Crypto Viz Rapport

DATE:	07/03/2024
REALISE PAR	**Amine Amarzouk, Samy Hadj-ali, Zakia, Chemoune, Christian Ndoradoumngue, Edouard Tan, Mélanie Manoharan**

INTRODUCTION

L'aventure de CryptoViz dans le domaine de la diffusion de données en temps réel s'est avérée être un périple riche en apprentissages et en défis. À travers ce document, nous partageons notre expérience avec Apache Kafka, les leçons tirées et les ajustements stratégiques qui ont caractérisé notre approche. Nous mettons un point d'honneur sur la manière dont nous avons orchestré nos topics Kafka et traité les flux de données des cryptomonnaies pour enrichir notre Dashboard en temps réel.

Objectif du projet

Crypto est un projet Big Data qui se concentre sur la collecte, le traitement et la visualisation des données financières associées aux cryptomonnaies. À cette fin, nous utilisons des technologies de pointe telles que Kafka et Python. Ces outils nous permettent de collecter et d'analyser en temps réel les données provenant de diverses plateformes de trading de cryptomonnaies, ouvrant ainsi la voie à une compréhension approfondie et à des insights précieux dans ce domaine en pleine expansion.

Organisation

Équipe de Développement:

Le projet a été mené par une équipe composée de trois développeurs expérimentés. Chaque développeur avait des responsabilités spécifiques dans la conception, le développement et le déploiement du système. Les rôles clés au sein de l'équipe de développement étaient les suivants :

- **Amine** : Responsable de l'écriture du scrapper pour récupérer les données de l'API cible, ainsi que du développement des producers et consumers Kafka pour stocker les données dans un fichier JSON. A travaillé sur la mise en place de l'API Flask pour récupérer les données stockées.
- **Christian** : En charge du développement de l'interface utilisateur à l'aide d'Angular. S'est occupé de créer des visualisations de données conviviales à l'aide de Chart.js pour afficher les données récupérées par l'API Flask.
- **Samy** : Responsable de la coordination entre le backend et le frontend, s'assurant que les données étaient correctement transmises et affichées dans l'interface utilisateur. Il était également en charge de l'intégration continue et du déploiement du système.

Équipe de Gestion de Projet:

L'équipe de gestion de projet comprenait trois chefs de projet, chacun ayant des responsabilités spécifiques pour assurer le bon déroulement du projet. Les rôles clés de l'équipe de gestion de projet étaient les suivants :

- **Zaki**a: Responsable de la supervision générale du projet, y compris la planification, la coordination des équipes de développement et de gestion, la communication avec les parties prenantes et la résolution des problèmes éventuels.
- **Edouard** : Chargé de superviser les aspects techniques du projet, notamment l'architecture du système, la sélection des technologies, la résolution des problèmes techniques et l'assurance de la qualité du code.
- **Mélanie** : Responsable de s'assurer que le produit final répondait aux besoins fonctionnels des utilisateurs. Il a travaillé en étroite collaboration avec l'équipe de développement pour garantir que les fonctionnalités nécessaires étaient correctement implémentées.

Communication et Collaboration:

Une communication claire et régulière entre les membres de l'équipe de développement et de gestion de projet a été essentielle pour garantir le succès du projet. Des réunions régulières ont été organisées pour discuter de l'avancement, résoudre les problèmes et prendre des décisions importantes. Un système de gestion de projet a été mis en place pour

suivre les tâches, les échéances et les problèmes. En outre, des canaux de communication ont été établis pour permettre une collaboration efficace entre les membres de l'équipe.

Cette version utilise le passé pour décrire les responsabilités et les actions des équipes de développement et de gestion de projet une fois le projet réalisé.

Méthodologie de travail

Pour garantir une avancée efficace du projet, nous avons utilisé une méthode de travail **Agile Scrum.**

Nous avons donc dans un premier temps identifié les compétences et rôles nécessaires pour le projet et nous nous sommes répartis les tâches et rôle en fonction de nos compétences et pouvant collaborer efficacement.

Nous avons avancé avec un point toutes les deux semaines en physique pour partager l'avancée du projet et parler des points de blocage potentiel. Pour tout autre moyen de communication, teams a été le moyen le plus privilégié.

En prenant compte nos points forts de l'agilité que nous procure la méthode, nous avons pu nous aider sur les autres points dont on n'était pas forcément assigné au départ.

Les documents et tâches sont tous répertoriés sur notre espace Notion pour être le plus transparent possible pour montrer l'avancée du projet en fonction de notre planning.

Kafka et format de données

- **Exploration d'Apache Kafka**:

Notre exploration d'Apache Kafka a été une plongée approfondie dans ses concepts et fonctionnalités. Nous avons étudié en détail la façon dont Kafka gère les flux de données, en mettant l'accent sur des éléments tels que les topics, les partitions, les brokers et les consumers. Nous avons également examiné comment Kafka peut être utilisé pour des cas d'utilisation spécifiques, notamment le streaming de données en temps réel, le traitement des données à grande échelle et la mise en œuvre de pipelines de données robustes.

Au cours de cette phase, nous avons réalisé des expérimentations pratiques avec Kafka, en configurant des clusters Kafka, en produisant et en consommant des messages, et en explorant les différentes options de configuration et de mise en œuvre. Cette immersion nous a permis de

comprendre en profondeur la puissance et la flexibilité de Kafka en tant que plateforme de streaming de données.

- **Préférence pour le Format JSON:**

Lorsque nous avons choisi le format de stockage de nos données, nous avons opté pour JSON en raison de sa flexibilité et de sa facilité d'utilisation. Contrairement au CSV, qui est plus rigide et moins adapté à la représentation de données complexes, le JSON offre une structure de données flexible et extensible.

En stockant nos données au format JSON, nous avons pu représenter des structures de données complexes telles que des objets imbriqués et des tableaux, ce qui était essentiel pour notre projet. De plus, le JSON est largement pris en charge par les outils d'analyse et de visualisation de données, ce qui facilitait l'intégration de nos données dans notre dashboard et d'autres systèmes.

- **Architecture des Topics Kafka:**

Lors de la conception de nos topics Kafka, nous avons pris en compte les besoins spécifiques de notre projet et les types de données que nous voulions traiter. Nous avons créé deux topics principaux : "datacrypto" pour les données des cryptomonnaies et "rssfeed" pour les actualités.

Chaque topic a été conçu pour répondre à des besoins spécifiques en matière de gestion et d'analyse des données. Par exemple, le topic "datacrypto" a été structuré pour inclure des informations telles que le nom de la cryptomonnaie, son prix actuel, sa capitalisation boursière et un horodatage, tandis que le topic "rssfeed" a été utilisé pour capturer les actualités liées aux cryptomonnaies. Cette segmentation nous a permis de gérer efficacement les flux de données et de faciliter l'analyse ciblée des informations.

Environnement de travail

- **Débuts avec une instance Google Cloud Platform**:

Lorsque nous avons commencé à construire notre environnement de projet, nous avons choisi Google Cloud Platform (GCP) en raison de sa robustesse, de sa scalabilité et de sa facilité d'utilisation. Nous

avions l'intention d'utiliser Apache Spark pour enrichir notre pipeline de données, ce qui nécessitait une infrastructure compatible avec les technologies Big Data telles que Hadoop YARN et HDFS.

Sur GCP, nous avons configuré notre environnement en déployant des instances de machines virtuelles (VM) et en mettant en place des services de stockage comme Google Cloud Storage. Nous avons également créé des clusters Hadoop et configuré des instances de Kafka pour créer un environnement complet de traitement et de diffusion de données.

- **Réorientation et Simplification:**

Au fur et à mesure que nous progressions dans notre projet, nous avons rencontré des défis inattendus qui nous ont amenés à réévaluer notre approche. Des problèmes tels que la perte d'accès à notre première VM et les difficultés rencontrées lors de l'intégration de Spark ont conduit à une réorientation de notre stratégie.

Nous avons décidé de simplifier notre architecture en nous concentrant exclusivement sur Kafka. Cela signifiait abandonner l'idée d'intégrer Spark et de mettre en place une infrastructure Hadoop complexe. En simplifiant notre architecture, nous avons pu réduire la complexité de notre système, ce qui nous a rendus plus agiles et plus réactifs aux changements et aux défis futurs.

Traitement et visualisation de données

- **Transformation des Données Cryptographiques**

Dans cette section, nous détaillons les fonctions de transformation de données utilisées pour traiter les informations relatives aux cryptomonnaies dans notre application. Chaque fonction effectue une transformation spécifique sur les données brutes afin de les rendre utilisables dans diverses fonctionnalités.

Les fonctions de transformation des données sont essentielles pour obtenir des informations précises et formatées sur les cryptomonnaies dans notre application. Voici une explication détaillée de chaque fonction :

L'extraction des données brutes depuis un fichier JSON pour récupérer la liste des cryptomonnaies disponibles est effectuée par une fonction. Cette fonction crée ensuite une liste de dictionnaires, chaque dictionnaire représentant une cryptomonnaie avec son identifiant et son nom.

Les informations actuelles sur les cryptomonnaies sont fournies en récupérant directement les données brutes des cryptomonnaies depuis le fichier JSON, grâce à une autre fonction.

Pour obtenir la capitalisation boursière des cryptomonnaies sur une période de temps spécifiée, une fonction permet de filtrer les données brutes. Cette fonction extrait la capitalisation boursière sur un intervalle de temps déterminé, en fonction de l'identifiant de la cryptomonnaie et de l'intervalle de temps spécifié.

De même, pour récupérer les valeurs des cryptomonnaies sur une période de temps spécifiée, une autre fonction filtre les données brutes. Elle obtient les valeurs sur un intervalle de temps déterminé, en fonction de l'identifiant de la cryptomonnaie et de l'intervalle de temps spécifié.

Enfin, le calcul de l'évolution en pourcentage des valeurs d'une cryptomonnaie sur une période de temps spécifiée est effectué par une dernière fonction. Cette fonction réalise ce calcul en fonction des valeurs fournies et de l'intervalle de temps spécifié.

Ces fonctions sont intégrées dans différentes parties de notre application pour fournir des données précises et formatées, permettant ainsi une visualisation et une analyse efficaces des cryptomonnaies. Elles jouent un rôle crucial dans le traitement des données et la présentation des informations pertinentes aux utilisateurs.

- **Personnalisation des préférences d'utilisateur**:

Nous avons intégré la possibilité pour les utilisateurs de définir leurs cryptomonnaies favorites au sein de notre interface. Cette fonctionnalité permet à chaque utilisateur de sélectionner les cryptomonnaies qui l'intéressent particulièrement et de suivre leurs évolutions de manière privilégiée.

L'utilisateur peut définir ses cryptomonnaies favorites dans son profil, lui permettant ainsi de recevoir des informations spécifiques et pertinentes concernant les actifs numériques qui l'intéressent le plus. Cette personnalisation améliore l'expérience utilisateur en lui fournissant des données qui correspondent à ses intérêts et à ses besoins spécifiques.

- **Filtrage du flux RSS en fonction des cryptomonnaies favorites:**

Nous avons également ajouté la fonctionnalité de filtrage du flux RSS en fonction des cryptomonnaies favorites de l'utilisateur. Lorsque l'utilisateur consulte les actualités dans notre interface web, le flux RSS est automatiquement filtré pour ne montrer que les actualités liées aux cryptomonnaies qu'il a sélectionnées comme favorites.

Ce filtrage personnalisé garantit que l'utilisateur ne reçoit que les informations qui l'intéressent le plus, ce qui lui permet de rester informé sur les actualités qui ont le plus d'importance pour lui. Cela offre

une expérience utilisateur plus pertinente et personnalisée, en fournissant des informations ciblées qui correspondent aux intérêts spécifiques de chaque utilisateur.

- **Visualisation des données:**

En complément de notre infrastructure de traitement et de diffusion de données, nous avons développé une interface web permettant de visualiser les informations collectées et traitées. Pour ce faire, nous avons créé une API REST en utilisant Python, avec des routes sécurisées et une authentification pour garantir la confidentialité des données.

Le backend de l'interface web repose sur Python, offrant une interface sécurisée pour accéder aux données collectées par Kafka. Nous avons mis en place des mécanismes d'authentification pour garantir que seuls les utilisateurs autorisés peuvent accéder aux informations sensibles.

En ce qui concerne le frontend, nous avons utilisé AngularTs pour développer une interface utilisateur intuitive et réactive. Nous avons utilisé des composants Angular tels que les tableaux pour afficher les données de manière organisée et les charts avec Chart.js pour visualiser graphiquement les données sous forme de graphiques et de diagrammes.

Nous avons opté pour Chart.js dans notre projet de visualisation des données en raison de sa légèreté, de sa facilité d'utilisation et de sa compatibilité avec les navigateurs modernes et les appareils mobiles. De plus, sa documentation complète et sa communauté active nous ont permis de rapidement intégrer et personnaliser nos graphiques. Enfin, sa modularité nous offre la possibilité d'ajouter facilement des fonctionnalités supplémentaires pour améliorer nos visualisations.

Pour assurer le temps réel des données affichées, nous avons mis en place un mécanisme de rafraîchissement des données à intervalles réguliers. Toutes les minutes, l'API récupère les données les plus récentes des topics Kafka, garantissant ainsi que les informations affichées dans l'interface web sont toujours à jour et pertinentes pour les utilisateurs.

Cette approche de visualisation des données en temps réel offre aux utilisateurs une vue claire et dynamique des tendances et des informations clés, facilitant ainsi la prise de décision et l'analyse des données dans un environnement convivial et interactif.

Architecture du projet

Notre projet utilise un système de récupération de données en plusieurs étapes pour garantir que les informations sont collectées de manière efficace, stockées de manière fiable et présentées de manière conviviale aux utilisateurs finaux.

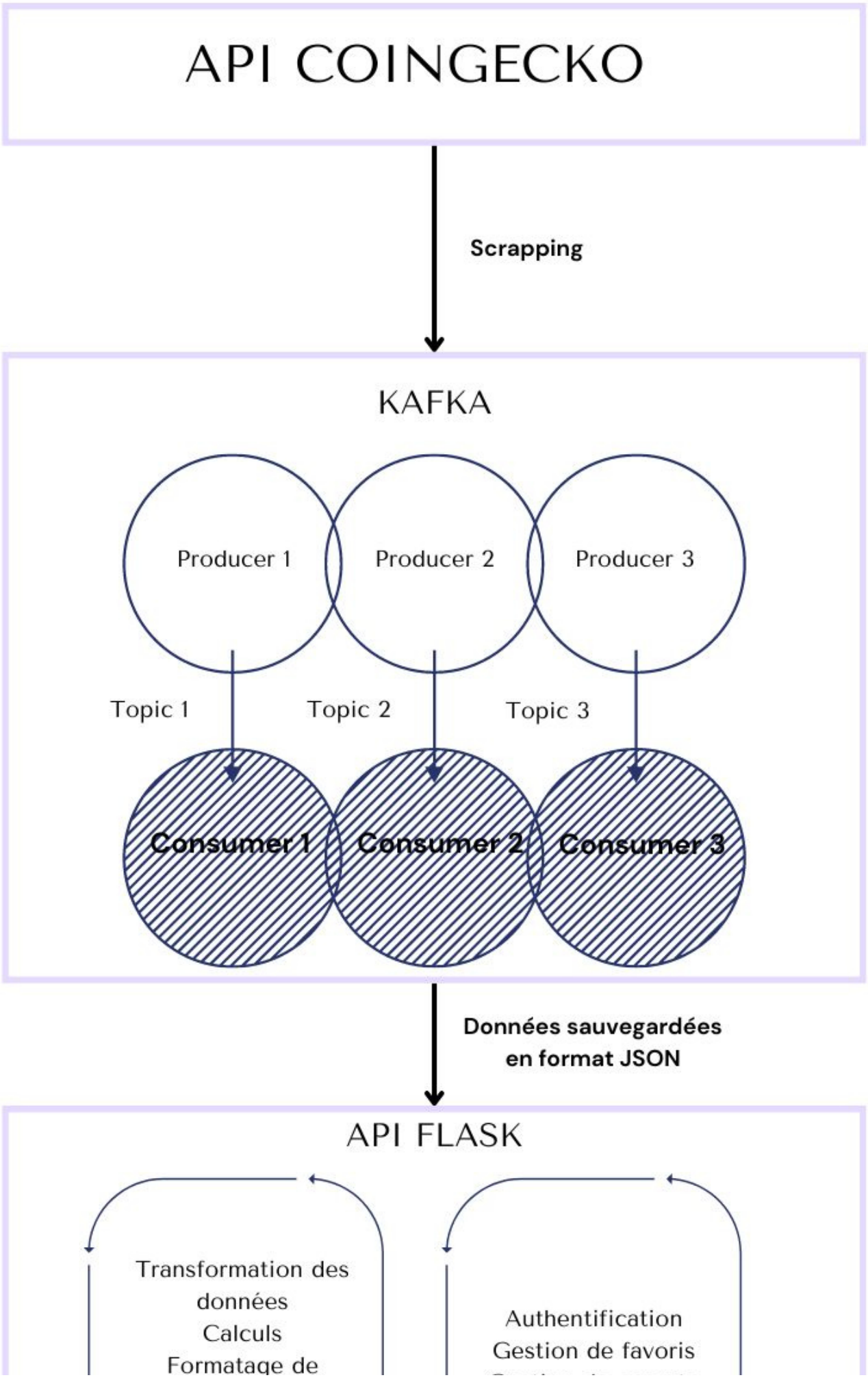
Dans un premier temps, nous utilisons un processus de scraping avec Kafka pour extraire les données de l'API source. Kafka est un outil de streaming distribué qui nous permet de gérer efficacement les flux de données et de les acheminer vers nos consommateurs. Ces consommateurs sont chargés de récupérer les données à partir des topics Kafka et de les stocker dans des fichiers JSON. Ce processus garantit une collecte et un stockage efficaces des données, même en cas de fluctuations de charge.

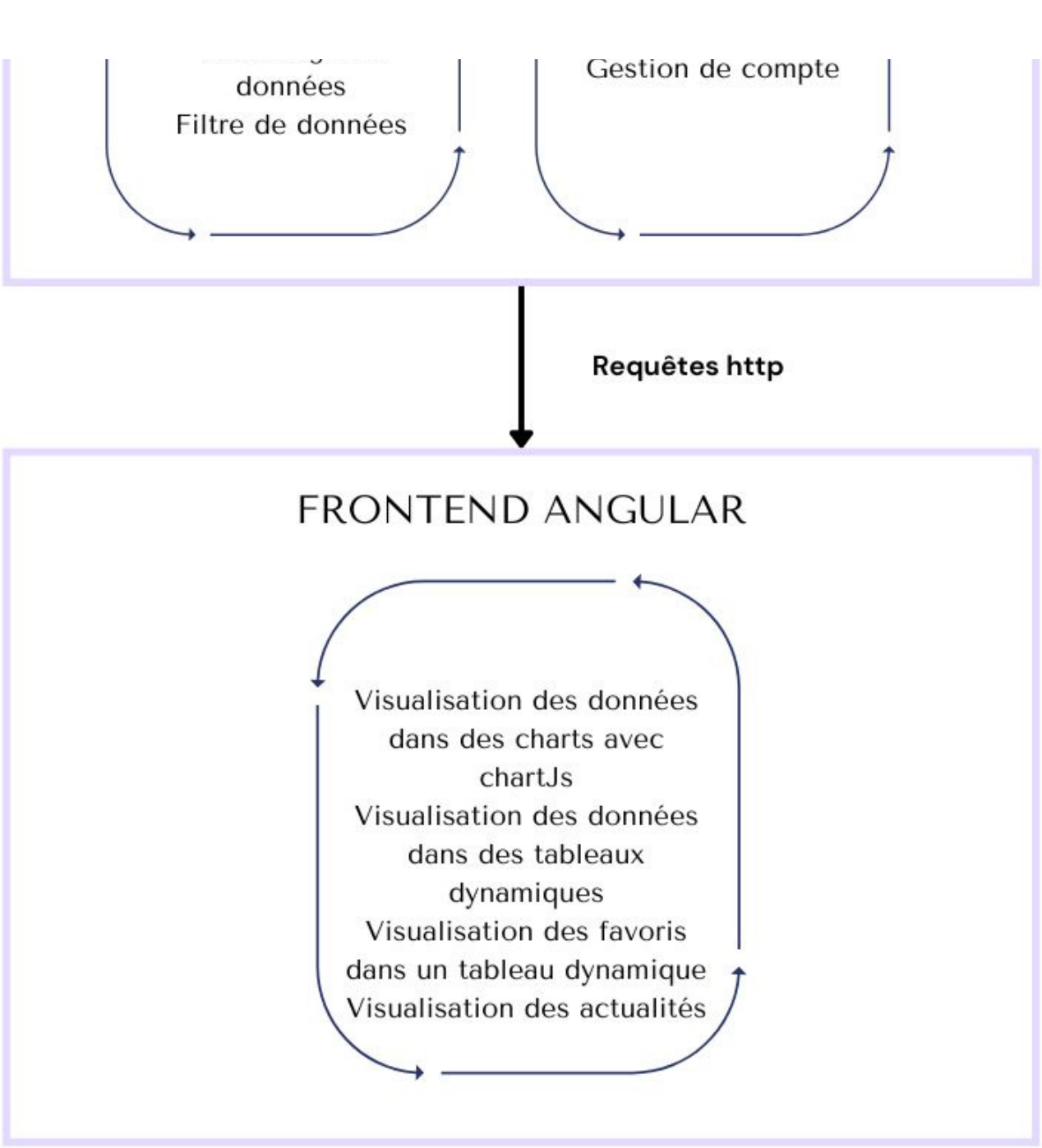
D'autre part, nous avons développé une API Flask qui agit comme un pont entre les données stockées et le frontend. Flask, un framework web en Python, est idéal pour créer des API RESTful simples et efficaces. Cette API récupère les données depuis les fichiers JSON et les formate selon les spécifications requises. Elle expose ensuite ces données sous forme d'endpoints accessibles par le frontend.

Enfin, pour offrir une expérience utilisateur dynamique et en temps réel, nous avons intégré notre API Flask avec un frontend Angular. Angular est un framework JavaScript populaire pour la construction d'applications web dynamiques. Nous utilisons des services Angular pour interagir avec notre API Flask, récupérer les données formatées et les afficher de manière interactive dans l'interface utilisateur. Ces services sont configurés pour appeler régulièrement l'API Flask afin de garantir que les données présentées aux utilisateurs sont toujours à jour.

En particulier, pour visualiser les données de manière graphique, nous utilisons Chart.js, une bibliothèque JavaScript flexible et puissante pour la création de graphiques. Nous configurons Chart.js dans notre frontend Angular pour créer des graphiques dynamiques et interactifs qui représentent les données collectées. Ainsi, les utilisateurs peuvent non seulement accéder aux données les plus récentes, mais aussi les analyser et les interpréter visuellement grâce aux graphiques générés par Chart.js.

En combinant le scraping avec Kafka, le stockage dans des fichiers JSON, l'utilisation d'une API Flask pour formater les données, l'intégration avec un frontend Angular et la visualisation des données avec Chart.js, notre projet offre une solution complète et robuste pour la collecte, le traitement, la visualisation et l'analyse des données provenant d'une API.


API COINGECKO
Scrapping
KAFKA
Producer 1
Producer 2
Producer 3
Topic 1
Topic 2
Topic 3
Consumer 1
Consumer 2
Consumer 3
Données sauvegardées
en format JSON
API FLASK
Transformation des
données
Calculs
Formatage de
Authentification
Gestion de favoris



CRYPTOVIZ

Axes d'amélioration

Malgré le succès de notre projet, nous avons identifié plusieurs axes d'amélioration qui pourraient être explorés à l'avenir pour optimiser davantage notre solution :

Conteneurisation de l'Application : Une première amélioration consisterait à conteneuriser l'application à l'aide d'outils tels que Docker et Kubernetes. Cela faciliterait le déploiement et

la gestion de l'application, en assurant une portabilité et une isolation accrues des différentes composantes.

Hébergement sur des Serveurs Plus Performants : Nous pourrions envisager de migrer notre application vers des serveurs plus performants ou vers une plateforme de cloud computing afin d'améliorer ses performances et sa disponibilité. Cela nous permettrait également de bénéficier de ressources supplémentaires pour faire face à des charges de travail plus importantes.

Ajout de Topics Supplémentaires : Nous pourrions étendre notre architecture Kafka en ajoutant des topics supplémentaires pour prendre en charge de nouveaux types de données ou de fonctionnalités. Par exemple, nous pourrions introduire des topics dédiés à la gestion des erreurs, à la surveillance du système ou à l'analyse en temps réel.

Optimisation des Processus et des Performances : Il serait également judicieux d'examiner de près nos processus de développement et nos algorithmes pour identifier les éventuelles inefficacités ou les goulots d'étranglement. En optimisant ces aspects, nous pourrions améliorer les performances globales de notre système et réduire sa consommation de ressources.

Mise en Place d'une Surveillance Avancée : Pour garantir le bon fonctionnement et la disponibilité continue de notre système, nous pourrions mettre en place des outils de surveillance avancée, tels que des tableaux de bord de monitoring en temps réel et des alertes automatisées. Cela nous permettrait de détecter rapidement les problèmes potentiels et d'y remédier avant qu'ils n'affectent les performances du système.

En mettant en œuvre ces axes d'amélioration, nous serions en mesure de renforcer encore davantage notre solution et de garantir sa pertinence et sa compétitivité à long terme.

Retour d’expérience

La réalisation de ce projet a été une expérience extrêmement enrichissante pour toute l'équipe. Elle nous a permis de plonger dans le monde fascinant du streaming de données et de comprendre les principes fondamentaux qui régissent ce domaine. L'utilisation de Kafka comme système de messagerie pour la gestion des flux de données a été particulièrement instructive. Nous avons pu apprécier la flexibilité et la scalabilité offertes par Kafka, ainsi que sa capacité à traiter efficacement de grands volumes de données en temps réel. De plus, la mise en œuvre d'une architecture basée sur des producteurs et des consommateurs Kafka

nous a permis de mieux comprendre les concepts de la programmation asynchrone et de la distribution des tâches.

Nous avons également développé une meilleure compréhension des défis liés à la gestion des données en temps réel, tels que la garantie de la cohérence et de l'intégrité des données, ainsi que la détection et la gestion des goulots d'étranglement. En travaillant sur ce projet, nous avons pu affiner nos compétences en matière de résolution de problèmes et renforcer notre capacité à concevoir et à mettre en œuvre des solutions robustes et évolutives.

Conclusion

En résumé, ce projet a été une aventure stimulante pour notre équipe, nous permettant d'explorer le domaine complexe du streaming de données. L'utilisation de Kafka comme système de messagerie nous a ouvert les portes vers une compréhension plus approfondie des enjeux et des perspectives dans le traitement des données en temps réel à grande échelle.

Au cours de ce projet, nous avons développé et affiné nos compétences techniques, en acquérant une compréhension approfondie des technologies telles que Kafka, Flask, Angular et Chart.js. Nous avons également renforcé nos compétences en matière de conception logicielle, de développement d'applications distribuées et de résolution de problèmes complexes.

De plus, nous avons identifié plusieurs axes d'amélioration potentiels qui pourraient être explorés à l'avenir pour optimiser davantage notre solution. En envisageant des initiatives telles que la conteneurisation de l'application, l'hébergement sur des serveurs plus performants, l'ajout de topics supplémentaires dans Kafka et l'optimisation des processus et des performances, nous sommes convaincus que nous pourrions renforcer encore davantage notre solution et garantir sa pertinence et sa compétitivité à long terme.

Enfin, ce projet nous a permis de renforcer notre collaboration en tant qu'équipe et de développer une approche rigoureuse et méthodique pour la gestion de projets de grande envergure. Nous sommes fiers des résultats que nous avons obtenus et nous sommes impatients d'appliquer les connaissances et les compétences acquises dans de futurs projets.

Nous tenons à exprimer notre gratitude envers toutes les personnes qui ont contribué au succès de ce projet, ainsi qu'à nos clients et partenaires pour leur confiance et leur soutien continu. Nous sommes convaincus que les enseignements tirés de cette expérience nous

serviront de base solide pour relever de nouveaux défis et atteindre de nouveaux sommets dans notre parcours professionnel.