**Titre RNCP 39586**

**INGENIEUR EN SCIENCE DES DONNÉES SPECIALISÉ EN** **APPRENTISSAGE AUTOMATIQUE**

**Bloc 1 :**

**COLLECTER TRANSFORMER T SÉCURISER DES DONNÉES**

Une image contenant dessin humoristique, clipart, conception

Le contenu généré par l’IA peut être incorrect.

**Sujet :**

**Développement d’un outil d’analyse automatisé des commentaires YouTube**

**Auteur :**

**FURTADO LEAL Carla**

Sommaire

[I. Collecte de données structurées et non structurées 3](#_Toc206425450)

[A. Une stratégie de collecte de données 3](#_Toc206425451)

[B. Un exemple de collecte de données 4](#_Toc206425452)

[C. Une méthode d’automatisation de collecte 6](#_Toc206425453)

[II. Stockage des données structurées et non structurées 7](#_Toc206425454)

[A. Une stratégie de stockage et un modèle de données 7](#_Toc206425455)

[B. Une base de données et une solution de stockage Big Data 9](#_Toc206425456)

[III. Structuration, transformation et enrichissement des données 9](#_Toc206425457)

[A. Outils et technologies de traitement des données 9](#_Toc206425458)

[B. Transformation des données 10](#_Toc206425459)

[C. Processus ETL, automatisation et orchestration du traitement des données 11](#_Toc206425460)

[IV. Sécurisation des données 13](#_Toc206425461)

[A. Une politique de sécurisation des données 13](#_Toc206425462)

[B. Un schéma d’architecture de sécurité 13](#_Toc206425463)

[Annexe 14](#_Toc206425464)

[IV. SÉCURISATION DES DONNÉES 14](#_Toc206425465)

[A. Politique de sécurisation et conformité RGPD 14](#_Toc206425466)

[B. Architecture de sécurité technique 15](#_Toc206425467)

# Collecte de données structurées et non structurées

## Une stratégie de collecte de données

Pour rappel, nous cherchons à répondre à la question :

***Comment valoriser les retours clients via l’analyse automatisé des commentaires YouTube ?***

Pour ce faire nous avons décidé de développer un outil d’analyse automatisé des commentaires You Tube. Ce qui implique l’élaboration d’une stratégie de collecte de données textuelles, indispensables à l’analyse de sentiments et Topic Modeling. Le Topic modeling fonctionne sur la base de données textuelles non labelisées et détermine les sujets sous-jacents latents dans ces données en se basant sur la co-occurrence des mots. L’analyse de sentiment, lorsqu’elle fait appel à un modèle d’apprentissage automatique supervisé, nécessite des données d’entrainement, des commentaires labelisés. Si on mène cette analyse en non supervisé, aucune donnée n’est nécessaire, car elle se base sur des modèles pré-entrainés, des lexiques ou des règles.

Ainsi, notre stratégie de collecte, doit nous permettre de construire et d’incrémenter une base de données de commentaires selon les besoins d’analyse et de construire un corpus d’entrainement représentatif pour les performances et la pertinence des analyses, basées sur des modèles d’apprentissage supervisé.

La collecte doit être ciblée et automatisée sur les vidéos renseignées par l’utilisateur, il ne s’agit pas de collecter tous les commentaires existant sur la plateforme. Elle doit également nous permettre d’avoir une base d’entrainement pour l’amélioration / monitoring / calibrage du modèle d’analyse de sentiments.

Les données utiles et nécessaires appartiennent donc à deux catégories :

* **Les données d’analyses** : commentaires YouTube extraits via l'API officielle
* **Les données d'entraînement** : corpus de textes français étiquetés pour le calibrage des modèles d'analyse de sentiment

Pour une vidéo donnée, nous interrogeons l’API YouTube V3 de Google Cloud, pour récupérer diverses métadonnées relatives à une vidéo donnée, dont les commentaires. Cette source nous permet de garantir la fiabilité des données et la conformité réglementaire (vs scrapping).

Il s’agit d’une source de données structurée, suivant le schéma de type JSON.

Lors de l’extraction seules les données utiles et nécessaires sont collectées :

* [Id](https://developers.google.com/youtube/v3/docs/comments?hl=fr#id) : l’identifiant unique du commentaire
* [channelId](https://developers.google.com/youtube/v3/docs/comments?hl=fr#snippet.channelId) : l’identifiant de la chaîne YouTube
* [publishedAt](https://developers.google.com/youtube/v3/docs/comments?hl=fr#snippet.publishedAt) : date de publication
* [textOriginal](https://developers.google.com/youtube/v3/docs/comments?hl=fr#snippet.textDisplay) : le texte du commentaire tel qu’il a été publié ou mis à jour.
* [likeCount](https://developers.google.com/youtube/v3/docs/comments?hl=fr#snippet.likeCount) : le nombre de like du commentaire
* viedoId : identifiant de la vidéo
* authorChannelId : l’identifiant de la chaîne de la personne qui écrit le commentaire
* title : titre de la vidéo
* description : description de la vidéo
* commentCount : le nombre de commentaires de la vidéo

Les identifiants permettent de rattacher chaque commentaire à une vidéo précise d’une chaîne donnée et de les identifier de manière unique. En ce qui concerne l’identifiant de la chaîne de l’utilisateur, elle n’est collectée uniquement pour supprimer le commentaire du youtubeur. A ces données nous ajoutons la date d’extraction, l’url de la vidéo

Afin de ne collecter que les données pertinentes, nous avons mis en place un filtre pour cibler les vidéos françaises (dont les commentaires dont essentiellement rédigés en français) et avec plus de 200 commentaires, grâce à commentCount. Le titre et la description sont utilisés pour s’assurer qu’il s’agit bien d’une vidéo française, seul le titre est stocké. Ces données sont ensuite stockées pour être analysées.

Ensuite, l’analyse automatisée nécessite également des données d’entrainement pour le calibrage du modèle d’analyse de sentiment. Ainsi, le meilleur modèle sélectionné, sera appliqué sur les données réelles, les commentaires. Les données d’entrainement doivent être similaires aux données réelles : langue, longueur, vocabulaire ...

En effet, plus les données ont des caractéristiques similaires aux données réelles meilleurs seront les performances. Kaggle est une plateforme qui met à disposition des jeux de données relatifs à sujets divers et variés, pour des compétitions, recherche scientifiques… C’est donc sur cette plateforme que nous avons trouvé nos données d’entrainement. Il s’agit de ~~post twitter rédigés en français et étiqueté positifs ou négatifs~~. Commentaires YouTube traduits de l’anglais vers le français à partie d’un modèle pré-entrainé.

Le web scrapping a également été envisagé. Nous n’aurions pas été soumis à un quota comme c’est le cas lorsque nous utilisons l’API. Nous avons tout de même privilégié l’API pour des raisons réglementaires et pour gain de temps. En effet, le scrapping des commentaires n’est pas autorisé, de plus développer un code de scrapping peut prendre du temps et le scrapping en lui-même est chronophage. Notons que nous somme également dépendant de la structure même du site, si elle vient à changer il faudra nécessairement revoir le programme. Ainsi, l’API allie simplicité et gain de temps, mais implique le respect de quotas.

## Un exemple de collecte de données

Nous avons sélectionné une vidéo de Squeezie (youtubeur populaire en France), dont les vidéos rencontrent généralement un fort engagement en termes de vues et de commentaires (plus de 200 commentaires).

Tout d’abord l’utilisation de l’API de google YouTube v3 nécessite d’avoir un compte Google cloud developper et un projet associé pour lequel nous avons généré des clés d’accès / des droit d’accès pour l’API.

L’interrogation de l’API se fait en précisant la ressource que l’on veut exploiter et la méthode à utiliser. Ici il s’agit de la ressource du fil de discussion, « commentThreads » exploité selon la méthode « list » pour extraire les données. Ensuite le paramètre « videoId », obtenu à partir de l’url, permet d’identifier la vidéo ciblée.

Une image contenant texte, capture d’écran, carte de visite, Police

Le contenu généré par l’IA peut être incorrect.

Figure 1: Exemple de structure d'appel d'API

Une image contenant texte, logiciel, Logiciel multimédia, capture d’écran

Le contenu généré par l’IA peut être incorrect.

Figure 2 : Extrait de code permettant de sélectionner les variables utiles

Le script *extraction.py* en annexe, permet d’interroger l’API via une classe qui prend l’url de la vidéo comme paramètre d’entrée.

Les commentaires récupérés sont initialement au format JSON, comme dans le schéma présenté précédemment. Nous récupèrons une liste de dictionnaires correspondant chacun à un commentaire et aux métadonnées associées. Pour l’analyse, les données sont ensuite transformées au format dataframe.

En termes d’exhaustivité de la collecte, le nombre de commentaires affiché est plus important que le nombre de commentaires extraits. Cela s’explique par l’absence des réponses aux commentaires dans l’extraction. Il est possible de récupérer les commentaires en ajoutant le paramètre « replies » dans « part », mais nous avons de ne pas les récupérer. En effet, les réponses aux commentaires permettent de capter les interactions entre les spectateurs, qui tendent à s’éloigner du sujet de la vidéo. Or nous cherchons à analyser les « réactions directes » au contenu et non les interactions entre les individus qui risquent de biaiser nos résultats ou de les faire diverger.

De plus, les commentaires modérés par la plateforme ne sont pas collectés. Seuls les commentaires publiés et donc approuvés par la plateforme sont collectés et analysés. Ce qui assure la qualité des données. En effet, la plateforme garantit un premier niveau de qualité en filtrant de type de commentaires indésirables, évitant doublons (tous les commentaires ont un id unique qu’ils conservent en cas de mise à jour) et en gérant le format des données.

De la même façon, l’utilisation de l’API implique le respect et la conformité aux principes du règlement générale sur la protection des données.

En ce qui concerne le jeu de données d’entrainement issu de Kaggle, il est téléchargé directement sur le site et se présente sous la forme d’une table comme celui qui nous avons présenté au-dessus. Il est nécessaire d’avoir un compte afin de télécharger.

## Une méthode d’automatisation de collecte

Ici nous avons décidé d’opter pour une mise à jour planifier plutôt qu’en temps réel. La mise à jour en temps réel permet de garantir la cohérence des données et de l’analyse à tout moment, mais dans notre cas cela n’est pas optimal. En effet, cela impliquerait d’interroger en continue l’api, or les appels sont limités. Nous pouvons dire que nous faisons du « quasi-temps réel » puisque dès lors que les données existent dans la base, elles sont mises à jour toutes les heures. D’autant plus que le moment où le nombre de commentaires évolue significativement c’est dans les premières heures / jours après la publication de la vidéo.

Lors de la première analyse, l’utilisateur entre une url dans l’application, ce qui conduit à une première extraction et à la création d’une base de données. C’est cette action qui automatise la collecte des données pour chaque analyse. Ensuite, tout au long de la vie de la vidéo, les données associées sont automatiquement mises à jour, indépendamment du fait que l’utilisateurs interagisse avec l’application ou non. Cette fonction de mise à jour fait l’objet d’un service à part qui tourne en parallèle de l’application, mais dont on suit le déroulement.

Ainsi, si la première extraction est soumise à une action, la mise à jour est totalement automatique. Elle s’appuie sur les différentes classent qui constituent le processus ETL. La première étape consiste à recenser l’ensemble des url analysés stockés dans nos bases de données. Ensuite le processus ETL est appelé pour chaque url ~~en précisant dans le code grâce à un paramètre qu’il s’agit d’une mise à jour et non d’une première extraction~~.

Une image contenant texte, capture d’écran, carte de visite

Le contenu généré par l’IA peut être incorrect.Cette fonction est déployée dans prefect flow, *un framework de gestion de workflows data, via la commande suivante :*

Ainsi, nous pouvons planifier son exécution tous les jours à intervalles d’une heure entre chaque exécution à partir d’une date donnée. Le code complet *synchronistaion.py* est disponible en annexe.

Les flows exécutés pendant les mises à jour sont monitorés via l’interface de prefect flow.

En termes de mise à jour, il était initialement prévu de n’ajouter que les nouveaux commentaires dans les bases de données. Cependant, à chaque commentaire est associé à un nombre de likes qui peut évoluer et le commentaire lui-même peut être modifié ou supprimé. Par conséquent, à chaque mise à jour nous faisons une update de ce qui existe déjà et ajoutons les nouvelles données.

# Stockage des données structurées et non structurées

## Une stratégie de stockage et un modèle de données

La stratégie de stockage adoptée repose une architecture conçue pour répondre aux besoins du client souhaitant analyser son contenu. Chaque chaîne y est identifiée comme une entité indépendante, ce qui permet de compartimenter les données selon une approche documentaire. Plus simplement, il y a une base par chaine You Tube et chacune contient un ensemble de collections qui recensent les données relatives à chaque vidéo. Cette approche garantit l’isolation entre les éventuels futurs utilisateurs et optimise l’accessibilité aux données. Elle s’inscrit dans une logique d’analyse séquentielle des vidéos, tout en conservant une certaine flexibilité pour des évolutions futures. Comme l’analyse de groupes de vidéos notamment pour les vidéos qui portent sur le même concept, les playlists. Dans cette éventualité nous pourrions regrouper tous les commentaires de plusieurs vidéos dans une base contenant 1 seule collection avec l’ensemble des documents. Cependant puisque nous sommes soumis à un quota pour la collecte cette option n’est pas encore possible.

Cette organisation permet l’utilisation des données selon 2 axes :

* **Analytique** : la méthode de stockage permet de stocker des données brutes et de faire des enrichissements successifs de méta données et des résultats d’analyse (sentiments, futures métriques de YouTube, autres idées d’an analyse…). Ainsi elle permet la production de tableaux de bords *de faire des agrégations, pour une même vidéo, des commentaires en fonction des résultats de l’analyse (sentiment et topic modeling) de manière simple et sans jointures en python.*
* **L’accessibilité** : les données sont disponibles et accessible pour chaque utilisateur sans problème de partitionnement. En effet, puisque l’analyse est menée par un client sur son contenu, il n’est pas nécessaire et même exclu de pouvoir croiser les informations entre les différentes chaines. Il est plus simple ainsi de sécuriser les données.

L’idée est que toutes les informations soient disponibles au même endroit pour une chaine et une vidéo donnée.

Notre stratégie de stockage s’articule donc autour de plusieurs bases de données qui correspondent chacune à des chaînes différentes (toujours dans l’éventualité des plusieurs utilisateurs). Dans une base on trouve une collection avec autant de documents que de commentaires analysés, appartenant à une vidéo. C’est dans les documents que l’on trouve les données. Si une base contient plusieurs collections, cela signifie pour cette chaine plusieurs vidéos ont été analysées.

Les bases sont nommées par rapport à l’identifiant de la chaine correspondante et les collections par rapport à l’identifiant de la vidéo. Ainsi, l’organisation peut s’illustrer par l’image suivante :

Une image contenant texte, capture d’écran, Police

Le contenu généré par l’IA peut être incorrect.

La structure / le modèle de chaque doucement suit la structure d’un document JSON avec des clés et des valeurs comme le montre le schéma suivant : \*

*~~Extrait de modèle de données avant maj et après maj pour monter la fléxibilité de mongodb~~:*

Une image contenant texte, logiciel, Logiciel multimédia, capture d’écran

Le contenu généré par l’IA peut être incorrect.

*Une image contenant texte, capture d’écran, Police

Le contenu généré par l’IA peut être incorrect.*

En ce qui concerne la manipulation, les données sont essentiellement manipulées avant d’être stockée dans la base données lors du processus ETL. Une fois stockées, elle se fait via python grâce à un connecteur qui permet la connexion avec la base de données. Les seules manipulations effectuées sont l’écriture /mise à jour dans la base de données, la lecture pour l’analyse et l’agrégation pour l’affichage des résultats. Ce sont des manipulations simples et qui ne nécessitent pas de jointures.

## Une base de données et une solution de stockage Big Data

Nous avons sélectionné MongoDB, un système de gestion de base de données (SGBD) orienté document (NoSQL), pour son affinité naturelle avec les documents JSON et la flexibilité du schéma qui permet l’enrichissement successif des collections et documents. En effet, dans notre cas il n’est pas nécessaire d’utiliser une base de données relationnelle car les chaînes sont indépendantes les unes des autres. Même si une personne peut avoir plusieurs chaînes, les contenus sont généralement différents et donc une analyse croisée ne serait pas pertinente. De plus, les bases de données relationnelles sont soumises à des schéma de données stricte, incompatible avec des données textuelles hétérogènes.

Notons que dans le classement des solution existantes, MongoDB est classé 5ème en termes de popularité et est le seul SGBD orienté document (de manière native) sur les 10 premiers.

De plus, les données sont facilement manipulables avec python grâce au connecter officiel PyMongo. Il permet d’effectuer les opérations CRUD et les requêtes d'agrégation directement depuis le code python. ~~A l’inverse PostgreSQL qui est moins souples en termes de calculs complexes.~~

Nous avons donc une solution de stockage Big Data qui fonction avec MongoDB conteneurisé et déployée via Docker pour garantir la portabilité et la reproductibilité. Scallable horizontalement.

# Structuration, transformation et enrichissement des données

## Outils et technologies de traitement des données

Le traitement des données textuelles s'appuie sur un ensemble de bibliothèques python spécialisées dans le traitement de texte et l’apprentissage automatique. Nos données étant déjà structurées, seuls les traitements nécessaires à l’analyse sont effectués.

Dans un premier temps, afin d’analyser les commentaires, il est nécessaire d’appliquer certaines transformations au texte brut. Ces techniques de prétraitement sont déjà implémentées dans plusieurs bibliothèques comme Spacy et NLTK. Ces 2 bibliothèques sont assez complètes en termes de traitements de données (stopwords, modèles de words embeding…). Cependant, nous avons privilégié l’utilisation de Spacy pour la majorité des traitements, pour sa rapidité et sa plus grande affinité avec à la langue française par rapport à NLTK. Nous avons associé Spacy à d’autres bibliothèques :

* Re - regex pour optimiser le nettoyage avec le traitement d’expressions régulières (certains patterns comme les liens, les hastag…)
* Unidecode pour convertir les caractères spéciaux en équivalent ASCII
* Langdetect permet de détecter la langue et de filtrer les données
* ~~NLTK pour la tâche de lemmatisation.~~
* String : pour la ponctuation

Ensuite, pour les vectoriser, c’est-à-dire représenter le texte sous forme numérique, il y a 3 types de méthodes :

* Les méthodes bag of words
* Les techniques d’embeding
* L’utilisation de modèles pré-entrainés

*\*Parmi les outils/ modèles de vectorisation de textes nous avons sélectionné la méthode Word2vec permet de concerver la sémentique. \**

**Enfin, nous avons décidé d’intégrer l’analyse de sentiment en tant que transformation / enrichissement des données. Encore une fois, les méthodes et modèles nécessaires sont déjà implémentées dans certaines bibliothèques python. Là aussi, il y a plusieurs méthodes possibles :**

* **Méthode qui se base sur le lexique : Vader, textblob, feel**
* **Machine learning classique : scklearn**
* **Modèles pré-entrainés de type transformers : hugging face**
* **Réseau de neurones**

**Ici notre choix sera déterminé par une analyse des performances des différents modèles. Toutefois, nous pouvons déjà dire que Vader n’est pas initialement adapté pour la langue française, mais s’il est toujours possible de traduire le texte, à la différence de Textblob. De plus, le choix du modèle peut également influencer le choix de la méthode de vectorisation.**

Enfin, en ce qui concerne la manipulation de données nous avons privilégié pandas car nous utilisons des dataframes.

## Transformation des données

L’ensemble des traitements sont détaillés dans le script *transformation.py* en annexe et se divisent donc en 3 parties.

* **Nettoyage textuel**

La première étape est le découpage du texte en token, ici en mots, tout en conservant l’ordre. Le token est notre unité d’analyse, comme il s’agit des mots, chaque token est généralement séparé par un espace. On filtre l’ensemble des tokens par rapport à une liste de mots et caractères indésirables qui comprend des stop-words, la ponctuation et les chiffres. Il s’agit de retirer les token qui ne portent aucune information tel que : *le, la, de, une, « , », ?, !...* On en profite également pour regrouper certaines expressions (ex : noms propres) qui sont porteuse de sens et en supprimer d’autre, tel que les url.

La seconde étape consiste à uniformiser nos tokens en mettant tout en minuscule, en retirant les accents et en réduisant certains mots à une racine. Cette dernière étape permet de réduire le nombre de tokens différents. En effet « Internet » et « internet » sont traités comme 2 tokens différents alors qu’il s’agit du même mot / de variables. En ce qui concerne la racinisation des mots, il existe 2 méthodes la « lemmatisation » qui réduit le mot à sa véritable racine et le « stemming » qui retire les terminaisons des mots, ce qui peut donner des mots qui n’ont pas de sens. Nous appliquons donc la lemmatisation qui réduit / regroupe les tokens sous une racine commune qui a du sens.

Une fois ces étapes terminées nous avons ajouté une étape qui filtre selon la taille du token, si le token ne contient que 3 caractères ou moins il est supprimé. Nous partons du principe qu’un token aussi court n’apportera pas de valeur à l’analyse.

*A titre d’exemple, la phrase « » deviendrait « ».* Une image contenant texte, capture d’écran, Police

Le contenu généré par l’IA peut être incorrect.

* ***La vectorisation***

*\**

* ***L’analyse***

*\**

*Chaque transformation ajoute une colonne dans la base MongoDB*. Pour le topic modeling nous avons décider ne pas l’intégrer au pipeline ETL en tant que transformation.

## Processus ETL, automatisation et orchestration du traitement des données

Notre processus ETL extrait, transforme et enrichie les données extraites de l’api, comme expliqué précédemment, puis charge les données dans mongodb. Le processus repose essentiellement sur les scripts python *extraction.py, transformation.py* et *load.py* qui contiennent toutes les fonctions essentielles. C’est le script *etl.py* qui permet d’orchestrer les différentes étapes et qui est en réalité le flow principal exécuté lorsque l’on utilise l’outil. Le point d’entrée du processus, est l’interface streamlit où l’utilisateur entre l’url d’une vidéo YouTube qui appartient à sa chaine. Suite à cette action notre processus ETL s’active, à condition que les données relatives à la vidéo n’existent pas déjà. Cette condition permet d’interroger directement la base de données et de gagner du temps si les données ont déjà été collectées. S’il s’agit de la première fois que cette url est saisie, les processus est actionné. Ce processus est également programmé pour s’exécuter toutes les heures de manière automatique pour mettre à jour la base de données.

Le processus complet suit le schéma ETL automatisé suivant :

*Une image contenant diagramme, capture d’écran, ligne, texte

Le contenu généré par l’IA peut être incorrect.*

1. ***Extraction****: récupération des données (YouTube API)*
2. ***Transform****: traitement NLP + enrichissement des données*
3. ***Load****: insertion dans la base MongoDB*

*Chaque étape clé représente un flow avec des task définies :*

*Une image contenant texte, capture d’écran

Le contenu généré par l’IA peut être incorrect.*

# Sécurisation des données

## Une politique de sécurisation des données

Les enjeux de sécurité sont inhérents au type de données traités et à l’architecture du projet. En effet, les commentaires sont des données « sensibles » dont le traitement doit être adapté en conséquence. Ils sont soumis à des contraintes légales (RGPD et conditions d’utilisation de l’API) auxquelles nous sommes soumis. Ainsi, notre politique de sécurité définit l’ensemble des mesures techniques et organisationnelles permettant de garantir l’intégrité du système (de bout en bout) et la confidentialité des données. \*\*contenu à caractère sensible, religion, politique …\*

*Sécurité dans les données*

Afin de préserver l’identité des auteurs de commentaires, nous avons limiter la collecte des données aux données strictement nécessaires. Donc la base de données ne contient aucune donnée d’identification tel que le pseudo, mis à part l’identifiant du commentaire.

*Sécurité dans l’usage*

Dans la même logique, les dashboard de résultats ne serons en mesure d’afficher que quelques commentaires et en aucun cas l’ensemble des commentaires et aucunes données d’identification ne sera présentée ou accessible pour l’utilisateur. Dans le même temps, l’analyse se fait au niveau macro, c’est-à-dire qu’il n’est pas possible de cibler un individu ou groupe/ minorité pour en faire l’objet de l’analyse

*Sécurité dans le stockage*

De plus, puisque les données sont mises à jour toutes les heures elles restent conformes à la réglementation RGPD tant YouTube la respecte. Si les données sont mises à jour ou supprimées sur YouTube elles le seront également dans nos bases. ***Mise à jour horaire garantie que les données ne sont pas stockées plus que nécessaire****.*

*Sécurité des flux*

*Sécurité dans l’organisation*

De plus, l’architecture de l’outil doit également garantir que seul l’auteur du contenu soit en mesure de lancer les analyses et accéder aux résultats. Donc la mise en place d’une authentification au niveau de l’interface streamlit. Il faut dans la même logique sécuriser l’ensemble des flux de données, notamment entre l’application streamlit et la base de données en s’assurant que ce qui entré est bien un url You Tube valide. La sécurisation de l’architecture passe par donc par la sécurisation des flux entre les entités du système et par la création des rôles et de droits différents.

Limiter l’activité de la cession streamlit

* ***Intégrité****: logs de traitement, backups automatisés*
* ***Accès restreint****: authentification pour requêtage de la base*
* ***Traçabilité****: versioning des données et des modèles*

*Les logs sont stockés localement et des sauvegardes de la base sont effectuées toutes les 24h.*

Sécuriser les flux de données

## Un schéma d’architecture de sécurité

plaintext

CopierModifier

[YouTube API]

|

v

[Script Python Extraction]

|

v

[Base PostgreSQL Docker] <------> [Traitements NLP Python]

|

v

[Dashboard utilisateur (Flask/Streamlit)]

* *Chiffrement des accès*
* *Accès à la base restreint aux conteneurs approuvés*
* *Hébergement sécurisé sur AWS (optionnel en production)*
* *CI/CD pour déploiement sûr avec tests*

# Annexe

[1] *extraction.py*

*[2]* \*dbengine ranking : comparaison avec les autres solutions

[3] synchronisation.py

Une image contenant texte, nombre, Police, capture d’écran

Le contenu généré par l’IA peut être incorrect.

<https://db-engines.com/en/ranking>

## IV. SÉCURISATION DES DONNÉES

### Politique de sécurisation et conformité RGPD

#### 1. Principes fondamentaux de sécurisation

La politique de sécurisation des données s'articule autour des principes fondamentaux du RGPD et des bonnes pratiques de sécurité informatique :

**Minimisation des données**

* Collecte exclusive des champs strictement nécessaires à l'analyse
* Suppression automatique des identifiants personnels non pertinents
* Rétention limitée dans le temps avec purge automatique après 24 mois

**Finalité et proportionnalité**

* Usage restreint aux fins d'analyse de sentiment dans un cadre pédagogique
* Interdiction de réutilisation des données à des fins commerciales
* Documentation claire des finalités dans les métadonnées de traitement

**Transparence et traçabilité**

* Journalisation complète des accès et modifications
* Versioning des données et des traitements appliqués
* Documentation technique accessible aux parties prenantes

#### 2. Classification et sensibilité des données

**Données publiques (Niveau 1)**

* Commentaires publics YouTube accessibles sans authentification
* Métadonnées vidéo (titre, description, statistiques)
* Mesures de protection : chiffrement en transit, accès contrôlé

**Données pseudo-personnelles (Niveau 2)**

* Pseudonymes des commentateurs (anonymisés par hachage)
* Horodatages précis pouvant permettre une ré-identification
* Mesures de protection : chiffrement au repos, anonymisation renforcée

**Données dérivées (Niveau 3)**

* Scores de sentiment et classifications automatiques
* Embeddings vectoriels et représentations transformées
* Mesures de protection : accès restreint, audit trail complet

#### 3. Droits des personnes concernées

**Implémentation des droits RGPD**

* **Droit d'accès** : API de consultation des données traitées par identifiant anonymisé
* **Droit de rectification** : procédure de correction des données erronées
* **Droit à l'effacement** : suppression automatique sur demande motivée
* **Droit à la portabilité** : export des données en format JSON structuré

### B. Architecture de sécurité technique

#### 1. Sécurisation de l'infrastructure

**Isolation par conteneurisation**

# Configuration Docker sécurisée

security\_opt:

- no-new-privileges:true

read\_only: true

user: "1001:1001"

cap\_drop:

- ALL

networks:

- internal\_network

**Chiffrement multicouche**

* **En transit** : TLS 1.3 pour toutes les communications API et base de données
* **Au repos** : AES-256 pour le stockage des données sensibles
* **En mémoire** : chiffrement des variables d'environnement contenant les secrets

**Contrôle d'accès granulaire**

-- Création de rôles avec privilèges minimaux

CREATE ROLE youtube\_reader;

GRANT SELECT ON comments, videos TO youtube\_reader;

CREATE ROLE youtube\_processor;

GRANT SELECT, INSERT, UPDATE ON comments TO youtube\_processor;

GRANT SELECT ON videos TO youtube\_processor;

#### 2. Sécurisation des API et des accès

**Authentification et autorisation**

* Clés API YouTube stockées dans des secrets managés (HashiCorp Vault)
* Authentification par certificats pour les accès base de données
* Rate limiting pour prévenir les attaques par déni de service

**Audit et monitoring de sécurité**

import logging

from datetime import datetime

def log\_data\_access(user\_id, action, resource, details=None):

security\_logger.info({

'timestamp': datetime.utcnow().isoformat(),

'user\_id': user\_id,

'action': action,

'resource': resource,

'details': details,

'ip\_address': request.remote\_addr

})

#### 3. Gestion des incidents et continuité

**Plan de réponse aux incidents**

1. **Détection** : monitoring automatisé des anomalies d'accès
2. **Containment** : isolation automatique des composants compromis
3. **Éradication** : analyse forensique et suppression des vulnérabilités
4. **Récupération** : restauration depuis les sauvegardes chiffrées
5. **Leçons apprises** : mise à jour des procédures et formations

**Sauvegarde et récupération**

* Sauvegardes chiffrées quotidiennes avec rétention sur 90 jours
* Tests de restauration mensuels pour valider l'intégrité
* Plan de continuité d'activité avec RTO < 4h et RPO < 1h

**Mise à jour et maintenance sécurisée**

* Cycle de mise à jour automatisé des dépendances
* Scan de vulnérabilités hebdomadaire avec Trivy et Snyk
* Rotation automatique des secrets et certificats

Cette architecture de sécurité garantit la protection des données tout au long de leur cycle de vie, depuis la collecte jusqu'à l'analyse, en respectant les exigences réglementaires et les meilleures pratiques de sécurité informatique.

***Gestion des erreurs et de la robustesse****L'automatisation intègre des mécanismes de résilience :*

* ***Retry automatique****: nouvelle tentative en cas d'échec temporaire de l'API*
* ***Logging détaillé****: traçabilité complète des opérations pour faciliter le débogage*
* ***Alerting*** : notification en cas d'échec critique nécessitant une intervention manuelle

***Métriques techniques***

* *Temps d'exécution par étape du pipeline*
* *Taux d'erreur des appels API*
* *Utilisation mémoire et CPU pendant les transformations*

***Métriques métier***

* *Nombre de commentaires traités par batch*
* *Distribution des sentiments détectés*
* *Qualité des topics extraits (coherence score)*

***Alerting***

* *Notification en cas d'échec critique du pipeline*
* *Alertes sur les déviations de qualité des données*
* *Monitoring des quotas API approchant les limites*