les identifiants et l'accès à la base sont corrects).

Un clic sur le bouton d'injection affiche dans la console de débogage de Node-Red, l'objet de la réponse SQL.

Comme pour l'API restful, on peut imaginer insérer un objet parser → json pour n'afficher que les données voulues.

4.1.2 Requête d'écriture

La création d'une écriture dans la base est aussi simple, en utilisant le champ msg.payload (qui peut provenir d'une fonction en amont).

4.1.2.1 Requête préparée

Voici le contenu de la fonction :

```
msg.payload=[6, 'MUSK Elon', 'Best book ever seen about me, no?', 5]
msg.topic = "INSERT INTO `tlivre` (`id`, `name`, `message`, `evaluation`) VALUES (?, ?, ?, ?)";
return msg;
```

Attention, Node-Red n'a pas de protection particulière contre les injections SQL. Il est donc dangereux de créer une application qui ne vérifie pas les champs reçus et il est nécessaire d'être très prudent lors du déploiement d'une application en production.

4.1.2.2 Requête paramétrée

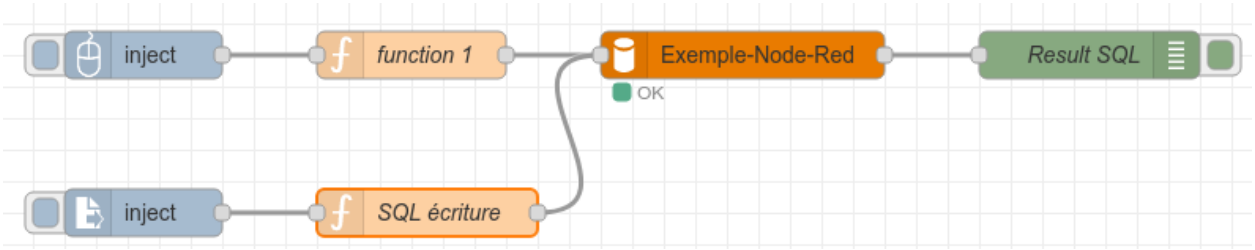
Le contenu change légèrement, comme en NodeJS :

```
msg.payload={}
msg.payload.uid = 6;
msg.payload.uname = "MUSK Elon";
msg.payload.cmt = "Best book ever seen about me, no?";
msg.payload.mark = 5;

msg.topic = "INSERT INTO `tlivre` (`id`, `name`, `message`, `evaluation`) VALUES
(:uid,:uname,:cmt,:mark) ON DUPLICATE KEY UPDATE `message`=:cmt;";
return msg;
```

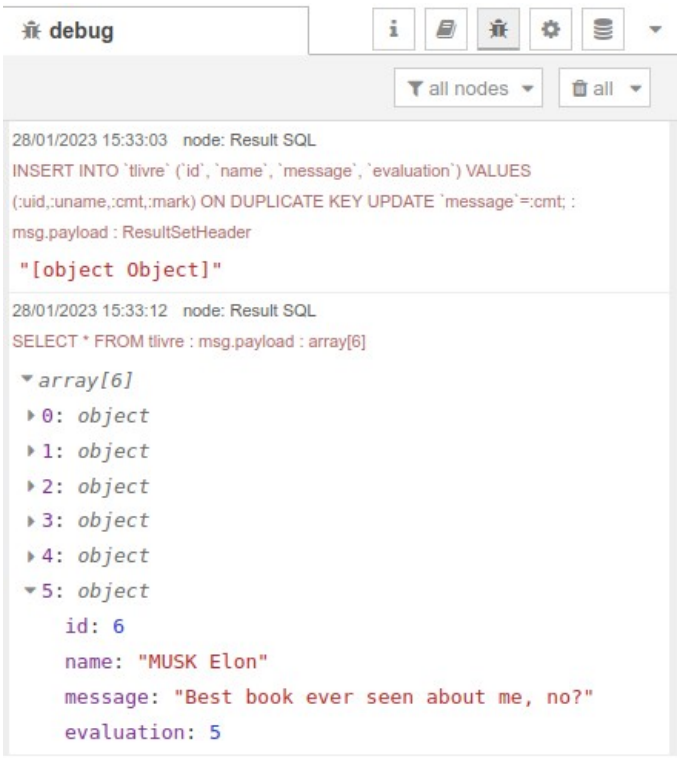
La requête permet en plus, de mettre à jour le champ commentaire (voir ON DUPLICATE KEY).

Ajoutez un bouton `common → Inject` pour cette fonction (videz le contenu et changez le symbole si vous le souhaitez) ;

Enfin, cliquez sur le bouton Deploy de l'éditeur.



Le résultat est le suivant :



Cliquez sur le bouton d'injection de la fonction en écriture, puis cliquez sur le bouton d'injection de la fonction "function 1" (lecture). Le nouvel enregistrement apparaît à la fin de la liste.

4.2 Récupération d'un formulaire HTML

Il existe un objet `Network → http in` qui permet de définir un lien et d'en récupérer les paramètres transmis.

Il faut pour cela, disposer d'un serveur web actif (Apache ou une solution XAMPP, LAMP…), sinon une solution légère est d'installer Python puis, dans le répertoire où se trouvera le fichier HTML, lancer la commande `python -m http.server 8000` (le port sera 8000).

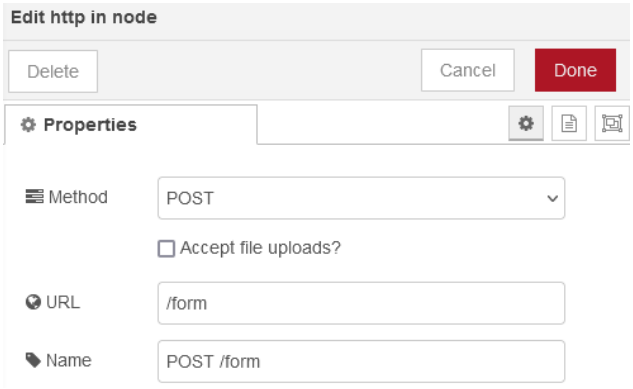
Le fichier HTML comporte un formulaire simple, contenant les champs nécessaires pour la base de données , il sera appelé avec le lien http://localhost[:port]/form.html :

form.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Test Node-Red</title>
</head>
<body>
    <form action="http://localhost:1880/form" method="post">
        <input name="uid" value="7">
        <input name="uname" value="DrNozman">
        <input name="cmt" value="Trop stylé, ce Node-Red">
        <input name="mark" value="5">
        <button>Envoyer</button>
      </form>
</body>
</html>
```

Maintenant, il faut interpréter les données : insérez un objet `Network → http` in et double-cliquez dessus.

Choisir la méthode '**POST**', indiquez l'url **/form** et nommez-le **POST /form** (plus lisible sur l'espace de travail).



Le contenu de la requête HTTP sera parsé dans la variable-objet **msg.payload** :

- msg.payload.uid
- msg.payload.uname
- msg.payload.cmt
- msg.payload.mark

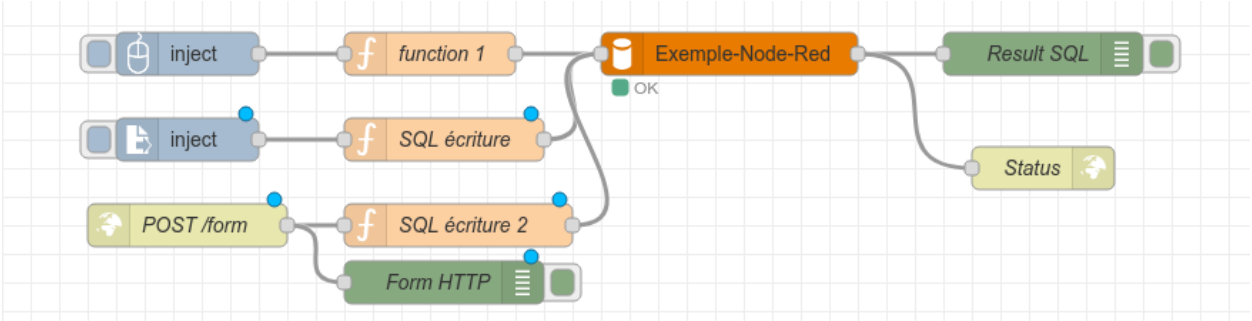
Copiez la fonction SQL écriture et collez la nouvelle juste en-dessous. Éditez son contenu (en supprimant la variable msg.payload) :

```
msg.topic = "INSERT INTO `tlivre` (`id`, `name`, `message`, `evaluation`) VALUES
(:uid,:uname,:cmt,:mark) ON DUPLICATE KEY UPDATE `message`=:cmt;";
return msg;
```

Ajoutez un objet `Network → http response` nommé '**Status**' et relié à la sortie de l'objet mysql.

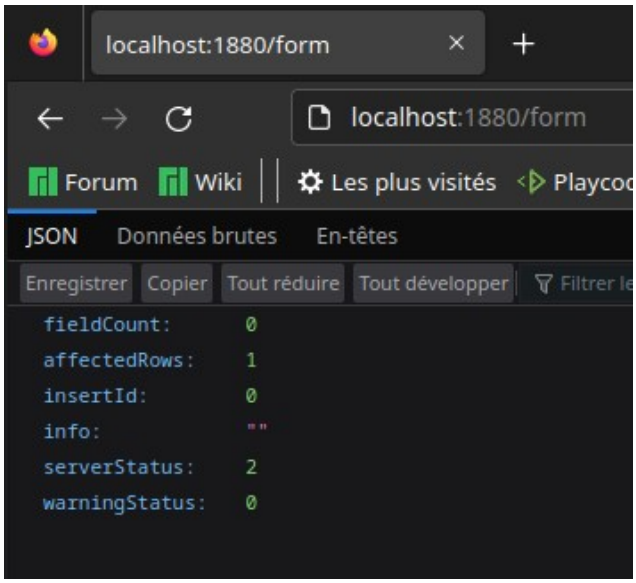
Ajoutez aussi un objet common → debug relié à la sortie de l'objet nommé 'POST /form' pour obtenir le résultat brut dans la console de débogage.

Reliez les objets entre-eux, comme suit :



Lancez la solution en utilisant le bouton Deploy, puis ouvrez le formulaire HTML dans un navigateur et cliquez sur le bouton Envoyer.

La base de données doit maintenant contenir le nouveau contenu (vérifiable avec la fonction de lecture de la base).

Côté Node-Red	Réponse navigateur
28/01/2023 16:16:40 node: Form HTTP msg.payload : Object { uid: "7", uname: "DrNozman", cmt: "Trop stylé, ce Node-Red", mark: "5" } 28/01/2023 16:16:40 node: Result SQL INSERT INTO `tlivre` (`id`, `name`, `message`, `evaluation`) VALUES (:uid,:uname,:cmt,:mark) ON DUPLICATE KEY UPDATE `message`=:cmt; : msg.payload : ResultSetHeader "[object Object]" 28/01/2023 16:16:52 node: Result SQL SELECT * FROM tlivre : msg.payload : array[9] array[9] 0: object 1: object 2: object 3: object 4: object 5: object 6: object 7: object 8: object id: 7 name: "DrNozman" message: "Trop stylé, ce Node-Red" evaluation: 5	

Node-Red est très utilisé dans les systèmes domotiques, les systèmes impliquant des IoT et dans certaines applications de contrôle-commande (instrumentation). Il existe notamment des extensions permettant de créer des tableaux d'indicateurs (extension node-red-dashboard).

Connaître ce genre d'outils est un avantage professionnel certain.