

## De quoi ça parle?

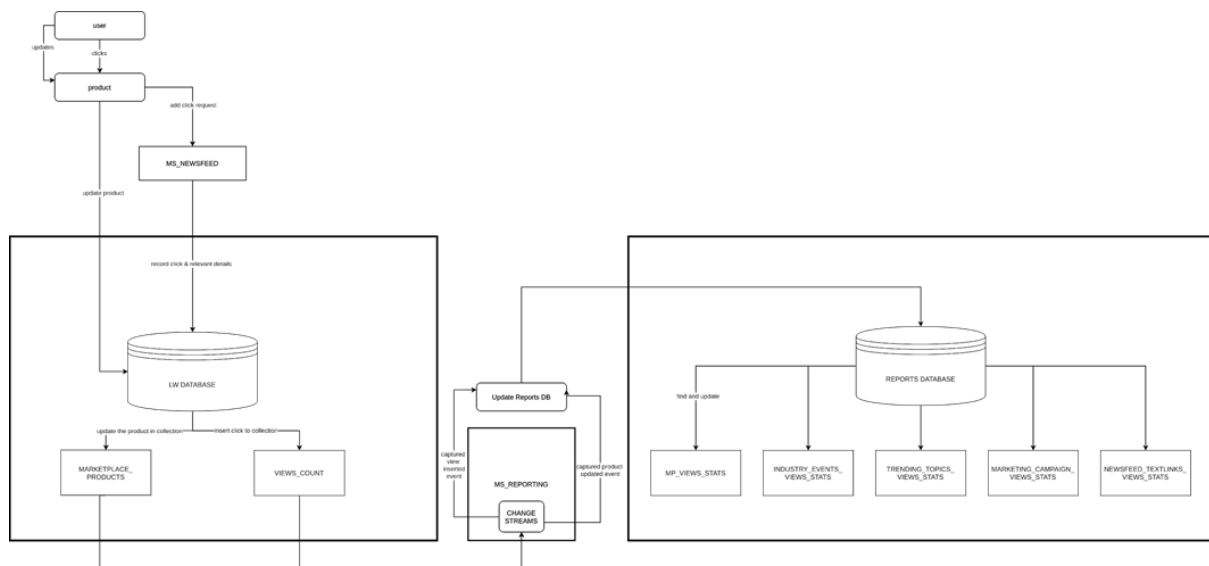
Cette étude de cas concerne le problème difficile mais intéressant que j'ai résolu pour Leafwire. Le projet est un sous-ensemble du travail que j'ai effectué pour le projet Leafwire. On m'a confié la responsabilité de gérer la section analytique de Leafwire. Leafwire surveille les interactions des utilisateurs pour offrir aux utilisateurs une meilleure expérience d'utilisation de l'application et pour produire des rapports. Toutes les données d'analyse ont été stockées dans une collection Mongo DB, qui correspond initialement à l'objectif de produire des rapports et de comprendre l'utilisation des utilisateurs.

## Énoncé du problème

Au fur et à mesure que la base d'utilisateurs commence à croître, les données d'analyse commencent à croître de manière exponentielle, j'ai remarqué que si cette croissance se poursuivait, cela aurait un impact considérable sur les performances, dans une mesure où la production de rapports consommerait 100 % d'utilisation du processeur et rendrait l'application inutilisable pour les utilisateurs. J'ai dû développer une solution au problème pour surmonter les inconvénients possibles dans un proche avenir. L'application prenait environ 25 secondes pour produire 1 rapport.

## Solution

J'ai créé une solution basée sur une architecture basée sur des microservices intégrant l'API leafwire à d'autres microservices.





### Microservice 1 : API du processeur de données

J'ai conçu ce microservice en utilisant en quelque sorte Mongo Change Stream. Ainsi, lorsqu'une opération de base de données a lieu dans la base de données d'origine, elle déclenchera automatiquement un événement. Ainsi, chaque fois qu'un enregistrement est créé dans la grande collection d'analyses, il déclenche un événement. Ce microservice traiterait les données et les ajouterait dans des tables distinctes. Il ajoutera le nombre combiné de données d'analyse par jour, et chaque jour aura un enregistrement du total des données d'analyse.

### Microservice 2 : API de création de rapports

J'ai conçu ce microservice pour produire des rapports à partir des données traitées et seule cette API peut accéder à la collecte de données de rapports séparée par le processeur de données. de la demande. En combinant ces deux microservices avec l'API leafwire, nous avons pu résoudre le problème et ma solution gère toujours des données qui augmentent au rythme de millions par semaine et le temps de traitement moyen des rapports est d'environ 0,2 seconde. Ainsi, avec ma solution, j'ai pu réduire considérablement le temps de

traitement (25 secondes à 0,2 secondes). Dans la capture d'écran ci-dessous, vous pouvez voir le temps de réponse pour le rapport généré (233 ms).

Name	Status	Type	Initiator	Size	Time
■ fetchMarketplaceStatistics	204	preflight	Preflight 	0 B	34 ms
■ fetchMarketplaceStatistics	200	xhr	xhr.js:160	3.0 kB	233 ms
■ fetchMarketplaceStatisticsCount	200	xhr	xhr.js:160	265 B	139 ms
■ fetchMarketplaceStatisticsCount	204	preflight	Preflight 	0 B	32 ms