ECOLE SUPERIEIURE PRIVEE DES TECHNOLOGIES DE
L'INFORMATION ET DE MANAGEMENT DE NABEUL

Mini Projet

Big Data

Application de Monitoring avec la Stack ELK et une Interface Web Flask

Elaboré par: **Daas Maissa**

Encadré par: **Mohamed Najeh Issaoui**

Année universitaire 2024 - 2025

Table des matières

Table des figures

Liste des tableaux

Introduction générale

Dans un monde de plus en plus axé sur le numérique, la gestion efficace des données et des événements générés par des applications est devenue une nécessité pour les entreprises modernes. Les systèmes de monitoring des logs jouent un rôle crucial en fournissant des informations en temps réel sur les performances des applications, en détectant les anomalies et en facilitant les prises de décision basées sur des données fiables.

Ce projet vise à concevoir et développer une solution complète de gestion et de visualisation des logs en s'appuyant sur la Stack ELK (Elasticsearch, Logstash, Kibana). L'objectif principal est de centraliser les logs provenant de diverses sources, de les indexer pour une recherche rapide, et de fournir des outils interactifs pour leur analyse. En complément, une interface web sera développée pour simplifier l'interaction avec le système, permettant notamment l'injection, la recherche et le filtrage des logs.

Au fil de ce rapport, nous détaillerons les étapes de création de cette application, les défis rencontrés, ainsi que les solutions mises en œuvre pour les surmonter. Notre rapport se compose de trois chapitres, en plus d'une introduction générale et d'une conclusion :

- Le premier chapitre intitulé « **Présentation générale du projet** » présente la problématique, la solution proposée et une analyse de l'existant.
- Le deuxième chapitre détaille l'architecture technique et fonctionnelle de la solution.
- Le troisième chapitre se concentre sur l'implémentation pratique, couvrant à la fois les aspects backend et frontend.

Première partie

Aspects Théoriques

Chapitre 1

Présentation générale du projet

1.1 Introduction

Ce chapitre présente une vue d'ensemble du projet en examinant la problématique à résoudre, en proposant une approche adaptée, et en offrant une analyse critique de l'existant afin de mieux comprendre les limites des méthodes actuelles et de justifier la pertinence de la solution proposée.

1.2 Problématique

Dans un environnement numérique complexe, La gestion des logs générés par des applications internes nécessite une centralisation efficace pour permettre une analyse en temps réel des performances et des anomalies, facilitant ainsi la prise de décision.

1.3 Solution proposée

La solution consiste à implémenter une application utilisant la Stack ELK pour centraliser, indexer, et visualiser les logs via une interface web simple, offrant également des fonctionnalités avancées de gestion et de filtrage.

1.4 Analyse de l'existant

Actuellement, La Stack ELK est une solution éprouvée et largement utilisée pour la gestion et l'analyse des logs, avec des intégrations solides permettant une visualisation via Kibana et une indexation efficace dans Elasticsearch.

1.5 Conclusion

Ce chapitre a présenté la problématique, la solution proposée et une analyse de l'existant. Le prochain chapitre abordera l'étude conceptuelle, incluant l'architecture, les acteurs et les diagrammes clés.

Chapitre 2

Étude conceptuelle

2.1 Introduction

Dans ce chapitre, nous présentons la modélisation du projet à travers les architectures technique et fonctionnelle, pour une vision claire et structurée du système.

2.2 Définition du Besoin

2.2.1 Analyse des données à traiter

2.2.1.1 Fichier CSV

Ce fichier CSV contient des logs système, utilisés pour la gestion et le diagnostic des performances ou de la sécurité des systèmes. Il est organisé sous forme tabulaire avec 15 colonnes, détaillées comme suit :

- **LineId** : Identifiant unique pour chaque entrée.
- **Label** : Colonne sans valeur dans ce fichier (indiquée par un tiret).
- **Timestamp** : Horodatage UNIX représentant l'heure et la date de l'événement.
- **Date** : Date lisible au format YYYY.MM.DD.
- **User** : Nom de l'utilisateur ou du serveur impliqué dans l'événement (ex. : `dn228`).
- **Month** : Mois de l'événement (ex. : `Nov`).
- **Day** : Jour de l'événement (ex. : `9`).
- **Time** : Heure précise de l'événement (ex. : `12:01:01`).
- **Location** : Chemin ou référence complète à l'utilisateur/serveur (ex. : `dn228/dn228`).
- **Component** : Processus ou module du système impliqué (ex. : `crond(pam_unix)`).
- **PID** : Identifiant du processus (Process ID).
- **Content** : Description textuelle de l'événement (ex. : *session closed for user root*).

- **EventId** : Identifiant unique pour chaque type d'événement (ex. : `E117`).
- **EventTemplate** : Modèle standardisé décrivant le type d'événement (similaire à la colonne **Content**).

Ces logs documentent les événements système, tels que l'ouverture ou la fermeture de sessions utilisateur, et l'exécution de tâches planifiées, ce qui permet de surveiller les performances et d'identifier des anomalies ou incidents de sécurité.

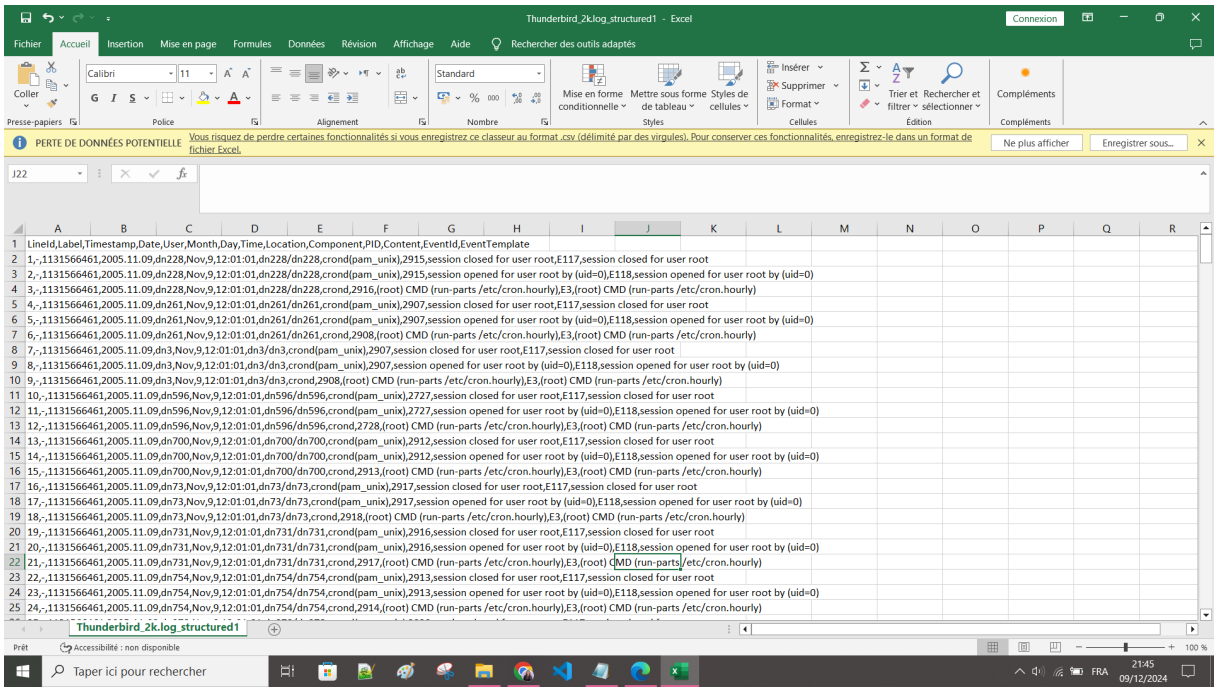


FIGURE 2.1 – « Fichier CSV »

2.2.1.2 Fichier JSON

Ce fichier JSON représente un journal d'événements contenant une liste chronologique d'activités et d'événements systèmes, utilisateur et réseau. Chaque objet du fichier décrit un événement distinct à l'aide de quatre propriétés :

- **lineid** : un identifiant unique pour chaque ligne d'événement, permettant de les différencier.
- **timestamp** : l'horodatage de l'événement, au format ISO 8601, précisant la date et l'heure exactes.
- **EventTemplate** : une brève description ou un modèle de l'événement, tel que "User Login", "System Boot", ou "Error : Disk Full".
- **eventid** : un identifiant spécifique de l'événement, comme "USR1" pour les connexions utilisateur ou "SYS1" pour les événements système.

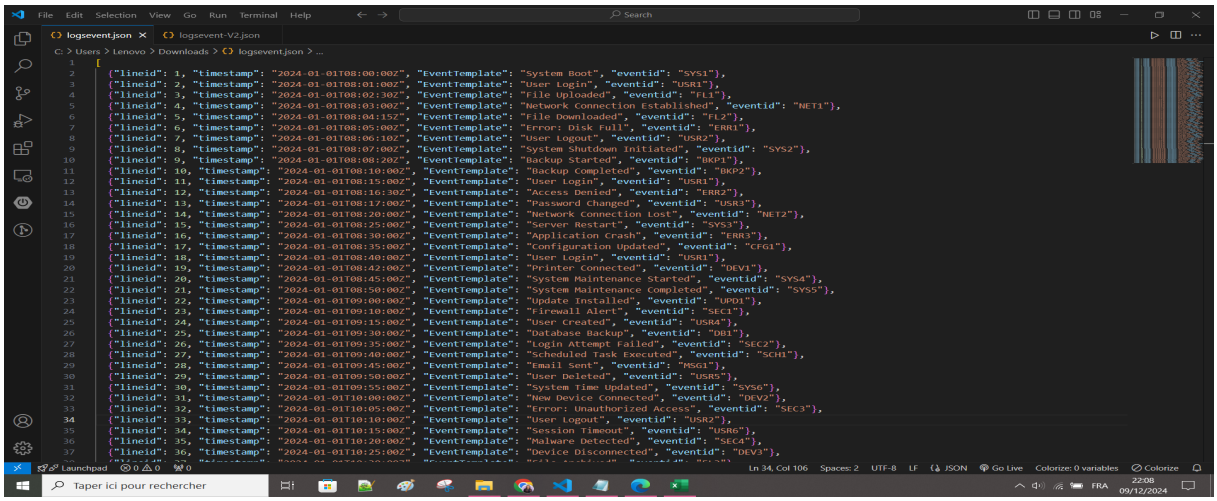


FIGURE 2.2 – « Fichier JSON »

2.2.2 Diagramme de cas d'utilisation

Le diagramme de cas d'utilisation illustre le comportement fonctionnel de notre mini-projet. Dans ce diagramme, l'utilisateur est l'acteur principal. Il peut effectuer les actions suivantes :

- Upload de fichiers CSV ou JSON : Importation de fichiers de logs dans le système pour traitement et analyse.
- Filtrage des logs et recherche : Les utilisateurs peuvent filtrer les logs selon divers critères tels que le timestamp, l'utilisateur, l'emplacement, l'ID d'événement, etc., afin de trouver des informations spécifiques.
- Visualisation des logs : Les logs sont affichés sous forme de graphiques interactifs, avec des options pour consulter les détails, organiser les données et les exporter.

Ce diagramme met en évidence les principales interactions entre l'utilisateur et les fonctionnalités du système.

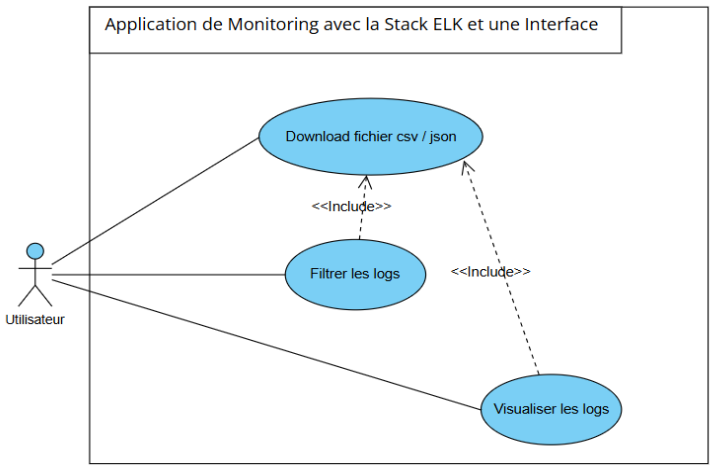


FIGURE 2.3 – « Diagramme de cas d'utilisation »

2.3 Architecture Technique Globale

Ce schéma illustre l'intégration des principaux composants du projet, notamment Logstash pour l'ingestion des données, Elasticsearch pour leur indexation, Kibana pour la visualisation, et l'application web Flask pour l'interaction utilisateur.

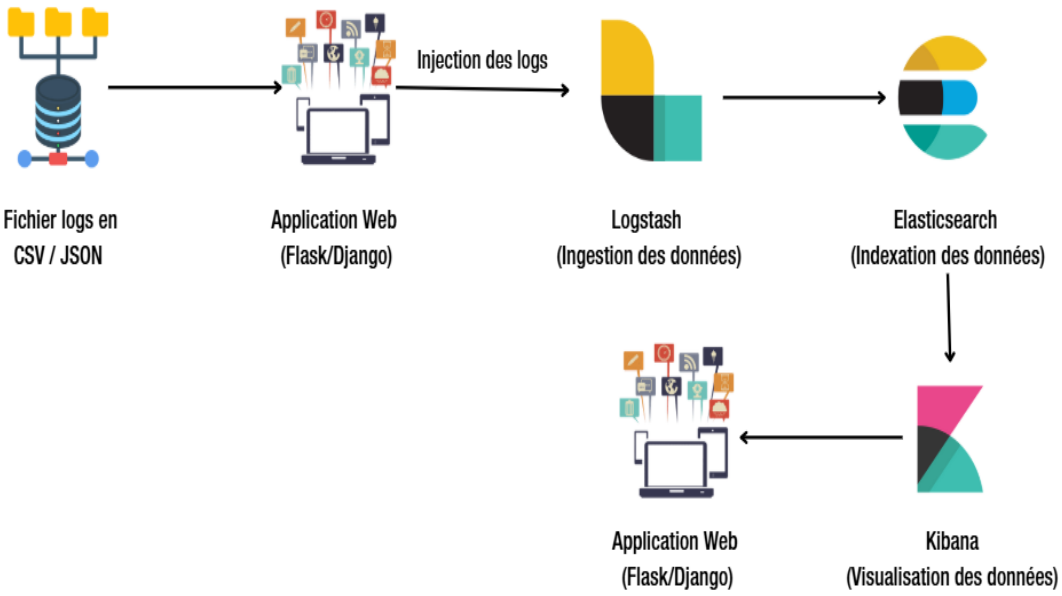


FIGURE 2.4 – « Architecture Technique Globale »

2.4 Architecture Fonctionnelle

Ce tableau présente l'architecture fonctionnelle du système de monitoring des logs. Il détaille les interactions entre les différents composants, tels que l'application web, Logstash, Elasticsearch et Kibana, illustrant comment les logs sont soumis, transformés, stockés et visualisés tout au long du processus.

Composant Source	Action	Composant Cible
Utilisateur	Soumet un fichier de logs (CSV, JSON, etc.)	Application Web
Application Web	Transfert des logs soumis à Logstash	Logstash
Logstash	Ingère et transforme les logs	Elasticsearch
Elasticsearch	Stocke et indexe les logs	Kibana
Kibana	Visualisation des logs sous forme de tableaux de bord	Utilisateur
Application Web	Permet à l'utilisateur de rechercher et filtrer les logs	Elasticsearch

TABLE 2.1 – Architecture Fonctionnelle

2.4.1 Architecture Détailée

Ce tableau décrit les composants clés du système de monitoring des logs, détaillant leur rôle et leur interaction. Il présente les tâches spécifiques de Logstash, Elasticsearch, Kibana et l'application web

Composant	Description
Logstash	- Configuration des pipelines pour lire les fichiers de logs - Application de filtres pour transformer les données - Envoi des données à Elasticsearch via `output`
Elasticsearch	- Définition des schémas d'index pour structurer les données (mapping des champs comme Timestamp, User, EventId, etc.). - Gestion des requêtes REST pour rechercher et filtrer les données.
Kibana	- Création de tableaux de bord pour visualiser les logs - Configuration des filtres et des champs à afficher dans l'interface.
Application Web Flask	- Backend pour gérer les requêtes utilisateurs (upload de fichiers). - Frontend pour l'affichage des résultats, les filtres, et l'interaction utilisateur.

TABLE 2.2 – Architecture Détailée

2.5 Conclusion

Dans ce chapitre, nous avons réalisé la modélisation de notre projet en répondant aux différentes questions liées à la conception et à la structure. Cette analyse a permis de développer le diagramme de cas d'utilisation ainsi que les architectures technique et fonctionnelle. Le prochain chapitre sera consacré à l'enchaînement des interfaces de notre application.

Deuxième partie

Réalisation et implémentation

Chapitre 3

Réalisation

3.1 Introduction

Ce chapitre est consacré à l'étude technique et à l'implémentation de notre système. Nous avons présenté les langages et outils de développement utilisés, ainsi que des captures d'écran pour illustrer l'aspect ergonomique et fonctionnel des interfaces créées.

3.2 Langages et outils de développement

3.2.1 Flask

FIGURE 3.1 – « Flask »

Flask est un micro-framework Python léger utilisé pour développer des applications web. Contrairement à d'autres frameworks plus complets, Flask offre une grande flexibilité et permet aux développeurs de choisir leurs outils et bibliothèques pour répondre aux besoins spécifiques du projet.[1]

3.2.2 Elasticsearch

FIGURE 3.2 – « Elasticsearch »

ElasticSearch est un moteur de recherche en texte brut. Il permet d'analyser, indexer et rechercher des données avec des performances proches de l'instantané. [2]

3.2.3 Logstash

FIGURE 3.3 – « Logstash »

Logstash est un outil informatique de collecte, analyse et ingestion de données. Il est généralement associé avec Elasticsearch et Kibana et il est capable d'intégrer une multitude de sources simultanément. [3]

3.2.4 kibana

FIGURE 3.4 – « Kibana »

Kibana est un outil open source de visualisation de données pour le moteur d'indexation ElasticSearch. Il permet de mettre en forme les données dans des tableaux de bord interactifs. [4]

3.3 Enchaînement des interfaces

3.3.1 Partie Front-End

3.3.1.1 Page de Téléchargement des Fichiers CSV/JSON

Cette interface initiale permet de télécharger des fichiers logs aux formats JSON et CSV, destinés à être exploités et traités ultérieurement.

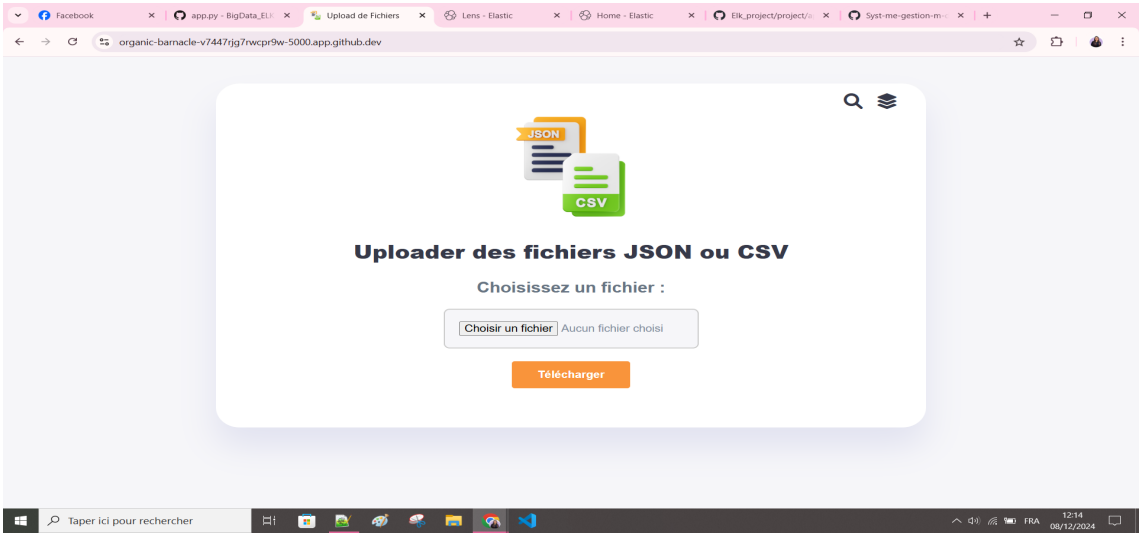


FIGURE 3.5 – « Page de Téléchargement des Fichiers CSV/JSON »

Cette interface comporte un menu avec deux boutons icônes : l'un redirige vers la page de recherche pour les fichiers JSON ou CSV, et l'autre vers le tableau de bord dédié aux fichiers CSV ou JSON.

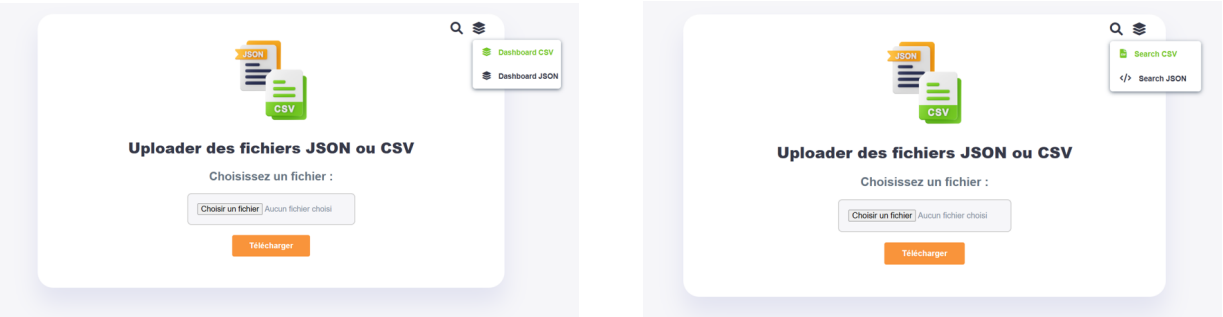


FIGURE 3.6 – « les 2 Boutons search et Dashboard »

3.3.1.2 Page de recherche

Cette interface inclut un champ de recherche où l'utilisateur peut saisir des données provenant d'un fichier CSV ou JSON, qui seront ensuite affichées ligne par ligne dans un tableau.

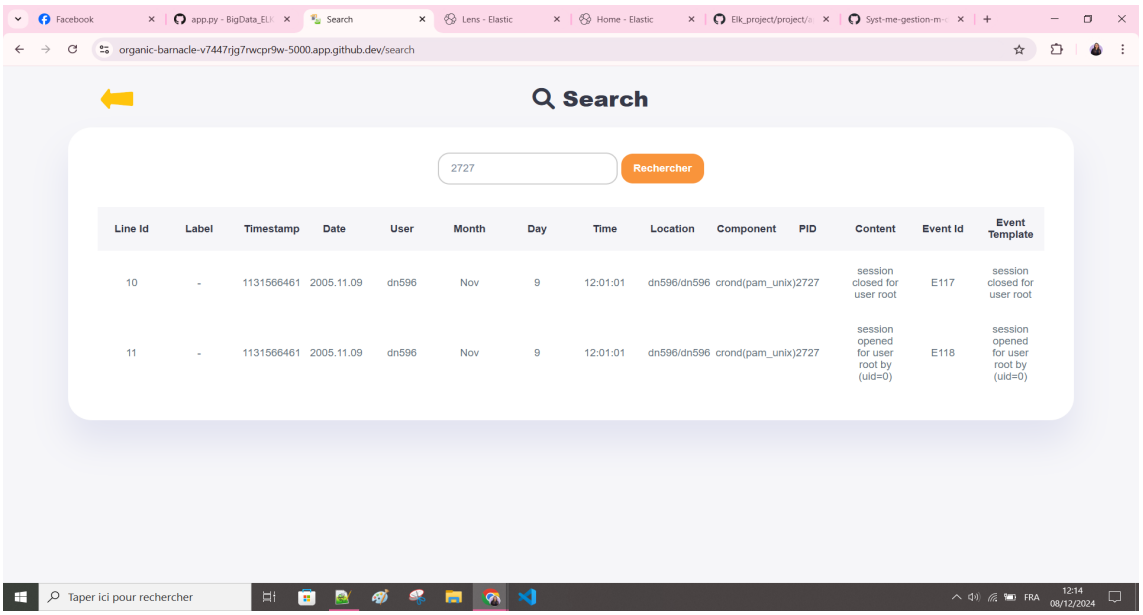


FIGURE 3.7 – « Page de recherche »

3.3.1.3 Dashboard

Cette interface constitue le tableau de bord de notre application de monitoring basée sur la stack ELK. Elle inclut une visualisation des fichiers logs au format JSON et CSV à l'aide de Kibana, accompagnée de graphiques analytiques.

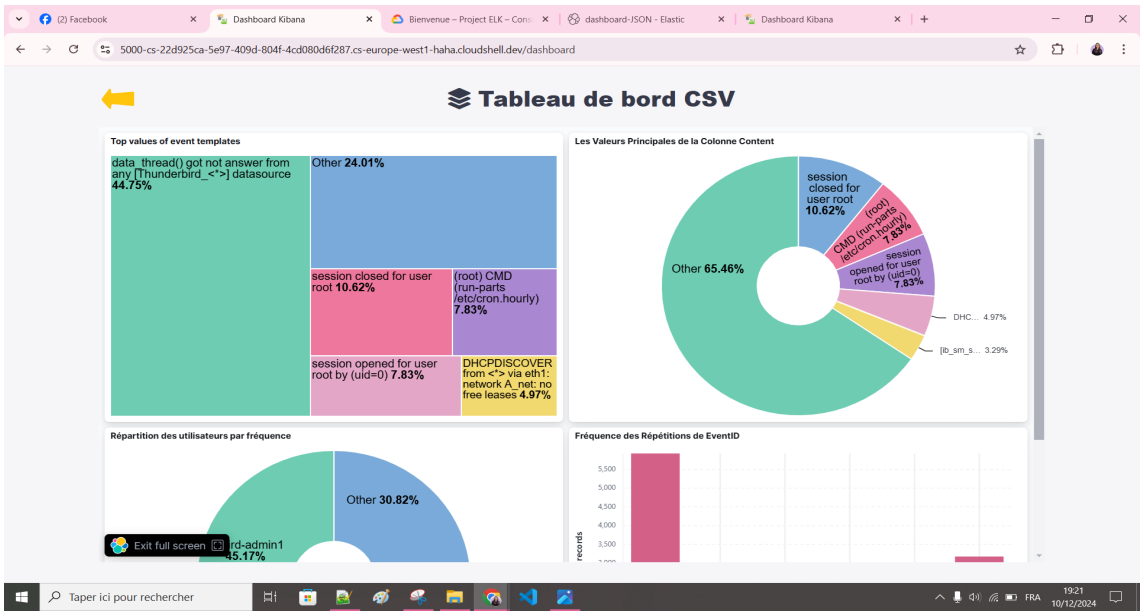


FIGURE 3.8 – « Dashboard CSV »

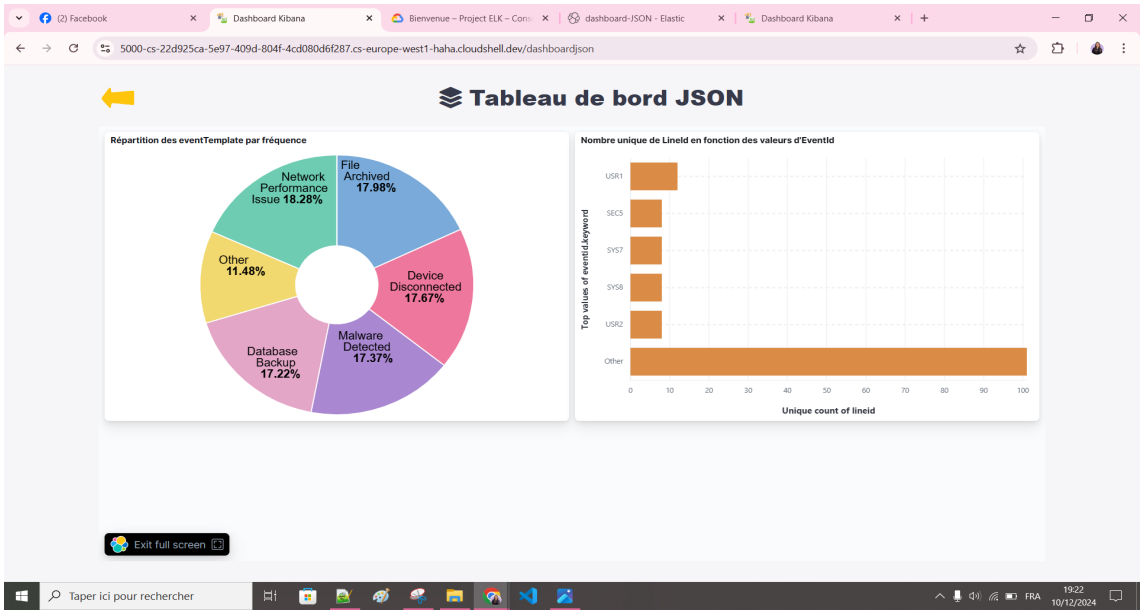


FIGURE 3.9 – « Dashboard JSON »

3.3.2 Partie Back-End

3.3.2.1 Architecture du Projet ELK avec Interface Flask

L'architecture du projet présente l'intégration de la stack ELK (Elasticsearch, Logstash, Kibana) avec une interface web Flask. Le projet utilise Docker pour orchestrer les services, et comprend la configuration de Logstash pour l'ingestion des fichiers de logs (CSV et JSON), l'indexation des données dans Elasticsearch, et la visualisation via Kibana, tout en offrant une interface Flask pour interagir avec les logs.

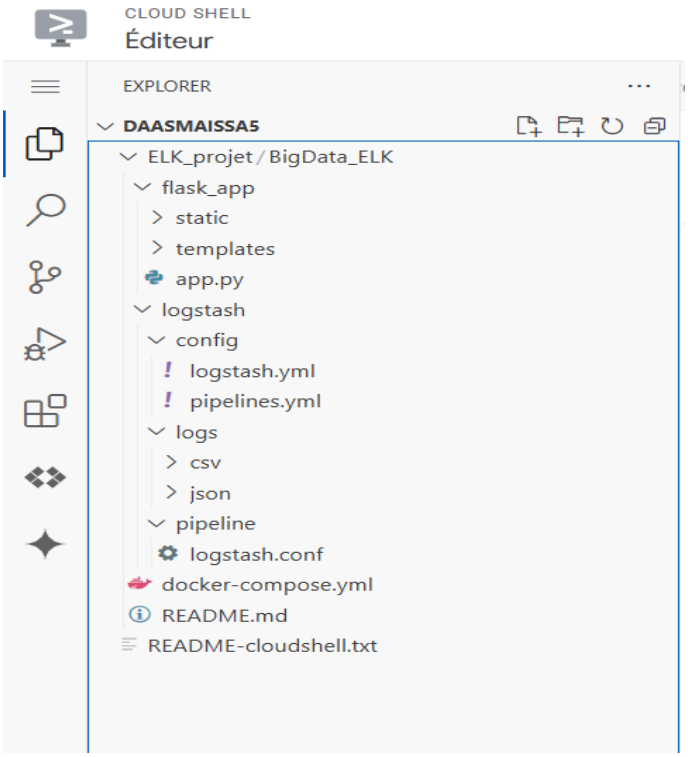


FIGURE 3.10 – « Architecture du Projet ELK avec Interface Flask »

3.3.2.2 Configuration de Logstash pour l'ingestion des fichiers CSV/JSON

La configuration de Logstash avec un fichier logstash.conf permet de lire les fichiers de logs (CSV ou JSON). Cela comprend la définition des chemins des fichiers à surveiller, l'application de filtres pour extraire ou transformer les données, et l'envoi de ces données vers Elasticsearch pour l'indexation et la recherche.

ELK_projet > BigData_ELK > logstash > pipeline > logstash.conf

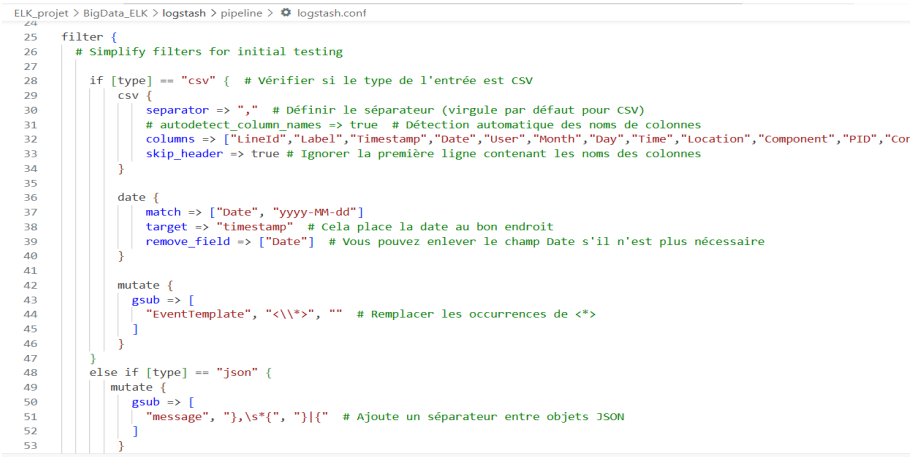
```
input {
  # lire fichier csv
    file {
    path => "/usr/share/logstash/logs/csv/*.csv"
    start_position => "beginning"
    sincedb_path => "/dev/null"
    mode => "tail"
    codec => plain { charset => "UTF-8" } # Pas besoin de multiline
    type => "csv"
    discover_interval => 15
    }

  # lire fichier (JSON)
    file {
        path => "/usr/share/logstash/logs/json/*.json"
        type => "json"
        start_position => "beginning"
        sincedb_path => "/dev/null"
        codec => json  # Utiliser json_lines pour des objets JSON séparés par des nouvelles lignes
        type => "json"
        discover_interval => 15
    }
}

filter {
  # Simplify filters for initial testing

    if [type] == "csv" {  # Vérifier si le type de l'entrée est CSV
        csv {
            separator => ","  # Définir le séparateur (virgule par défaut pour CSV)
```

FIGURE 3.11 – « Code de logstash.conf pour Lire les fichiers de logs (CSV ou JSON) »



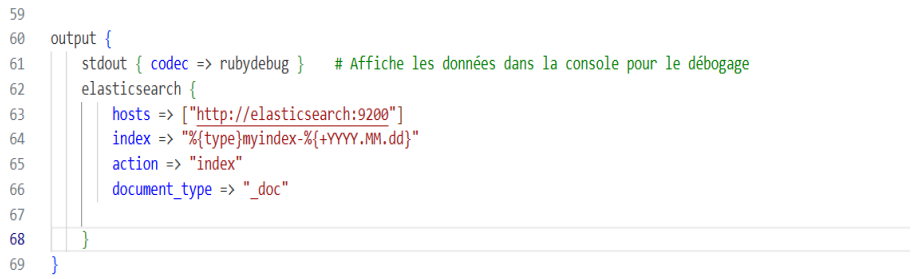
```
ELK_projet > BigData_ELK > logstash > pipeline > logstash.conf
  filter {
    # Simplify filters for initial testing

    if [type] == "csv" {  # Vérifier si le type de l'entrée est CSV
        csv {
            separator => ","  # Définir le séparateur (virgule par défaut pour CSV)
            # autodetect_column_names => true  # Détection automatique des noms de colonnes
            columns => ["LineId","Label","Timestamp","Date","User","Month","Day","Time","Location","Component","PID","Cor
            skip_header => true # Ignorer la première ligne contenant les noms des colonnes
        }

        date {
            match => ["Date", "yyyy-MM-dd"]
            target => "timestamp"  # Cela place la date au bon endroit
            remove_field => ["Date"]  # Vous pouvez enlever le champ Date s'il n'est plus nécessaire
        }

        mutate {
          gsub => [
            "EventTemplate", "<\\*>", ""  # Remplacer les occurrences de <*>
          ]
        }
    }
    else if [type] == "json" {
        mutate {
          gsub => [
            "message", "},\s*{", "}|{"  # Ajoute un séparateur entre objets JSON
          ]
        }
```

FIGURE 3.12 – « Code de logstash.conf pour extraire les données »

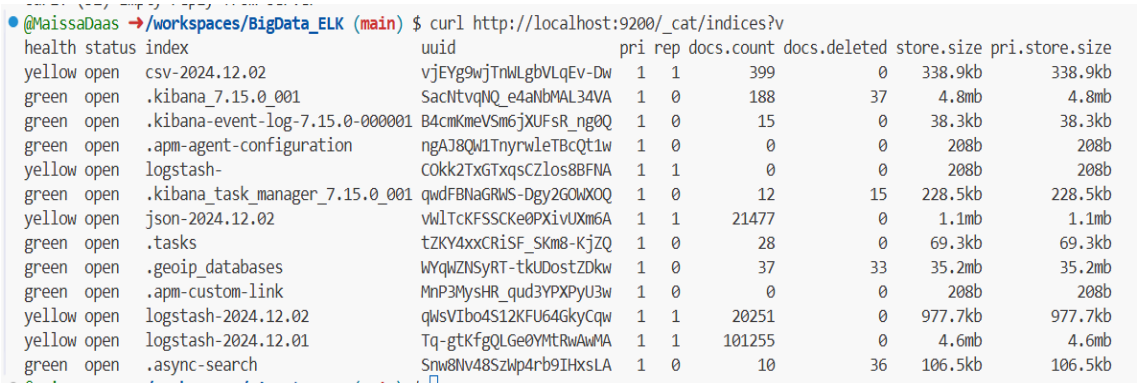


```
59
60  output {
61      stdout { codec => rubydebug }    # Affiche les données dans la console pour le débogage
62      elasticsearch {
63          hosts => ["http://elasticsearch:9200"]
64          index => "%{type}myindex-%{+YYYY.MM.dd}"
65          action => "index"
66          document_type => "_doc"
67
68      }
69  }
```

FIGURE 3.13 – « Code de logstash.conf pour envoi dles données vers Elasticsearch »

3.3.2.3 Creations des indices

Cette image montre les indices créés par Logstash et existants dans Elasticsearch, accompagnés d'informations détaillées sur leur statut, le nombre de documents et les paramètres associés.



```
@MaissaDaas →/workspaces/BigData_ELK (main) $ curl http://localhost:9200/_cat/indices?v
health status index                           uuid                   pri rep docs.count docs.deleted store.size pri.store.size
yellow open   csv-2024.12.02                  vjEYg9wjTnWLgbVLqEv-Dw   1   1        399            0    338.9kb        338.9kb
green  open   .kibana_7.15.0_001              SacNtvqNQ_e4aNbMAL34VA   1   0        188           37      4.8mb          4.8mb
green  open   .kibana-event-log-7.15.0-000001 B4cmKmeVSm6jXUFsR_ng0Q   1   0         15            0     38.3kb         38.3kb
green  open   .apm-agent-configuration        ngAJ8QW1TnyrwleTBcQt1w   1   0          0            0      208b           208b
yellow open   logstash-                       COkk2TxGTxqsCZlos8BFNA   1   1          0            0      208b           208b
green  open   .kibana_task_manager_7.15.0_001 qwdFBNaGRWS-Dgy2GOWXOQ   1   0         12           15    228.5kb        228.5kb
yellow open   json-2024.12.02                 vWlTcKFSSCKe0PXivUXm6A   1   1      21477            0      1.1mb          1.1mb
green  open   .tasks                          tZKY4xxCRiSF_SKm8-KjZQ   1   0         28            0     69.3kb         69.3kb
green  open   .geoip_databases                WYqWZNSyRT-tkUDostZDkw   1   0         37           33     35.2mb         35.2mb
green  open   .apm-custom-link                MnP3MysHR_qud3YPXPyU3w   1   0          0            0      208b           208b
yellow open   logstash-2024.12.02             qWsVIbo4S12KFU64GkyCqw   1   1      20251            0    977.7kb        977.7kb
yellow open   logstash-2024.12.01             Tq-gtKfgQLGe0YMtRwAwMA   1   1     101255            0      4.6mb          4.6mb
green  open   .async-search                    Snw8Nv48SzWp4rb9IHxsLA   1   0         10           36    106.5kb        106.5kb
```

FIGURE 3.14 – « Creations des indices»

3.3.2.4 Visualisation des Graphiques dans Kibana

Cette image présente les graphiques et visualisations générés dans Kibana à partir des données indexées dans Elasticsearch, offrant une vue d'ensemble des logs et permettant une analyse détaillée des événements.

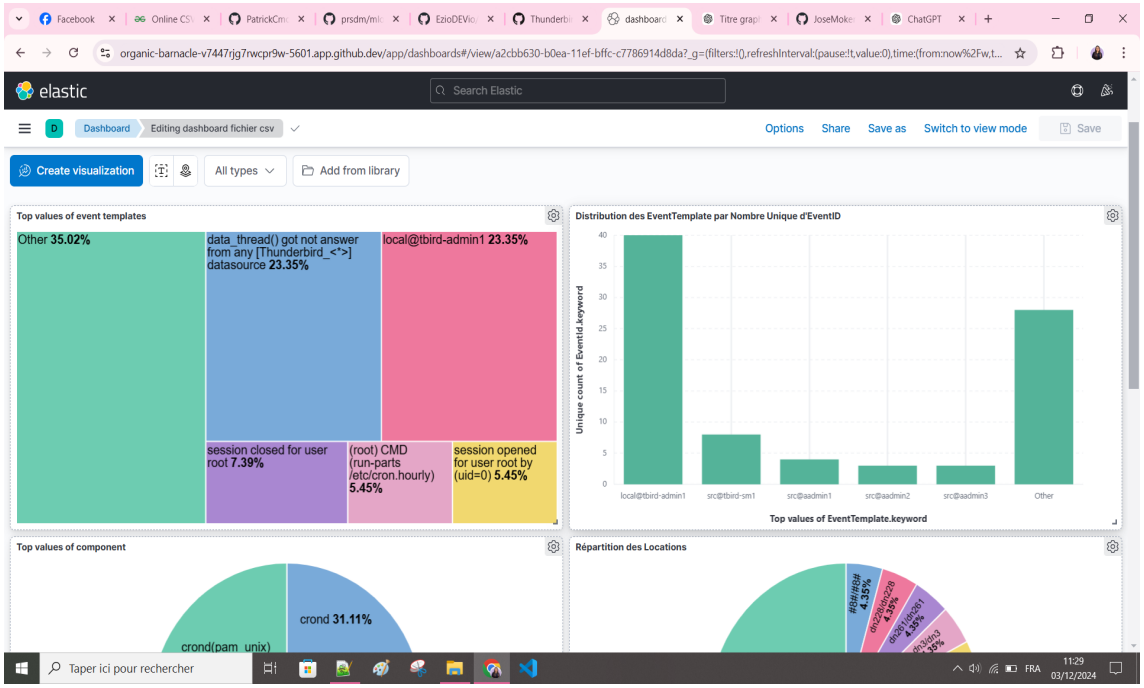


FIGURE 3.15 – « Visualisation des Graphiques dans Kibana »

3.4 Conclusion

Dans ce dernier chapitre, nous avons présenté les langages et outils de développement utilisés, ainsi que toutes les interfaces de notre projet, avec une explication détaillée, en précisant la fonctionnalité de chaque bouton pour en faciliter l'utilisation.

Conclusion générale

Le projet de mise en place d'un système de monitoring des logs à l'aide de la Stack ELK (Elasticsearch, Logstash, Kibana) et d'une interface web Flask a permis de développer une solution complète pour la gestion, l'analyse, et la visualisation des logs générés par diverses applications internes. Ce système répond aux besoins exprimés par l'entreprise en centralisant et en traitant les logs dans un environnement scalable et performant.

La première étape du projet a consisté en l'analyse des données et la définition des cas d'utilisation pour l'application web. Cela a permis de cerner les fonctionnalités nécessaires telles que l'injection de logs, les filtres de recherche, et la gestion des formats de fichiers compatibles (CSV, JSON). L'architecture technique et fonctionnelle a été soigneusement conçue pour intégrer les composants de la Stack ELK, en assurant une communication fluide entre Logstash, Elasticsearch, et Kibana, tout en intégrant l'interface utilisateur via Flask .

L'implémentation a permis de réaliser une solution efficace, avec une application web permettant aux utilisateurs d'interagir facilement avec les logs, de les rechercher, de les filtrer et de les visualiser. Kibana a été utilisé pour offrir une visualisation dynamique et interactive des logs, facilitant ainsi l'analyse des événements critiques et des performances des applications surveillées.

En conclusion, ce projet a non seulement répondu aux attentes de l'entreprise, mais a également permis d'optimiser les performances et la gestion des logs, en offrant une interface utilisateur intuitive et des fonctionnalités avancées pour l'analyse des données. Cette solution pourra évoluer et être enrichie dans le futur, notamment avec l'intégration de technologies supplémentaires comme Redis, afin d'améliorer encore davantage les performances du système.

Bibliographie

[1] Définition de Flask : https ://fr.wikipedia.org/wiki/Flask_(*framework*)

[2] Définition de Elasticsearch : https ://www.next-decision.fr/editeurs-bi/restitution/elastic-kibana

[3] Définition de Logstash : https ://fr.wikipedia.org/wiki/Logstash

[4] Définition de kibana : https ://www.next-decision.fr/editeurs-bi/restitution/elastic-kibana