Table des matières

Table des figures

Introduction

Ce travail consiste en l'étude et la mise en place d'une infrastructure pour capter le flux Twitter et suivre les avis des internautes sur l'évolution du cours du Bitcoin. Une partie substantielle du travail concerne la recherche d'un environnement de développement adéquat, son apprentissage et son installation. En particulier nous nous sommes concentrés sur l'outil Node-RED d'IBM, les bases de données NoSQL et les APIs cognitifs d'IBM Watson.

Nous aborderons le captage des tweets, leur stockage ainsi que leur analyse.

Démarrage du projet et recherches

Pour démarrer le projet, nous devons définir le problème : comment savoir de quelle manière le Bitcoin va évoluer. Une solution plausible est de suivre les avis des internautes selon les spéculations émises sur Twitter.

Du moment que le problème est défini, nous devons réfléchir aux différents APIs de Twitter et d'IBM Watson que nous pouvons utiliser pour régler ce problème.

Nous allons découper ces recherches en différentes étapes :

1. La récupération des tweets – Recherche sur les APIs de Twitter
2. Le stockage des tweets dans une base de données NoSQL (MongoDB)
3. L'analyse des tweets – Recherche sur les APIs d'IBM Watson

Pour le développement, nous utiliserons l'environnement graphique Node-RED d'IBM.

Différentes étapes de recherches

Récupération des tweets - Recherches des différentes APIs de Twitter

La source de données de notre système est Twitter. Cependant ce média contient beaucoup de bruit, c'est-à-dire des messages n'apportant aucune information pour notre problème. Le premier travail est donc de filtrer ces messages.

Twitter offre plusieurs APIs permettant aux développeurs de pouvoir récupérer des tweets sous format JSON afin de les utiliser. L'API le plus intéressant que propose Twitter est *Streaming API* qui permet la récupération des tweets en continu. Cet API permet de récupérer 1% de tous les tweets mondiaux et permet aussi de suivre des comptes particuliers (environ 500 comptes).

Analyse des messages : APIs d'IBM Watson

IBM propose un catalogue des différents produits et services dédiés aux développeurs. Une partie de ce catalogue est dédié à Watson, donc à de l'intelligence artificielle concernant notamment le langage naturel.

Les APIs que nous avons envisagés sont :

- Tone Analyser
  Cet API nous permet d'analyser le contenu d'un tweet selon le ton, c'est-à-dire l'émotion véhiculée par le message. On obtient un score pour les différentes émotions : la colère, la peur, la joie, la tristesse, l'esprit d'analyse, la confiance ainsi que le niveau d'hésitation.

- Natural Language Understanding
  Cet API permet de faire ressortir les catégories, concepts, entités, mots-clefs, relations ainsi que le sentiment exprimé dans un message.

- Sentiment Analysis
  Cet API permet de ressortir le « sentiment global » d'un message et retourne un score selon le niveau de « positivité » ou de « négativité » du message.

Les outils

Node-RED

Pour pouvoir réaliser notre projet, nous avons décidé d'utiliser l'environnement de développement graphique open source Node-RED d'IBM qui offre une interface simple pour connecter les APIs d'IBM et en particulier la récupération des tweets, leur stockage et analyse.

Qu'est-ce que Node-RED ?

C'est l'œuvre de deux chercheurs, Nick O'Leary et Dave Conway-Jones, qui, à la base, cherchaient un outil pour eux et qui l'ont ensuite offert en open source.

C'est une interface graphique pour créer des « flows » formés de composants de traitement, les « nodes » qu'on interconnecte visuellement. On peut également programmer des fonctions en Javascript dans un node function qui possède son propre éditeur.

Node-RED se base sur Node.js qui est le côté serveur de Javascript. Grâce à Node.js nous pouvons faire des fonctions en Javascript directement sur notre interface Node-RED.

La puissance de Node-RED vient de la combinaison de deux facteurs principaux :

- C'est une méthodologie de programmation visuelle qui se base sur des nodes et des flows (nous verrons plus en détails ces deux notions dans la suite de ce document).
- La variété et la puissance des nodes proposés dans la palette pour la création de flows et qui sont en lien avec le monde réel, comme par exemple le node de Twitter.

Pourquoi utiliser Node-RED ?

Node-RED permet de réunir et de connecter les composants nécessaires à notre application sous forme de « nodes », en particulier les node-modules des APIs d'IBM Watson. La facilité d'utilisation de cet environnement nous a permis de tester facilement les différents APIs afin de déterminer celui qui correspond le mieux à nos attentes.

La programmation dans Node-RED

Les nodes

Node-RED considère un node comme une boite noire, c'est un composant du flow qui a un but défini. Il reçoit des données, exécute une action précise et transmet les données traitées au node suivant dans le flow.

Lorsqu'on installe Node-RED on dispose d'une palette de base relativement complète avec les principaux nodes, mais il est possible d'ajouter des nodes-modules supplémentaires développés par la communauté Node-RED.

Voici la palette de base que nous propose Node-RED à l'installation :

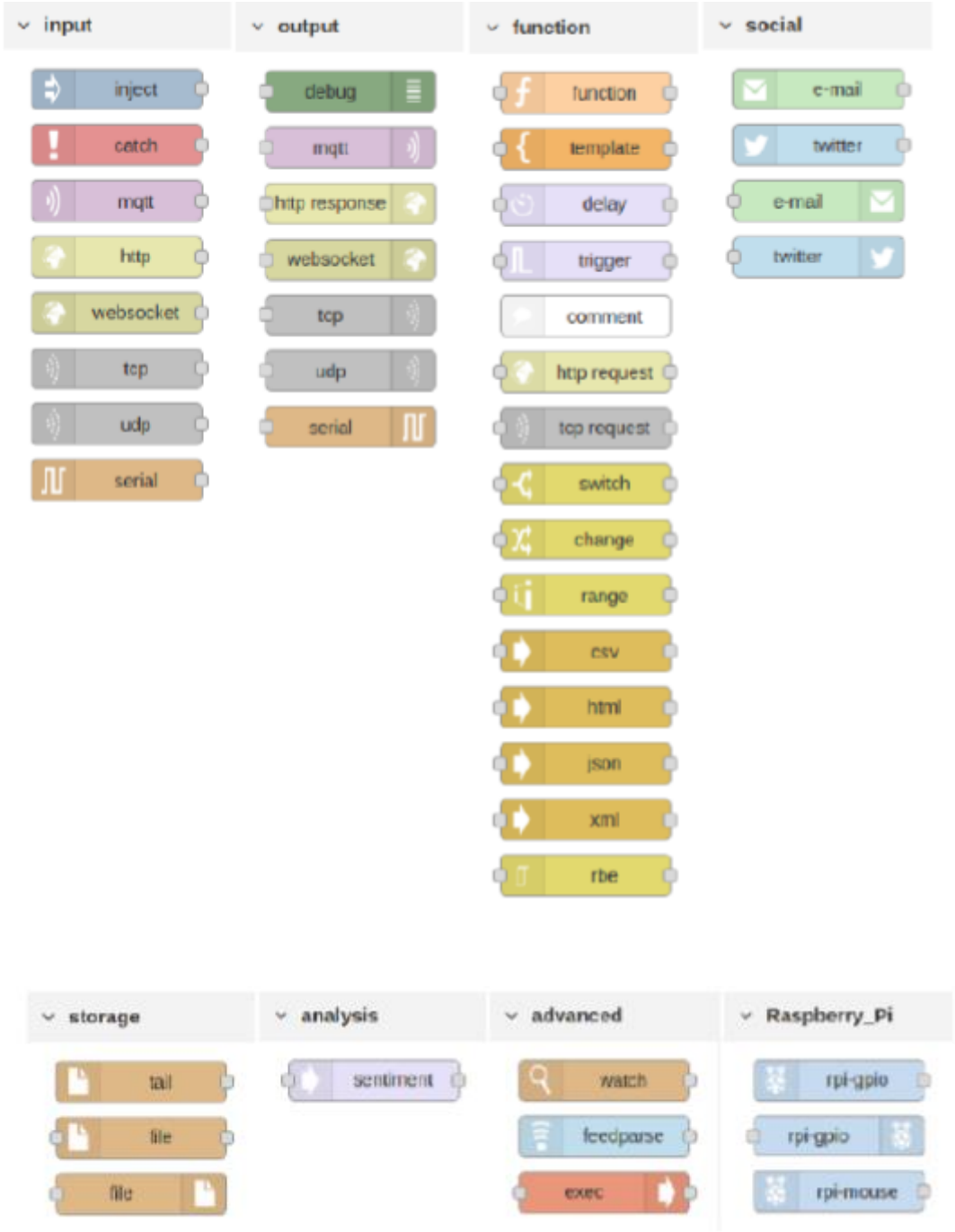


Figure 1 : Palette de base de Node-RED

Les différents nodes sont classés selon leur usage dans un flow. La palette est donc organisée de manière simple afin d'aider l'utilisateur dans sa création sur Node-RED.

Les nodes interconnectés communiquent au travers de messages qui passent dans le flux d'un node au suivant.

Propriétés de base des messages

Tous les nodes ont une propriété commune : les messages.

Grâce aux messages transmis d'un node à un autre, nous pouvons construire un flow complet. Le node de type « inject » permet d'introduire de l'information dans le système. Pour faire passer un message d'un node à un autre, il faut les relier graphiquement pour créer un flow.

La caractéristique principale d'un node est de consommer un message en input et de produire un message en output. Ces messages transmettent l'information dans le flow d'un node à un autre et ce sont ces messages qui vont être traités et analysés. Ces messages sont des objets en Javascript qui ont trois propriétés de base :

- Un message topic
  C'est-à-dire le « titre » du message. Dans le cas de Twitter, le message topic sera « tweet/ » et le pseudo de l'internaute qui a tweeté.
- Un payload
  C'est le cœur du message. C'est la partie que l'on analyse dans notre cas.
- Une identification interne
  C'est une identification qui est mise par Node-RED.

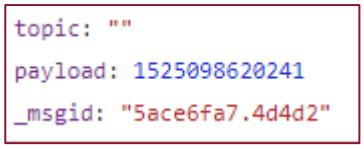


Figure 2 : Résultat d'un message

En plus de ces trois propriétés de base certains nodes peuvent produire des informations supplémentaires qui dépendent du node. Par exemple, lorsque l'on passe un tweet comme message, nous aurons plus d'informations que les trois propriétés de bases :

Figure 3 : Résultat d'un tweet

Dans la partie « tweet », on retrouve en plus des informations sur le user, les hashtags, une indication si c'est un retweet, la date de création du tweet et encore d'autres informations.

Les différents types de nodes

Il existe trois principaux différents types de nodes :

1. Les Input Nodes

Ces nodes injectent de la data dans notre flow.

L'image de droite présente les différents types d'input de la palette de base à laquelle nous allons ajouter les nodes Twitter et MongoDB qui permettent aussi de récupérer des données qui vont ensuite pouvoir être traitées dans notre flow.

Dans Node-RED, quand on parle d'input nodes, on pense au premier node qui est le « Inject ». Par défaut, il fournit un timestamp.

L'input node est toujours le premier node dans un flow, c'est lui qui alimente le flow grâce aux messages qu'il injecte.

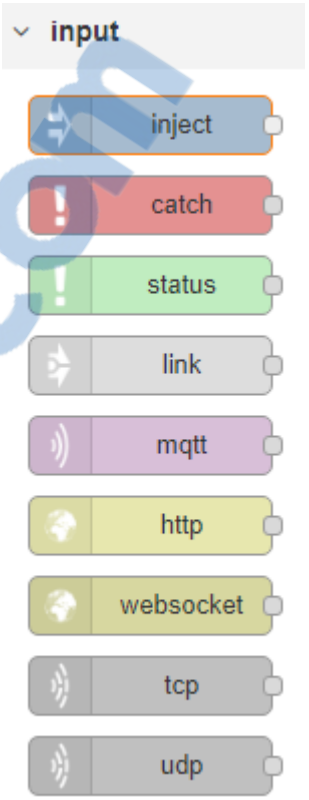


Figure 4 : Inputs nodes

2. Les Outputs Nodes

Les nodes d'output ressortent un résultat final de manière visuelle sur l'interface ou le stocke dans une base de données.

Le principal output node est le « Debug », car il permet la visualisation des résultats et des erreurs qui peuvent se produire.

Dans les outputs nodes disponibles, nous avons aussi le « http response » qui permet de visualiser les résultats sur une page web.

MongoDB est considéré comme un output node qui stocke les résultats produits. Twitter peut aussi s'utiliser comme output node pour produire un tweet à partir du résultat obtenu.

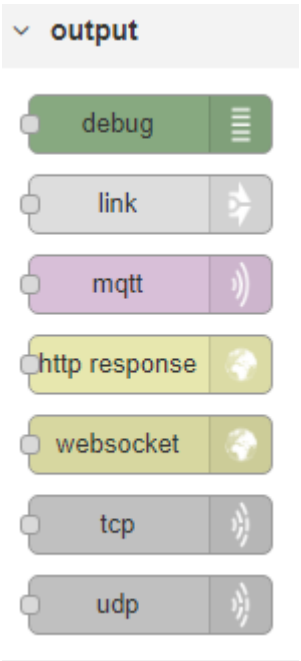


Figure 5 : Outputs nodes

3. Les Processing Nodes

Les Nodes dit Processing font les traitements sur les données entre les input et output nodes.

On traite les données grâce à des fonctions ou à des nodes déjà conçus pour formater des données, compter le nombre de messages passés dans un node, splitter les données, les trier et encore d'autres possibilités.

Le Processing nodes le plus utilisé est le « function » grâce auquel on peut librement coder en Javascript dans l'éditeur.

Les nodes « csv » et « xml » permettent de changer le format de sortie du message.

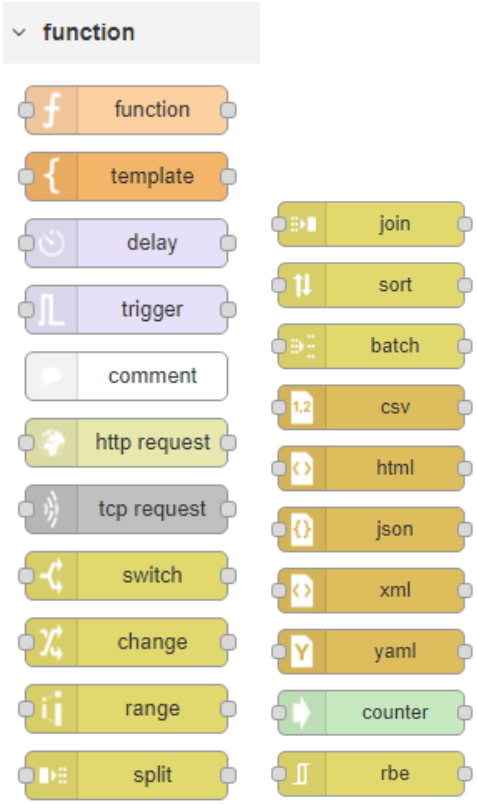


Figure 6 : Processing nodes

En plus de ces trois types principaux de nodes, il existe deux autres catégories :

- Les Credentials Nodes

  Ce sont les nodes qui se connectent à un système externe en demandant une API key ou une authentification avec un username et un password.

- Les User-Created Nodes

  Ce sont les nodes que l'on programme, comme les nodes function dont le code peut être stocké et réutilisé.

Node Function

Un node function est un node générique pour écrire en Javascript un traitement de message et en particulier de son payload.

Par défaut, un node function ne fait que renvoyer le messager d'entrée vers sa sortie :

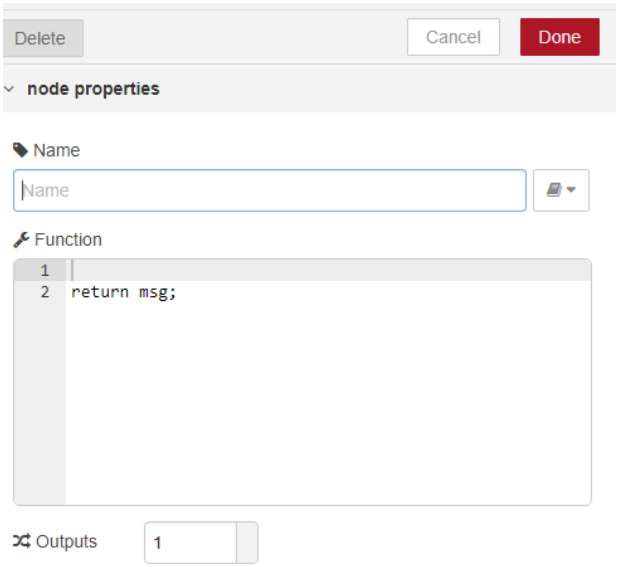


Figure 7 : Editeur d'un node function

On voit qu'il y a la notion d'« output » dans l'écran de programmation de ce node. C'est un paramètre modifiable pour définir le nombre de « canaux de sortie » permettant d'envoyer différents messages à des nodes distincts.

Dans le code et le schéma ci-dessous, on définit trois outputs dans les paramètres du node function. Les messages de sortie dépendent d'une condition et la syntaxe est *[sortie 1, sortie 2, sortie 3]*.

```
1. if (msg.payload == "high") {
2. return [ msg, null, null ];
3. } else if (msg.payload == "med") {
4. return [ null, msg, null];
5.} else {
6. return [null, null, msg];
7. }
```

Figure 8 : Exemple de node function avec multiple outputs

Il faut voir les outputs du *test function* sur l'interface graphique de haut en bas, le plus haut sera le premier. Dans le code, il faudra renvoyer trois paramètres dans notre *return*, un pour chaque sortie de notre node. Le premier est dédié au premier output du *test function*. Dans la ligne 2, nous renvoyons le message uniquement dans le premier output, nous faisons de même à la ligne 4, en le renvoyant dans le deuxième output et de même à la ligne 6 avec le dernier output.

Cet exemple démontre les possibilités octroyées par Node-RED dans la conception des nodes-function.

Context et Global Context dans un node function

Le context et le global context sont des propriétés utilisées dans les nodes function pour stocker des informations.

La différence entre le context et le global context est que l'information est stockée dans le node pour un context et dans le flow pour un global context.

Là, nous avons l'exemple de l'utilisation d'un context au sein d'un node. Ce context compte le nombre de fois où l'on va passer dans le node. On déclare un compteur et on l'initialise à zéro s'il n'existe pas, ensuite on l'incrémente et on le set pour le stocker.

```
// initialise the counter to 0 if it doesn't exist already
var count = context.get('count')||0;
count += 1;
// store the value back
context.set('count',count);
// make it part of the outgoing msg object
msg.count = count;
```

Figure 9 : Exemple de context avec un compteur

Pour obtenir une information dans un autre node du flow, on utilise un global context :

```
global.set("foo","bar");  // this is now available to other nodes
```

Figure 10 : Exemple de global context

Dans cet exemple, on set la variable « foo » avec la valeur « bar ». Si dans un autre node, on fait un get sur le « foo » :

```
var myfoo = global.get("foo");  // this should now be "bar"
```

Figure 11 : Exemple de récupération d'un global context

En reprenant notre exemple de compteur qui est dans le context, si ajoute cette ligne :

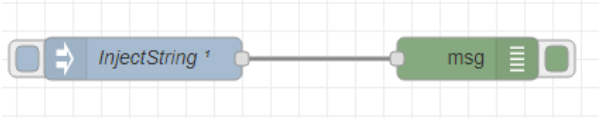
global.set(« compteur », count) ;

Dans un autre node, on récupère ce compteur en faisant un get avec la variable « compteur ».

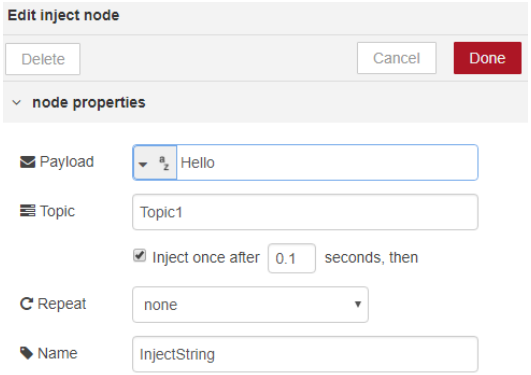
Les interactions entre les différents nodes pour former un flow

Pour illustrer les interactions entre les nodes, voici un exemple : créer une entrée de type String qui produit « hello », créer une fonction qui lui concatène « world » et afficher le résultat :

1. Création du node Inject avec le text « hello » et affichage du résultat :

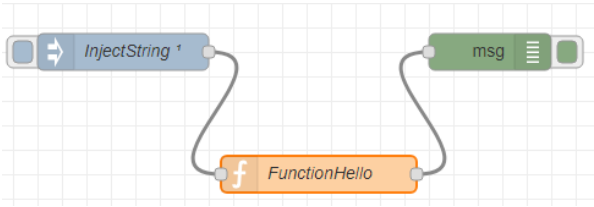


Paramétrage de notre « Inject » :

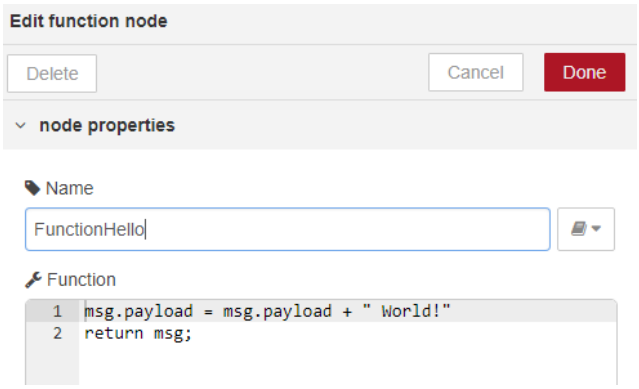


Résultat après notre output dans le debug :

2. Ajout du node-function pour la concaténation avec le mot « world » et affichage du résultat :



Code du function node :



Le résultat ressorti dans le output :

MongoDB

Qu'est-ce que MongoDB ?

MongoDB est une base de données NoSQL (Not Only SQL), c'est-à-dire que c'est une base qui ne suit pas le modèle relationnel.

Il existe différentes familles de base NoSQL :

- **Clef/Valeur**
  Ce type de base de données est basée sur les clefs qui sont uniques, chaque donnée est identifiée par une clef qui a sa propre valeur et il n'y pas de langage pour pouvoir faire des requêtes.
- **Colonnes**
  Normalement, les données dans une base sont représentées sous forme de lignes avec un identifiant pour une ligne entière, mais les base de données en colonnes permettent de se focaliser sur les attributs en distribuant les identifiants en colonne. Ainsi on peut traiter les informations des attributs sans avoir les autres informations.

- **Documents**
  C'est la base qui ressemble le plus à une base de données relationnelles. Elle s'appuie sur le principe des clefs comme la première, mais chaque clef est reliée de manière unique à un document qui contient plus d'informations.
  Ce type de bases permet les requêtes pour retrouver les informations.
  MongoDB est une base basé document.
- **Graphes**
  Pour ce dernier type, on s'intéresse à la corrélation entre les différents éléments, par exemple si on veut calculer la distance entre deux personnes connectées sur un réseau social.
  Ces bases permettent un visuel graphique des données présentes.

Pourquoi utiliser une base de données NoSQL ?

Nous utilisons une base de données NoSQL pour trois principales raisons :

- La rapidité
- Le stockage des données sans contraintes d'intégrité des données
- La performance

Architecture

Ci-après, nous présentons l'architecture générale de notre solution. Concernant la représentation graphique, nous avons naturellement choisi de la représenter dans le graphisme de Node-RED :

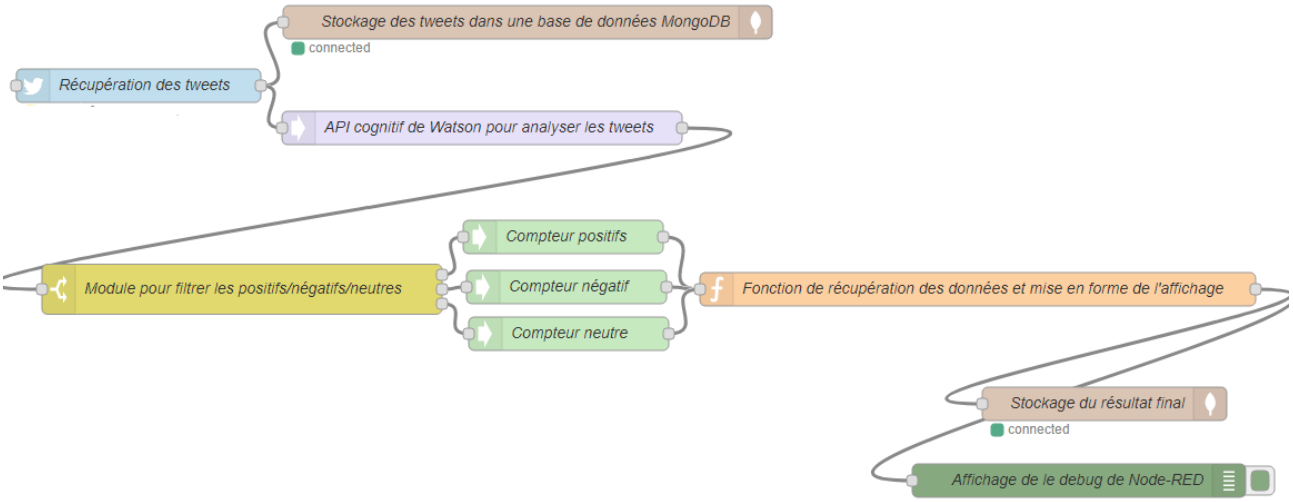
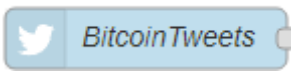


Figure 12 : Architecture de la solution

Mise en place du projet

La mise en place du projet se fait en plusieurs étapes qu'on décompose selon les node-modules qu'on a utilisé :

Node Twitter



Le premier node est celui de Twitter pour récupérer les tweets qui nous intéressent selon les mots clefs que l'on va passer en paramètre.

On récupère tous les tweets publics qui ont les mots-clefs :

- bitcoin up price
- bitcoin down price
- bitcoin increase
- bitcoin decrease

L'ordre des mots n'a pas d'importance, il faut qu'il y ait l'une des quatre combinaisons présente dans le tweet pour le récupérer.

Node MongoDB

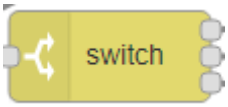
On stocke les tweets qui ressortent de Twitter dans une collection pour garder une trace des tweets avant de les analyser.

Node Sentiment Analysis

Ce node met un score sur les tweets que l'on récupère. Les scores vont de -9, si le tweet est vraiment négatif à 9 si le tweet est vraiment positif.

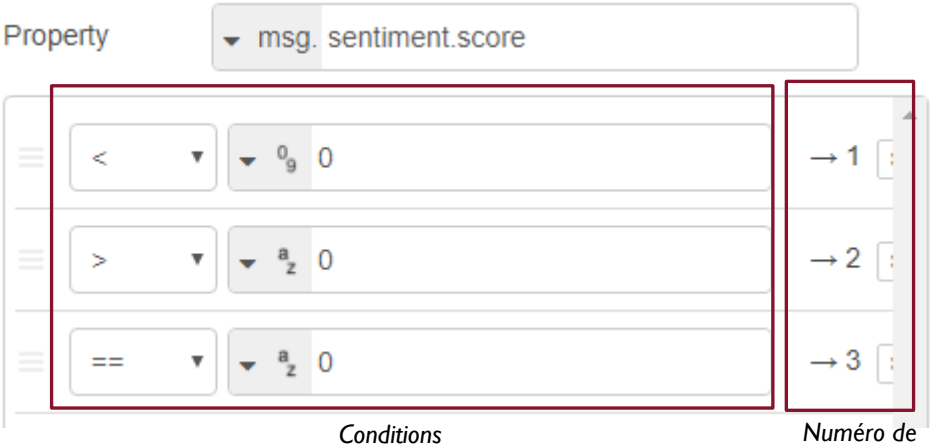
Ce score ne se base pas sur les mots-clefs mais sur la phrase entière et sur la sémantique.

Node Switch

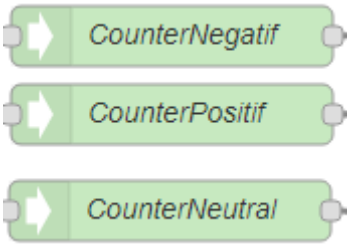


Ce node sépare les tweets positifs, négatifs et neutres en se basant sur le score.

On a donc trois outputs, un pour chaque type selon les conditions :



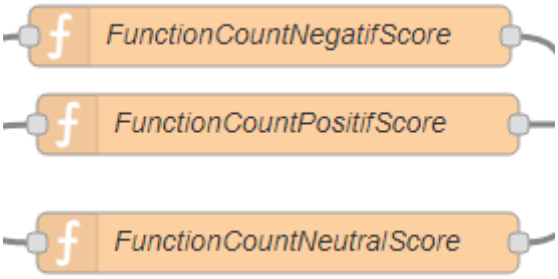
Node Counter



Pour chacune des conditions, on a un compteur pour savoir combien de tweets négatifs, positifs et neutres sont transmis.

Node Function

Node Function pour chacun des scores



Pour chacune des fonctions, on a le même code, les seules différences sont les noms des variables et le message que l'on ressort.

On déclare la variable du compteur en context, pour pouvoir la stocker dans le node.

On met le compteur et le score en global context pour les récupérer dans notre fonction finale.

Le code se compose comme suit :

```
1  //Déclaration de la variable du compteur et initialisation à 0 si elle n'existe pas
2  var count = context.get('count')||0;
3  //incrémentation du compteur avec le score reçu du noeud précédent
4  count += msg.sentiment.score;
5  //stockage de la valeur
6  context.set('count',count);
7  global.set("countNeg", msg.payload);
8  global.set("scoreNeg", count);
9  //Création du payload
10 msg.payload = "Nb nég : " + msg.payload + " score total : " + count;
11 //Retour du message final
12 return msg;
```

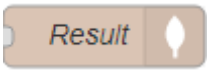
Node Function final

Dans cette fonction finale, on va faire différentes opérations :

- Mise en place de la date et de l'heure avec le formatage adapté
- Récupération des variables globales que l'on a stocké au préalable
- Création du compteur du nombre total de tweets ainsi que du compteur final du score
- Mise en place du payload de sortie

```
//Création des variables pour la date et l'heure
var d = new Date();
var dd = d.getDate();
var mm = d.getMonth()+1; //Janvier est le 0, donc on fait +1 pour avoir le bon mois
var yyyy = d.getFullYear();
var hh = d.getHours();
var min = d.getMinutes();

//Ajout du 0 avant le jour quand c'est avant 10
if(dd<10) {
    dd = '0'+dd
}
//Ajout du 0 avant le mois quand c'est avant 10
if(mm<10) {
    mm = '0'+mm
}

//Ajout du 0 avant l'heure quand c'est avant 10
if(hh<10) {
    hh = '0'+hh
}
//Ajout du 0 avant les minutes quand c'est avant 10
if(min<10) {
    min = '0'+min
}

//Formatage de la date pour l'affichage
d = dd + '/' + mm + '/' + yyyy + " - " + hh + ":" + min;
//Récupération du compteur et du score négatif ou mise à 0 s'il n'existe pas
var countNeg = global.get("countNeg")||0;
var scoreNeg = global.get("scoreNeg")||0;
//Récupération du compteur et du score positif ou mise à 0 s'il n'existe pas
var countPos = global.get("countPos")||0;
var scorePos = global.get("scorePos")||0;

//Récupération du compteur et du score neutre ou mise à 0 s'il n'existe pas
var countNeutral = global.get("countNeutral")||0;
var scoreNeutral = global.get("scoreNeutral")||0;
//Compteur du nombre de tweets
var nbTweet = countNeg + countPos + countNeutral;
//Compteur du score final
var scoreFinal = scoreNeg + scorePos + scoreNeutral;

//Création du payload
msg.payload = d + " : Nombre de tweets total : " + nbTweet + " Score totale : " + scoreFinal;
msg.payload = msg.payload + " & " + "Nombre de tweets négatifs : " + countNeg + " Score négatif : " + scoreNeg;
msg.payload = msg.payload + " & " + "Nombre de tweets positifs : " + countPos + " Score positif : " + scorePos;
msg.payload = msg.payload + " & " + "Nombre de tweets neutres : " + countNeutral + " Score neutre : " + scoreNeutral;
//Retour du message final
return msg;
```

Node Mongo



On ajoute un deuxième node MongoDB pour stocker le résultat après notre analyse dans une nouvelle collection.

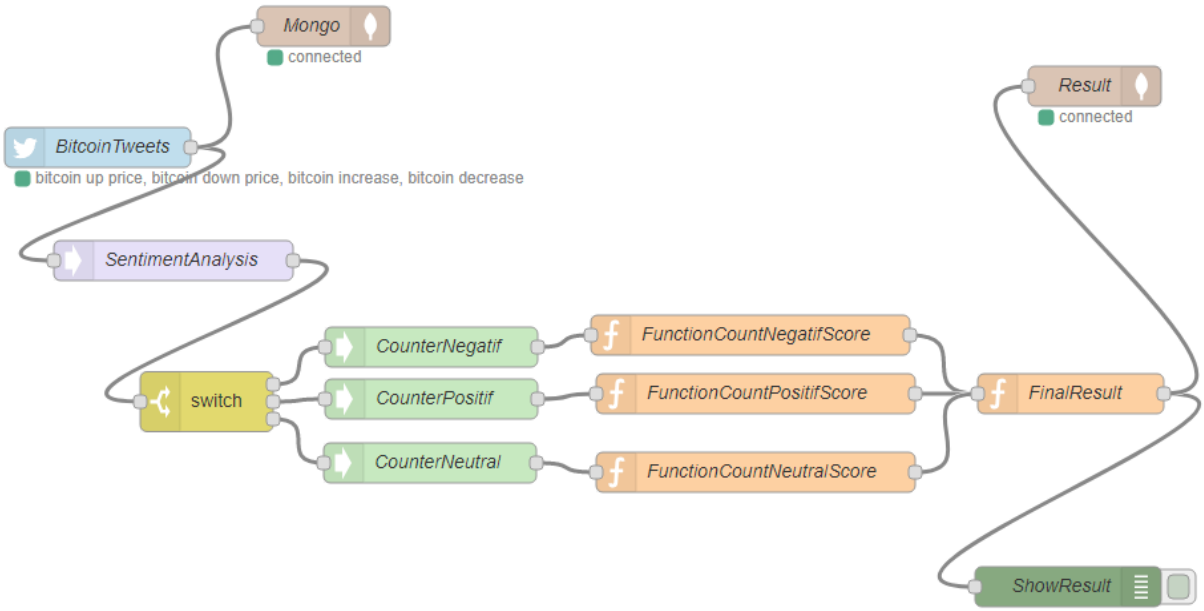
On a une collection qui comporte tous les tweets et une deuxième qui a les résultats, on peut donc grâce aux deux collections voir le tweet et le score qui lui a été attribué.

Node Debug

Le dernier node qu'on ajoute est un debug qui affiche le résultat que l'on a paramétré dans notre dernière fonction :

```
"30/04/2018 - 21:17 : Nombre de tweets total : 9 Score totale : 3 & Nombre de
tweets négatifs : 1 Score négatif : -2 & Nombre de tweets positifs : 3 Score
positif : 5 & Nombre de tweets neutres : 5 Score neutre : 0"
```

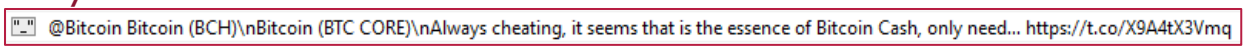
Notre architecture finale :



Résultat du projet

Les résultats de ce projet ne peuvent être concluants que si les scores attribués par l'IA de Watson sont similaires à ceux qu'un humain aurait attribué.

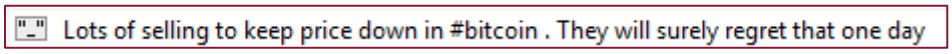
Analyse des scores



Score attribué par Sentiment Analysis : -3

Score qu'on aurait attribué : -3

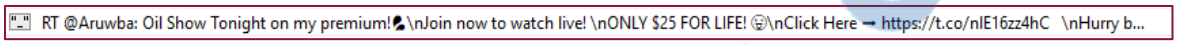
Raison : Les mots « always cheating » met en avant une certaine négativité. Nous aurions accordé le même score.



Score attribué par Sentiment Analysis : -2

Score qu'on aurait attribué : -4

Raison : Les mots « surely regret » mettent en avant une vraie spéculation de la descente du cours.



Score attribué par Sentiment Analysis : 0

Score qu'on aurait attribué : 0

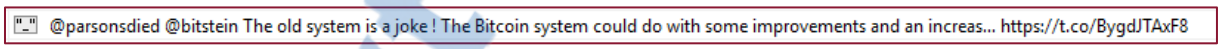
Raison : Malgré les filtres appliqués pour la récupération des tweets, nous récupérons du bruit.

Dans la majorité des cas, ces tweets publicitaires ont le score de 0 ce qui n'influencent pas notre score.

Score attribué par Sentiment Analysis : 2

Score qu'on aurait attribué : 0

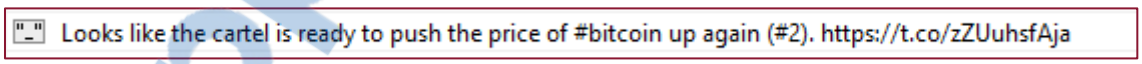
Raison : Dans les tweets scorés à 2, on trouve des tweets publicitaires qui faussent nos résultats.



Score attribué par Sentiment Analysis : 2

Score qu'on aurait attribué : 0

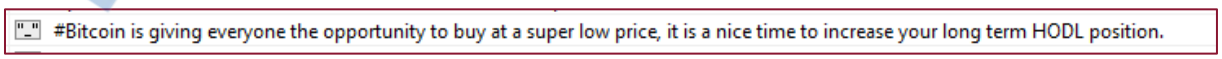
Raison : Certains tweets sont plus pertinents, mais n'amènent pas d'informations quant à la spéculation du cours, juste une constatation.



Score attribué par Sentiment Analysis : 2

Score qu'on aurait attribué : 2

Raison : Les tweets pertinents ont la majorité du temps des scores cohérents à celui qu'un humain aurait attribué.



Score attribué par Sentiment Analysis : 9

Score qu'on aurait attribué : 9

Raison : Peu de tweets ont un score aussi élevé, mais ceux-ci sont rankés correctement.

Conclusion des résultats obtenus

Les scores sont dans la plupart des cas attribués correctement. Le node « Sentiment Analysis » peut être considéré comme fiable et permet une vraie notation du sentiment globale du tweet.

Le problème est que trop de bruit passe depuis Twitter et les scores attribués sont parfois positifs ou négatifs même si une grande partie reste neutre. Il y a un effet en chaîne lors d'un retweet publicitaire qui a un score positif ou négatif qui peut influencer nos résultats.

Le node Twitter ne possède pas un filtre adéquat pour garder uniquement les tweets « intéressants ». Ce node devrait avoir un filtre plus fin. Une amélioration possible serait de pouvoir écrire les mots-clefs des tweets que nous ne voulons pas récupérer. Nous avons pu constater que les tweets publicitaires contenaient souvent l'expression « sign up », en éliminant seulement les tweets contenants ces mots, nous pourrions déjà obtenir des résultats plus intéressants pour notre analyse.

Notre source de données n'est finalement pas assez fiable pour pouvoir faire une analyse entre nos résultats et le cours réel du Bitcoin.

Conclusion

Durant ce projet, nous avons appris à utiliser de nouveaux outils et des nodes des APIs cognitifs d'IBM Watson pour créer une application qui nous permettrait de répondre à notre besoin qui est de savoir si les spéculations émises sur le cours du Bitcoin peuvent avoir une corrélation avec le cours réel de cette cryptomonnaie.

En utilisant les nodes-modules disponibles, nous avons pu récupérer les tweets selon les paramètres que nous avions définis selon les mots-clefs pertinents à notre besoin, les stocker dans une base de données NoSQL grâce au node-module de MongoDB et les analyser grâce à celui d'IBM Watson, « Sentiment Analysis ».

Malgré les résultats non concluants de ce projet, Node-RED s'avère être intéressant pour la création d'application. Son interface visuelle facilite les traitements et nous a permis de construire l'architecture nécessaire à ce travail et peut devenir une révolution dans le domaine de création d'application.

Nous pouvons en conclure que notre hypothèse de spéculation concernant le cours du Bitcoin grâce aux avis émis sur Twitter pourrait s'avérer vraie si nous pouvions récupérer uniquement les tweets « intéressants » à notre besoin.

Sources

Sites internet

1. NODE-RED. *Node-RED* [en ligne]. [Consulté le 20 décembre 2017]. Disponible à l'adresse : https://nodeRED.org/

2. IBM. *Documentation des APIs pour les services* Watson [en ligne]. [Consulté le 22 février 2018]. Disponible à l'adresse : https://console.bluemix.net/developer/watson/documentation

3. TWITTER. *Documentation des APIs pour les services de Twitter* [en ligne]. [Consulté le 10 janvier 2018], Disponible à l'adresse : https://developer.twitter.com/en/docs

4. NODE-RED. *Documentation sur l'utilisation des nodes Watson* [en ligne]. [Consulté le 15 mars 2018], Disponible à l'adresse : https://flows.nodered.org/node/node-red-node-watson

5. GITHUB. *Exemples d'utilisation des nodes Watson* [en ligne]. [Consulté le 15 mars 2018]. disponible à l'adresse : https://github.com/watson-developer-cloud/node-red-labs/tree/master/basic_examples

6. DIGITAL OCEAN, 2014. *A comparison Of NoSQL Database Management Systems And Models* [en ligne]. [Consulté le 18 février 2018]. Disponible à l'adresse : https://www.digitalocean.com/community/tutorials/a-comparison-of-nosql-database-management-systems-and-models

7. OPENCLASSROOMS. *Maîtrisez les bases de données NoSQL* [en ligne]. Consulté le 18 février 2018], Disponible à l'adresse : https://openclassrooms.com/courses/maitrisez-les-bases-de-donnees-nosql

8. MONGODB, 2018. *Top 5 Considerations When Evaluating NoSQL Databases* [en ligne]. [Consulté le 18 février 2018]. Disponible à l'adresse : https://www.mongodb.com/collateral/top-5-considerations-when-evaluating-nosql-databases

9. NODE-RED GUIDE, 2016. *Node-RED Programming Guide : A brief Introduction to Node-RED* [en ligne]. [Consulté le 16 janvier 2018]. Disponible à l'adresse : http://noderedguide.com/nr-lecture-1/

10. NODE-RED GUIDE, 2016. *Node-RED Programming Guide : Building your first flows* [en ligne]. [Consulté le 16 janvier 2018]. Disponible à l'adresse : http://noderedguide.com/node-red-lecture-2-building-your-first-flows-15/

11. NODE-RED GUIDE, 2016., *Node-RED Programming Guide : Basic nodes and flows* [en ligne]. [Consulté le 16 janvier 2018]. Disponible à l'adresse : http://noderedguide.com/node-red-lecture-3-basic-nodes-and-flows/

12. NODE-RED GUIDE, 2016. *Node-RED Programming Guide : A tour of the core nodes* [en ligne]. [Consulté le 16 janvier 2018]. Disponible à l'adresse : http://noderedguide.com/node-red-lecture-4-a-tour-of-the-core-nodes/

13. NODE-RED GUIDE, 2016. *Node-RED Programming Guide : The Node-RED programming model* [en ligne]. [Consulté le 16 janvier 2018]. Disponible à l'adresse : http://noderedguide.com/node-red-lecture-5-the-node-red-programming-model/

14. NODE-RED GUIDE, 2016. *Node-RED Programming Guide : Intermediate flows* [en ligne]. [Consulté le 16 janvier 2018]. Disponible à l'adresse : http://noderedguide.com/node-red-lecture-6-intermediate-flows-2/

15. NODE-RED GUIDE, 2016. *Node-RED Programming Guide : Dashboard and UI techniques* [en ligne]. [Consulté le 16 janvier 2018]. Disponible à l'adresse : http://noderedguide.com/lecture-7-dashboards-and-ui-techniques-for-node-red/

16. NODE-RED GUIDE, 2018. *Node-RED Programming Guide : Examples of advanced flows* [en ligne]. [Consulté le 16 janvier 2018]. Disponible à l'adresse : http://noderedguide.com/node-red-lecture-8-advanced-flows-with-node-red/

Vidéos

1. Rodric Yates, 2014. Node Red in 5 minutes [enregistrement vidéo]. *YouTube* [en ligne]. 7 juin 2014. [Consulté le 13 février 2018]. Disponible à l'adresse : https://www.youtube.com/watch?v=f5o4tIz2Zzc

2. Mr Sandeep, 2017. Node Red | MongoDB connection using Node-RED [enregistrement vidéo]. *YouTube* [en ligne]. 4 octobre 2017. [Consulté le 25 avril 2018]. Disponible à l'adresse : https://www.youtube.com/watch?v=zRRQCEov-4Q

Images

1. Image du logo de Node-RED- Disponible à l'adresse : https://upload.wikimedia.org/wikipedia/commons/2/2b/Node-red-icon.png

2. Image d'un digital brain IA. Disponible à l'adresse : https://www.xconomy.com/wordpress/wp-content/images/2017/06/Digital-brain-AI-stock-Depositphotos-1100x733.jpg

3. Image du logo de MongoDB. Disponible à l'adresse : https://user.oc-static.com/upload/2017/09/08/1504876015749_MongoDB.png

4. Image du logo d'IBM Bluemix. Disponible à l'adresse : https://situationpublishing.com/wp-content/uploads/2016/12/IBMLogo.png

5. Image du logo de Twitter. Disponible à l'adresse : https://png.icons8.com/?id=60469&size=1600

6. Image du logo d'IBM. Disponible à l'adresse : http://pluspng.com/img-png/ibm-png-ibm-logo-png-transparent-4464.png

7. Images des guides de Node-RED. Disponible à l'adresse : http://noderedguide.com/

8. Images d'illustration du context et du global context. Disponible à l'adresse : https://nodered.org/docs/writing-functions

Annexe 1 : Guide d'installation

Table des matières

Table des figures

Introduction

Dans ce guide d'installation, nous parlerons uniquement de l'installation de Node-RED en local.

Une version existe aussi dans IBM Cloud, mais a des limites de stockage.

Installation de Node-RED

1. Installation de Node.JS

   Pour fonctionner, Node-RED utilise le principe des nodes et des flows grâce à NodeJS.

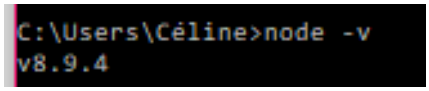
   L'installation se fait grâce à un package que l'on télécharge en suivant ce lien : https://nodejs.org/en/download/

   a. On télécharge la version adéquate à notre environnement :

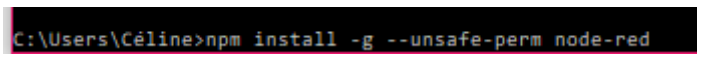
Figure 13 : Ecran de téléchargement de Node.JS

   b. On installe nodejs en suivant les étapes d'installation.

2. Vérification de l'installation et de la version de nodejs via notre console :

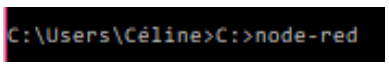


3. Installation de Node-RED :
   Pour cette installation, nous passons par les lignes de commandes :

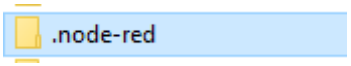


4. Création d'un dossier de stockage Node-RED :

   On crée un dossier pour stocker nos node-modules ainsi que nos flows.



   Cette commande va créer un nouveau dossier dans notre répertoire :



Dedans, on y trouve :

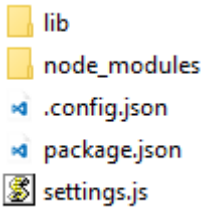


Figure 14 : Structure de notre répertoire .node-red

5. Lancement de Node-RED

   En passant par notre cmd, nous lançons une dernière commande qui permet de lancer Node-RED et d'avoir accès à notre éditeur graphique :

```
C:\Users\Céline>node-red
17 Feb 15:15:32 - [info]

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

17 Feb 15:15:32 - [info] Node-RED version: v0.18.3
17 Feb 15:15:32 - [info] Node.js  version: v8.9.4
17 Feb 15:15:32 - [info] Windows_NT 10.0.16299 x64 LE
17 Feb 15:15:35 - [info] Loading palette nodes
17 Feb 15:15:46 - [warn] ------------------------------------------------------
17 Feb 15:15:46 - [warn] [node-red/rpi-gpio] Info : Ignoring Raspberry Pi specific node
17 Feb 15:15:46 - [warn] [node-red/tail] Not currently supported on Windows.
17 Feb 15:15:46 - [warn] ------------------------------------------------------
17 Feb 15:15:46 - [info] Settings file  : \Users\Céline\.node-red\settings.js
17 Feb 15:15:46 - [info] User directory : \Users\Céline\.node-red
17 Feb 15:15:46 - [warn] Projects disabled : set editorTheme.projects.enabled=true to enable
17 Feb 15:15:46 - [info] Flows file     : \Users\Céline\.node-red\flows_OrdiHPCéline.json
17 Feb 15:15:46 - [info] Creating new flow file
17 Feb 15:15:46 - [info] Starting flows
17 Feb 15:15:46 - [info] Started flows
17 Feb 15:15:46 - [info] Server now running at http://127.0.0.1:1880/
```

6. Accès à Node-RED
   Pour accéder à notre Node-RED Editor, on se rend sur http://localhost:1880 :

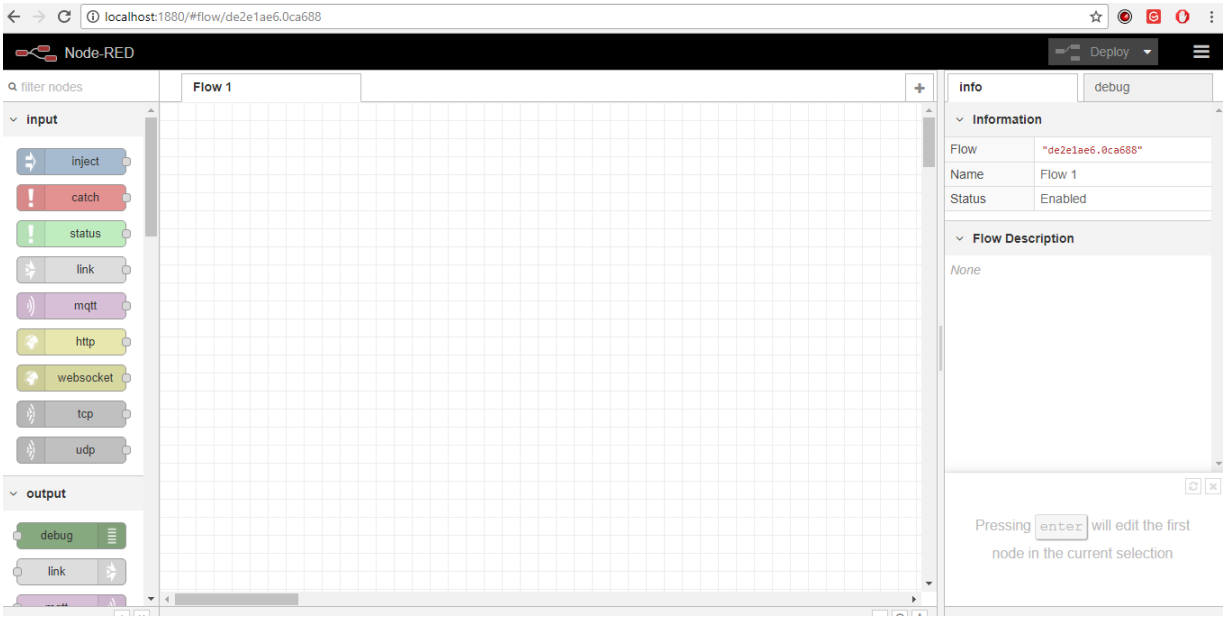


Figure 15 : Editeur Node-RED

Dans notre éditeur, une palette de base est déjà installée.

Installation des node-modules

En plus de la palette de base que Node-RED fournit, nous pouvons ajouter des node-modules supplémentaire, deux méthodes sont possibles pour cette installation :

Par ligne de commande

Par l'interface graphique de Node-RED

Pour cette deuxième option :

1. On se rend dans le menu :

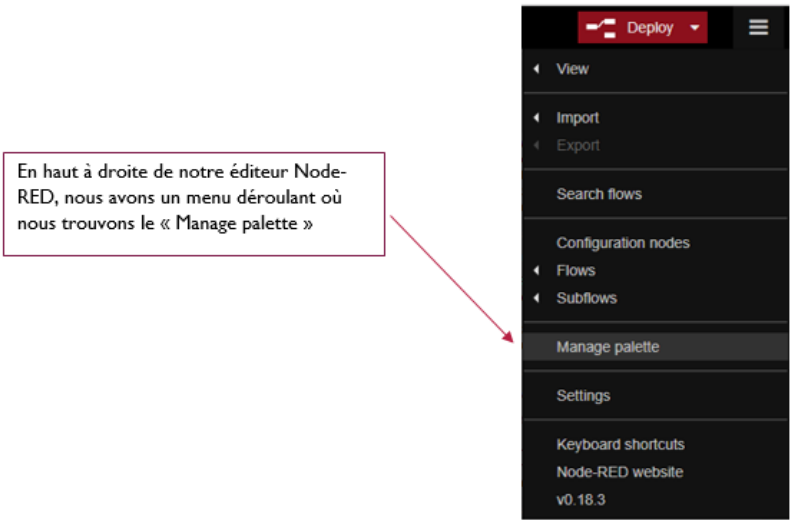


Figure 16 : Menu Node-RED

2. On a les nodes qui sont dans notre palette :

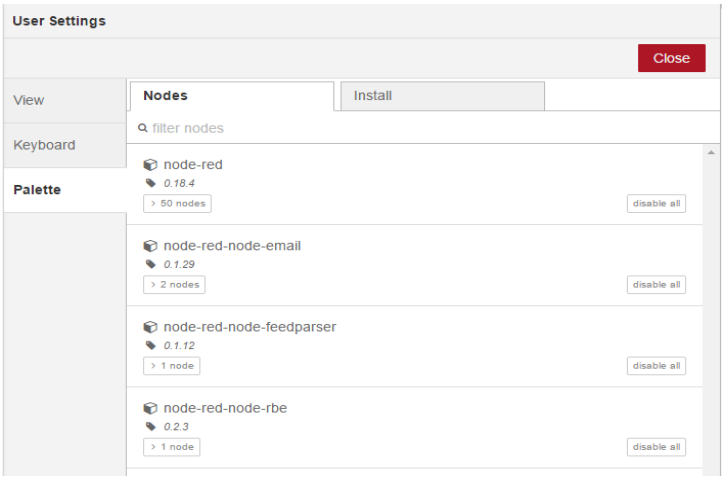


Figure 17 : Ecran "Manage Palette" avec les nodes installés

3. On a les nodes que l'on peut installer :

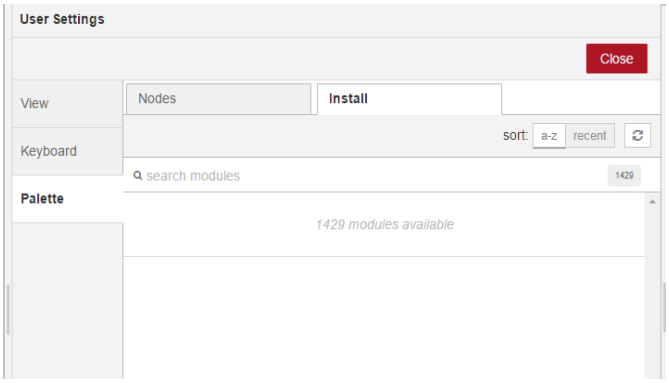


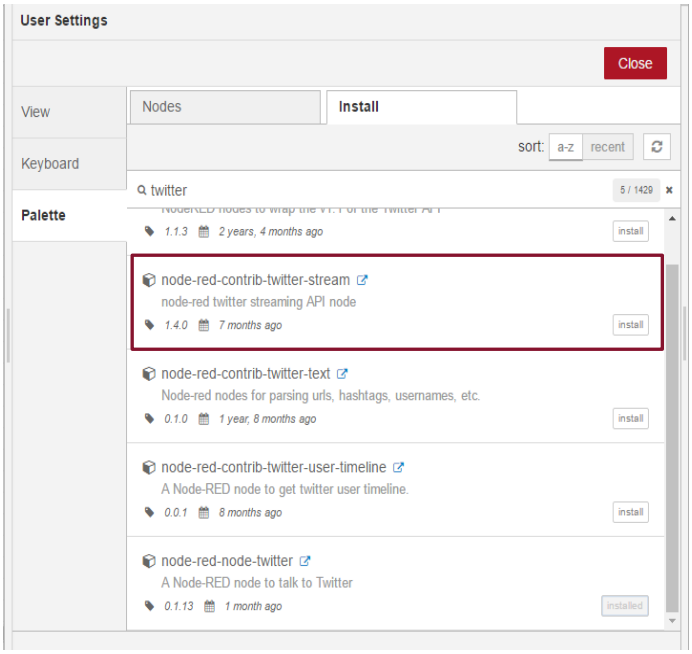
Figure 18 : Ecran "Manage Palette" avec les nodes disponibles

Twitter

Pour l'installation par ligne de commande :

```
C:\Users\Céline>npm install node-red-twitter
npm WARN saveError ENOENT: no such file or directory, open 'C:\Users\Céline\package.json'
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN enoent ENOENT: no such file or directory, open 'C:\Users\Céline\package.json'
npm WARN Céline No description
npm WARN Céline No repository field.
npm WARN Céline No README data
npm WARN Céline No license field.

+ node-red-twitter@1.0.5
added 3 packages in 3.02s
```

Pour l'installation par l'interface graphique :



Après l'ajout, dans notre palette :

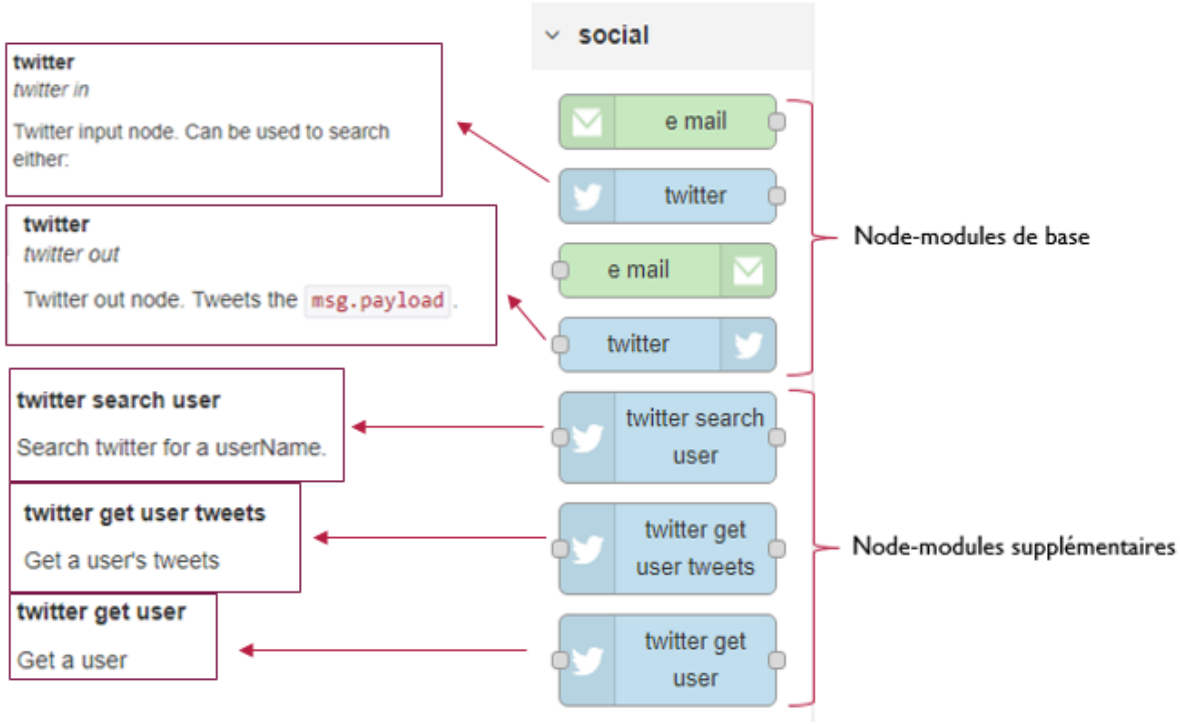


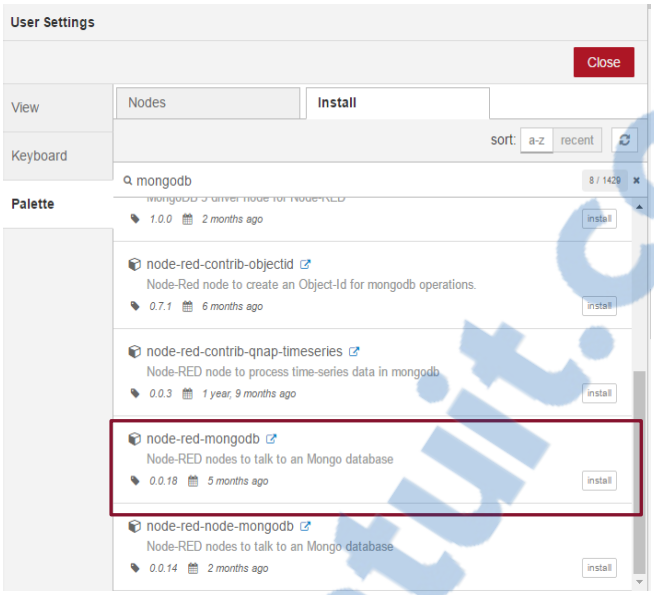
Figure 19 : Nodes Twitter

MongoDB

Pour l'installation avec les lignes de commande :

```
C:\Users\Céline>npm install node-red-mongodb
npm WARN saveError ENOENT: no such file or directory, open 'C:\Users\Céline\package.json'
npm WARN enoent ENOENT: no such file or directory, open 'C:\Users\Céline\package.json'
npm WARN Céline No description
npm WARN Céline No repository field.
npm WARN Céline No README data
npm WARN Céline No license field.

+ node-red-mongodb@0.0.18
added 17 packages in 9.673s
```

Pour l'installation sur l'interface graphique :



Dans notre palette :

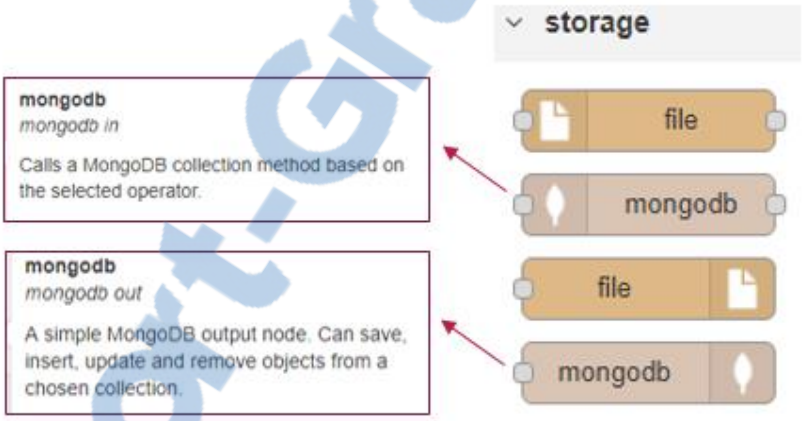
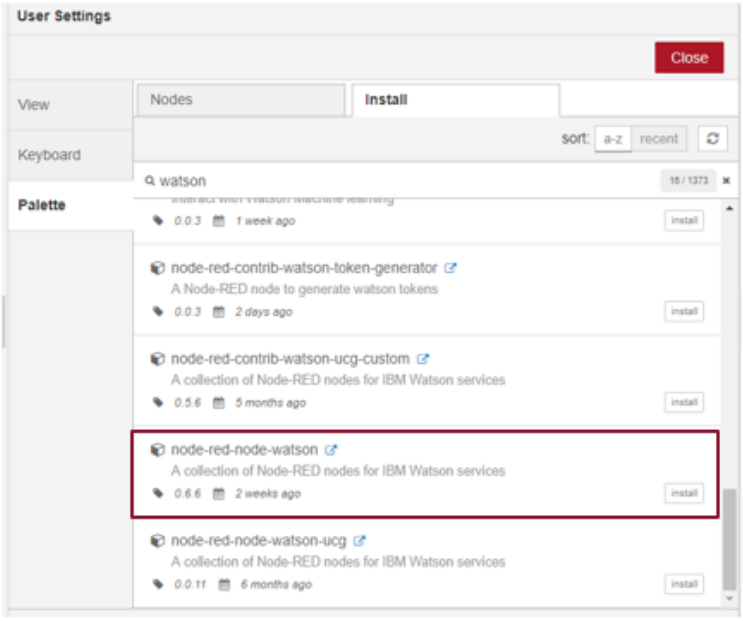


Figure 20 : Nodes MongoDB

IBM Watson

Pour les APIs d'IBM Watson, nous passerons que par l'interface graphique :



Dans notre palette :

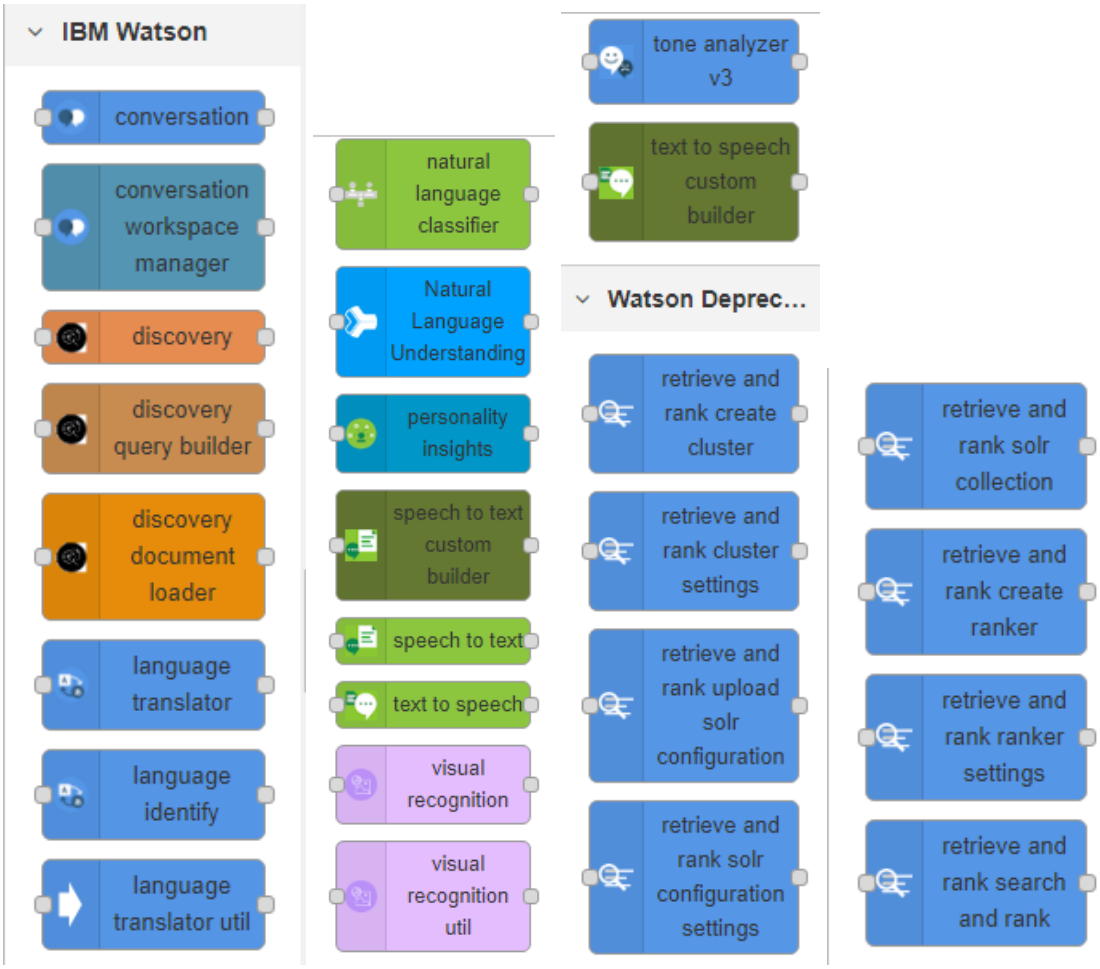


Figure 21 : Nodes IBM Watson

Ce sont tous les nodes disponibles d'IBM Watson dans Node-RED, pour plus d'informations sur les fonctionnalités des nodes voir dans les sources.

Installation de Studio 3T

Qu'est-ce que Studio 3T ?

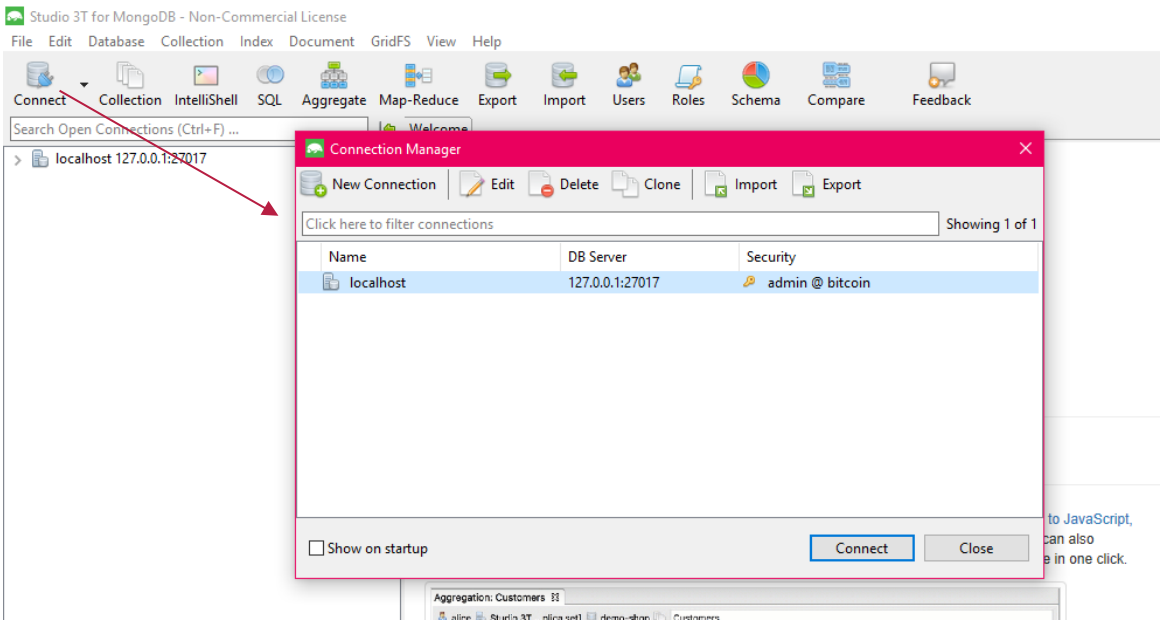
Sudio 3T est le GUI de MongoDB pour la consultation et la création de collections, users, bases. Cet outil nous permet aussi les exports en format JSON, CSV ou SQL.

Installation de Studio 3T

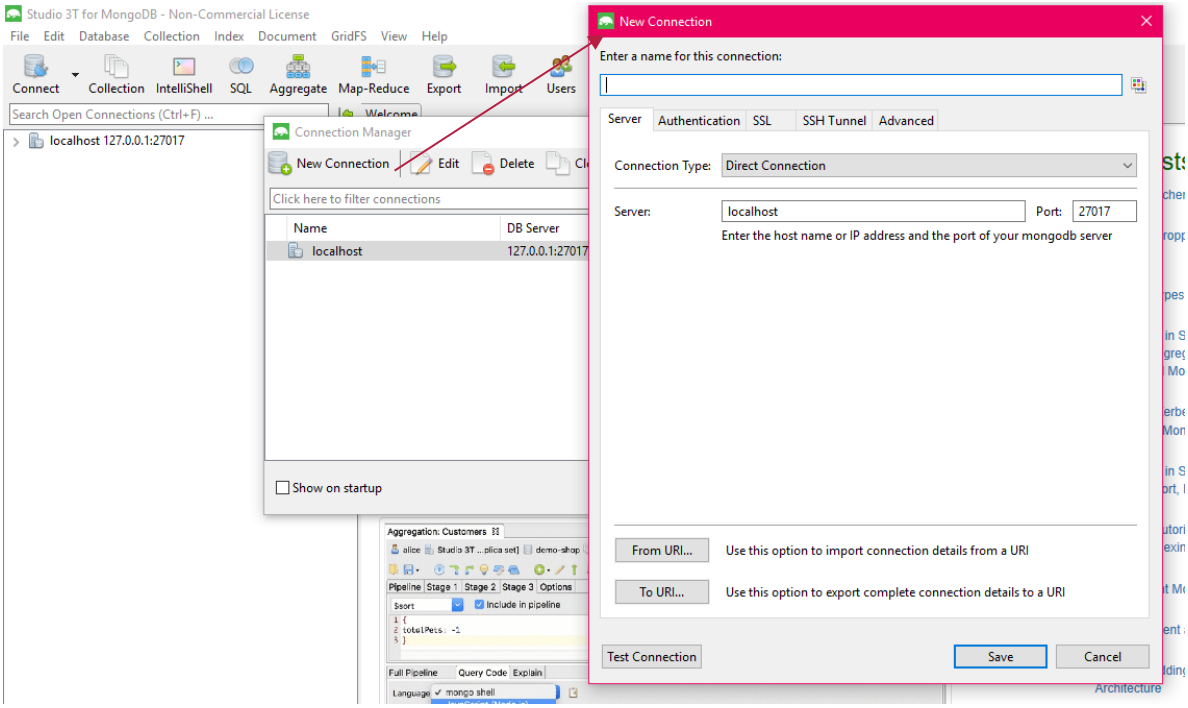
L'installation se fait en suivant ce lien : https://studio3t.com/ et en suivant les instructions d'installation.

Paramétrage pour l'utilisation avec Node-RED

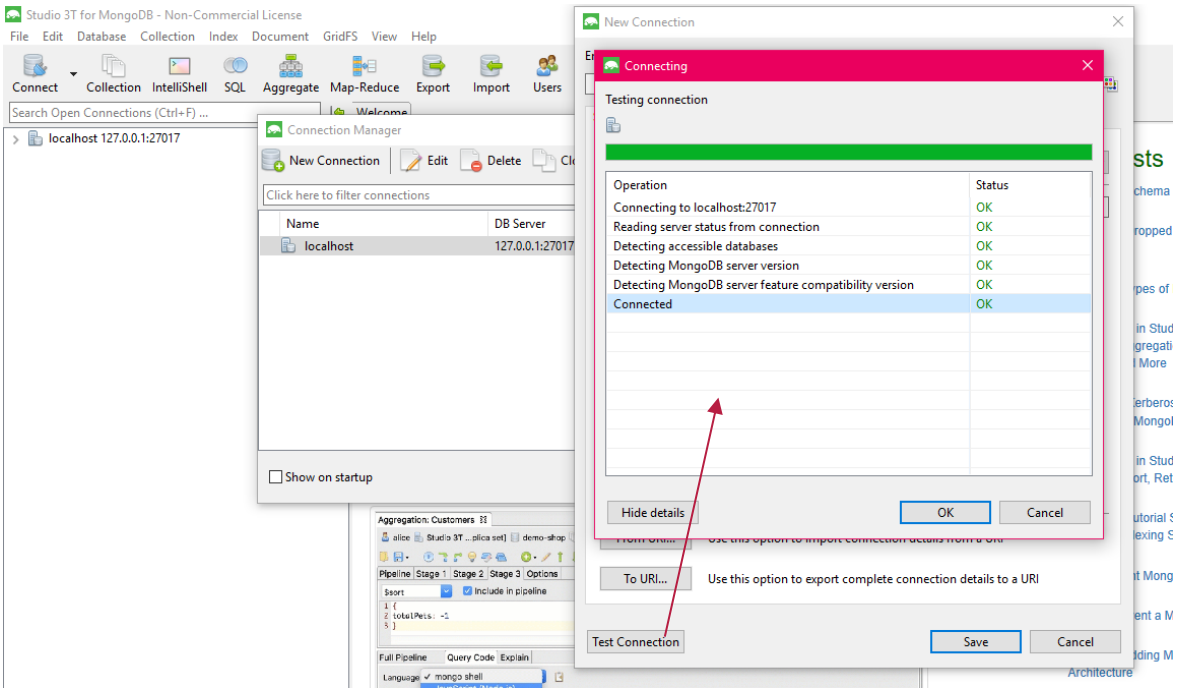
1. Création d'une nouvelle connexion au serveur
   En allant sur la page d'accueil et dans l''icône « Connect »



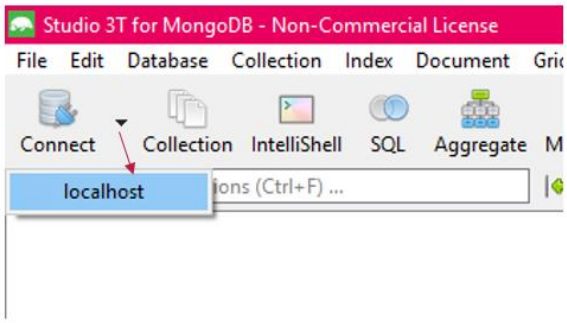
Création de la connexion avec notre ip local :



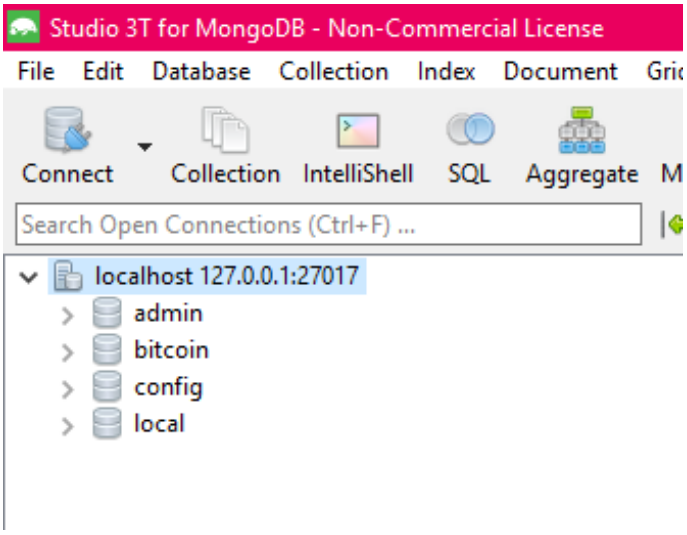
Test de la connexion :



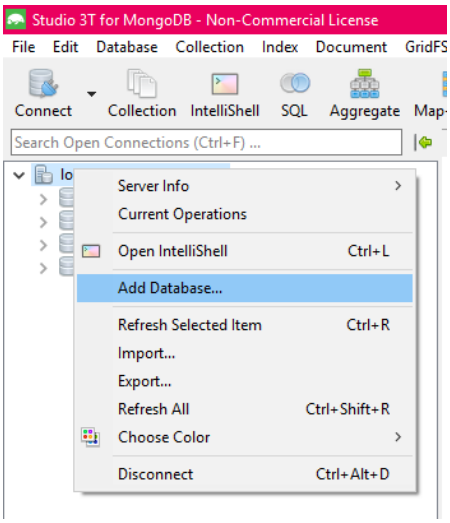
Connection à notre serveur en cliquant dessus :



Visualisation des différentes bases du serveur :



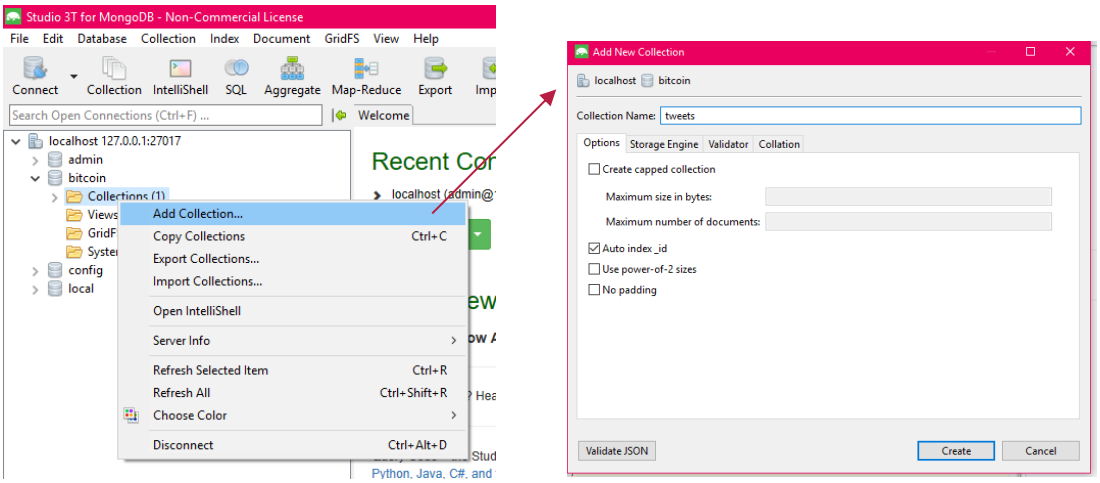
La base « bitcoin » a été ajoutée pour notre projet.
Pour ajouter une base de données, on fait un clic droit sur notre serveur et on choisit « Add Database » :



Après avoir cliqué, il faut donner un nom à notre base et valider.

2. Création de collections

La manipulation est la même que pour ajouter une base de données, un clic droit sur notre base et de séléctionner « Add Collection » :

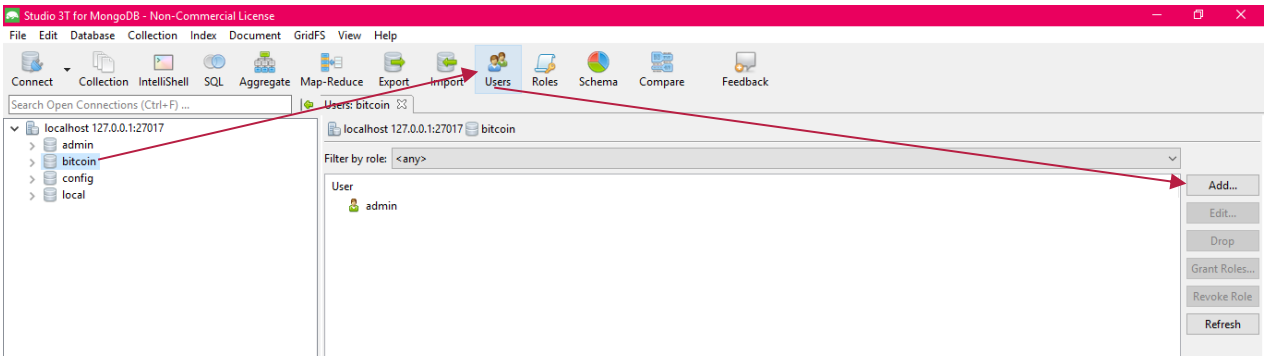


On met uniquement le nom de la collection avant de valider.

3. Création d'un user

La création d'un user sécurise une base de données et est essentiel pour se connecter dans Node-RED avec MongoDB.

On crée un user « admin » qui aura tous les droits. On séléctionne notre base « bitcoin », on clique sur « Users » et on clic sur « Add… ».



Dans les users, on peut aussi définir des rôles si on en a besoin pour notre utilisation.

Utilisation de Node-RED

Pour utiliser Node-RED, deux commandes sont impératives.

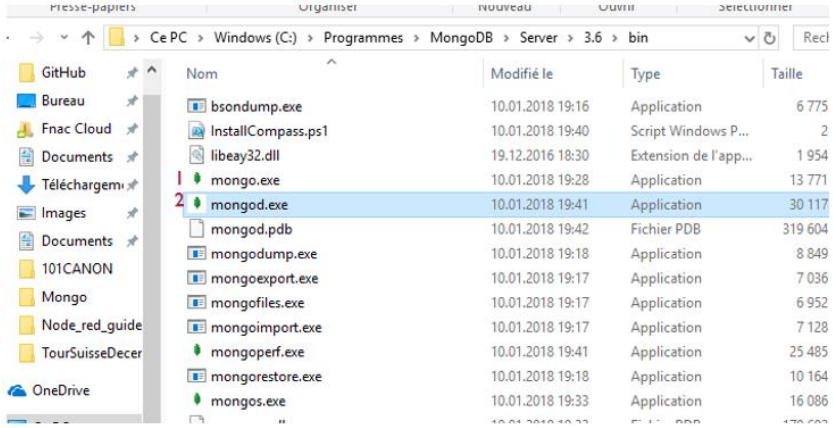
La première pour lancer Node-RED[1] et la deuxième pour lancer MongoDB :

Deux méthodes sont possibles :

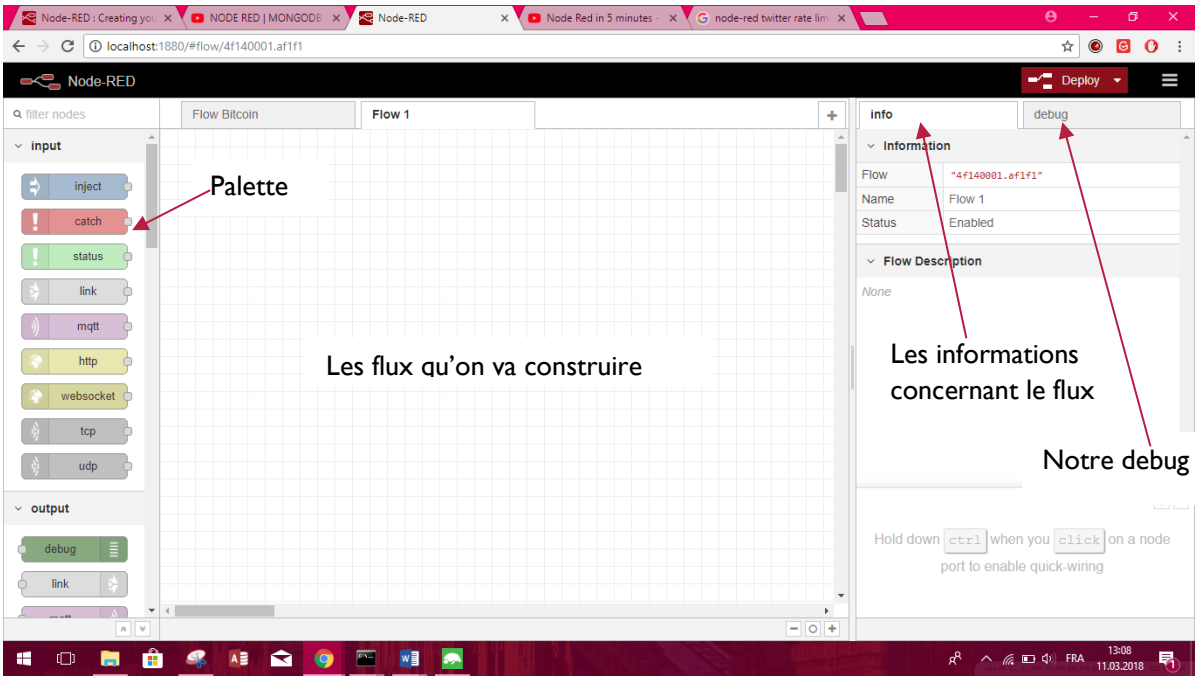
1. Par ligne de commande :
   On se met dans le bon répertoire, dans mon cas : « C:\Program Files\MongoDB\Server\3.6\bin » et :

```
C:\Program Files\MongoDB\Server\3.6\bin>mongod --dbpath C:\data\db
```

2. Par les exécutables de MongoDB (en suivant la numérotation) :

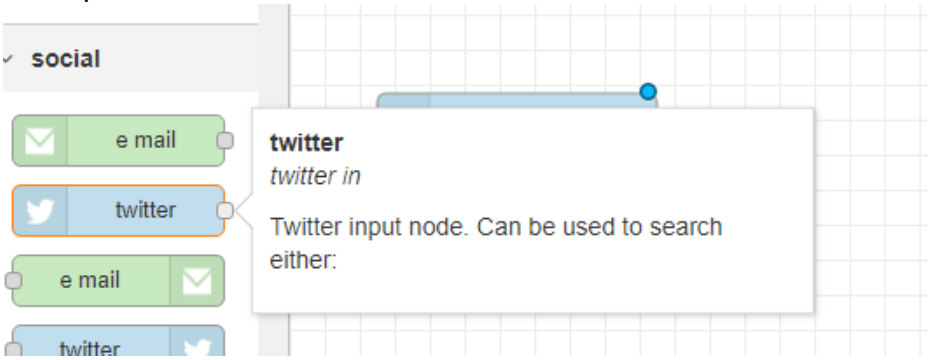


1. Les différents éléments qui composent notre éditeur :

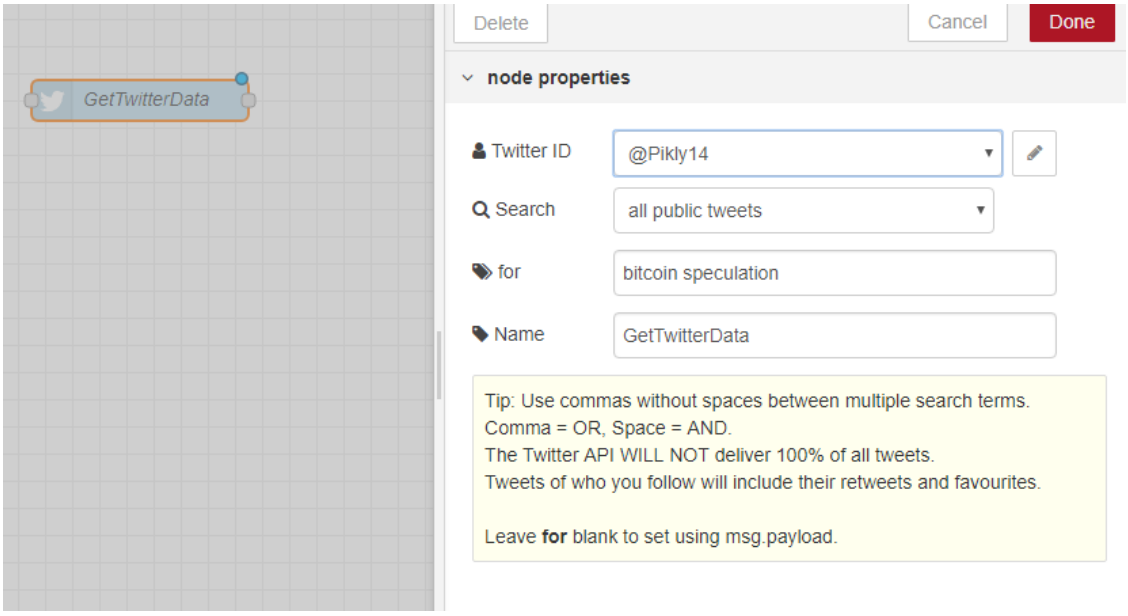


[1] Cf : Lancement de Node-RED

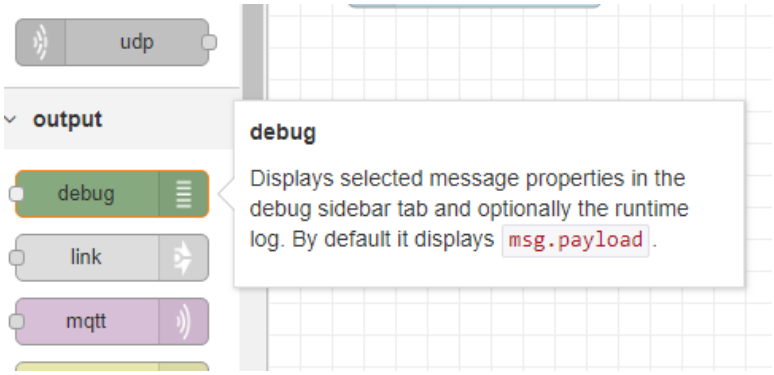
2. On insère un node Twitter en faisant un drag and drop sur notre flow.
On va prend le node Twitter suivant :



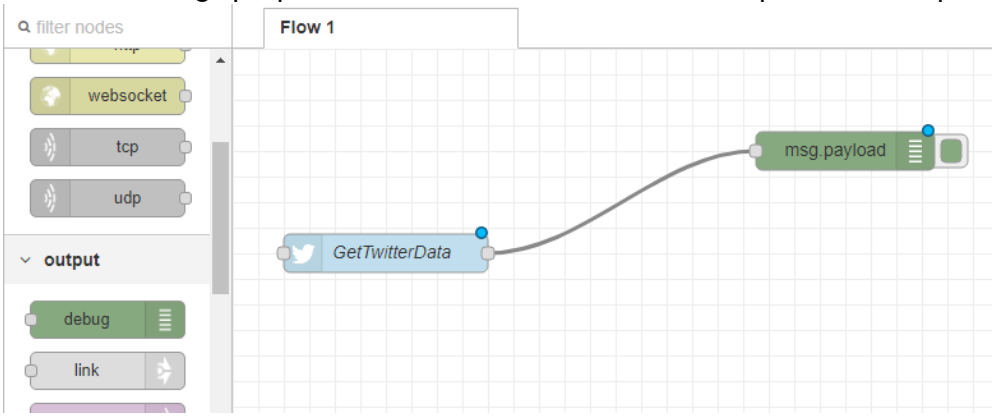
Pour utiliser le node, dans les paramètres, on se connecte à notre compte Twitter en cliquant sur le l'icône .
On spécifie qu'on veut recupérer tous les tweets, donc « all public tweets ».
On définit des mots clefs, ici « bitcoin » et « spéculation ». L'espace entre les mots-clefs signifie « et », la virgule signifie « ou ».
On nomme ensuite notre node.



3. On ajoute dans notre flow un node de debug pour avoir un visuel de nos tweets.

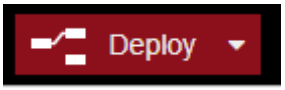


Ce node ne nécessite pas de paramétrage spécifique dans ce cas.
Dans l'interface graphique, on les relie de manière visuelle afin qu'ils communiquent.



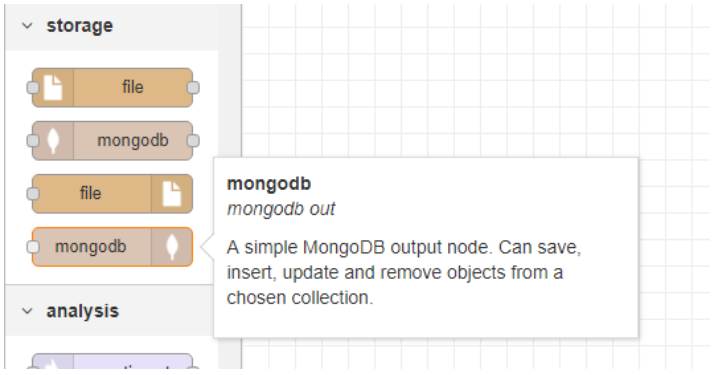
Les ronds bleus en haut de nos nodes signifient que le flow n'a pas été déployé depuis la dernière modification.

4. Déploiement du flow
Pour déployer le flow, on clic sur le bouton qui se trouve en haut à droite de l'écran :

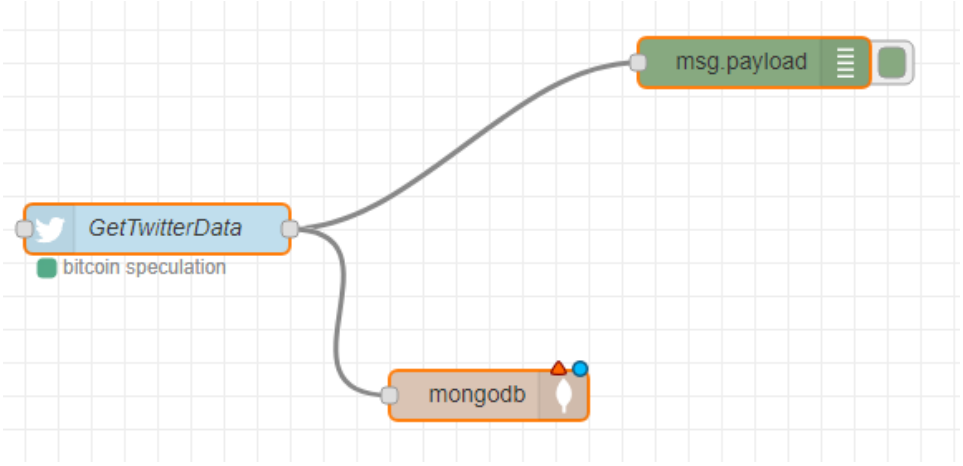


Si tout va bien on a le message ci-dessous et les petits ronds bleus qui se trouvaient sur nos nodes ont disparus :

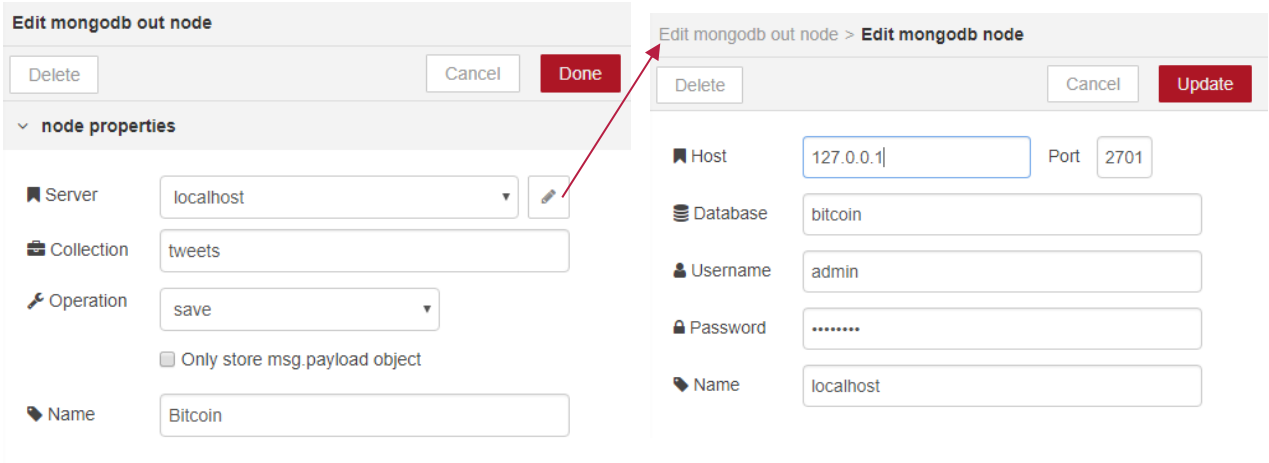
5. Dans notre flow, on va ajouter un node MongoDB pour stocker nos tweets. On prend le node suivant dans notre palette :



On relie graphiquement nos nodes sur notre interface :

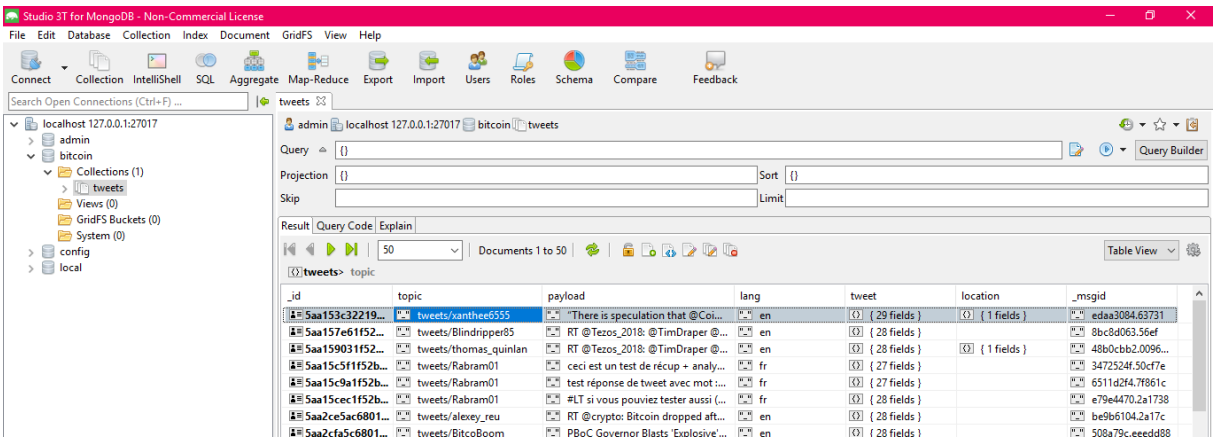


On va paramétrer notre node MongoDB :
On ajoute la connexion à notre serveur (à droite) avec le user et la base de données qu'on a créé puis on indique la collection où on va insérer nos données (à gauche).

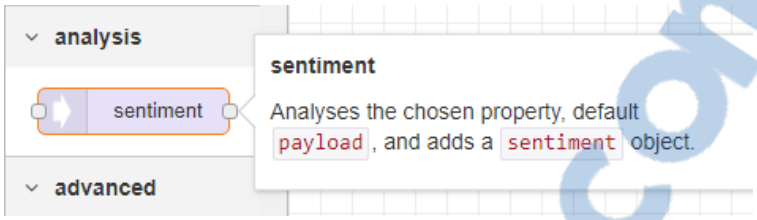


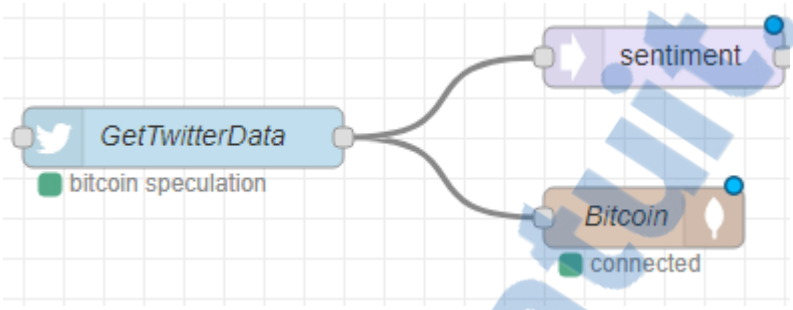
Après ce paramétrage, on déploie notre flow.

Dans l'interface graphique, on a toujours un visuel des tweets, mais notre base de données se remplit aussi :



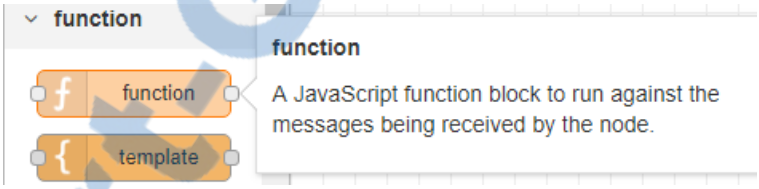
6. On peut ensuite ajouter un node « Sentiment Analysis » qui nous permet de pouvoir mettre un score global sur le tweet :

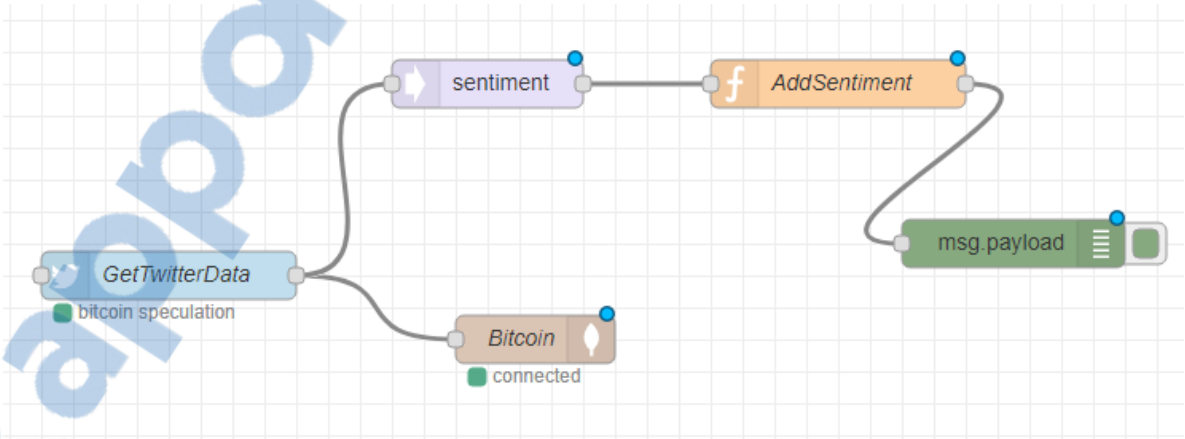




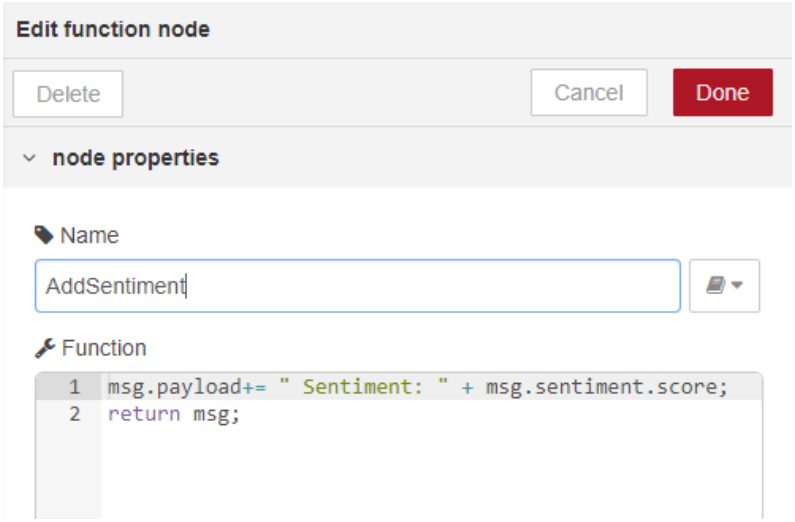
Ce node n'a pas besoin de paramétrage.

7. On ajoute un node function pour formater le message de retour en ajoutant le score en plus du tweet passé :





On ajoute le code suivant dans le paramétrage du node :



On ajoute au payload qui est passé dans le message le score du node « Sentiment Analysis ».

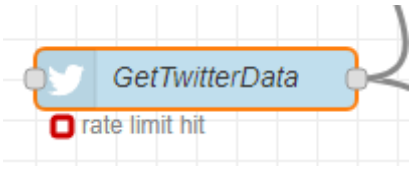
8. On déploie notre modèle et on a le visuel de nos tweets et le score attribué dans notre interface :

On retrouve les mêmes tweets dans la base de données :

_id	topic	payload	lang	tweet	location	_msgid
5aa2dfe1c...	tweets/Line_TB_Test	Love bitcoin speculation, test	en	{ 27 fields }		40700a55.932c6
5aa2e010c...	tweets/Line_TB_Test	Hate speculation bitcoin test	en	{ 27 fields }		b2dd27c.73023..
5aa2e1e0c...	tweets/BitcoBoom	PBoC Governor Blasts 'Explosive' Crypto Speculation - https://t.co/sHByHENkaq - G...	en	{ 28 fields }		c21ad40d.d81c..
5aa2e5e2c...	tweets/007_Xia	@boxmining @demian_wilhelm I would say good news, because people keeps conf...	en	{ 29 fields }	{ 1 fields }	5b4e32df.5d41a
5aa2effec6...	tweets/hakadonkiria	Up one level\nBitcoin atm crook county oregon we are your resource for full\nInfo ...	en	{ 28 fields }	{ 1 fields }	59de1100.8c30a
5aa2f252c...	tweets/BitcoinEdu	@rchllwlsn Bitcoin is a volatile currency. It will rise and fall. Nothing like Bitcoin has ...	en	{ 28 fields }	{ 1 fields }	58410ee6.7253b
5aa2f2aec6...	tweets/mindstatex	@sashandiggers @Egon_01 We are stuck at stage 1 because the focus has become ...	en	{ 29 fields }	{ 1 fields }	313b3300.e4caa
5aa2f793c...	tweets/chug	Le BITCOIN: Arnaque ou révolution? https://t.co/eH5CyKdkor\nSortir du storytellin...	fr	{ 29 fields }		7036990d.4aea4
5aa2fcdec6...	tweets/paperboynewsy	RT @tradebuddyonlin: (Gemini Could Finally Support Litecoin and Bitcoin Cash Aft...	en	{ 28 fields }	{ 1 fields }	be31ef03.16415
5aa2fd12c...	tweets/Vancity888	@steveshmidt @AdamPOlsen @BCGreens You really said that? Prices are unnaturall...	en	{ 29 fields }	{ 1 fields }	a583be51.ddd4
5aa2fe2fc6...	tweets/JonathanCCook	The old fruit warehouses of eastern Washington state are good for \n1) bitcoin mini...	en	{ 33 fields }	{ 1 fields }	8620bb8.438dd..
5aa30251c...	tweets/MichaelQuintonP	Cryptos are not an investment like real property or gold, they are %100 speculation ...	en	{ 29 fields }	{ 1 fields }	acb1fa9e.3f2d3
5aa30641c...	tweets/CryptoChaoss	"I learned early there is nothing new in Wall Street. There can't be, speculation is as ...	en	{ 30 fields }	{ 1 fields }	7aa2db58.b8a9..
5aa307c1c...	tweets/CryptoChaoss	"I learned early there is nothing new in Wall Street. There can't be, speculation is as ...	en	{ 30 fields }	{ 1 fields }	f701b1bb.6dc2c
5aa3084cc...	tweets/CryptoTrackerBt	$RKFHF - 6-K - Important when you invest in growth companies!\nC.an\nI finish thi...	en	{ 29 fields }	{ 1 fields }	3fe5869a.0c5e2
5aa30a4bc...	tweets/HectorRed_	RT @CryptoChaoss: "I learned early there is nothing new in Wall Street. There can't ...	en	{ 28 fields }		9de8104f.568b2
5aa30a7bc...	tweets/NewsForBitCoin	Bitcoin speculation continues to surge https://t.co/fTAjUJGgEV	en	{ 28 fields }		cfdd6d12.63957

Problèmes de Node-RED

« Rate limit hit »



Ce problème est dû aux quotas que Twitter nous autorise à récupérer. Quand ce problème arrive, on attend et le problème se résoud.

Les APIs Twitter ont des limitations en termes de nombre de requêtes par période de temps ou en termes du nombre de résultats. Par exemple on a le droit à 450 recherches par période de 15mn, après le compteur est remis à zéro et on le droit à 3200 tweets maximum dans un historique d'utilisateur.

Sources

Sites internet

1. NODE-RED. *Installation de Node-RED* [en ligne]. [Consulté le 22 décembre 2017]. Disponible à l'adresse : https://nodered.org/docs/platforms/windows

2. SENSETECNIC, *Tutorial : use the Watson cognitive service with FRED (Cloud Node-RED)* [en ligne]. [Consulté le 28 mars 2018]. Disponible à l'adresse : http://developers.sensetecnic.com/article/tutorial-use-the-watson-cognitive-service-with-fred/

3. NODE-RED. *node-red-node-watson* [en ligne]. [Consulté le 10 mars 20018]. Disponible à l'adresse : https://flows.nodered.org/node/node-red-node-watson

4. GITHUB. *node-red-labs* [en ligne]. [Consulté le 10 mars 2018]. Disponible à l'adresse : https://github.com/watson-developer-cloud/node-red-labs

5. Twitter. *Rate limits* [en ligne]. [Consulté le 15 février 2018]. Disponible à l'adresse : https://developer.twitter.com/en/docs/basics/rate-limits

6. NODE-RED. *node-red-node-twitter* [en ligne]. [Consulté le 19 janvier 2018]. Disponible à l'adresse : https://flows.nodered.org/node/node-red-node-twitter

7. NODE-RED. *node-red-contrib-twitter* [en ligne]. [Consulté le 19 janvier 2018]. Disponible à l'adresse : https://flows.nodered.org/node/node-red-contrib-twitter

8. NODE-RED. *node-red-node-mongodb* [en ligne]. [Consulté le 19 janvier 2018]. Disponible à l'adresse : https://flows.nodered.org/node/node-red-node-mongodb

9. NODE-RED. *node-red-node-watson* [en ligne]. [Consulté le 19 janvier 2018]. Disponible à l'adresse : https://flows.nodered.org/node/node-red-node-watson

10. STUDIO 3T. *Studio 3T* [en ligne]. [Consulté le 10 janvier 2018]. Disponible à l'adresse : https://studio3t.com/

11. IBM. *Internet of Things – IoT* [en ligne]. [Consulté le 9 mars 2018]. Disponible à l'adresse : https://www.ibm.com/cloud/internet-of-things

12. NODEJS. *Downolods* [en ligne]. [Consulté le 22 décembre 2018]. Disponible à l'adresse : https://nodejs.org/en/download/

Annexe 2 : Document de vision

Historique des révisions

Date	Version	Description	Auteur
27.11.17	1.0	Création du document	Céline Paniz
07.12.17	1.1	Continuation de l'écriture du document	Céline Paniz
15.12.17	1.2	Continuation et finition de l'écriture du document	Céline Paniz
03.01.18	2.0	Reprise du document	Céline Paniz
07.01.18	2.1	Continuation de la reprise du document	Céline Paniz
05.05.18	2.2	Finition du document et vérification	Céline Paniz

Table des matières

1. Introduction

1.1 Objectif

Ce document a pour but de mettre en avant les objectifs du projet. Il mettra en avant les différents éléments qui entrent en ligne de compte pour l'établissement du projet. Nous aborderons donc les fonctions principales de l'application à venir ainsi que les différentes parties prenantes à ce projet.

Ce projet aura pour but d'analyser les sentiments des internautes concernant les opportunités d'investissement dans le Bitcoin été en particulier l'analyse du sentiment vis-à-vis de la hausse ou de la baisse supposée du cours de cette crypto-monnaie.

1.2 Portée

Application permettant la vision des spéculations des internautes sur Twitter grâce à un API cognitif d'IBM Watson.

1.3 Définitions, acronymes et abréviations

IBM Bluemix : Plateforme Cloud[1]

API : Application Programming Interface[2]

1.4 Références

- Liste des différentes APIs cognitifs d'IBM Watson

1.5 Présentation

Le contenu du document de vision est structuré en différentes parties, tout d'abord, nous aborderons le positionnement du produit à venir, ensuite les utilisateurs ainsi que les intervenants du projet. Finalement, nous parlerons du produit en lui-même, ses fonctions, ses contraintes, le tout avec des annexes explicatifs pour illustrer le projet.

2. Positionnement

2.1 Opportunités métier

Dans de nombreux établissements financiers tels que les banques, les employés aimeraient savoir si investir dans le Bitcoin est une bonne optique dans le but de pouvoir orienter leurs clients dans leurs investissements. Bien que partant de l'investissement, il aurait aussi dans l'idée de pouvoir connaître l'avis de la majorité, sachant que si la majorité des internautes parlent d'une baisse du cours, il serait aussi judicieux de se retirer. De plus, la vision des spéculations à venir serait une plus-value, bien qu'étant totalement spéculative, l'avis de la majorité peut souvent s'avérer correct. Ce serait une opportunité dans ce domaine-là, car cet outil permettrait d'aider les employés de banque dans leur travail, et même les orienter eux-mêmes s'ils décident d'investir personnellement.

2.2 Expression du problème

Actuellement, se faire un avis quant à la hausse ou la baisse du Bitcoin est d'une trop grande complexité. La création d'un outil qui permettrait d'analyser le sentiment des internautes quant à cette variation permettrait aux différents acteurs investissant dans cette crypto-monnaie de se faire un avis et de pouvoir investir ou de retirer son argent selon les résultats des analyses quant à la tendance actuelle. De nombreuses personnes tentent maintenant d'investir dans différentes crypto-monnaie, mais la

[1] Pour plus d'information : https://openclassrooms.com/
[2] Pour plus d'information : https://openclassrooms.com/

problématique reste la même, personne ne sait si l'investissement serait judicieux ou non. Avoir une vision d'une partie de la population très investie sur les variations du cours et prédisant des hausses ou des baisses serait un avantage non négligeable, bien que les sources ne soient que spéculatives.

Le problème	Un manque de vision quant à l'avis d'une population sur la variation du Bitcoin
Affecte	Les employés de banque dans l'orientation des clients pour leurs investissements et eux-mêmes dans leurs investissements personnels
L'impact du problème est	Il est difficile de pouvoir prédire la montée ou la descente du Bitcoin de soi-même
Une solution satisfaisante serait	Créer une application qui permettrait la vision des spéculations de nombreux internautes sur Twitter afin d'avoir un vision de la majorité

2.3 Descriptif du positionnement du produit

Pour	- Les employés de banque - Toute personne intéressée à investir - Toute personne ayant investis mais voulant savoir si investir plus ou se retirer
Qui	Aimerait avoir un avis d'une communauté concernant leurs spéculations quant à l'avenir (à court terme) de la variation du cours du Bitcoin
Le produit	Une application
Qui	Permettra une vision des spéculations facilement
A la différence de	D'un avis personnel
Notre produit	Permettra un sentiment de la tendance des avis/spéculations des internautes de Twitter

3. Description des parties prenantes et des utilisateurs

3.1 Données démographiques du marché

Les banques qui pourraient être intéressé pour orienter leur clientèle (s'ils décident de faire investir leurs clients dans ce nouveau marché), ainsi que les particuliers désirant avoir le sentiment de la tendance actuelle avant d'investir dans le Bitcoin ou de retirer leurs investissements si les spéculations émises sont à la baisse.

3.2 Récapitulatif des utilisateurs

Nom	Description	Rôle
Potentiel investisseur dans le Bitcoin	Futur utilisateur de l'application	• Décrire le besoin qu'il a vis-à-vis de la vision des flux qu'il aimerait pouvoir visualiser • Avoir une vision résumée de la tendance des spéculations émises sur Twitter

3.3 Environnement de l'utilisateur

3.4 Profil d'utilisateur

Description	Futur utilisateur de l'application
Type	Personne qui s'intéresse aux nouvelles technologies ainsi qu'à l'investissement
Responsabilités	Décrire le besoin du type d'information nécessaire à visualiser
Critère de succès	Application qui est utile au besoin
Implication	Pas d'implication particulière dans le développement mise à part une description de sa vision de l'application future
Livrables	Pas de livrables définis

3.5 Besoins principaux de la partie prenante ou de l'utilisateur

Besoin (métier)	Priorité	Concerne	Solution actuelle	Solutions proposées
Avoir un aperçu résumé de la vision globale des internautes, de la tendance émise par les internautes	**1**	**Utilisateur**	**Aucune**	**Application**

4. Présentation du produit

4.1 Perspective du produit

Le produit est une application utilisant des APIs cognitifs des composants de l'IA d'IBM Bluemix. Cette application permettra une visualisation globale des avis des internautes concernant le futur du cours du Bitcoin. Cette application ne se base que sur des spéculations, mais pourra apporter une vision différente à l'utilisateur quant aux différents points de vus existants et en ressortir l'avis majoritaire. Ce produit aura le but de mettre en avant la tendance actuelle de ce que les internautes postent et de pouvoir en ressortir la hausse ou la baisse du cours du Bitcoin après analyse des tweets.

4.2 Récapitulatif des fonctions

Avantage pour l'utilisateur	Caractéristiques correspondantes
Vision claire et concise des spéculations faites par les utilisateurs de Twitter	Utilisation des APIs d'IBM Watson pour pouvoir ressortir les résultats par rapport au flux Twitter récupérée.

4.3 Hypothèses et dépendances

Les différentes hypothèses qui pourraient faire en sorte que le document de vision soit modifié sont les suivantes :

- Si le flux Twitter ne peut pas être récupérer en temps réel :
  - On ne sera plus en temps réel, mais il aura un décalage quant aux tweets récupérés, ce qui pourrait entraîner le fait que les utilisateurs n'aient pas une vision correcte au niveau temporel.
- Si le besoin des futurs utilisateurs se modifie :
  - Les fonctions ainsi que la perspective du produit se modifieraient aussi

4.4 Coût et prix

Aucun coût ni prix ne sera défini, car ce projet se place dans le cadre d'un travail de Bachelor pour la HEG.

5. Fonctions du produit

5.1 Visualisation des spéculations concernant la hausse ou la baisse du Bitcoin en temps réel

L'utilisateur aura la possibilité de pouvoir consulter à tout moment l'avis de la communauté Twitter concernée par la tendance sur le cours du Bitcoin.

Cette fonction permettra une vision de l'avis général des internautes et pourra aider à une prise de décision concernant un futur investissement dans cette crypto-monnaie. Tout en sachant que les données évaluées ne sont que spéculatives et ne sont pas à prendre comme une vérité absolue, mais peut aider si la majorité pense comme le futur investisseur. La mise en avant d'une tendance aidera à la prise de décision.

6. Contraintes

Les contraintes actuelles sont la mise en place d'une base de données ayant la capacité de pouvoir récupérer des tweets en temps réel sans être surchargée.

De plus, l'utilisation des APIs d'IBM est limitée selon différents degrés et pourraient entraîner un problème si le projet prend plus d'ampleur avec le temps.