République Tunisienne
Ministère de l'Enseignement Supérieur
et de la Recherche Scientifique
Université de Sousse

Institut Supérieur des Sciences
Appliquées et de Technologie
de Sousse

DEPARTEMENT INFORMATIQUE

RAPPORT DE PROJET DE FIN D'ETUDES

En vue de l'obtention du:
Diplôme National d'Ingénieur Informatique
Option: Génie Logiciel-Architecture Logicielle

Système intelligent de détection et de résolution des anomalies sur des données BigData

Elaboré par :

Zayati Oussama

Soutenu le 15/07/2021 devant le jury :

Président :	**M.MTIBAA Riadh**	**ISSAT de Sousse**
Examinateur :	**Mme.KANZARI Dalel**	**ISSAT de Sousse**
EncadrÁÎô :	**Mme.BELGACEM Selma**	**ISSAT de Sousse**
Encadrant Industriel :	**M.SHILI Anouar**	**MedicusClinic**

Année Universitaire : 2020 / 2021

Code Sujet: FI-GL21-018

Dédicace

“

À ceux qui n'ont cessé de me soutenir, m'encourager et me guider tout au long de ma vie,
à mes chers parents ***Mohsen*** *et* ***Sana****, que Dieu vous préserve et vous accorde une longue vie. Que ce modeste travail soit l'exaucement de vos voeux tant formulés, le fruit de vos innombrables sacrifices, bien que je ne vous en acquitterai jamais assez,*

À mes chères frères ***Wissem*** *et* ***Yassine****, je vous souhaite un avenir plein de joie, de la réussite, du bonheur et de la sérénité,*

À ***la mémoire de mon frère et mon grand père****, que ce travail soit une prière pour le repos de vos âmes,*

À tous ***mes amis*** *avec lesquels j'ai partagé des moments de joie et de bonheur,*

À tous ceux qui me sont chers, à vous tous

Merci.

”

- Oussama

Remerciements

Tout d'abord, je remercie Allah le tout puissant de m'avoir donné le courage et la patience nécessaires à mener ce travail à son terme.

Je tiens à remercier tout particulièrement mon encadrante **Mme. Belgacem Selma**, pour l'aide compétente qu'elle m'a apportée, pour sa patience et son encouragement. Son œil critique m'a été très précieux pour structurer le travail et pour améliorer la qualité des différentes sections.

Je tiens à remercier également mon promoteur **M. Shili Anouar** pour son aide immense, la qualité de son suivie ainsi que pour tous les conseils et les informations qu'il m'a prodigués avec un degré de patience et de professionnalisme sans égal.

Je tiens aussi à exprimer toute ma gratitude envers **mes enseignants de l'ISSATSo** pour les connaissances qu'ils m'ont communiqué tout au long de ma formation.

Je tiens à exprimer mes remerciements aux membres du jury qui me font le grand honneur d'avoir accepté d'évaluer mon travail.

Pour finir, je souhaite remercier toute personne ayant contribué de prés ou de loin à la réalisation de ce travail.

Résumé

Ce projet de fin d'études s'est déroulé à la société MedicusClinic en vue de l'obtention du diplôme d'ingénieur en informatique de l'Institut Supérieur des Sciences Appliquées et de Technologie de Sousse.

Vu le nombre croissant des composants logiciels et matériels d'un système informatique, une plateforme de monitoring est mise en place, dans le cadre de ce projet, pour la surveillance des états de ces composants et leur protection contre les attaques. Cette tâche de monitoring est élaborée en utilisant les techniques de Big Data pour la collection des Logs qui contiennent les informations sur l'état du système et les techniques de Machine Learning pour détecter les anomalies et proposer des solutions.

Mots clés : Monitoring, Logs, Anomalies, Big Data, Machine Learning

---

Abstract

This project was conducted at MedicusClinic company in order to obtain the software engineering diploma from the higher institute of applied sciences and technology of Sousse.

Given the growing number of software and hardware components of an information system, a monitoring platform is set up for the surveillance of these components state and for possible attacks prediction. The monitoring task is performed using Big Data techniques for the collection of Logs containing the system state information and Machine Learning techniques to detect anomalies and offer solutions.

Keywords : Monitoring, Logs, Anomalies, Big Data, Machine Learning

---

Table des matières

Table des figures

Liste des tableaux

Introduction générale

Les technologies de l'information sont devenues indispensable dans les entreprises des différents secteurs. L'entreprise ne peut jamais fonctionner comme elle l'est aujourd'hui sans des systèmes informatiques qui gèrent et surveillent le fonctionnement de ses équipements en optimisant la consommation des ressources et en détectant les pics d'activité. Ces capacités prennent en charge la croissance de l'activité de l'entreprise et renforcent sa compétitivité. Généralement, ces systèmes sont critiques et doivent être hautement disponibles au point qu'une panne peut entraîner des effets secondaires irréversibles et finir par facturer des coûts supplémentaires à l'entreprise .

Ces systèmes sont développés et installés soit par le service informatique de l'entreprise, soit par un organisme tiers. L'un des principaux défis du développement de tels systèmes est de traiter de grands volumes de données en temps réel. Ce phénomène est causée par la croissance exponentielle de la technologie et est connu sous le nom de Big Data.

En même temps, Le Machine Learning a envahit le monde en montrant ce que l'intelligence artificielle peut nous offrir en terme d'assistance numérique, travail permanent et réduction des erreurs.

Bien que le Big Data et le Machine Learning ont évolués indépendamment, ils sont devenus interdépendants. Lorsque les organisations collectent les données pour les analyser, le Machine Learning intervient, pas seulement pour détecter des défaillances , mais aussi pour proposer des solutions pour résoudre ces problèmes. Et c'est le point où le rôle du Machine Learning dans le domaine de Big Data entre en scène.

Dans ce contexte, l'objectif de ce stage est la conception et le développement d'une plateforme permettant la collection, le Monitoring et la détection des anomalies des Logs. Le processus de collection va être réalisé en appliquant des techniques Big Data. Et le processus de détection des anomalies et la proposition des solutions va être réalisé en appliquant des techniques de Machine Learning.

Ce stage est réalisé dans le cadre du projet de fin d'études de la formation d'ingénieur en Informatique à l'Institut Supérieure des Sciences Appliquées et de Technologie de Sousse pour l'année universitaire 2020/2021, et a été effectué au sein de la société « Medicus Clinic ».

Le présent rapport qui documente le travail effectué dans le cadre de ce stage est organisé en quatre chapitres comme suit :

Le premier chapitre s'intitule "**Présentation générale du projet**." Il s'agit d'un chapitre introductif, qui présente l'entreprise d'accueil, la problématique, la solution proposée et les objectifs de notre projet. Il comporte également une étude de l'existant et une description du processus de développement de l'application.

Le deuxième chapitre "**Spécification des besoins**" définit les acteurs de notre application et spécifie les besoins fonctionnels et non fonctionnels auxquels notre application doit répondre tout en présentant ses principales fonctionnalités.

Le troisième chapitre "**Conception »**" présente une description des schémas conceptuels et de l'architecture adoptée pour la solution proposée avec une description du comportement dynamique de l'application.

Le quatrième chapitre "**Réalisation**" présente l'environnement et les outils de développement ainsi que la visualisation des résultats de notre travail à travers les principales interfaces du logiciel.

Notre projet sera achevé par une conclusion dans laquelle nous indiquerons les différents atouts de l'application réalisée et les perspectives de son amélioration.

Chapitre 1

Présentation générale du projet

Introduction

Dans ce chapitre, nous allons présenter l'organisme d'accueil ainsi que la problématique et les objectifs du projet, la solution proposée et le processus de développement utilisé.

1.1. Présentation de l'organisme d'accueil

Fig. 1.1 : Logo de MedicusClinic

STE MEDICUS CLINIC est une société de promotion de solutions technologiques web et mobile orientées spécifiquement vers le secteur médical à distance, en offrant des services d'assistance avancée aux différents intervenants médicaux.

Services :
Développement Web / Mobile : La société met en œuvre des applications SaaS (Software As A Service) de tout type de taille à l'aide de plates-formes ouvertes évolutives.

DevOps et hébergement : La gestion du cycle de vie des applications est basée sur une intégration continue complétée par des pipelines de déploiement et de suivi.

1.2. Concepts généraux

1.2.1. Logs

Des fichiers journaux qui enregistrent la trace des évènements qui se produisent lors de l'exécution d'un processus. [1]
Un évènement est caractérisé par les propriétés suivantes :
- Timestamp : Date et heure de l'évènement.
- Data :

Données échangées avec l'environnement du processus.

Informations sur la configuration de l'environnement du processus.

Réponse reçue d'un serveur.

Type d'éventuelle erreur survenue.

1.2.2. Anomalie

- Dans le cas général, une anomalie est un défaut de conception ou de réalisation qui cause un dysfonctionnement du système.
- Dans le cadre du domaine Big Data, une anomalie est une observation rare qui présente des caractéristiques différentes des caractéristiques des autres données de même type. [2]

1.2.3. Monitoring

En informatique, le Monitoring est une activité de surveillance du système, permettant de : [3]
- Mesurer avec précision les activités du système.
- Détecter les anomalies en temps réel.
- Eviter les pannes.
- Alerter l'utilisateur des dysfonctionnements avant que le problème ne survienne.

1.3. Présentation du projet

1.3.1. Cadre général du projet

Les technologies de l'information sont devenues indispensables dans les entreprises pour leur permettre de gérer et de surveiller les états de leurs équipements. Le traitement du grand volume de flux de données présente un défis pour ces entreprises. De plus, la détection d'un dysfonctionnement dans leurs équipements doit être automatique et en temps réel. Dans ce cadre, l'objectif de ce projet est la création d'une plateforme permettant l'analyse d'un grand nombre de données pour la détection des anomalies dans un système informatique.

1.3.2. Problématique

Le nombre croissant des composants logiciels et matériels d'un système informatique augmente la quantité d'information et complexifie la tâche de l'administrateur. En effet, la gestion de flux d'information et le suivi de l'état des machines et des différents services doivent être continus et en temps réel.
Bien qu'il existe un examen journalier des logs systèmes, les anomalies peuvent être détectées avec un retard couteux au niveau financier et organisationnel.
Ainsi, l'absence d'un outil de supervision automatique temps réel engendre :
- la difficulté de détection des pannes.
- la nécessité d'intervention manuelle par l'équipe de vérification pour identifier les pannes.
- l'absence d'alerte automatique en cas de problème.

Devant ce constat et dans le cadre de ce projet, nous avons fixé l'objectif de concevoir et de réaliser un système de supervision et de monitoring informatique.

1.3.3. Solution proposée

Le Monitoring et la mesure d'indices de performance des applications informatique sont un enjeu majeur pour les entreprises. L'évolution des technologies , y compris celles du Machine Learning, appliquées pour la qualification, le stockage et le traitement de gros volumes de données (Big Data), contribue à l'amélioration du processus de Monitoring. Un Monitoring performant facilite la détection des pannes système, la détection des bugs et la détection des fraudes en temps réel.
Ainsi, pour développer ce processus performant de Monitoring, nous proposons d'appliquer la pile ELK (Elasticsearch, Logstash et Kibana) permettant la récupération, la centralisation et la visualisation des données[4]. Nous proposons aussi d'y intégrer une méthode de Machine Learning pour la détection des anomalies sur les données de Logs.

1.3.4. Objectifs

- Avoir une image instantanée de l'état global du système contenant :
  - Mesures de performances (exemple : temps de réponse)
  - Mesures de disponibilité (exemple : composants matérielles et logiciels fonctionnels)
  - Mesures d'intégrité (exemple : état persistant des fichiers logs)
- Eviter les pannes :
  - Anticiper les défaillances en prévoyant les dysfonctionnements avant que le problème ne survienne.
  - Définir des seuils de tolérance à ne pas franchir.
- Lutter contre les attaques :
  - Protéger le système contre les cybercriminels : Le pare-feu et l'antivirus ne suffisent pas pour bloquer toutes les attaques. Le monitoring permet d'ajouter un niveau de sécurité en détectant les intrusions et les comportements inhabituels.
- Faire évoluer le système informatique :
  - Le monitoring permet d'établir en temps réel une image claire et précise du système informatique en analysant les flux de données, identifiant la source du problème et en prenant les mesures de protection nécessaires. Ainsi, le système devient adaptable aux situations critiques avec une meilleure gestion de la sécurité et des coûts.

1.4. Etude de l'existant

Cette section représente une analyse des logiciels similaires existants permettant de dégager leurs points faibles qui pourraient être améliorés à travers notre logiciel.

1.4.1. Analyse de l'existant

Dans ce paragraphe, nous présentons les différentes solutions existantes étudiées dans le cadre de notre projet.

- **New Relic One** : New Relic One est une plateforme d'observabilité qui permet d'instrumenter, dépanner et optimiser l'intégralité de son stack de logiciels. [5]
- **Dynatrace** : Dynatrace est un éditeur de logiciels spécialisé en gestion de la performance applicative, à destination des directions des systèmes d'information et du business numérique. [6]
- **Datadog** : Datadog est un service de surveillance des applications, des serveurs, des bases de données, des outils et des services, grâce à une plateforme d'analyse de données en mode Software As A Service. [7]

1.4.2. Critique de l'existant

Le tableau 1.1 récapitule les avantages et les inconvénients des solutions présentées.

Système existant	Avantages	Inconvénients
New Relic One	- Dispose d'un tableau de bord avec une surveillance en temps réel. - Capable de reconnaître les problèmes	- Pas de suivi des Logs. - Pas de protection contre les attaques.
Dynatrace	- Surveillance de l'infrastructure y compris les serveurs, les conteneurs et le cloud - Suivi de chaque transaction effectuée.	- N'est pas multi utilisateurs. - Pas de protection contre les attaques.
Datadog	- Les utilisateurs peuvent créer des alarmes et des notifications en cas de dépassement de seuil. - Agrège les données provenant des applications.	- Pas de suivi des Logs. - Pas de protection contre les attaques.

Tab. 1.1 : Critique de l'existant [8]

L'objectif de ce projet est donc d'intégrer tous les points forts des solutions existantes et en même temps compenser leurs points faibles.

1.5. Processus de développement

1.5.1. Développement incrémental

Nous avons choisi comme processus de développement de notre logiciel le modèle incrémental. L'apport du modèle incrémental est le partitionnement du modèle du logiciel en un ensemble d'incréments individuellement cohérents, fonctionnels et livrables au client. [9]

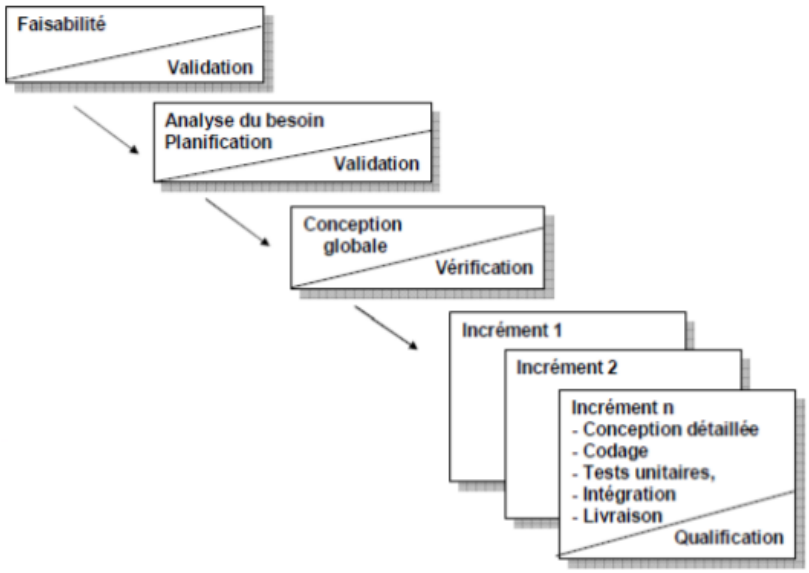


FIG. 1.2 : Modèle incrémental [9]

1.5.2. Incréments du logiciel

Les différents modules de notre logiciel sont :

- **Centralisation et visualisation des Logs** : collecter les Logs depuis plusieurs ressources, les formater dans un format standard, et les stocker en utilisant des techniques Big Data. Puis, les visualiser en utilisant des statistiques et des graphiques.
- **Gestion des alertes** : Lors d'un dysfonctionnement ou bien d'une attaque, le logiciel notifie l'utilisateur via un Mail ou une notification.
- **Détection des anomalies** : détecter les problèmes en temps réel et proposer des solutions grâce à une méthode de Machine Learning sans aucune intervention de l'utilisateur.

1.5.3. Planning prévisionnel des taches

Le diagramme de Gantt est un outil graphique qui représente la gestion du projet dans le temps ce qui facilite sa réalisation.

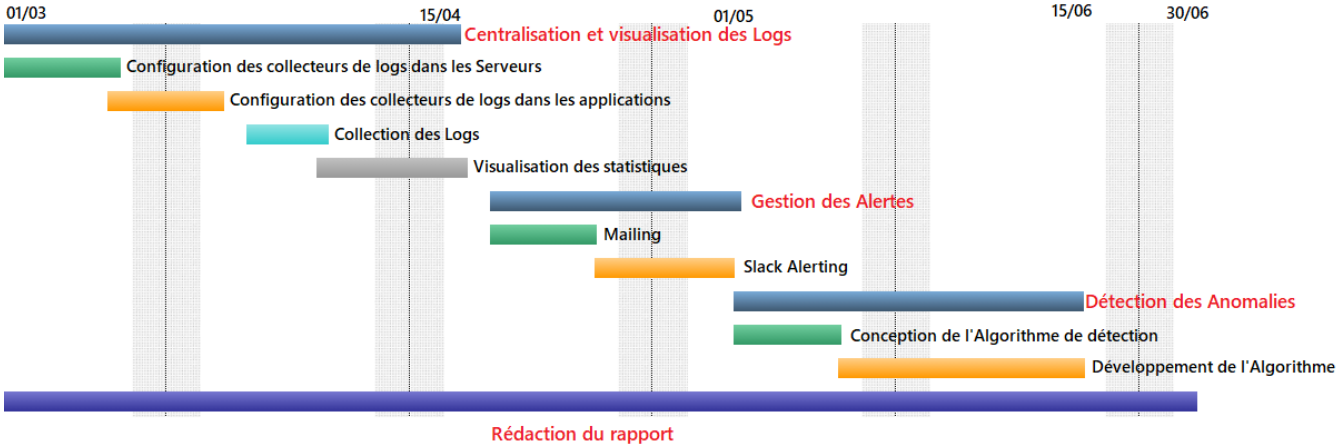


Fig. 1.3 : Diagramme de Gantt

Conclusion

Dans ce chapitre, nous avons exposé une description générale du projet : organisme d'accueil "MedicusClinic", concepts clés, cadre du projet, problématique, solution proposée et objectifs à réaliser. Aussi, nous avons présenté une étude des applications alternatives existantes pour dégager l'originalité nécessaire de notre application. Nous avons clôturé notre chapitre par une description du processus de développement qui sera appliqué.
Dans le prochain chapitre, nous analysons les besoins du client pour déduire les principales fonctionnalités de notre logiciel.

Chapitre 2

Spécification des besoins

Introduction

Dans ce chapitre nous allons identifier les acteurs de notre système, analyser les besoins fonctionnels et non fonctionnels exprimés par le client, et dégager les principales fonctionnalités de notre logiciel et les décrire à travers des diagrammes de cas d'utilisation.

2.1. Glossaire

Mots techniques utilisés dans le cadre du projet :

- **Système informatique** : un ensemble de serveurs et d'applications web hébergées.
- **Anomalie** : dysfonctionnement au niveau d'un serveur, exemple : excès d'utilisation de la RAM ou du CPU.
- **Attaque** : réception d'un nombre excessif de requêtes au niveau d'un même serveur simultanément.
- **Alerte** : résultat de la détection d'une anomalie ou d'une attaque envoyé sous forme d'une notification à l'administrateur et à l'investigateur via un logiciel local à la société, ou bien sous forme d'un mail à l'administrateur.

2.2. Identification des acteurs

Les acteurs sont des utilisateurs humains ou des systèmes informatiques qui interagissent dans notre logiciel. Les acteurs primaires de notre logiciel sont :

- **Utilisateur simple** : consulter l'état du système informatique à travers un Dashboard contenant des statistiques et des graphiques (voir section 4.4).
- **Investigateur** : Cet acteur intervient en cas de détection d'une anomalie pour analyser le problème détecté par un module intelligent qui va être un composant de notre logiciel, vérifier la solution proposée par ce module intelligent et l'appliquer par la suite. Puis, générer un rapport.
- **Administrateur** : Cet acteur peut exploiter les services fournis pour les acteurs "utilisateur simple" et "investigateur" avec d'autres services administratifs supplémentaires.

2.3. Besoins fonctionnels

Notre système doit offrir les fonctionnalités suivantes :

- **Visualisation, adaptée selon le privilège de l'utilisateur, des informations sur l'état des différents composants du système informatique en temps réel** : surveillance de l'état du système à travers l'affichage d'un ensemble de statistiques et de graphiques :
  - pour les serveurs : taux d'utilisation de la RAM et du CPU ou débit de transfert des données
  - pour les applications : type des requêtes reçues, temps de réponse, erreurs renvoyées, liste des utilisateurs
- **Analyse du flux de données provenant des différents composants**.
- **Identification des anomalies et proposition de solutions** : un module intelligent, utilisant une méthode de Machine Learning, détecte les anomalies et propose des solutions selon des règles définies d'une manière automatique (voir section 3.1.4).
- **Protection contre les attaques** : bloquer la source d'un nombre élevé de requêtes reçues simultanément(voir section 3.1.5).
- **Paramétrage du logiciel** :
  - fixer des seuils du taux d'utilisation de la RAM/CPU, dont le dépassement déclenche l'envoi d'une notification ou d'un mail.
  - saisir des règles de décisions permettant au module intelligent de proposer des solutions pour la résolution des anomalies.
- **Gérer les comptes utilisateur** : ajouter, supprimer, mettre à jour, attribuer des rôles et des privilèges.

2.4. Besoins non fonctionnels

Notre système doit garantir :

- **Performance** : Notre plateforme doit gérer le Big Data en termes de volume, de vitesse, de variété et de variabilité.
- **Sécurité** : La sécurité est une priorité dans notre plan de développement logiciel puisque nous traitons les données des différents appareils connectés.
- **Réutilisabilité** : La conception des modules de notre système doit garantir la possibilité de les réutiliser pour d'autres applications.
- **Ergonomie** : Les interfaces et les graphiques de notre plateforme doivent être compréhensibles et facile à manipuler.

2.5. Diagrammes des cas d'utilisation

2.5.1. Diagramme général

La figure 2.1 illustre le diagramme de cas d'utilisation globale de notre application.

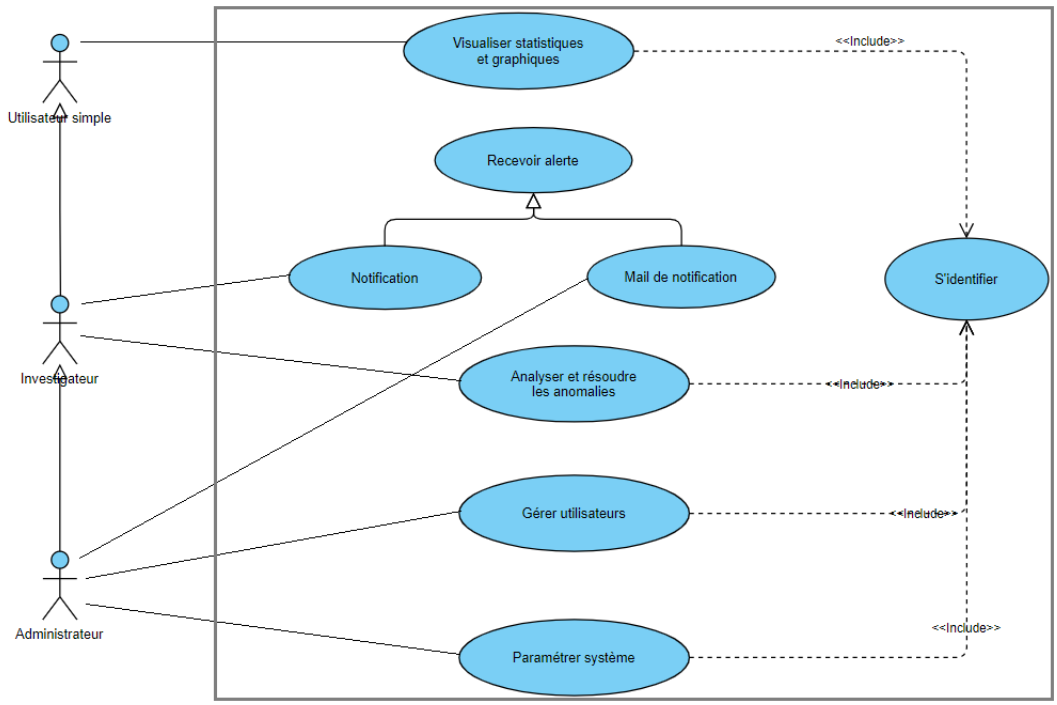


FIG. 2.1 : Diagramme de cas d'utilisation général

2.5.2. Diagramme de visualisation des statistiques et des graphiques

La figure 2.2 illustre le diagramme de cas d'utilisation de visualisation des statistiques et des graphiques

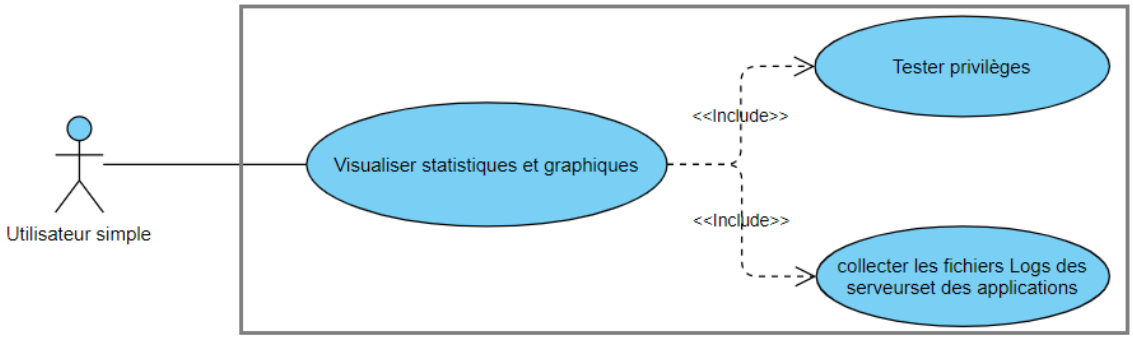


FIG. 2.2 : Diagramme de cas d'utilisation : Visualiser statistiques et graphiques

SOMMAIRE	
Titre	Visualiser statistiques et graphiques
Acteurs primaires	Utilisateur simple
Pré condition	L'utilisateur doit être authentifié
Post condition	Logs analysés
Scénario nominal	1. le système collecte les statistiques à partir des fichiers logs des serveurs et des applications. 2. le système teste le privilège de l'utilisateur. 3. l'utilisateur choisit le serveur ou l'application. 4. le système affiche les graphiques des statistiques contenant des informations dépendant du privilège de l'utilisateur.

TAB. 2.2 : Cas d'utilisation : Visualiser statistiques et graphiques

2.5.3. Diagramme d'analyse et de résolution des anomalies

La figure 2.3 illustre le diagramme de cas d'utilisation d'analyse et de résolution des anomalies.

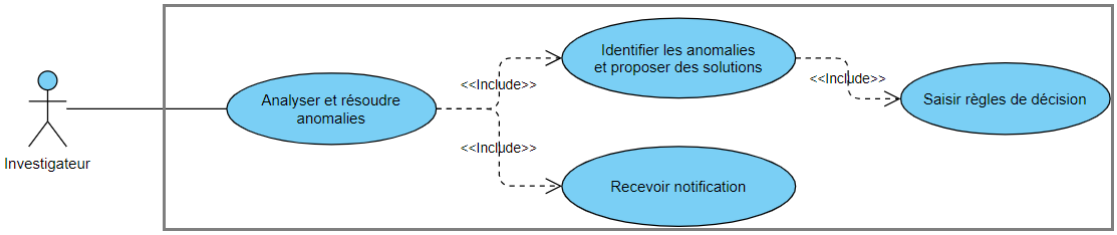


FIG. 2.3 : Diagramme de cas d'utilisation : analyser et résoudre les anomalies

SOMMAIRE	
Titre	Analyser et résoudre les anomalies
Acteurs primaires	Investigateur
Pré condition	L'utilisateur doit être authentifié
Post condition	Rapport généré et Anomalie résolue
Scénario nominal	1. le système (module intelligent) analyse les statistiques et détecte les anomalies. 2. le système (module intelligent) propose des solutions en appliquant un ensemble de règles. 3. le système envoie une notification à l'investigateur. 4. l'investigateur consulte l'anomalie. 5. l'investigateur vérifie et applique la solution proposée. 6. l'investigateur génère rapport.

TAB. 2.4 : Cas d'utilisation : Analyser et résoudre les anomalies

2.5.4. Diagramme de paramétrage du système

La figure 2.4 illustre le diagramme de cas d'utilisation de paramétrage du système.

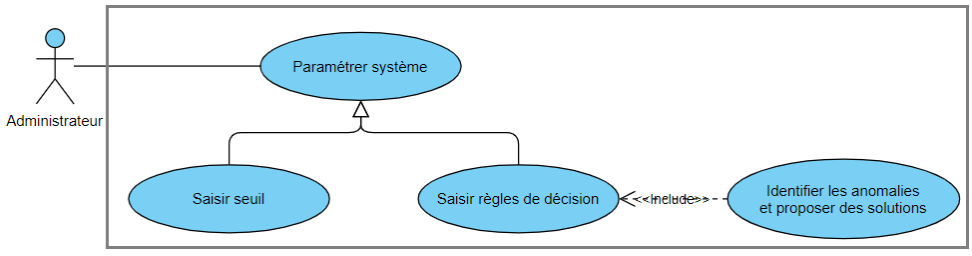


FIG. 2.4 : Diagramme de cas d'utilisation : Paramétrer système

SOMMAIRE	
Titre	Paramétrer système.
Acteurs primaires	Administrateur.
Pré condition	L'utilisateur doit être authentifié.
Post condition	Règles saisies ou seuil fixé.
Scénario nominal	1. le système affiche un formulaire. 2. l'administrateur saisit les règles de décision. 3. le système (module intelligent) applique ces règles pour proposer des solutions pour la résolution des anomalies.

Tab. 2.6 : Cas d'utilisation : Paramétrer système

Conclusion

Dans ce chapitre, nous avons décrit les phases de spécification des besoins de notre projet. Nous avons commencé par l'identification des différents acteurs. Ensuite, nous avons présenté les besoins fonctionnels et non fonctionnels du projet. Finalement, nous avons détaillé les diagrammes des différents cas d'utilisation.
Dans le prochain chapitre, nous allons détailler la modélisation interne statique et dynamique de notre logiciel.

Chapitre 3

Conception

Introduction

Dans ce chapitre nous allons préparer le terrain pour la réalisation. Pour ce faire, nous allons spécifier la vue statique architecturale suivie de la vue dynamique comportementale de notre logiciel en appliquant des patrons d'architecture et des diagrammes UML.

3.1. Vue statique structurelle de l'application

3.1.1. Architecture logique du logiciel

L'architecture logique permet d'identifier les composants logiciels essentiels à la mise en œuvre de la solution et de décrire les relations entre ces composants.

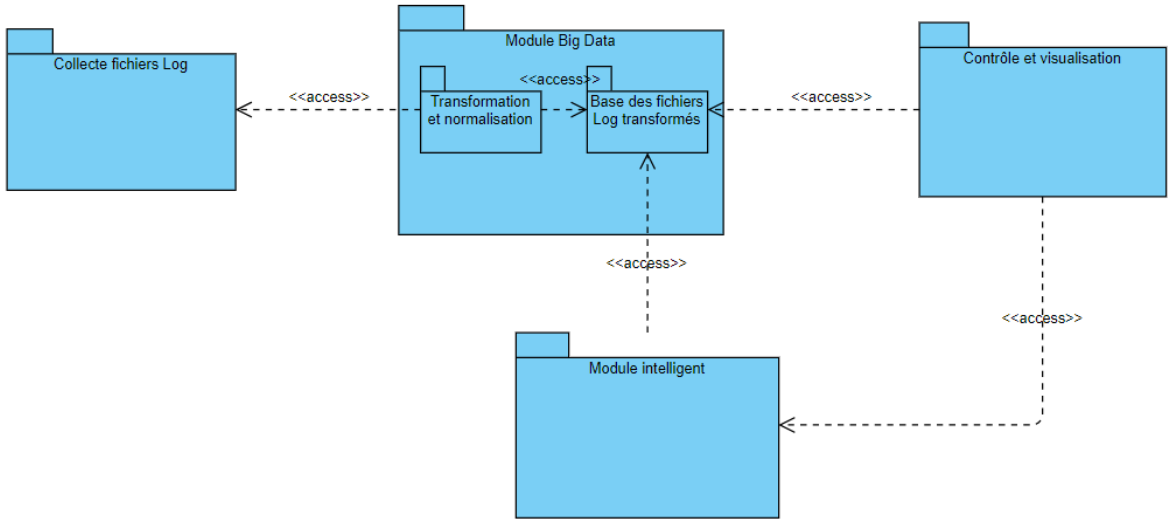


Fig. 3.1 : Vue globale du logiciel

Dans le cas de notre système, comme le montre la figure 3.1, l'architecture globale est composée de quatre paquetages ayant chacun un rôle bien défini :

- **Collecte fichiers Log** : collecte des fichiers Log provenant des différents serveurs et applications du système informatique.
- **Module Big Data** :
  - **Transformation et normalisation** : transformer les fichiers Log dans un seul format.
  - **Base des fichiers Log transformés** : espace de stockage des Logs après leur transformation.
- **Module intelligent** : module appliquant une méthode Machine Learning pour la détection des anomalies à partir des informations enregistrées dans les fichiers Log transformés. En appliquant des règles de décision (voir tableau 3.2), des solutions seront proposées.

- **Contrôle et visualisation** :
  - contrôle du système.
  - visualisation personnalisée des statistiques (voir section 4.4) selon le privilège de l'utilisateur.
  - détection des attaques (voir algorithme 3) et blocage de leurs sources d'une manière automatique.
  - gestion des comptes utilisateurs.

3.1.2. Archtiecture du paquetage : Collecte fichiers Log

La figure 3.2 illustre le diagramme de classes du paquetage "Collecte fichiers Log".

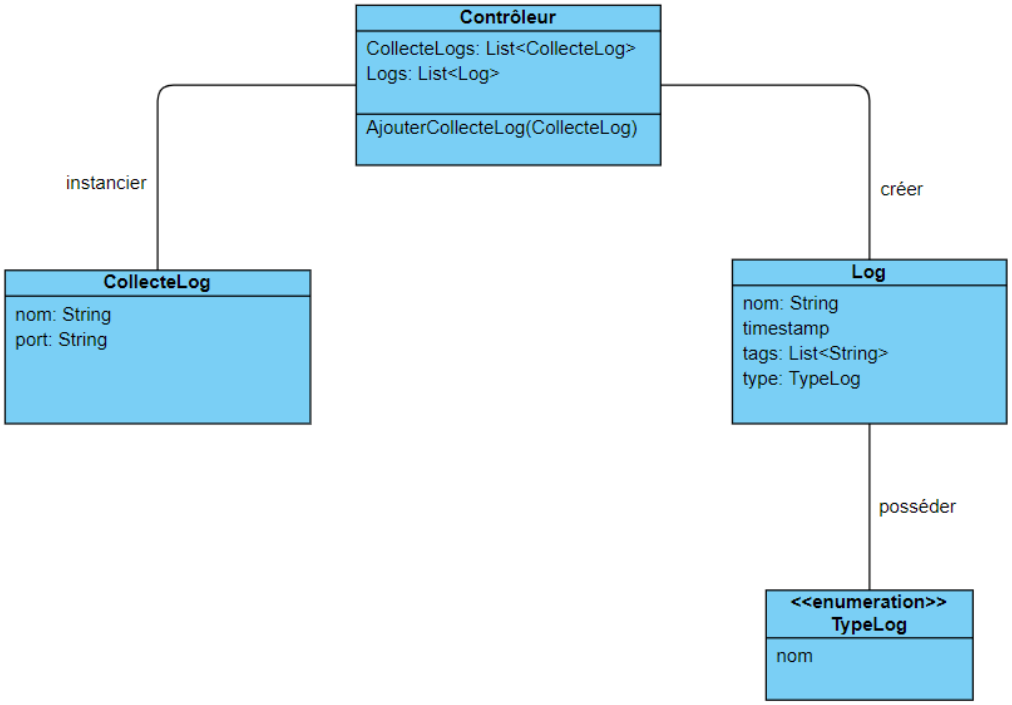


FIG. 3.2 : Diagramme de classes du paquetage : Collecte fichiers Log

Les classes du diagramme 3.2 sont expliquées en détail comme suit :

- **Contrôleur** : classe qui récupère les fichiers Log et leur attribue des instances de la classe "Log".
- **CollecteLog** : classe instanciée pour chaque serveur et pour chaque application afin d'obtenir les fichiers Log associés.
- **Log** : Classe responsable de la sérialisation des Logs collectés des différents serveurs et applications.

- **TypeLog** : Enumeration utilisée par la classe Log permettant de décrire le type des Logs que le système va collecter. Les différents types des Logs sont : *MetricBeat*(Indicateurs de performances, exemples : Processeur - RAM) - *PacketBeat*(Données réseaux) - *WinlogBeat*(Logs des évenements Windows)- *HeartBeat*(Disponibilité)

3.1.3. Archtiecture du paquetage : Module Big Data

La figure 3.3 illustre l'architecture Kappa permettant de créer une application de traitement de flux pour gérer les données en temps réel.

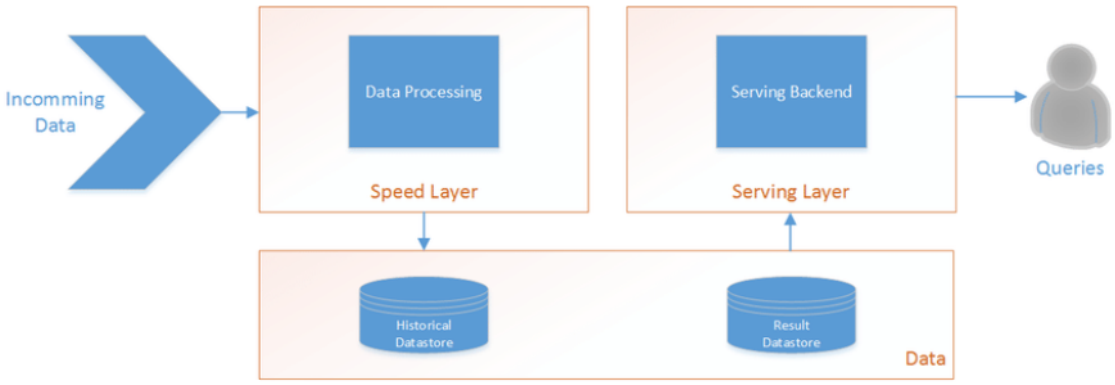


FIG. 3.3 : Architecture Kappa [10]

C'est une architecture qui se compose de deux couches différentes :

- **Speed Layer** : Cette couche est responsable du traitement, de la surveillance et de la diffusion continue des données.
- **Serving Layer** : Dans cette couche, les données sont préparées pour être présentées dans la couche suivante et pour coordonner entre ce que l'utilisateur demande et ce qu'il va recevoir. [11]

La figure 3.4 illustre le diagramme de classes du paquetage ”Module Big Data”.

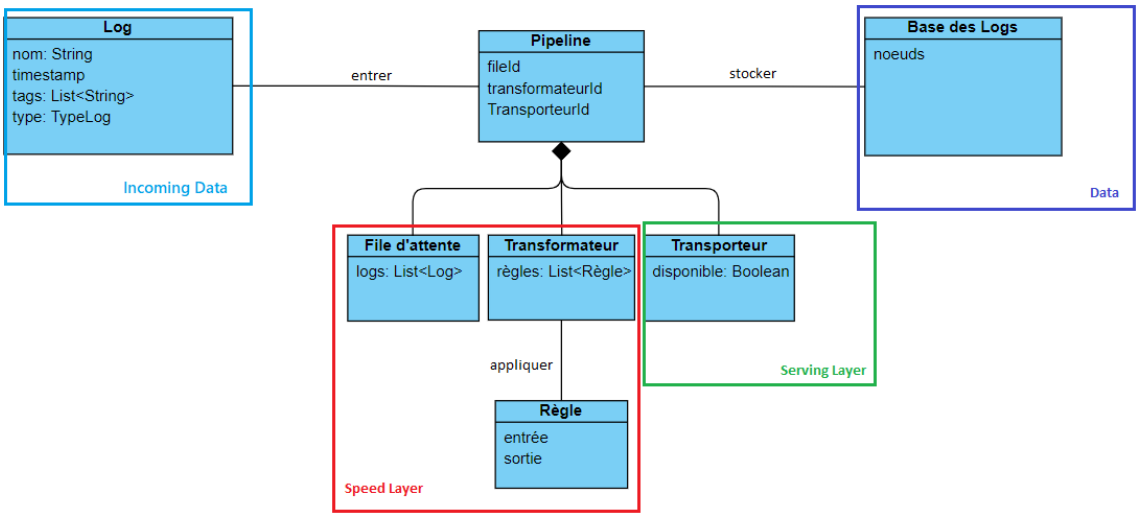


FIG. 3.4 : Diagramme de classe du paquetage : Module Big Data

Les classes du diagramme 3.4 sont expliquées en détail comme suit :

- **Log** : Les Logs provenanat du paquetage ”Collecte fichiers Log”.
- **Pipeline** : Le moteur responsable de la standardisation des données. Il est composé d'une file d'attente, d'un transformateur et d'un transporteur.
- **File d'attente** : Structure contenant les Logs en attente.
- **Transformateur** : Responsable de la standardisation les Logs selon des règles prédéfinis.
- **Règle** : Définit comment transformer les Logs. (voir tableau 3.4)
- **Transporteur** : Responsable de transférer les Logs vers l'espace de stockage.
- **Base des Logs** : Représente l'espace de stockage des Logs standardisés.

3.1.4. Archtiecture du paquetage : Module Intelligent

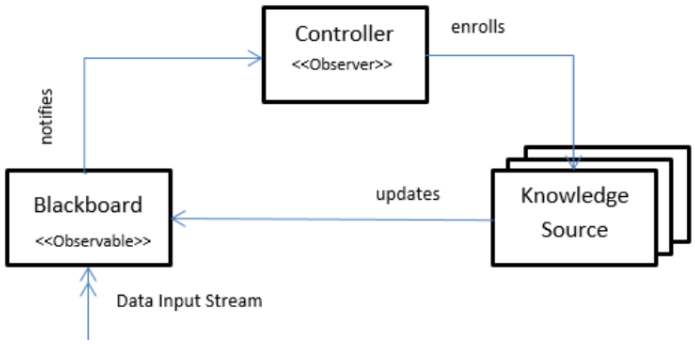


FIG. 3.5 : Architecture Blackboard [12]

La figure 3.5 illustre l'architecture Blackboard définissant trois composants principaux :

- **Blackboard** : mémoire globale représentant un espace de stockage des données.
- **Knowledge Source** : caractérisé par un ensemble de conditions de déclenchement et un code exécutable qui récupère les données du Blackboard et ajoute sa contribution au processus de résolution du problème.
- **Controller** : sélectionne, configure et exécute les "Knowledge Sources" [12]

La figure 3.6 illustre le diagramme de classes du paquetage "Module Intelligent".

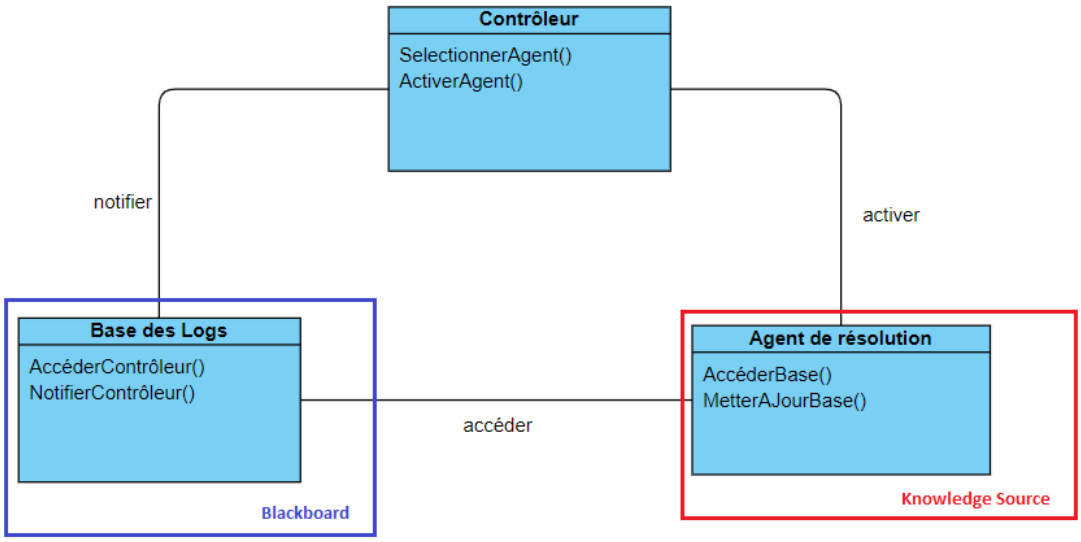


FIG. 3.6 : Diagramme de classe du paquetage : Module Intelligent

Les classes du diagramme 3.6 sont expliquées en détail comme suit :

- **Base des Logs** : Représente l'espace de stockage des Logs.
- **Contrôleur** : Récupère les données de la base et exécute les "Agents de résolution".

- **Agent de résolution** : Responsable de développer la résolution du problème en appliquant l'algorithme 1. Dans notre cas, cette classe utilise l'algorithme d'apprentissage non-supervisé K-means.

Algorithm 1 : Algorithme de détection et de proposition de solution

Entrée : Statistique S
Début

1. appliquer algorithme K-means(voir algorithme 2)
2. **Si** anomalie détectée
   (1) chercher solution dans le tableau des règles d'alertes
   (2) **Si** solution trouvée
      (1) retourner anomalie détectée + **solution**
   (3) **Sinon**
      (1) retourner anomalie détectée et solution non trouvée

Fin

Algorithm 2 : Algorithme K-means [13]

Entrée : Statistique S, k classes
Début

1. initialiser k centres de classes G_k
2. **Répéter**
3. **Allocation :** Affecter chaque individu à la classes dont le centre est le plus proche
4. **Représentation :** Recalculer les centres de classes à partir des individus rattachés (nouvelles valeurs des barycentres).
5. **Jusqu'**à Convergence
   Nombre d'itérations fixé
   Aucun individu ne change de classe
   Stabilisation des centres de clusters(les centroids ne bougent plus lors des itérations)

Fin

K-means est un algorithme d'apprentissage non supervisé de clustering permettant de regrouper en k clusters distincts les observations. Ainsi les données similaires se retrouveront dans un même cluster.

- Avantages :
  - K-means est simple, facile à comprendre et à mettre en oeuvre.
  - Scalabilité : Capacité à traiter les très grandes bases.
  - Applicable à tout type de données.

- Inconvénient :
  - Le nombre de classes doit être fixé au départ.

Dans notre projet le nombre de classes est connu (2 : anomalie - pas d'anomalie), donc l'inconvénient de l'algorithme K-means ne présente aucun problème dans notre cas. D'où cet algorithme convient parfaitement à notre projet.

- Le type de l'anomalie dépend de la grandeur statistique traitée.
- Les composantes d' un vecteur O =<t,V> qui représente une observation où t représente le temps et V représente la valeur de la grandeur statistique (taux d'utilisation de la RAM / taux d'utilisation du CPU / taux du débit du transfert de données).
- Nous avons choisi comme centres initiaux des deux classes d'observations les deux points :
  - G1(0 , 50%) pour la classe non-anomalie.
  - G2(0 , 90%) pour la classe anomalie.
- Nous avons choisi comme distance entre deux points disctincts : **La distance Euclidienne**

Le tableau 3.2 représente deux exemples de règles de proposition de solution utilisées par le module intelligent

Anomalie	**Solution ou Indication**
Problème d'utilisation RAM/CPU (dépassement de seuils adaptables : le module intelligent analyse l'évolution de ce taux et agit en cas de pic : exemple seuil maximal 90%)	Fermer certaines applications
Débit élevé de transfert de données	Répétition non limitée d'un bloc d'instructions contenant un transfert de données.

Tab. 3.2 : Tableau de deux exemples de règles de proposition de solution

3.1.5. Archtiecture du paquetage : Contrôle et visualisation

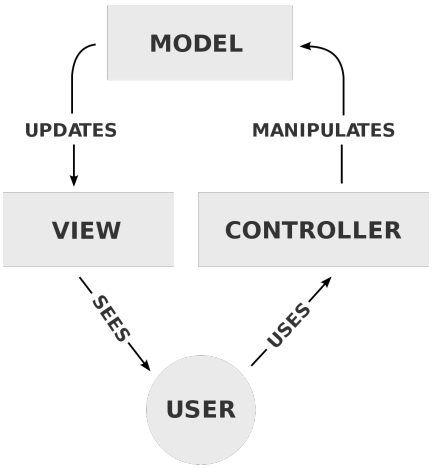


Fig. 3.7 : Architecture MVC

La figure 3.7 illustre l'architecture MVC composée de trois types de modules ayant trois responsabilités différentes : les modèles, les vues et les contrôleurs.

- **Modèle** : contient une représentation des données et les classes qui gèrent ces données et y appliquent des traitement (voir figure 3.8).
- **Vue** : l'ensemble des interfaces d'interaction avec l'utilisateur y compris les interfaces de visualisation des statistiques et des graphiques, Les types des statistiques visualisées sont :
  - pour les serveurs : taux d'utilisation de la RAM et du CPU ou débit de transfert des données
  - pour les applications : type des requêtes reçues, temps de réponse, erreurs renvoyées, liste des utilisateurs
- **Contrôleur** : représente le programme principale qui gère le flux de données dans le paquetage et déclenche les différentes fonctions de visualisation.

La figure 3.8 illustre le diagramme de classes de la partie Modèle du patron MVC du paquetage "Contrôle et Visualisation".

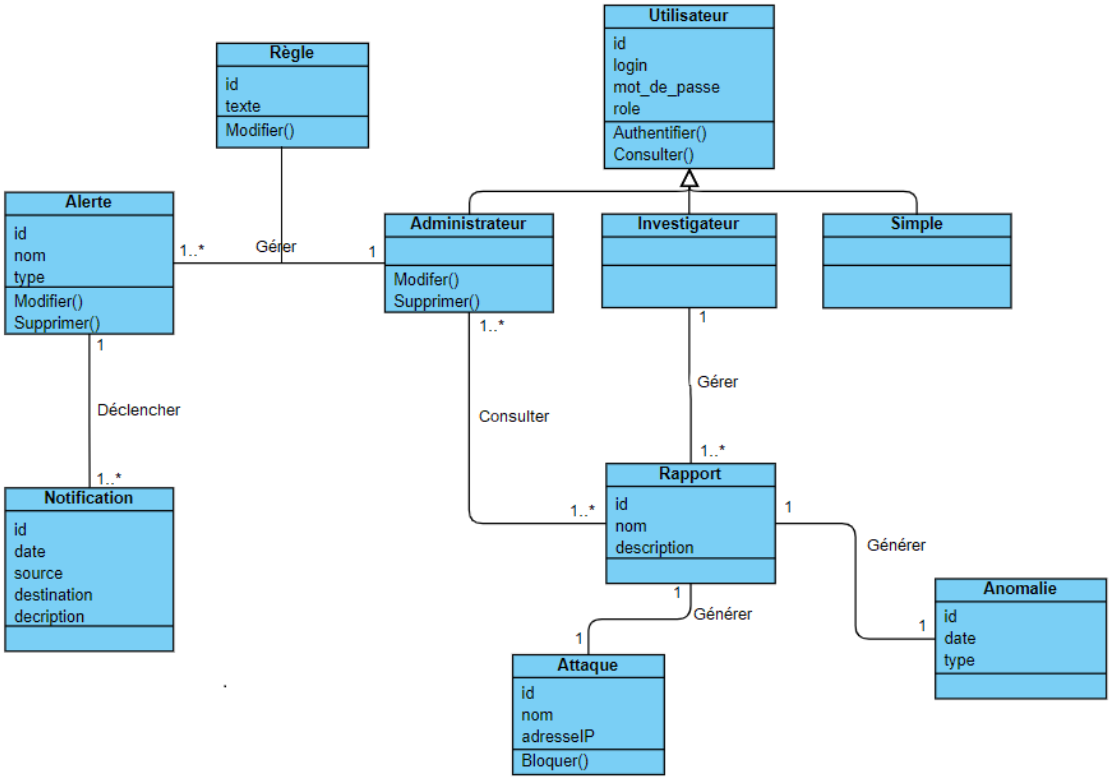


FIG. 3.8 : Diagramme de classe du Modèle du paquetage : Contrôle et visualisation

Les classes du diagramme 3.8 sont expliquées en détail comme suit :

- **Utilisateur** : Classe qui représente les acteurs de notre logiciel (Utilisateur simple, Investigateur, Administrateur).
- **Règle** : Définit par l'administrateur pour décrire quand déclencher une Alerte lors d'une anomalie et comment la résoudre.
- **Alerte** : Créée par l'administrateur pour déclencher une notification en cas d'anomalie.
- **Notification** : Déclenchée pour notifier l'investigateur et l'administrateur en cas d'anomalie.
- **Rapport** : Classe qui représente les détails d'une anomalie ou d'une attaque.
- **Anomalie** : Classe instanciée en cas de détection d'une anomalie par le module intelligent.

Nous avons développé un algorithme de détection et de blocage des attaques (algorithme 3) qui fait partie de la classe "Attaque". Lors de la réception d'un nombre élevé de requêtes simultanément et provenant de la même source ayant la même adresse IP, notre algorithme bloque cette source d'une manière permanente.

Algorithm 3 : Algorithme de blocage des attaques

Entrée : Requête R, Liste des requêtes précédentes L
Début

1. **Extraire** l'adresse IP, A, de la requête R.
2. **Créer** une liste L_1, à partir de la liste L, contenant les adresses IP des requêtes reçu depuis 10 secondes.
3. **Compter** le nombre de requêtes de la liste L_1 ayant la même adresse IP que A.
4. **Si** nombre > 10
   (1) **Blocage permanent des requêtes ayant l'adresse A**

Fin

3.2. Vue dynamique comportementale de l'application

3.2.1. Diagrammes de séquences détaillés

Les diagrammes de séquences détaillés sont des diagrammes d'interactions, entre les objets qui composent le système, mettants l'accent sur la chronologie de l'envoi des messages. Nous fournissons alors les diagrammes suivants afin d'assurer une description détaillée du comportement de notre système :

3.2.1.1) Collecte des Logs et préparation des statistiques

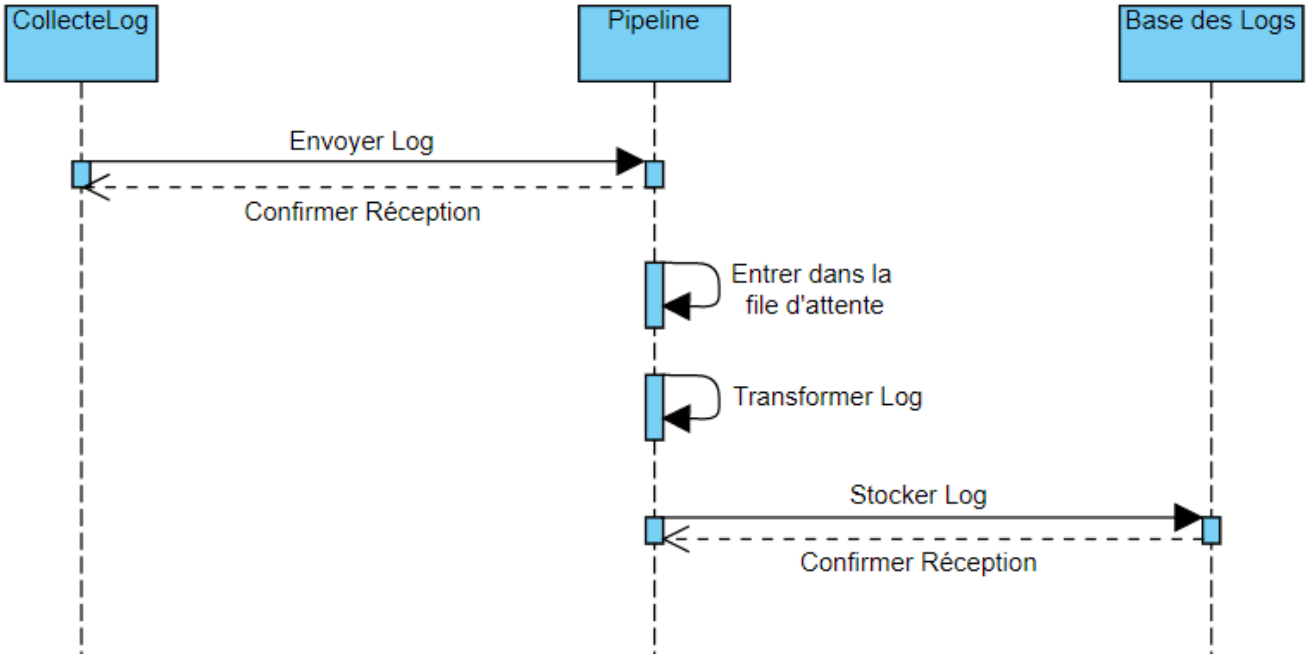


FIG. 3.9 : Diagramme de séquences modélisant la collecte des Logs et la préparation des statistiques

Le diagramme 3.9 montre le processus de traitement prenant lieu dès la réception des Logs. En effet, chaque Log subit un changement selon son type (voir tableau 3.4) à travers le ”Transformateur” du ”Pipeline” tout en passant par la ”File d'attente”.

3.2.1.2) Analyse et résolution des anomalies

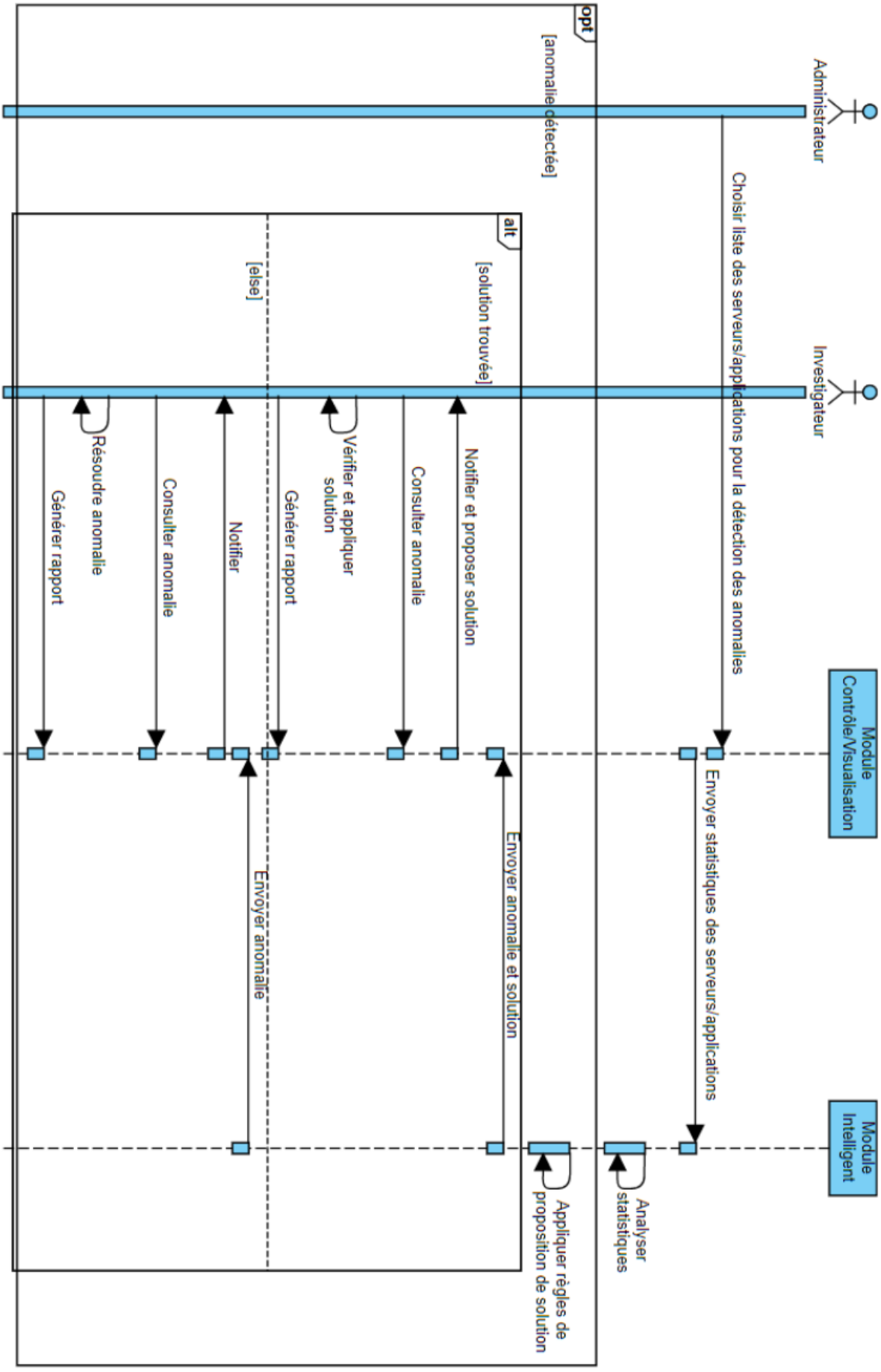


FIG. 3.10 : Diagramme de séquences modélisant l'analyse et la résolution des anomalies

Le diagramme 3.10 montre le processus d'analyse et de résolution des anomalies. En effet, l'administrateur doit choisir le composant (serveur ou application) qui va être contrôlé, c-à-d pour lequel la détection des anomalies va être effectuée. Le module intelligent de notre système analyse ensuite les statistiques calculés à partir des logs en temps réel et en cas de détection d'anomalie, il cherche une solution à partir des règles prédéfinis par l'administrateur. Puis, il notifie l'investigateur qui intervient pour résoudre cette anomalie en vérifiant et appliquant la solution proposée.

3.2.2. Diagramme d'activités

Le diagramme 3.11 illustre l'enchaînement des fonctionnalités principales de notre logiciel.

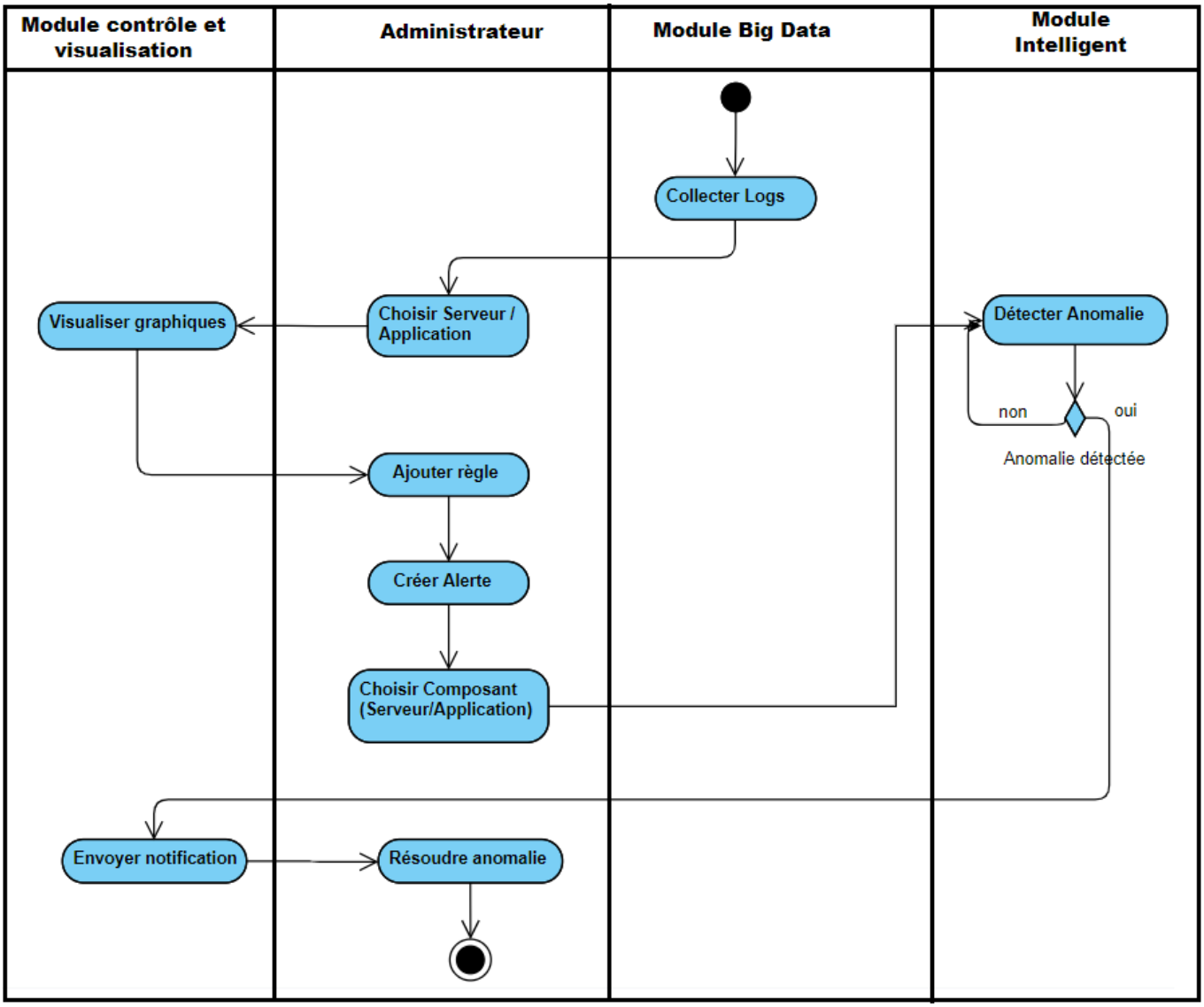


FIG. 3.11 : Diagramme d'activités partitionné d'un scénario d'utilisation complet

3.2.3. Diagramme état-transition des Logs

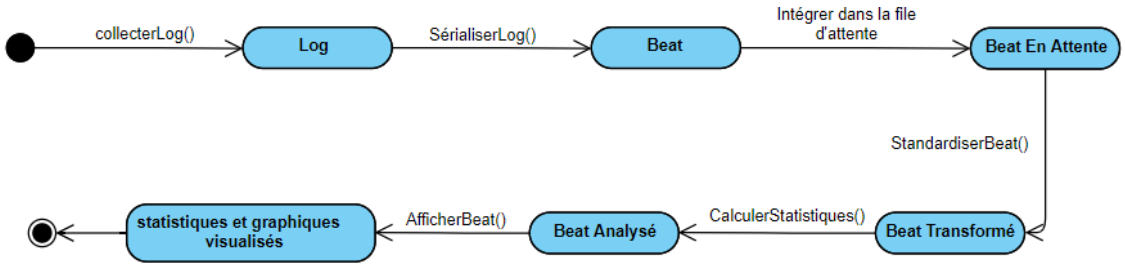


FIG. 3.12 : Diagramme d'état-transition des Logs

Le diagramme 3.12 décrit l'enchaînement des états possibles des Logs au cours de l'exécution de notre logiciel.

- **Beat** : Les logs deviennent des beats après leur sérialisation.
- **Beat en Attente** : Les Logs entrent dans le système Big Data et attendent leurs tours dans la file d'attente.
- **Beat Transformé** : Les Logs sont transformés selon leurs types (voir tableau 3.4).

Type Log	Transformation
MetricBeat	Conservation des champs, de l'attribut "Tags" de la classe Log, reliés aux indicateurs de performances (CPU - RAM)
PacketBeat	Conservation des champs, de l'attribut "Tags" de la classe Log, reliés aux informations réseaux
WinLogBeat	Conservation des champs, de l'attribut "Tags" de la classe Log, reliés aux informations du système d'exploitation
HeartBeat	Conservation des champs, de l'attribut "Tags" de la classe Log, reliés à la disponibilité

TAB. 3.4 : Tableau des transformations des Logs

- **Beat Analysé** : Les Logs sont analysés par le module intelligent pour la détection des anomalies.
- **Beat visualisé** : Des statistiques sont calculés à partir des fichiers Log et des graphiques résultantes sont affichés (voir section 4.4).

Conclusion

Dans ce chapitre, nous avons présenté à la fois l'architecture globale et détaillée de notre logiciel. Nous avons décrit la vue dynamique du système en utilisant différents types de diagrammes UML (diagrammes de séquences détaillé, diagramme d'activités, diagramme d'états-transitions).
Le chapitre suivant portera sur la phase de réalisation du projet qui sera illustrée par une description des technologies ainsi que le matériel utilisés, une architecture physique et des captures d'écran des différents interfaces réalisés.

Chapitre 4

Réalisation

Introduction

Nous passons lors de ce dernier chapitre à la réalisation de la solution proposée. Dans un premier lieu, nous allons présenter les technologies et les outils d'implémentation utilisés pour développer notre logiciel. Dans un second lieu, nous allons décrire l'architecture physique du logiciel. Enfin, nous allons illustrer le travail réalisé par des captures d'écran.

4.1. Technologies

Dans cette section, nous définissons les technologies qui composent l'environnement de développement de notre logiciel.

FIG. 4.1 : Technologies utilisées

- **VM VirtualBox** : VM VirtualBox est un logiciel de virtualisation publié par Oracle.
- **Visual Studio Code** : Visual Studio Code est un éditeur de code développé par Microsoft.
- **Docker** : Docker est un logiciel permettant de lancer des applications dans des conteneurs logiciels.
- **Beats** : Beats est une plateforme qui accueille des agents réservés au transfert de données.
- **Logstash** : Technologie Big Data : Logstash est un outil informatique de collecte, de transformation et de stockage de logs.
- **Elasticsearch** : Elasticsearch est un logiciel utilisant Lucene pour l'indexation et la recherche de données. Il fournit un moteur de recherche distribué et multi-entité à travers une interface REST.
- **Kibana** : Kibana est un greffon de visualisation de données. Il fournit des fonctions de visualisation de contenu indexé dans une grappe Elasticsearch.
- **X-Pack** : Technologie Machine Learning : X-Pack est une extension d'Elastic Stack qui fournit des fonctionnalités d'authentification, de notification, et des outils de Machine Learning.

4.2. Outils d'implémentation

Le tableau 4.1 montre les caractéristiques des composants de l'environnement matériel.

Machine	Ordinateur Portable	Machine Virtuelle
Processeur	Intel Core i7-8750H 2.20GHz	Intel Core i7-8750H 2.20GHz
RAM	16 Go	8 Go
Système d'exploitation	Windows 10	Ubuntu 20.04
Utilisation	Rédaction des documents	Implémentation du projet

TAB. 4.1 : Caractéristiques matérielles

4.3. Architecture physique

4.3.1. Architecture de la pile ELK

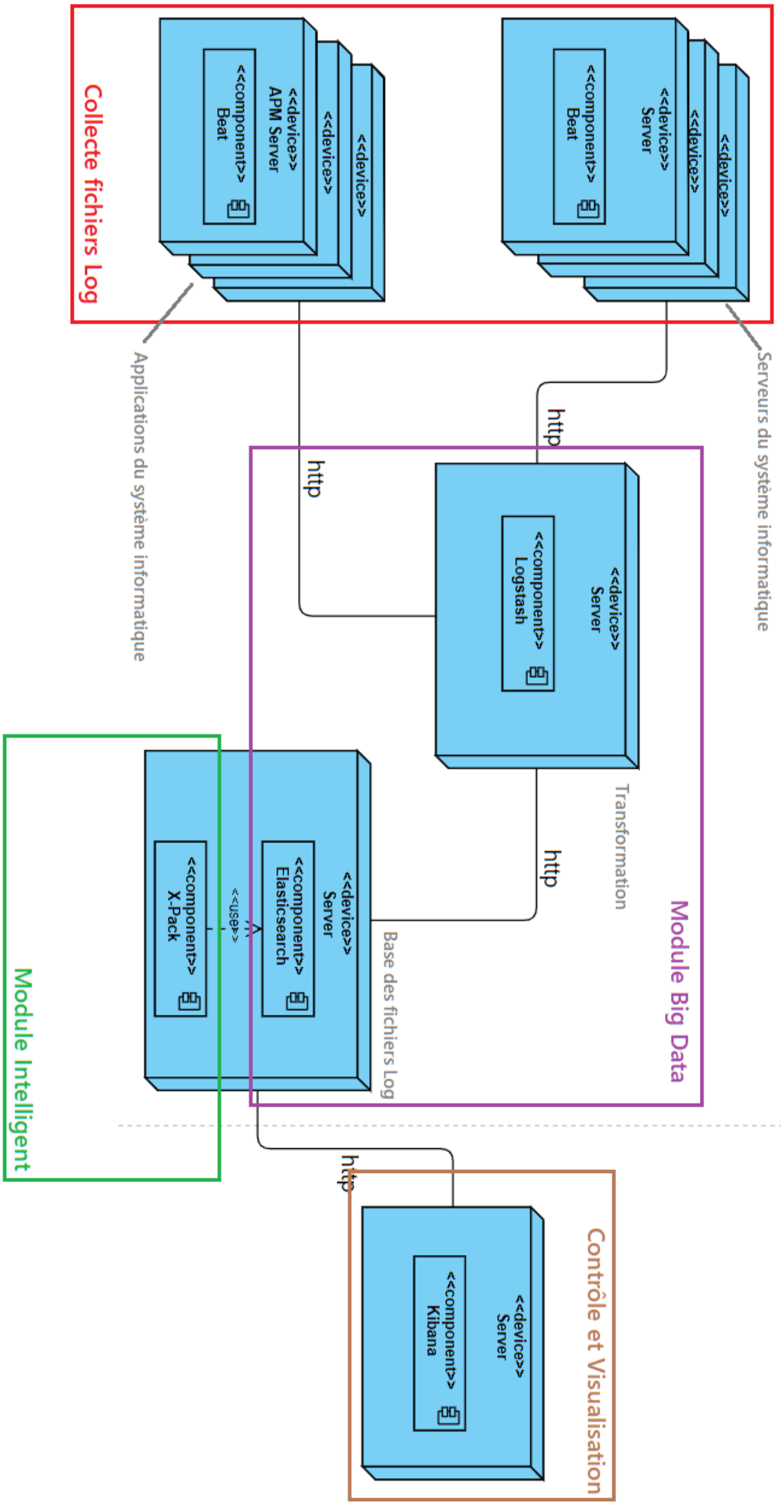
FIG. 4.2 : Pile ELK

La figure 4.2 illustre l'architecture de la pile ELK fournissant les composants de base de notre logiciel, en effet :

- **Beats** : représente le module "Collecte fichiers Logs" dans notre logiciel : outil installé dans tous les serveurs et les applications pour la collecte des Logs en utilisant le protocole HTTP pour l'envoie des données.
- **Logstash** : c'est l'outil représentant le module Big Data dans notre logiciel : responsable de la transformation et le stockage des Logs.
- **Elasticsearch** : représente la base des fichiers Logs après leurs transformations et fournit aussi un moteur de recherche.
- **Kibana** : c'est l'outil de visualisation des statistiques et des graphiques.

4.3.2. Diagramme de déploiement

Le diagramme de déploiement permet de décrire la composition matérielle et logicielle de l'application en montrant l'utilisation des technologies décrites dans le cadre du projet. Autrement dit, ce diagramme est sous la forme d'une superposition de l'architecture logique et de l'architecture physique du logiciel.


Collecte fichiers Log
<<device>>
Server
<<component>>
Beat
<<device>>
<<device>>
APM Server
<<component>>
Beat
<<device>>
<<device>>
Applications du système informatique
Serveurs du système informatique
http
http
http
http
Module Big Data
<<device>>
Server
<<component>>
Logstash
Transformation
<<device>>
Server
<<component>>
Elasticsearch
<<component>>
X-Pack
<<use>>
Base des fichiers Log
Module Intelligent
Contrôle et Visualisation
<<device>>
Server
<<component>>
Kibana

L'architecture physique de notre logiciel utilise la pile ELK, dont chaque composant est déployé sur un serveur pour assurer la bonne performance des différents modules.
Premièrement, nous avons intallé, pour chaque serveur et chaque application de notre système informatique, un agent "Beat" pour l'envoie de leurs fichiers Log à notre module Big Data.
Deuxièmement, Nous avons installé "Logstash" (Module Big Data) dans un serveur séparé pour recevoir les Logs, les transformer et les stocker dans la base.
Ensuite, nous avons déployé la base des fichiers Log "Elasticsearch" et le Module intelligent "X-Pack" dans un même serveur pour que l'accès aux données soit plus rapide.
Enfin, "Kibana", le reponsable de la visualisation, est installé dans un serveur différent pour afficher les résultats des requêtes des utilisateurs de notre logiciel.

4.4. Interfaces de l'application

Dans cette partie, nous allons présenter à travers des captures d'écran, les principales fonctionnalités de notre application.

4.4.1. Vue globale de l'état d'un serveur

La figure 4.3 montre un aperçu globale de l'état d'un serveur de notre système informatique à l'aide de mini-graphiques qui présentent des statistiques (taux d'utilisation de la RAM et du CPU, débit de transfert des données...).

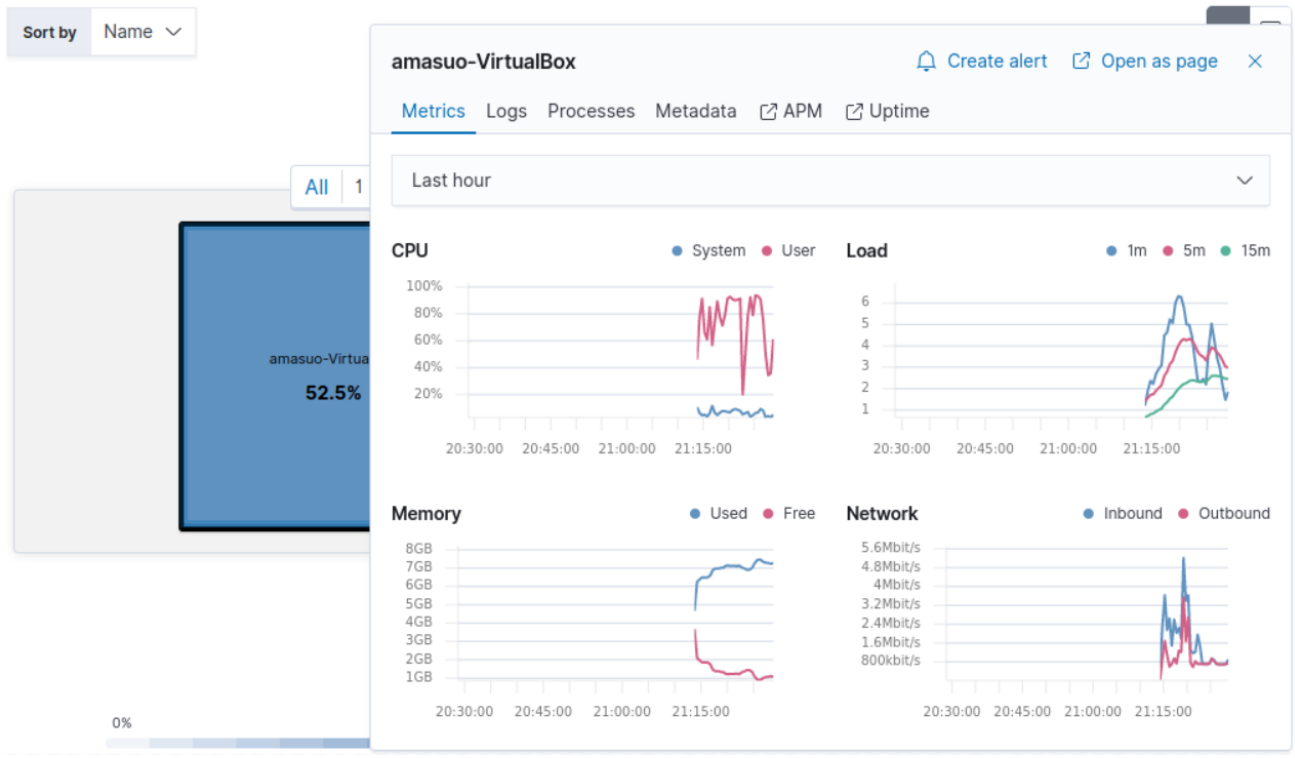


Fig. 4.3 : Vue globale de l'état d'un serveur

4.4.2. Vue détaillée de l'état d'un serveur

La figure 4.4 montre des informations détaillées sur l'état d'un serveur de notre système informatique.

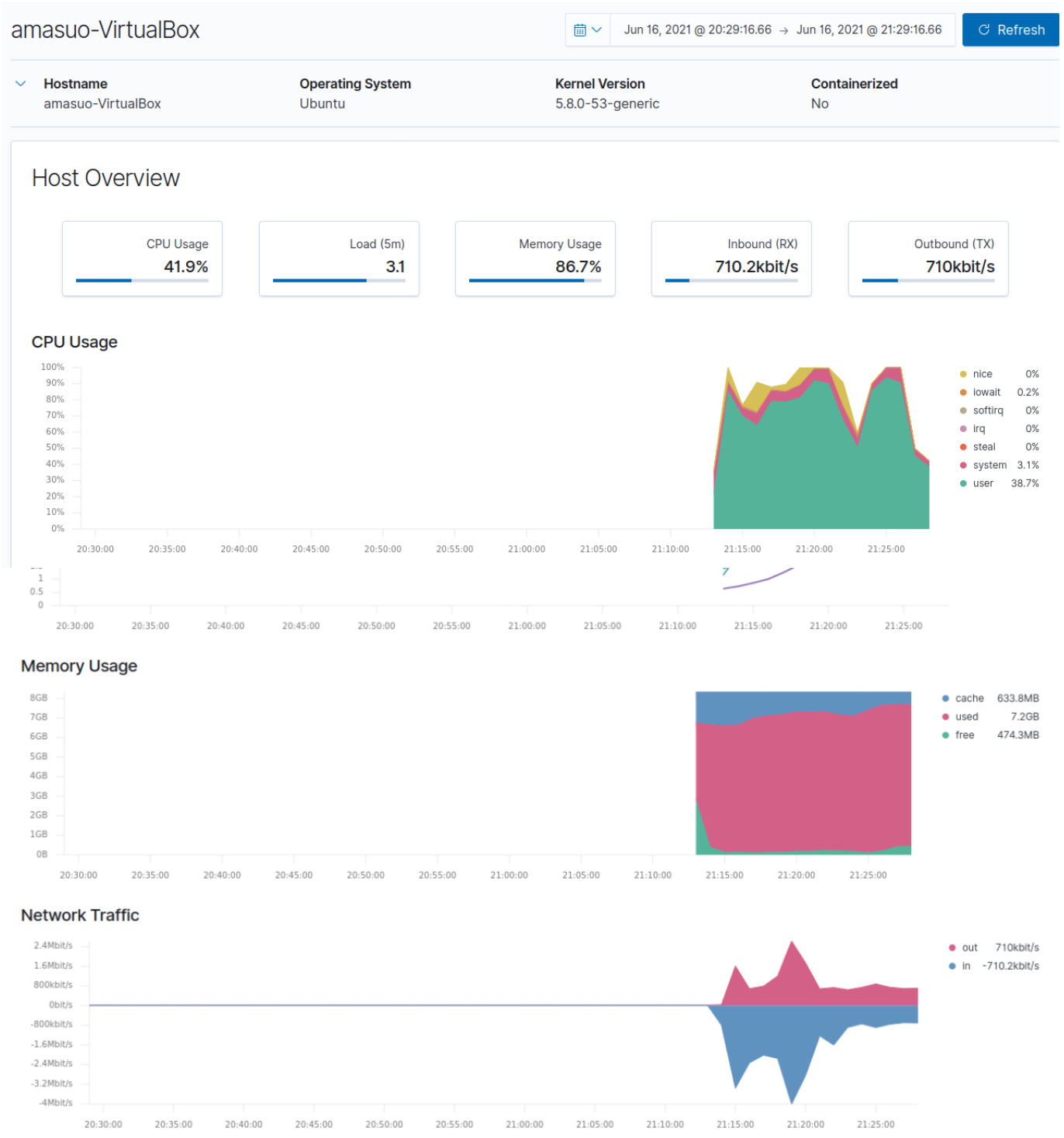


FIG. 4.4 : Vue détaillée de l'état d'un serveur

4.4.3. Vue détaillée de l'état d'une application

La figure 4.5 montre des informations détaillées sur l'état d'une application de notre système informatique.

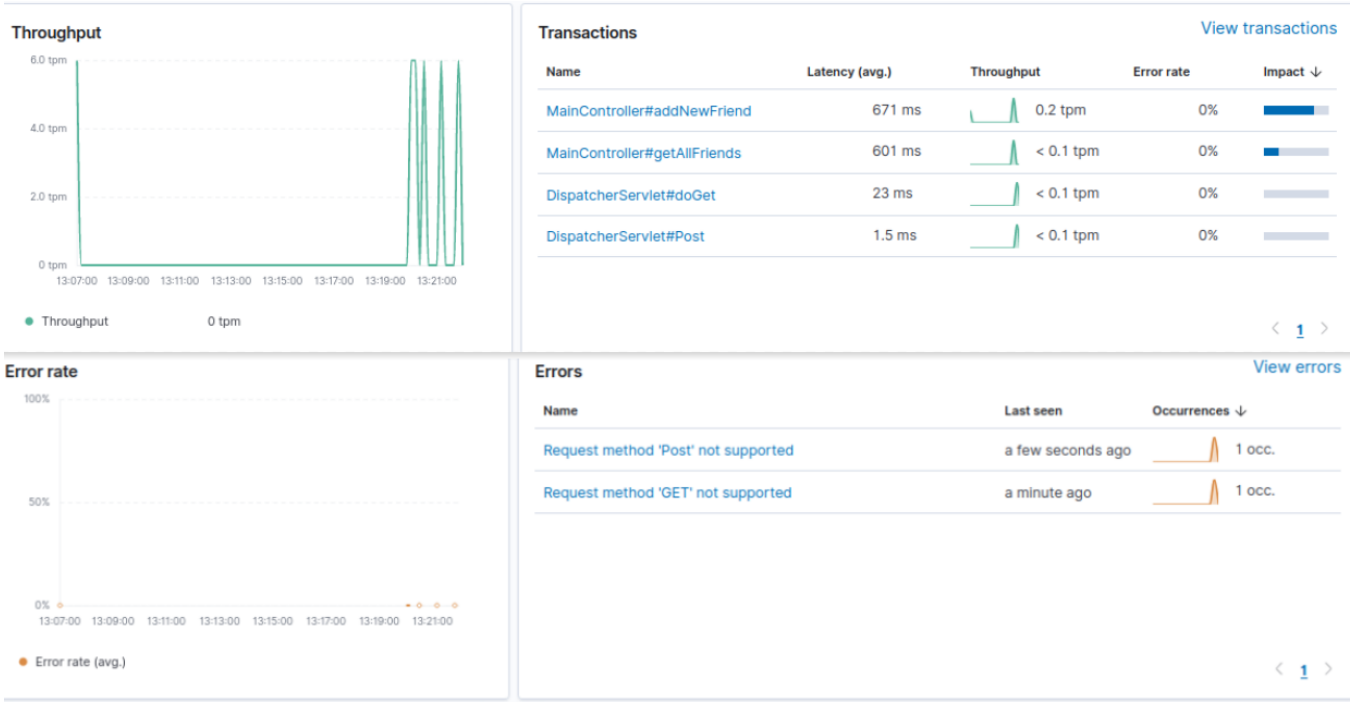


FIG. 4.5 : Vue détaillée de l'état d'une application

4.4.4. Création d'un Privilège

La figure 4.6 montre l'interface de la création d'un privilège destinée spécifiquement à l'administrateur. Parmi les privilèges, on trouve : le privilège d'accès aux interfaces des statistiques(pour le simple utilisateur), le privilège d'accès au module intelligent (pour l'investigateur), le privilège du paramétrage du système (pour l'administrateur) ...

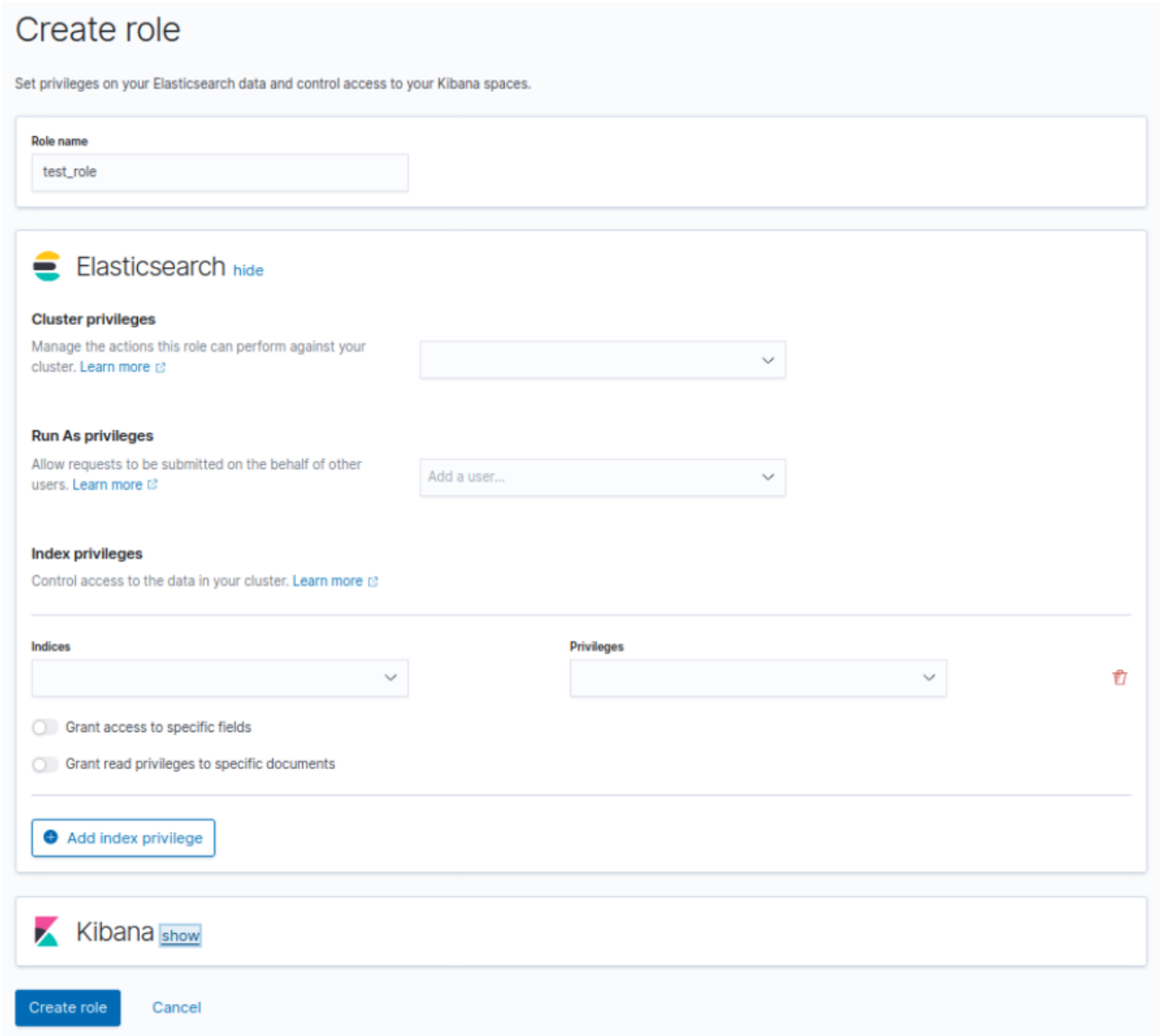


Fig. 4.6 : Creation d'un Privilège

4.4.5. Création d'une Alerte

La figure 4.7 montre l'interface de la création d'un seuil dont le dépassement déclenche une alerte. Cette interface est destinée spécifiquement à l'administrateur.

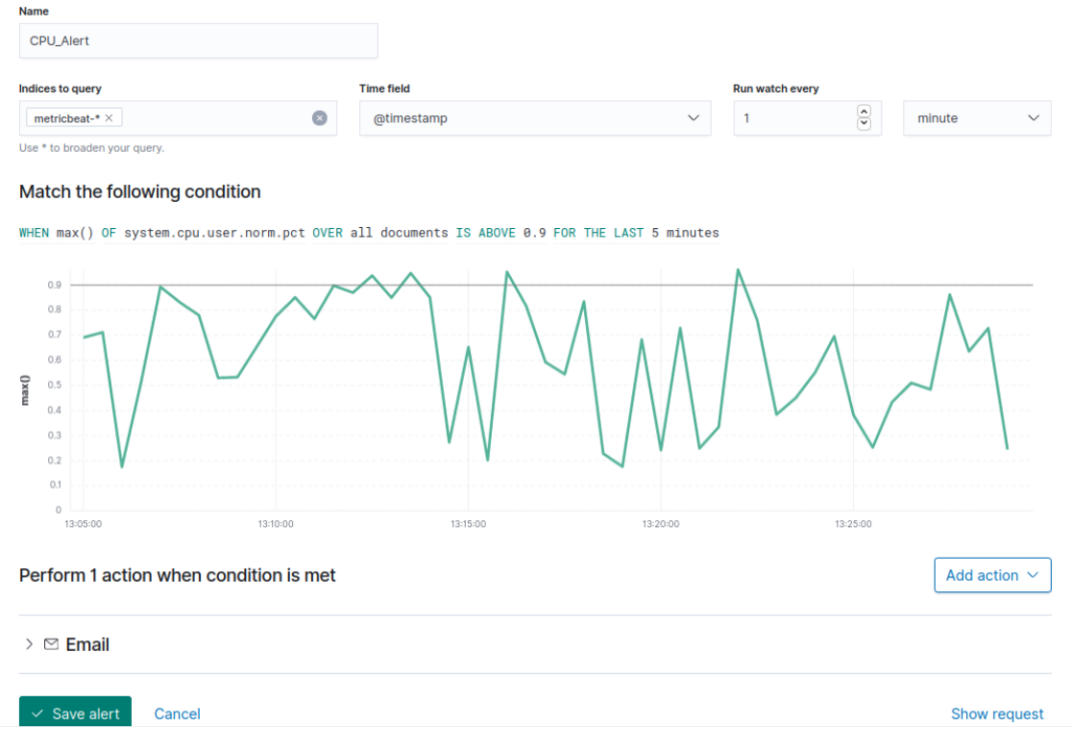


FIG. 4.7 : Création d'une Alerte

4.4.6. Liste des tâches du module intelligent

La figure 4.8 illustre la liste des tâches créées par l'administrateur pour la détection des anomalies. En effet, l'administrateur doit choisir le composant (serveur/application) pour lequel la détection des anomalies doit être effectuée. Après plusieurs sélections, la liste des tâches de détection d'anomalies que le module intelligent doit réaliser est formée.

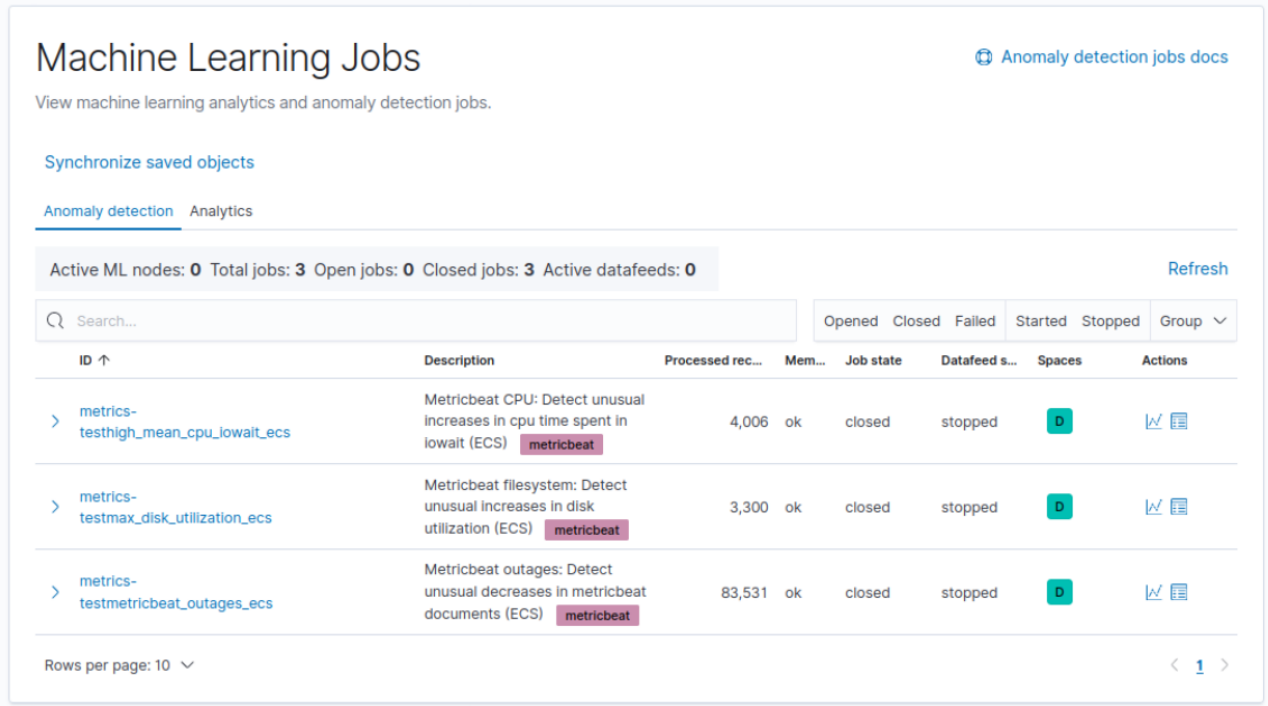


Fig. 4.8 : Liste des tâches du module intelligent

4.4.7. Détection d'une Anomalie

La figure 4.9 montre la capacité de détection d'une anomalie par notre module intelligent. En effet, le module intelligent a bien localisé un pic sur la courbe de taux d'utilisation du CPU dans le temps. Ce pic montre l'excès d'utilisation du CPU qui correspond à une anomalie.

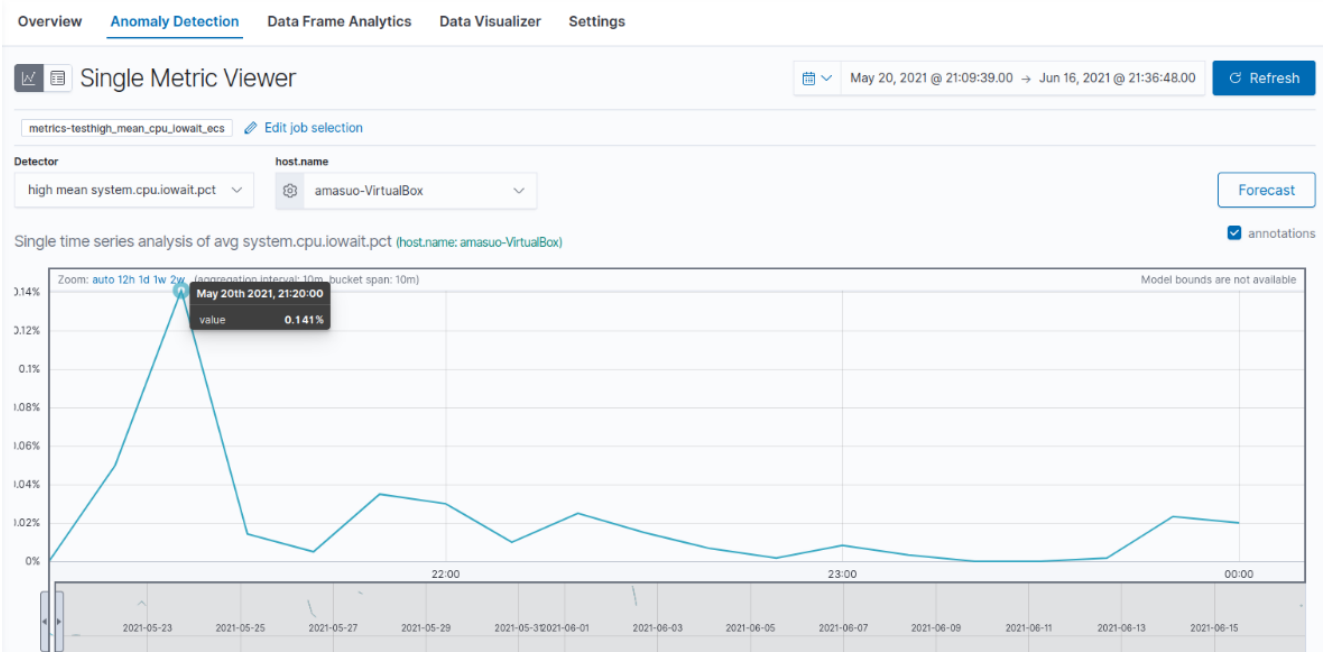


Fig. 4.9 : Détection d'une Anomalie

Conclusion

Au cours de ce dernier chapitre nous avons présenté l'environnement matériel et logiciel pour l'implémentation du logiciel, l'architecture physique à l'aide d'un diagramme de dépoloiement et les résultats de développement à l'aide des imprimes écran de quelques interfaces de notre plateforme.

Conclusion générale

Le présent rapport décrit les différentes étapes suivies pour mettre en place une plateforme de monitoring d'un système informatique. Notre logiciel est conçu pour l'utilisation par trois types d'acteurs partageant le droit de visualisation des statistiques permettant de suivre en temps réel l'état des composants logiciels et matériels du système informatique. De plus, il permet de notifier les investigateurs en cas de détection d'anomalies pour intervenir d'une manière immédiate. Aussi, notre logiciel fournit une fonctionnalité de protection contre les attaques des cybercriminels. Il facilite également les tâches administratives comme le déploiement des règles de détection d'anomalies et de proposition de solutions.

Pour atteindre notre objectif, nous avons commencé par une étude préalable pour bien comprendre le contexte du projet.
Nous avons présenté également les besoins fonctionnels et non fonctionnels qui nous ont permis de préparer une étude conceptuelle en modélisant notre système à l'aide des diagrammes de cas d'utilisation.
Ensuite, nous avons élaboré la conception de notre logiciel en commençant par une architecture globale qui présente les différents paquetages, suivi par des diagrammes de classes de chaque paquetage. Puis, nous avons modélisé la vue dynamique de l'application à travers les diagrammes de séquences, d'activité et d'état-transition.
Enfin, nous avons abordé l'étape de la réalisation durant laquelle nous avons traduit la modélisation conceptuelle en une implémentation physique.

Ce stage a constitué une expérience très riche sur le plan technique et professionnel. Sur le plan technique, ce projet a été une bonne occasion pour explorer de nouveaux domaines informatique qui sont en pleines croissances comme le Big Data et le Machine Learing, aussi pour découvrir et maitriser des technologies avancés comme la Pile ELK. Sur le plan professionnel, ce projet a été une véritable occasion pour vivre l'expérience de travail au sein d'une société.

En conclusion, les fonctionnalités de notre logiciel répondent parfaitement aux besoins fonctionnels et non fonctionnels de notre client. Néanmoins, nous pouvons prochainement automatiser totalement la tâche de proposition de solutions en cas de détection d'anomalies et éviter l'intervention de l'administrateur pour saisir les règles de résolution. Le module intelligent devient ainsi capable de proposer d'une manière autonome des solutions en se basant sur un mécanisme d'apprentissage intelligent.

Webographie

[1] *Log*. https ://www.ionos.fr/digitalguide/web-marketing/analyse-web/les-fichiers-log-lenregistrement-des-processus-informatiques/. Mar. 2021.

[2] *Anomalie*. https ://fr.wikipedia.org/wiki/Bug_(informatique). Mar. 2021.

[3] *Monitoring*. https ://fr.wikipedia.org/wiki/Surveillance_(informatique). Mar. 2021.

[4] *ELK Stack*. https ://www.elastic.co/. Mar. 2021.

[5] *NewRelicOne*. https ://newrelic.com/fr. Mar. 2021.

[6] *Dynatrace*. https ://www.dynatrace.com/. Mar. 2021.

[7] *Datadog*. https ://www.datadoghq.com/. Mar. 2021.

[8] *Comparaison solutions*. https ://www.itcentralstation.com/products/comparisons. Mar. 2021.

[10] *Architecture Kappa*. https ://www.researchgate.net/figure/Kappa-Architecture-based-on-7_fig7_334363591. Avril 2021.

Bibliographie

[9] Selma Belgacem. *Chapitre "Processus de développement", Cours Génie Logiciel*. Institut Supérieur des Sciences Appliquées et de Technologie de Sousse, 2020.

[11] Yosra Skah. *Conception et implémentation d'un système de traitement des évènements complexes*. Institut Supérieur des Sciences Appliquées et de Technologie de Sousse : Rapport de projet de fin d'études, Juillet 2019.

[12] Selma Belgacem. *Chapitre "Les patrons d'architecture", Cours Génie Logiciel*. Institut Supérieur des Sciences Appliquées et de Technologie de Sousse, 2020.

[13] Bochra Rabouche. *Chapitre "Apprentissage non supervisé", Cours Data Science*. Institut Supérieur des Sciences Appliquées et de Technologie de Sousse, 2021.