



# Documentation technique

## Administration base de données

Jérémy RICHARD, Louis ARDILLY et Clément RAMOS LAGE

# Sommaire

I. Contexte	<b>3</b>
II. Introduction	<b>3</b>
III. Outils utilisés	<b>4</b>
IV. Architecture du projet	<b>7</b>
V. Alternatives considérées	<b>9</b>
VI. Flux de données	<b>10</b>
VII. Données brutes	<b>12</b>
VIII. Architecture des bases de données	<b>14</b>
IX. Interface	<b>15</b>
X. Évolution du projet	<b>16</b>

# I. Contexte

Ce document est une documentation technique mise en place dans le cadre du projet de fin de module intitulé "Administration base de données".

Dans le cadre du projet nous devons réaliser un projet par groupe de 3 sur le thème de notre choix (validé par l'enseignant) avec les contraintes suivantes :

- Au moins deux sources de données API différentes
- Au moins une source de données en temps réelle (plusieurs données par heure) - Utilisation de deux paradigmes de base de données différents (un seul est possible mais il faudra bien le motiver)
- Déploiement d'un système HDFS pour stocker des données non organisées (images, vidéos, longs textes, pages html...)
- Le système de base de données mis en place doit pouvoir permettre une récupération de toute les données en temps réel (réponse en maximum 1 seconde, idéalement 100ms)

Ce document est un élément de rendu obligatoire, il sera aussi un document de référence dans la description du projet.

## II. Introduction

Pour la réalisation de ce projet nous avons choisi de nous orienter vers l'analyse des cryptomonnaies. Nous avons choisi ce sujet car l'apparition des cryptomonnaies est une vague qui déferle sur le monde moderne et nous ne pouvons pas la laisser passer. Ce projet est l'occasion pour nous d'accroître nos compétences dans ce domaine.

Avant toute chose qu'est-ce qu'une cryptomonnaie ? Une cryptomonnaie est une devise numérique décentralisée, qui utilise des algorithmes cryptographiques et un protocole nommé blockchain pour assurer la fiabilité et la traçabilité des transactions. Les cryptomonnaies sont entièrement virtuelles, elles peuvent être stockées dans un portefeuille numérique protégé par un code secret appartenant à son propriétaire. Des plateformes d'échanges (Binance, Coinbase, Bitstamp, etc.) servent à acheter et revendre de la cryptomonnaie en ligne.

L'univers complexe des cryptomonnaies est aussi un sujet de discorde, les sceptiques sont nombreux, une meilleure compréhension du sujet nous permettra de donner un avis plus éclairé sur le sujet.

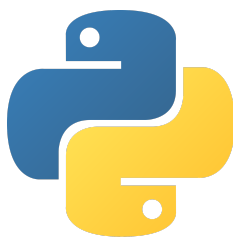
Depuis maintenant plusieurs années, les cryptomonnaies se démocratisent et prennent une grande place dans l'économie internationale. Des pays comme le Salvadore en ont même déjà fait leur monnaie nationale.

Nous avons donc choisi de mettre en place une architecture de type Big-Data pour récupérer, traiter et stocker nos données. Nous allons donc travailler avec deux flux de données, un flux numérique et un flux textuel.

Les données seront récupérées à l'aide d'API et de scrapping suite à cela elles seront traitées à l'aide de différents outils et stockées à l'aide d'une base de données NoSQL.

### III. Outils utilisés

Pour la réalisation de ce projet différents outils sont utilisés, principalement utilisés dans des projets Big-Data chaque outil à son utilité.



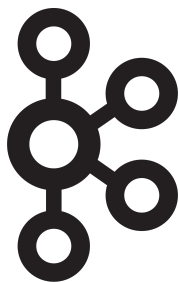
**Python:** C'est un langage de programmation très utilisé dans le domaine de la data.

Premier point très important, python est disponible sur l'ensemble des systèmes d'exploitation, de plus ce langage dispose d'une multiplicité de package et de librairies qui répondent aux différents besoins du

développeur.

Et pour finir Python dispose d'une communauté très active facilitant la recherche d'information ou la correction d'erreur.

Dans notre cas nous avons choisi python pour sa vitesse d'exécution mais aussi et surtout pour les différentes librairies disponibles. De plus, python est un langage que l'ensemble du groupe apprécie et connaît, une veille n'était donc pas nécessaire et cela nous a fait gagner du temps.



**Apache Kafka:** C'est une plateforme distribuée de diffusion de données en continu, capable de publier, stocker, traiter et souscrire à des flux d'enregistrement en temps réel. Elle est conçue pour gérer des flux de données provenant de plusieurs sources et les fournir à plusieurs utilisateurs. En bref, elle ne se contente pas de déplacer un volume colossal de données d'un point A à un point B : elle peut le faire depuis n'importe quels points vers n'importe quels autres points, selon vos besoins et même simultanément. Dans notre cas Kafka permet la diffusion de nos données en continu et nous permet de gérer la diffusion de nos données provenant de plusieurs sources vers un seul point.

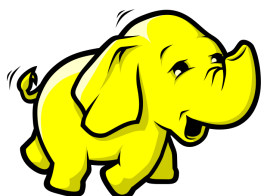
Kafka nous permet aussi de déplacer de gros volume de données rapidement et nous fait donc gagner un temps considérable de diffusion et de pré-traitement.



**MongoDB:** C'est un système de base de données NoSQL orientée document. Elle fait partie du paradigme Document oriented. C'est un système de BDD complet et complexe. Les données sont stockées sous forme de collections et de documents.

MongoDB présente plusieurs avantages majeurs. Tout d'abord, cette base de données NoSQL orientée document se révèle très flexible et adaptée aux cas d'usage concrets d'une entreprise.

Notons aussi la possibilité de créer des index pour améliorer la performance des recherches. N'importe quel champ peut être indexé. MongoDB est un choix pour nous pertinent car notre BDD est vouée à évoluer avec l'ajout de traitement supplémentaire sur nos données, de plus concernant les données textuelles le volume peut très fortement évoluer sur une courte période en fonction de l'actualité.



**Hadoop:** Hadoop est un framework logiciel dédié au stockage et au traitement de larges volumes de données. Les systèmes de données Hadoop ne sont pas limités en termes d'échelle, ce qui signifie qu'il est possible d'ajouter davantage de hardware et de clusters pour supporter une charge plus lourde sans passer par une reconfiguration ou l'achat de licences logicielles onéreuses.

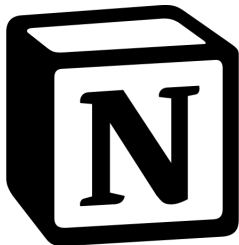
Dans notre cas, Hadoop sera implémenté à notre architecture principalement pour stocker de gros volumes de données. Nous

stockerons sur hadoop l'ensemble de nos données textuelles après leur traitement et l'analyse de sentiment. Cela nous permettra de ne pas surcharger notre BDD mongoDB car nos données textuelles ne seront pas utilisées à des fins d'affichage mais simplement utilisées pour l'analyse de sentiment.



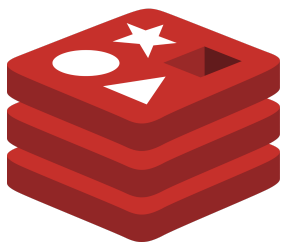
**GitFlow:** C'est un modèle de branching Git alternatif qui utilise des branches de fonctionnalité et plusieurs branches primaires.

Dans notre cas, le modèle gitflow nous a permis d'organiser le développement de notre projet. De plus, cela nous permet de voir en temps réel les différentes branches et fonctionnalités en cours de développement.



**Notion:** C'est une application de prise de notes. Dans notre cas l'application notion a été très intéressante dans la réalisation de notre projet. Cette application nous a permis de prendre en note l'ensemble de nos idées pour structurer notre projet, mais elle nous a aussi permis de mettre en place une todo app nous permettant de définir l'ensemble des tâches à faire afin

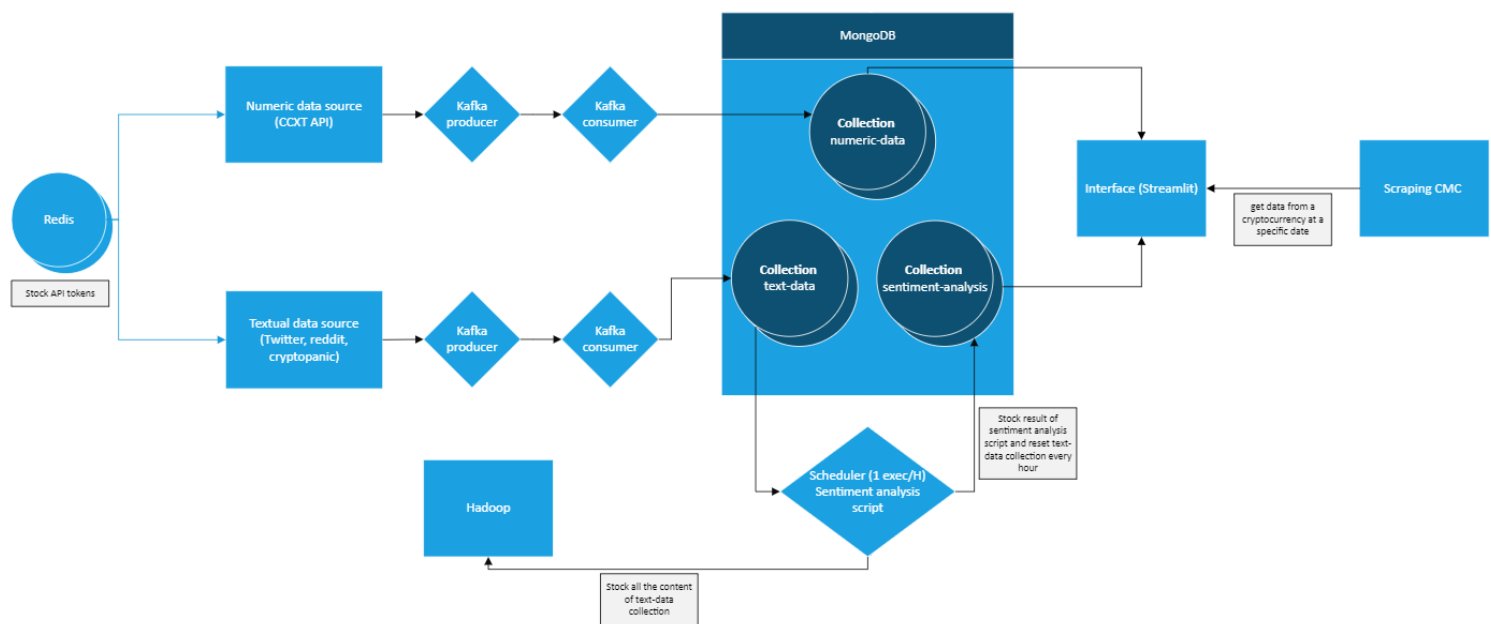
de répartir efficacement la charge de travail aux différents membres du groupe. La communication et le partage des informations est primordiale à la réussite d'un projet en groupe.



**Redis:** C'est un système de base de données clé-valeur hautes performances qui stocke les informations en mémoire pour un accès rapide. Il peut être utilisé pour gérer les sessions de vos utilisateurs ou le cache de votre site par exemple. Dans notre cas, nous utilisons des redis pour stocker nos token d'API que nous transformons

avec la librairie base64 avant de les stocker afin que les tokens ne soient pas stockés en dur dans la base. De plus, sa facilité d'utilisation avec python nous a confortés dans le choix de cet outil.

## IV. Architecture du projet



Pour voir le document en plein écran regarder le fichier “architecture\_admin\_bdd.png” dans le dossier documentation présent sur le repository github.

Concernant l’architecture nous avons choisi de stocker nos tokens d’API dans une base de données redis (key-value), cela nous permet de stocker de façon sécurisée nos tokens et de nous connecter à nos différentes API sans garder nos tokens en dur dans le code.

Après la récupération des tokens dans la BDD redis nous nous connectons à nos sources de données, nous les avons séparées en deux sources principales :

- Les sources textuelles
- Les sources numériques

Une fois la connexion effectuée nous récupérons nos données et les diffusons à l’aide de Kafka. Kafka nous permet de diffuser nos volumes de données très rapidement en effectuant un pré-traitement de celles-ci afin de n’envoyer que les données dont nous avons besoin. Suite à cela nos données sont stockées dans la base de données mongoDB à l’aide du consumer kafka.

Pour ce qui est des données numériques, elles sont stockées dans une collection nommée “numeric-data”, tandis que les données textuelles sont

stockées dans la collection "text-data". La dernière collection nommée "sentiment-analysis" nous permet de stocker le résultat de notre algorithme d'analyse de sentiments qui nous sera utile par la suite dans l'interface finale.

Nous avons choisi de stocker l'ensemble de nos données dans une base de données mongoDB car le volume de données peut évoluer très rapidement, nous avons également choisi mongodb car nous avons 4 sources de données différentes, malgré un formatage avant insertion la structure des données reste tout de même différente. Mongodb nous permettrait également de changer voir d'ajouter facilement de nouveaux traitements à nos données. De plus, mongoDB nous offre beaucoup de facilité grâce à la librairie python pymongo nous permettant d'effectuer des requêtes mongo sur python.

Concernant les données textuelles elles subiront un traitement spécial, en effet chaque heure l'ensemble des données seront traitées par un script python qui effectuera une analyse de sentiment. Cette analyse sera stockée dans la collection "sentiment-analysis". Une fois le traitement et le stockage du résultat effectué, l'ensemble des données de la collection text-data sont envoyées vers hadoop afin de les stocker dans le système HDFS puis sont supprimées de la collection afin de ne pas surcharger la base de données mongoDB. Nous avons fait ce choix car les données textuelles ne serviront que pour l'analyse de sentiment, elles ne sont donc d'aucune utilité dans notre base de données mongoDB. Concernant le choix de Hadoop, le système HDFS est très performant dans le stockage de gros volume de données ce qui est notre cas avec nos données textuelles. Pour finir l'interface utilisateur, c'est la partie visible de chaque utilisateur, elle devra donc être simple et compréhensible afin de pouvoir toucher le plus grand nombre d'utilisateurs. Nous avons choisi d'utiliser Streamlit pour sa simplicité et sa mise en place, de plus Streamlit fonctionne avec python ce qui nous permet de garder un seul langage dans la réalisation de ce projet et donc de nous faire gagner du temps. L'interface affichera donc nos données numériques et permettra de voir en temps réel l'évolution d'une cryptomonnaie, elle affichera aussi le résultat de notre analyse de sentiment qui permettra potentiellement d'analyser une tendance. Pour finir il sera possible de rechercher les données historiques pour chaque cryptomonnaie grâce à un script de scrapping permettant de récupérer les données historiques d'une cryptomonnaie sur Coin Market Cap.



## V. Alternatives considérées

Pour la bonne réalisation d'un projet l'ensemble des alternatives doivent être réfléchi afin de choisir la solution la plus adaptée au besoin du projet. Nous avons donc réfléchi à plusieurs alternatives dans un premier temps concernant notre architecture mais aussi concernant les outils que nous allons utiliser par la suite.

Nous devons donc dans un premier temps réfléchir à la diffusion de nos données afin de choisir pour chaque source si nous souhaitons diffuser nos données en streaming (diffusion en temps réel) ou en batch (diffusion à des créneaux précis). Dans notre cas, nous avons choisi que les données soient diffusées en streaming car les API nous offrent la possibilité de récupérer les données en temps réel. Bien entendu l'intégration d'une solution en batch est possible et peut être un axe d'amélioration de notre projet.

Justement une seconde alternative c'est présenter à nous concernant les sources de données, nous avons le choix entre utiliser des API tel que twitter, reddit, cryptopanic et ccxt qui nous fournissent les données en temps réel ou utiliser des fichier csv contenant déjà un gros volume de données, suffisant pour la réalisation de notre projet. Nous avons choisi la solution au plus long terme car les fichiers csv ont leurs limites car ils ne sont pas tenu à jour et les données présentes ne varient pas dans la grande majorité des cas. Un flux streaming nous permet donc d'avoir des valeurs en continu tant que l'API est fonctionnelle, le seul inconvénient étant que nos données arrivent en temps réel et il est plus compliqué d'aller chercher des valeurs dans le passé.

Le choix de Kafka est aussi un choix réfléchi dans une optique de long terme, nous aurions pu choisir comme alternative un simple script python qui envoie et stock nos données en BDD, seulement Kafka nous offre la possibilité d'augmenter très fortement le volume de données que nous diffusons.

Le choix de la base de données à été le plus difficile car plusieurs alternatives s'offraient à nous. Tout d'abord le choix entre mongoDB et SQL, dans notre cas nous avons choisi mongoDB car elle offrait une plus grande flexibilité comparé à SQL. Notre volume de données pouvant beaucoup augmenter et la possibilité que nous ajoutons de nouveaux traitements à stocker en BDD nous a orienté vers mongoDB. L'alternative aurait été SQL qui est beaucoup plus structuré mais nous souhaitons garder une flexibilité afin d'ajouter de nouvelles données dans le futur.

La dernière alternative était elasticSearch pour le stockage de nos données textuelles, seulement nous n'avons pas de recherche à effectuer dans nos données car elles sont déplacées vers Hadoop chaque heure et nous servent simplement à effectuer notre analyse de sentiment. ElasticSearch perdait donc tout intérêt à être intégré dans notre projet.

La dernière alternative que nous avons explorée concernait l'ajout d'Hadoop dans notre architecture. Le système HDFS proposé par Hadoop correspondait exactement à nos besoins en termes de stockage de gros volumes de données. L'alternative à hadoop à laquelle nous avons réfléchi était de garder toutes nos données dans notre base de données mongoDB mais cela représentait plusieurs inconvénient, tout d'abord le volume dépendant grandement du flux d'actualité le volume peut être très différent entre 2 heures mais surtout car le volume augmentera au fil des jours et sera surement très lourd au bout de quelques semaines ce qui affecterait les performance de notre BDD. Nous avons donc choisi d'utiliser hadoop pour stocker nos données textuelles afin de garder les données dans le cas où nous souhaiterions les traiter différemment ou visualiser l'évolution des données dans le temps.

## **VI. Flux de données**

Les données que nous manipulons proviennent de différentes sources et nous offrent un large panel de possibilités concernant leur récupération, leur traitement et leur stockage.

L'ensemble des sources que nous utilisons sont gratuites, le nombre de requêtes est donc limité. Cet inconvénient était donc à prendre en compte afin de ne pas limiter nos possibilités.

Tout d'abord, nos sources de données sont séparées en deux, les données numériques contenant l'ensemble des données de type numérique par exemple le prix, le nombre de monnaies en circulation etc... . Ensuite nous avons les données textuelles contenant l'ensemble des données de type texte comme les tweets, les subreddit et les news, qui nous permettrons d'effectuer une analyse de sentiments et de donner une tendance à l'aide de cette analyse.

### Les données numériques proviennent de :

- **CCXT:** La bibliothèque est utilisée pour se connecter et échanger avec des bourses de crypto-monnaies. Elle fournit un accès rapide aux données du marché pour le stockage, l'analyse, la visualisation, le développement d'indicateurs, le trading algorithmique, le backtesting de stratégies.

Il est destiné à être utilisé par les codeurs, les développeurs, les traders ayant des compétences techniques, les data-scientists et les analystes financiers pour construire des algorithmes de trading.

- **Scraping:** Le scraping, ou "web scraping", est le processus d'extraction de grandes quantités d'informations d'un site Web. Cela peut impliquer le téléchargement de plusieurs pages Web ou du site entier. Le contenu téléchargé peut inclure uniquement le texte des pages, le code HTML complet ou à la fois le code HTML et les images de chaque page.

L'API CCXT nous permet de récupérer les informations du marché "Binance" qui est un des leaders dans son domaine. Pour ce qui est du scraping nous récupérerons les données du site Coin Market Cap.

### Les données textuelles proviennent de :

- **Twitter API:** L'API twitter offre un accès aux données de twitter tel que les tweets, les retweets etc... Elle permet aussi d'effectuer des actions automatisées sur le réseau social. Dans notre cas l'API twitter nous permet de récupérer les tweets et les retweets concernant les cryptomonnaies que nous ciblons. Les données que nous récupérerons nous permettront par la suite d'effectuer une analyse de sentiment que nous pouvons corréliser avec les fluctuations de prix des différentes crypto-monnaies.
- **Reddit API:** L'API reddit nous offre un accès aux différentes données de reddit et plus particulièrement l'ensemble des données concernant chaque post d'un subreddit précis (Un subreddit est une communauté d'utilisateur centrés sur un sujet spécifique). Dans notre cas nous allons nous baser sur 3 subreddits différents :
  - Bitcoin
  - Ethereum
  - Solana

Les données de ces différents subreddits nous permettront d'effectuer une analyse de sentiment similaire à celle de twitter. Cela

nous permettra d'avoir un plus grand échantillon et d'avoir un résultat plus précis.

- **CryptoPanic:** C'est un agrégateur crypto minimaliste qui ressemble finalement beaucoup à Reddit. Le site répartit les informations en différentes catégories, telles que les informations sur les médias (vidéo), les blogs ou simplement les sites d'information. Nous utilisons l'API de cet agrégateur ce qui nous permet de récupérer l'ensemble des news sur les différentes cryptomonnaies. L'API fournit un système de filtrage nous permettant de ne récupérer que les informations concernant les cryptomonnaies que nous souhaitons traiter. Tout comme les deux précédentes sources de données nous allons stocker les données afin d'effectuer une analyse de sentiment.

Les trois sources de données sont gratuites et des restrictions sont donc appliquées. Le nombre de requêtes est limité, nous devons donc gérer cet inconvénient dans nos scripts afin de ne pas dépasser le nombre de requêtes disponibles et subir une pénalité de plusieurs minutes.

Concernant l'API twitter nous avons été dans l'obligation d'élever notre compte développeur afin d'obtenir l'accès aux données de twitter avec un assouplissement des restrictions

## VII. Données brutes

Voici à quoi ressemble les données brutes que nous récupérons :

**Scraping :**

```
[{'Name': 'Bitcoin', 'Symbol': 'BTC', 'Market Cap': '$883,011,296,029.51', 'Price': '$46,707.01', 'Circulating Supply': '18,905,325'}, {'Name': 'Ethereum', 'Symbol': 'ETH', 'Market Cap': '$466,037,604,378.56', 'Price': '$3,922.59', 'Circulating Supply': '118,808,565'}, {'Name': 'Solana', 'Symbol': 'SOL', 'Market Cap': '$55,490,684,848.85', 'Price': '$180.10', 'Circulating Supply': '308,113,743'}]
```

**CCXT :**

```
{'symbol': 'BTC/USD', 'timestamp': 1643057499730, 'datetime': '2022-01-24T20:51:39.730Z', 'high': 36893.39, 'low': 32946.89, 'bid': 36776.95, 'bidVolume': 0.076, 'ask': 36798.82, 'askVolume': 0.015607, 'vwap': 34632.1693, 'open': 35293.5, 'close': 36781.7, 'last': 36781.7, 'previousClose': 35295.07, 'change': 1488.2, 'percentage': 4.217, 'average': 36037.6, 'baseVolume': 2761.724405, 'quoteVolume': 95644507.1122, 'info': {'symbol': 'BTCUSD', 'priceChange': '1488.2000', 'priceChangePercent': '4.217', 'weightedAvgPrice': '34632.1693', 'prevClosePrice': '35295.0700', 'lastPrice': '36781.7000', 'lastQty':
```

```
'0.00994200', 'bidPrice': '36776.9500', 'bidQty': '0.07600000', 'askPrice':  
'36798.8200', 'askQty': '0.01560700', 'openPrice': '35293.5000', 'highPrice':  
'36893.3900', 'lowPrice': '32946.8900', 'volume': '2761.72440500',  
'quoteVolume': '95644507.1122', 'openTime': '1642971099730', 'closeTime':  
'1643057499730', 'firstId': '28888013', 'lastId': '28973743', 'count': '85731']}]
```

### Reddit API :

```
[{'subreddit': 'Bitcoin', 'title': 'Mentor Monday, January 24, 2022: Ask all your  
bitcoin questions!', 'selftext': "Ask (and answer!) away! Here are the  
general rules:\r\n\r\n* If you'd like to learn something, ask.\r\n* If you'd like to  
share knowledge, answer.\r\n* Any question about Bitcoin is fair  
game.\r\n\r\nAnd don't forget to check out /r/BitcoinBeginners\r\n\r\nYou  
can sort by new to see the latest questions that may not be answered  
yet."}]
```


### Twitter API :


```
[{'Symbol': 'BTC', 'Tweet': 'I think $BTC will go a little bit further. follow me  
for TA and news on'}, {'Symbol': 'BTC', 'Tweet': 'choosing the right  
investment strategy can reduce your risk and make the most of your  
money. i have made over 2.3 profit this year regardless the market  
condition not just by buying the dip but understanding &  
implementing trade with signals supplied by'}, {'Symbol': 'BTC', 'Tweet':  
'Sources: MLB, PA to continue talks amid progress 📺'}, {'Symbol': 'BTC',  
'Tweet': '👉'}, {'Symbol': 'BTC', 'Tweet': '👉'}]
```


### CryptoPanic API :

```
[{'code': 'ETH', 'title': 'Ethereum', 'slug': 'ethereum', 'url':  
'https://cryptopanic.com/news/ethereum/'}, {'code': 'LUNA', 'title': 'Terra',  
'slug': 'terra-luna', 'url': 'https://cryptopanic.com/news/terra-luna/'}, {'code':  
'LUNA', 'title': 'Terra', 'slug': 'luna-terra', 'url':  
'https://cryptopanic.com/news/luna-terra/'}, {'code': 'FTM', 'title': 'Fantom',  
'slug': 'fantom', 'url': 'https://cryptopanic.com/news/fantom/'}]
```

## VIII. Architecture des bases de données

numeric_data comment					
 id	dataType	N-N	default	comment	
price	FLOAT	N-N	default	comment	
symbol	STR	N-N	default	comment	
datetime	DATETIME	N-N	default	comment	
high	FLOAT	NULL	default	comment	
low	FLOAT	NULL	default	comment	
average	FLOAT	NULL	default	comment	

text_data comment					
 id	INTEGER	N-N	default	comment	
symbol	str or list	N-N	default	comment	
text	STR	N-N	default	comment	
source	STR	N-N	default	comment	

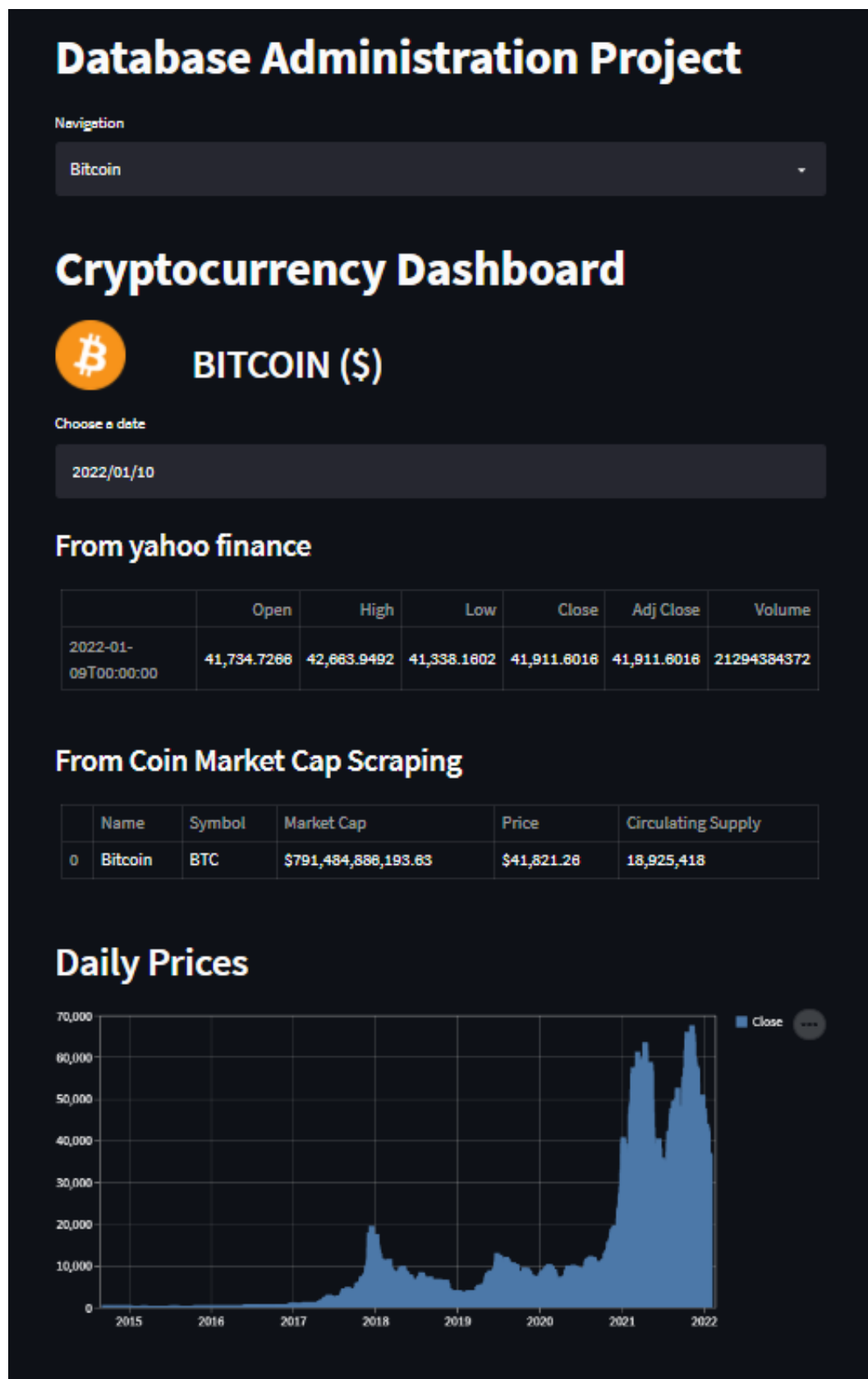
sentiment_analysis comment					
 id	dataType	N-N	default	comment	
symbol	dataType	N-N	default	comment	
polarity	dataType	NULL	default	comment	
subjectivity	dataType	NULL	default	comment	
datetime	dataType	NULL	default	comment	

Sur cette image vous pouvez observer la structure des trois collections que nous utilisons sur notre base de données MongoDB.

Nous avons fait le choix de créer un index simple sur la colonne text de la collection "text-data" car nous avons deux requêtes nécessitant un temps d'exécution assez long. La première requête étant la suppression de l'ensemble des lignes de la collection text-data chaque heure. Et la seconde requête étant la vérification de doublon dans la collection nous devons parcourir l'ensemble de la collection pour cette vérification, l'indexation de la colonne text nous permet donc d'améliorer grandement la vitesse d'exécution de nos 2 requêtes principales. Comme nous le savons l'indexation améliore grandement la vitesse d'exécution de certaines de nos requêtes mais peut amener à réduire un petit peu la vitesse d'insertion de nos données, il faut donc peser le pour et le contre et dans notre cas l'amélioration de la vitesse d'exécution est bien plus bénéfique que la légère perte de vitesse d'insertion. Il nous est donc plus bénéfique de créer un index.

Pour donner une estimation nous supprimons environ 20000 lignes par heure contre environ 300 données insérées par minute.

## IX. Interface



## Real Time Price

	date_time	price	symbol	high	low	
0	2022-01-28T10:55:15.314Z	36,583.8600	BTC/BUSD	37,519.1300	35,524.5200	36,
	date_time	price	symbol	high	low	
0	2022-01-28T10:55:15.031Z	36,578.3700	BTC/USD	37,506.7800	35,538.8700	36,4
	date_time	price	symbol	high	low	
0	2022-01-28T10:55:14.272Z	36,565.0100	BTC/USDC	37,500.0000	35,594.2700	36,
	date_time	price	symbol	high	low	
0	2022-01-28T10:55:15.248Z	36,551.2100	BTC/USDT	37,494.5600	35,546.2500	36,

## Sentiment Analysis

Polarity is float which lies in the range of  $[-1,1]$  where 1 means positive statement and -1 means a negative statement. Subjective sentences generally refer to personal opinion, emotion or judgment whereas objective refers to factual information. Subjectivity is also a float which lies in the range of  $[0,1]$ .

### Polarity

↑ 0.15

### Subjectivity

↑ 0.35

Nous avons fait le choix de mettre en place une interface simple et compréhensible par un utilisateur lambda n'ayant aucune connaissance des cryptomonnaies. Cette interface est vouée à évoluer avec l'ajout de nouveaux graphiques et avec l'ajout d'un visuel pour nos traitements futurs.

## X. Évolution du projet

Nous avons déjà réfléchi à plusieurs axes d'amélioration que nous pourrions mettre en place dans le futur. Le projet étant réalisé dans le cadre scolaire nous avons donc une deadline ce qui nous a obligé à mettre en place l'essentiel, nous avons donc encore énormément d'axes potentiels de progression. Avant tout, nous pourrions augmenter le volume de données en récupérant des données d'autres sources et d'autres formats tels que des images et des vidéos par exemple qui nous permettrait d'effectuer un traitement OCR (Optical Character Recognition).



L'ajout de nouvelles sources nous permettrait d'ajouter de nouveaux traitements à nos données tel que du machine learning avec un algorithme de prédiction de tendance ou de prix pour les cryptomonnaies. De plus, nous pourrions héberger notre architecture afin de ne plus avoir besoin de docker.

Nous pourrions aussi faire un parallèle entre l'évolution des cryptomonnaies et l'évolution de la bourse afin de savoir s'il y a un lien entre ces deux entités différentes.

Et pour finir grâce aux données que nous aurons récolter et à l'algorithme de prédiction que nous mettrons en place nous pourrions créer un bot de trading automatique qui achètera et vendra automatiquement des cryptomonnaies grâce à la prédiction.

Bien entendu nous n'avons pas réfléchi à toutes les possibilités d'évolution car le domaine des cryptomonnaies est très vaste et il existe énormément de possibilités et de traitements différents que nous pourrions appliquer à nos données.