Création d'un système générique d'analyse de performance des applications

Ioana-Miruna Fotea

CONSERVATOIRE NATIONAL DES ARTS ET MÉTIERS

CENTRE RÉGIONAL DE TOULOUSE

MÉMOIRE

présenté en vue d'obtenir

LE DIPLÔME D'INGÉNIEUR CNAM

SPECIALITE : Informatique

OPTION : **Architecture et Ingénierie des Systèmes et des Logiciels**

par

Miruna Ioana FOTEA

Création d'un système générique d'analyse de performance des applications

Date de soutenance envisagée :29/10/2021

Jury

Membres :	M. POLLET Yann	Professeur au CNAM Paris
	M. CHAARI Lotfi	Maître de Conférences - HDR, INP Toulouse
	M. CRÉGUT Xavier	Maître de Conférences en informatique à l'ENSEEIHT
	M. BONNEL Jérémy	Chef de Projet TMA Rationnalisée – Sopra Steria Group

Remerciements

Tout d'abord, je souhaite remercier toutes les personnes qui m'ont permis d'être qui je suis aujourd'hui. Je tiens à adresser également un grand merci à l'entreprise Sopra Steria Group de de m'avoir réservé un accueil chaleureux.

Je tiens à remercier Monsieur Jérémy BONNEL mon tuteur de stage et chef de projet, pour s'être rendu disponible, pour m'avoir permis d'intégrer l'équipe du projet TMA Rationnalisée, pour les encouragements et pour la confiance qu'il m'a accordée.

Je voudrais remercier Monsieur Xavier CRÉGUT, mon tuteur et professeur au CNAM, qui m'a conseillée, accompagnée et encouragée, tout au long du déroulement de ce mémoire.

Merci aussi Anne-Sophie LARCEBEAU, Mathilde CUSNIR, Charles ALLARD et Guilhem ESCANDE et le reste de l'équipe pour la bonne ambiance, ainsi que pour leurs qualités humaines et pédagogiques et qui m'ont permis de monter en compétences tout en profitant de leur expérience techniques et managériales.

J'adresse mes remerciements aux professeurs de l'IPST-CNAM, ceux dont la bienveillance a été ressentie tout au long de ces années.

Merci à Gautier pour sa disponibilité et sa bienveillance, à Nathanael, Loïc et au reste de mes collègues pour les encouragements et les bons moments.

Je tiens à remercier ma famille et Alexandru qui pendant ces longs moments m'ont supportée, rassurée et guidée.

Et enfin mes remerciements à mes amis, et tout particulièrement à Milagros pour sa réactivité et sa disponibilité.

Table des matières

Introduction

Dans le domaine bancaire, au sein de l'entreprise Sopra Steria, une des exigences les plus importantes de la qualité est la performance des applications. Avec la présence de plus en plus marquée des applications temps réel, le niveau de performance d'une application bancaire est devenu un critère important. Pour cette raison la possibilité d'évaluation des performances d'une application est essentielle. Il existe différents types d'outils et types de tests qui peuvent être implémentés et réalisés sur des applications. La réalisation des tests de performance en elle-même demande beaucoup de temps et de ressources.

La qualité d'un logiciel est un critère essentiel sur lequel les clients sont intransigeants. Ce processus est composé de plusieurs phases qui permettent de mettre en place des actions qui vont faciliter la réalisation d'un produit qualitatif. Le respect de ce processus représente une garantie de qualité pour le client. Les différentes phases du processus appliqués dans l'entreprise sont décrites dans le document.

Pour un projet contenant plusieurs applications très différentes en termes d'organisation, d'utilisation et de technologies, la méthode de test de performances est propre à chacune d'entre elles. La réalisation de certains jeux de test peut prendre beaucoup de temps pour qu'il soit effectué à chaque évolution. D'où le besoin d'avoir une application qui peut mesurer les performances d'une manière simple, rapide et facile à implémenter sur toutes les applications indépendamment de leur type.

Le système décrit dans le document est conçu pour qu'il puisse analyser les résultats de test de toute application qui respecte des critères de log déterminés et la présence de tests unitaires. Ces deux critères sont le point commun de toutes nos applications sur le projet. L'utilisation du système nécessite l'installation d'un outil léger et l'utilisation d'un navigateur web pour pouvoir être exploité. Cette simplicité permet d'évaluer les performances de toutes les applications du système, avec un effort minimal et en peu de temps.

La réalisation des analyses est rapide et les rapports générés offrent une vision globale immédiate sur l'état des performances de l'application.

Dans un premier temps est décrit le cahier de charges qui permet d'identifier les fonctionnalités principales du système. Dans un deuxième temps est présenté l'état de l'art sur les méthodes d'évaluation des performances, les logs et leur analyse, pour finaliser avec les outils disponibles sur le marché et qui sont adaptés au besoin.

Une autre phase d'analyse est nécessaire pour identifier les composants et l'architecture adapté au système, pour ensuite décrire les outils choisis et la mise en œuvre pour chaque composant de l'architecture identifié.
Une évaluation de la solution est également présentée à la fin du document.

1. Contexte industriel

1.1. L'entreprise & son contexte

La société Sopra Steria est une Entreprise au Service du Numérique (ESN) née de la fusion entre Sopra et Steria deux ESN françaises en 2014. Sopra a été créée en 1968 et Steria en 1969. Sopra Steria est un leader européen dans la transformation numérique avec 4,3 milliards d'euros de chiffre d'affaires en 2020, ce qui la place au top 5 en Europe et en deuxième position au top des ESN, en France. Elle comptabilise plus de 46 000 collaborateurs dans plus de 30 pays, essentiellement implantés en Europe. Sopra Steria s'inscrit dans les trois principaux secteurs suivants : Le Secteur Public, La Banque et l'Aerospace / Défense / Sécurité. C'est une ESN indépendante, c'est-à-dire que c'est une entité juridique qui ne dépend de personne, son créateur Pierre PASQUIER est toujours aux commandes du groupe.

1.1.1. Activités

Sopra Steria s'inscrit dans diverses activités de la technologie digitale. Actuellement, les activités se concentrent autour de la réalité augmentée, la mobilité, les objets connectés, l'intelligence artificielle, le Big Data, le Cloud, les Smart Cities et la Cyber sécurité.
Du côté bancaire, Sopra Steria est en charge de la gestion du risque, de la protection des données bancaires et de la conformité réglementaire.
En ce qui concerne le marché du secteur public, les enjeux sont surtout centrés sur l'amélioration de l'efficacité des institutions publiques et de leurs organisations, notamment dans l'optimisation des fonctions de ressources humaines, de finance et du patrimoine de l'État. Sopra Steria offre des services d'administration en ligne ou de l'amélioration des services existants.
Le marché Aérospace et défense s'inscrit dans les objectifs d'accompagnement pour la performance industrielle, la chaîne d'approvisionnement, les systèmes embarqués, le service client, la recherche et développement (réalité augmentée) ainsi que la gestion du trafic aérien.

1.1.2. Organisation Midi-Pyrénées

La région Midi Pyrénées compte à elle seule plus de 2 100 collaborateurs avec 11 divisions dont les principales sont : Aeroline SIG (Système d'Informations de Gestion), Aeroline STIE, Conseil et Tertiaire regroupant des projets bien spécifiques. Les divisions Aeroline SIG et STIE (informatique Scientifique Technique Industrielle et Embarquée) s'occupent du secteur Aéronautique. La division Tertiaire gère principalement les secteurs Télécoms, Financiers et publiques. Chaque division est elle-même divisée en agence focalisée sur la division Tertiaire. J'ai effectué mon stage dans l'agence Banque au sein de cette division.

Figure 1 – Clients Vertical Banque

L'agence Banque a comme clients Crédit Agricole, notamment par le biais de la filiale Crédit Agricole Payment Services, Crédit Mutuel, La Banque Postale, La Banque de France, Neuflize OBC, le Groupe BPCE, BNP et Société Générale.

Cette agence fait partie de la division appelée Vertical Banque qui est étendue sur plusieurs sites en France et à l'étranger. En France, les centres principaux sont : à Toulouse, Albi, Rodez, Paris, Bordeaux, Lyon, Lille et Strasbourg.

De l'agence Banque font partie plusieurs projets dont la TMA Rationalisée. Les projets sont divisés en fonction des clients.

1.1.3. Environnement de poste

Depuis 2018, j'ai intégré la société Sopra Steria pour travailler sur le projet TMA Rationalisée. Ce projet permet de gérer la maintenance, dans de nombreux langages, d'environ 160 applications du Service Informatique (SI) du Crédit Agricole qui se nomme Crédit Agricole Payment Services (CAPS). Le projet TMA Rationnalisée est notamment réparti sur plusieurs sites, et ceux dont le projet concernent sont : Toulouse, Rodez, Albi et Madrid.

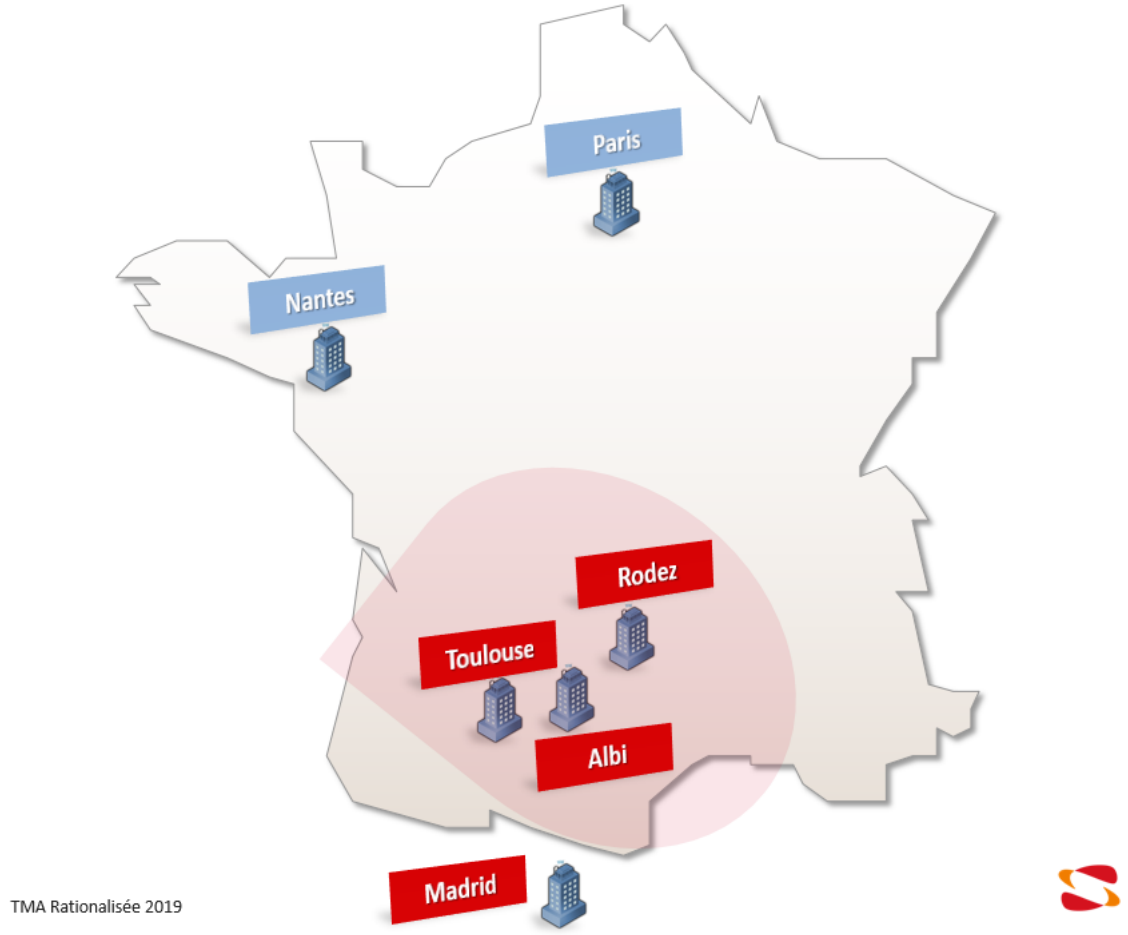

Figure 2 – TMA Rationalisée

L'organisation à l'intérieur de la TMA rationalisée est divisée en plusieurs Streams ou groupes selon la technologie utilisée par les applications de ce Stream.

Les Streams sont les suivants :

- Stream MVS (Multiple Virtual Storage) s'occupe des applications en Cobol sur l'environnement MainFrame d'IBM
- Stream C s'occupe des applications basées sur les technologies C
- Stream NT (Nouvelles Technologies) TLS (Toulouse) s'occupe d'une partie des applications basées principalement sur les technologies Java et qui est situé à Toulouse.
- Stream NT CSE (Centre de Services Espagne) qui s'occupe de l'autre partie des applications Java, mais qui est situé à Madrid.
- Stream BI s'occupe des applications de type décisionnelles (business intelligence)

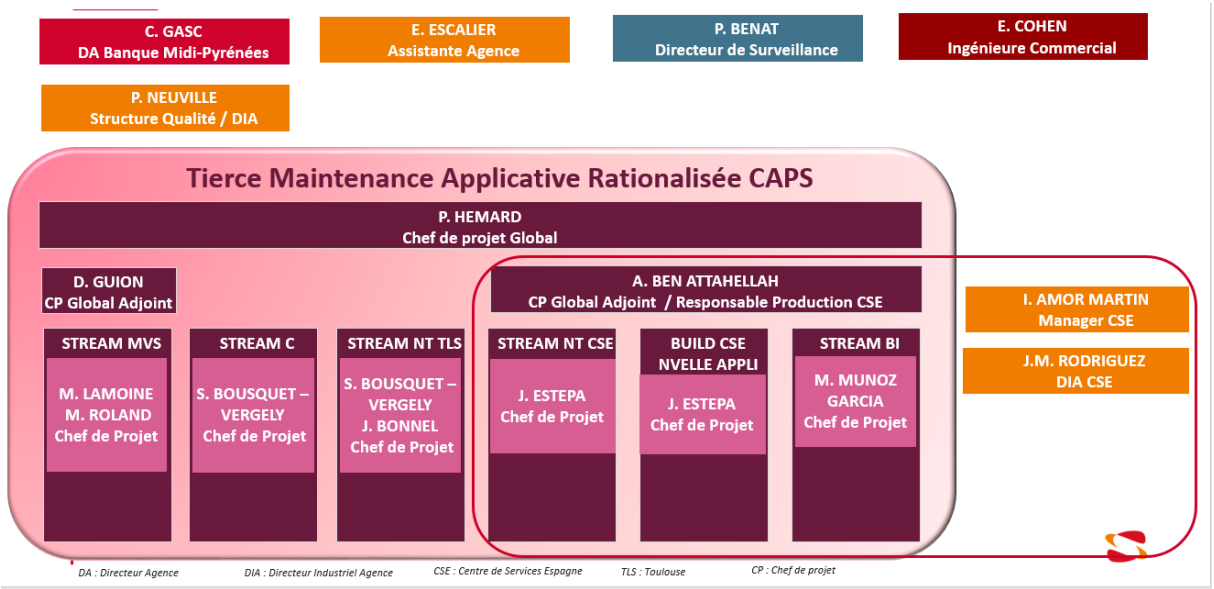

Figure 3 – Organisation Agence Banque

Je fais partie du Stream NT. Il est subdivisé en deux groupes. Le premier est présent dans les locaux de Toulouse (Stream NT TLS) et le second quant à lui est situé à Madrid (Stream NT CSE).

Au sein du Stream NT TLS, les projets sont gérés par une personne responsable de l'application. En fonction de la demande du client, de la difficulté ou de l'urgence de l'évolution ou des incidents, plusieurs personnes interviennent sur ce projet.
Dans le cadre de mes missions, j'ai travaillé dans des équipes composées de 2 à 7 personnes avec des compétences différentes sur plusieurs types d'applications.
Nous traitons deux types de demandes : les demandes d'évolutions qui doivent être livrées au client dans les temps établis lors de la demande officielle et les demandes d'incidents qui doivent être réglées dans un temps limité, en fonction du niveau d'urgence.

Les évolutions

L'organisation des projets est basée sur une méthodologie cycle en cascade. Ce modèle comporte 7 phases :

- Analyse des besoins, étape qui se déroule entre le responsable de l'application et le client au moment de la demande de ce dernier
- Chiffrage des temps de mise en œuvre
- Création des spécifications suite à la démarche réalisée avec le client
- Conception détaillée
- Implémentation de la demande
- Tests de validation
- Installation en environnement de qualification (simili environnement de recette)

Chacune de ces phases doivent produire un ou plusieurs livrables définis à l'avance et à une date d'échéance fixée. Le critère de validation du client est ce qui permet de passer aux différentes phases du projet à travers les différents livrables. Ainsi, si le client est satisfait, il est possible de passer à la phase suivante, sinon on remonte à la phase précédente.
Au niveau du suivi, nous utilisons un système de tickets créé dans l'outil Jira, une fois que la demande du client est validée. Sur chaque ticket est présent le nom de l'application, le nombre de jours disponibles pour mettre en place la solution et la personne qui l'a prise en charge.

Les incidents

Les demandes d'incidents sont de 3 types :

- Bloquant si une application ou une fonctionnalité est inutilisable avec un impact direct sur le client final (Caisse Régionale et/ou client de la banque)
- Majeur quand l'incident rend l'application ou une fonctionnalité principale inutilisable
- Moyen quand l'incident concerne une partie de l'application qui ne fonctionne pas correctement.
- Mineur quand l'incident est un léger disfonctionnement d'une fonctionnalité et peut être traité avec une solution de contournement

Les jours de résolutions des incidents sont établis en fonctions du classement des applications du projet TMA :

- Or
- Argent
- Bronze

Les applications classées "or" sont les plus critiques car elles traitent des données essentielles au fonctionnement du système de la banque. Donc le niveau de criticité est plus important. Par exemple un incident classé bloquant pour une application « or » a un délai de résolution de 0,5 jours.
Les applications classées "argent " sont utilisées pour le fonctionnement interne du système de la banque, mais leur fonctionnement reste indispensable au client.
Les applications classées "bronze" sont des applications peu critiques avec des délais de résolutions plus longs que les applications "or" ou "argent", par exemple un incident classé urgent par le client a un délai de résolution de 3 jours.

1.1.4. Qualité logicielle

Pour l'entreprise, maintenir un niveau de qualité de travail est un point essentiel, tout en respectant les délais lors d'une phase de livraison.
Par définition, la qualité d'un logiciel est une appréciation globale du système, qui consiste à livrer un produit qui soit conforme aux exigences client en termes de fonctionnalités, mais aussi d'interopérabilité, de sécurité, de fiabilité, de performances et de maintenabilité.

Différentes procédures sont mises en place pour garantir la qualité d'un produit.
La première procédure d'assurance qualité comprend la préparation d'un plan de test, la planification et l'exécution de tests et la gestion des données de tests. Les plans de test sont préparés en avance de phase et validés avec le client et les responsables projet. La gestion de données est une phase qui est différente de projet à projet en fonction du classement des applications. Le plus souvent les données de test sont des données de préproduction anonymisés selon la législation, afin de nous permettre de couvrir tous les cas de tests possibles.

Au moment de la finalisation des phases de développement, une phase de test est prévue pour vérifier que les différents attributs d'un système et les différents aspects de son utilisation fonctionnent comme prévu et qu'ils n'exécutent aucune des fonctions qu'ils ne sont pas supposés exécuter. L'objectif des tests de logiciels est de détecter les problèmes, mais aussi de s'assurer que les problèmes détectés sont entièrement corrigés sans aucun effet secondaire.

En fonction du type de projet, plusieurs types de test peuvent être réalisés, comme : tests fonctionnels, test de performance, tests de compatibilité, tests d'utilisation, test de fiabilité, tests de sécurité, tests de maintenabilité et tests de portabilité.

1.2. Contexte du projet

Dans le contexte bancaire, la qualité des logiciels est essentielle. De plus le niveau de performance des applications est toujours plus demandé par le client, surtout pour les applications temps réel. Par exemple, un paiement par carte doit être validé quasi instantanément. Pour cette raison, le maintien d'un bon niveau de performance est important pour notre société.

À présent nos applications sont développées en plusieurs versions de Java, qui vont de Java 8 jusqu'à Java 11. Sur la majorité des applications les tests unitaires sont réalisés avec JUnit et automatisés grâce à Maven. Cette variété de versions et de technologies rend la réalisation des tests spécifiques à chaque application, et donc l'analyse des performances difficile.
Les tests unitaires permettent de vérifier le bon fonctionnement d'une méthode, mais dans notre cas, le temps de réponse de ces derniers, peuvent devenir un indicateur de performance de l'application.

Après avoir effectué une modification dans une application, il est nécessaire de rejouer tous les tests automatiques prévus pour celle-ci et évaluer l'amélioration ou la dégradation des performances apportée par la modification, ce qui implique d'avoir un historique des tests précédents afin de comparer les résultats. Cette étape peut être longue et difficile à faire.

1.2.1. Identification du besoin

Le but de ce système est de permettre aux développeurs d'évaluer de manière automatique les performances de l'application après une modification.

Le système reçoit les fichiers de log résultant de Maven comme données en entrée. Il est capable de détecter le type de log et d'analyser son contenu.

L’analyse de ces logs est présentée à l’utilisateur via une IHM sous forme de rapport et il est alors possible de l’exporter sous format PDF ou CSV.

Le système est capable de garder l’historique des résultats et d’en donner l’indice de comparaison avec les précédents. Tous les temps d’exécution des tests unitaires sont sauvegardés dans la base de données avec les noms des méthodes correspondantes, la date d’exécution, le nom de l’application et la version.

Le système permet de réitérer l’analyse des tests. Les analyses peuvent être faites entre les résultats de tests de la même application avec une version différente, ou entre les résultats de tests de la même application, avec la même version.

Le rapport contient la différence entre les temps de réponse des tests réalisés sur la version actuelle ainsi que la précédente de la même application, et également des indicateurs statistiques de ces données (moyenne, minimum et maximum). Une courbe des résultats des tests des différentes versions, est également visible.

Le système peut être intégré sur toutes les applications de manière simple. L’intégration doit être compatible avec Maven.

2. Analyse Conceptuelle

L'analyse conceptuelle a pour but d'identifier les concepts clés du système à partir du cahier des charges exprimé précédemment.
Dans cette partie seront expliquées les exigences de haut niveau ainsi que les cas d'utilisations et les processus métier du système.

2.1. Description des processus métier

Afin d'avoir une idée plus précise des différents processus métier du système, l'utilisation de la norme de modélisation des processus métier (BPMN) permet d'avoir une vision plus globale, de haut niveau. Cette vision aide à mieux identifier les besoins du client.

Les processus métier permettent de modéliser les flux d'informations déduits de l'analyse du besoin du client exprimés précédemment.

Processus global

Avec le BPMN qui suit sont exprimées, d'un point de vue utilisateur, les étapes nécessaires que doit contenir le système. Ces étapes sont liées entre elles afin de garantir le fonctionnement du système, comme décrit par le client. Dans ce cas, le processus commence quand l'utilisateur souhaite réaliser une analyse ou en relancer une, et le processus se termine soit avec la visualisation du rapport, soit avec le téléchargement de ce dernier.

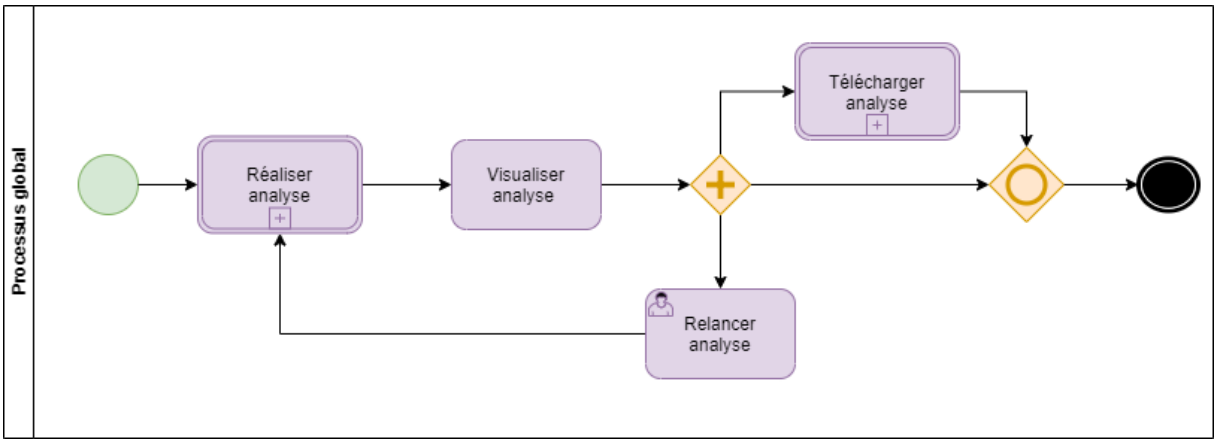

Figure 4 – BPMN – Processus global

Réaliser analyse

Le processus de réalisation de l'analyse est plus complexe car plusieurs composants sont impliqués. Pour pouvoir réaliser une analyse sur les résultats de tests unitaires, le processus doit récupérer les données des tests réalisés sur la version en cours et les données des tests réalisés sur la version précédente pour créer le rapport. Le processus se termine par l'affichage du rapport contenant l'analyse.

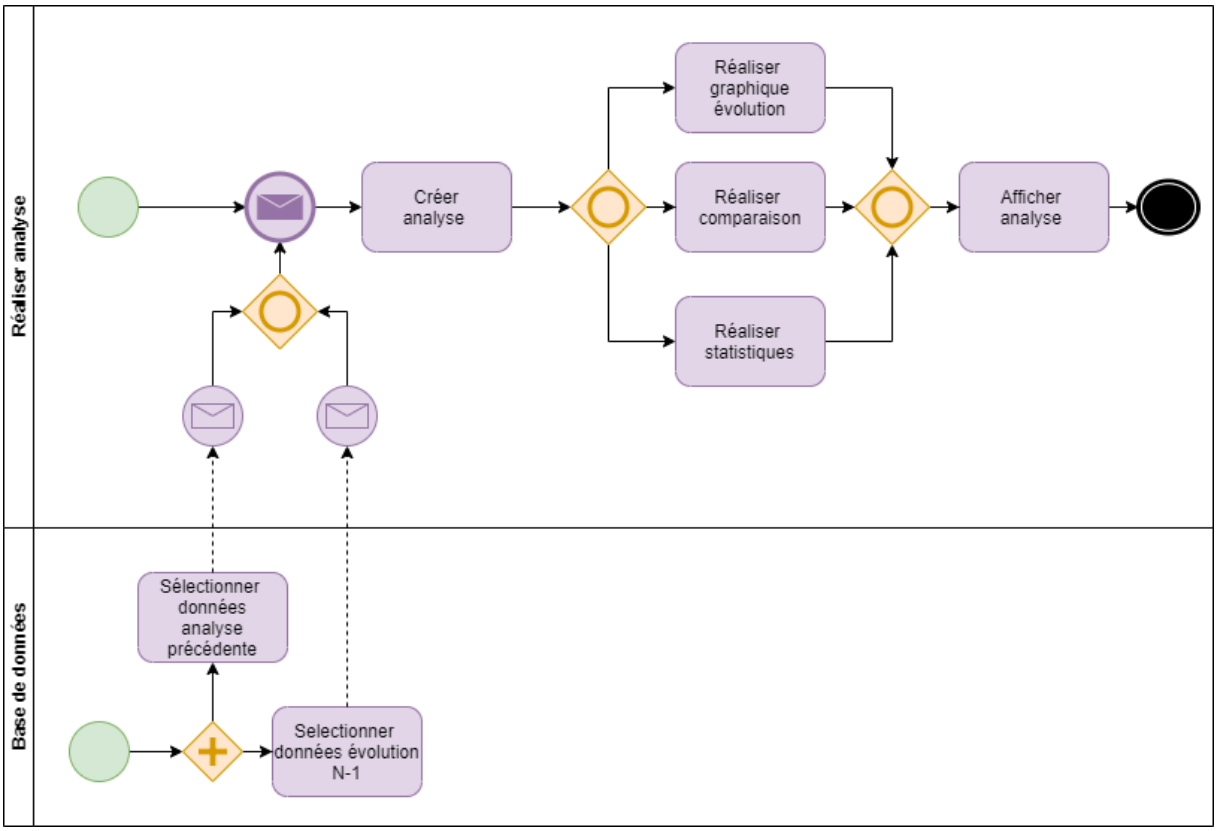

Figure 5 – BPMN – Réaliser analyse

Télécharger analyse

Ce processus illustre quelles sont les étapes de téléchargement d'un rapport. Le processus est démarré quand l'utilisateur décide de télécharger le rapport contenant l'analyse réalisée, et il se termine avec le téléchargement de cette dernière dans le format souhaité.

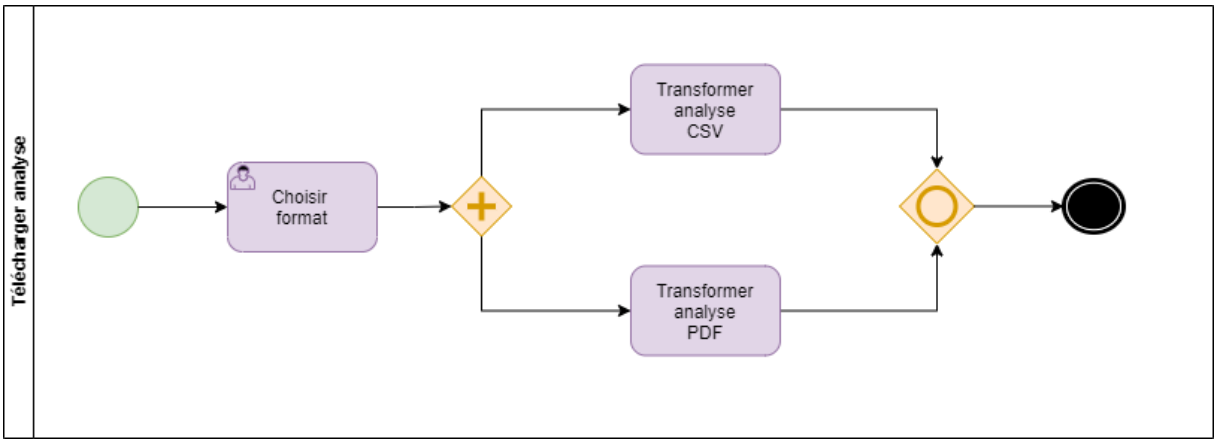

Figure 6 – BPMN – Télécharger analyse

2.2. Les acteurs du système

Après avoir identifié les étapes générales qui composent le système, pour obtenir le résultat désiré, il est nécessaire d'identifier les acteurs humains et non-humains qui vont interagir avec le système.
Dans notre cas, les acteurs du système sont les développeurs qui s'occupent de la gestion d'un projet et souhaitent évaluer les performances de ce dernier.

2.3. Diagramme de cas d'utilisation

Avec ce type de diagramme sont illustrés les cas d'utilisation qui permettent à l'acteur de se servir du système. Le cas d'utilisation « Analyser résultats » est décrit dans le tableau ci-dessous. Les autres cas sont décrits dans les tableaux en annexe.

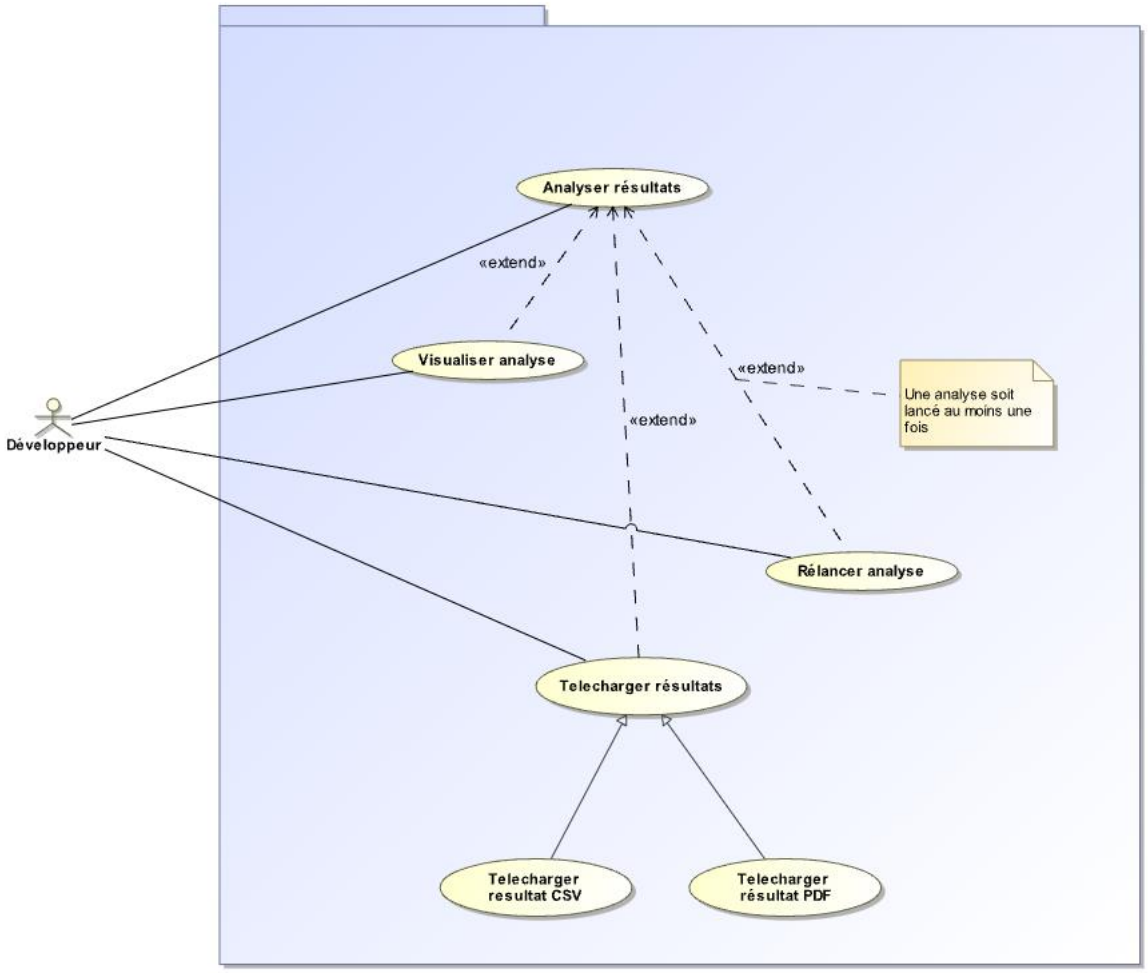

Figure 7 – Diagramme de cas d'utilisation

Cas d'utilisation « Analyser résultats »

Cas d'utilisation :	Analyser résultats
Acteurs :	Développeurs des applications
Scénario nominal détaillé :	Le développeur réalise l'analyse de des performances d'une l'application.
Conditions préalables :	Le développeur doit avoir installé le composant de collecte du système sur l'application qu'il gère. Le développeur doit être connecté au réseau interne à l'entreprise.
Postconditions :	/
Flux Autre(s) :	/
Spécifications supplémentaires :	L'application doit être paramétrée dans le système.

Figure 8 – Description cas d'utilisation « Analyser résultats »

2.4. Description des exigences HLR (High Level Requirements)

Il est possible de déduire les exigences de haut niveau du système à partir des cas d'utilisation et des besoins du client.

Dans le tableau ci-dessous sont présentes les exigences du système de haut niveau. Les exigences renseignent comment les parties prenantes visualisent le système, son comportement, son interaction avec les utilisateurs et l'environnement dans lequel s'exerce toute l'activité de ce dernier.

N°	Description	FCT/ NFCT[1]
1	Le système doit pouvoir intégrer comme données entrantes les fichiers de log résultants de Maven.	FCT
2	Le système doit pouvoir s'intégrer avec les applications.	FCT
3	Le système doit pouvoir réaliser une analyse des logs et la présenter sous forme de graphique avec en plus des indicateurs statistiques des temps de réponse (temps moyen, temps minimum et temps maximum)	FCT
4	Le système doit avoir une IHM d'affichage des rapports.	FCT
5	Le système doit pouvoir permettre d'exporter un rapport sous format PDF et CSV.	FCT
6	Le système doit être capable d'historiser tous les temps de réponse dans une base de données.	FCT
7	Le système doit pouvoir donner un indice de comparaison avec les résultats des tests précédents.	FCT
8	Tous les temps d'exécution des tests unitaires sont sauvegardés dans la base de données avec les noms des méthodes correspondantes, la date d'exécution, le nom de l'application et la version.	FCT
9	La IHM doit avoir un seuil configurable d'acceptation des temps de réponse des tests. Le seuil détermine si le test est dégradé ou pas.	FCT
10	La IHM doit permettre de visualiser la différence des résultats des tests sur une courbe afin de donner une vision globale des résultats.	FCT
11	La IHM doit permettre de refaire une analyse et d'afficher les résultats.	FCT

Figure 9 – Matrice des exigences de haut niveau

[1] FCT=exigence fonctionnelle, NFCT= exigence non fonctionnelle

2.5. Analyse des données

En fonction des exigences de haut niveau, il est possible d'avoir une première vision sur les données qui vont circuler dans le système. A ce propos, la représentation logique de l'organisation des informations et de leurs relations seront exprimées grâce au modèle conceptuel des données. Le MCD est exprimé en format UML.

Les données faisant partie du système sont : « Application » qui contient le nom de chaque application et sa version, « Rapport » qui représente les analyses générées par le système, avec un identifiant pour chaque rapport et les différentes statistiques, « Tests » qui contient les informations des logs générés pour chaque application avec identifiant fichier et le *timestamp* des tests réalisés et enfin « Méthode » qui contient les noms des méthodes testées ainsi que leur temps d'exécution.

Un rapport peut être généré à partir de plusieurs tests. Un test est composé des méthodes testées et il fait référence à l'application.

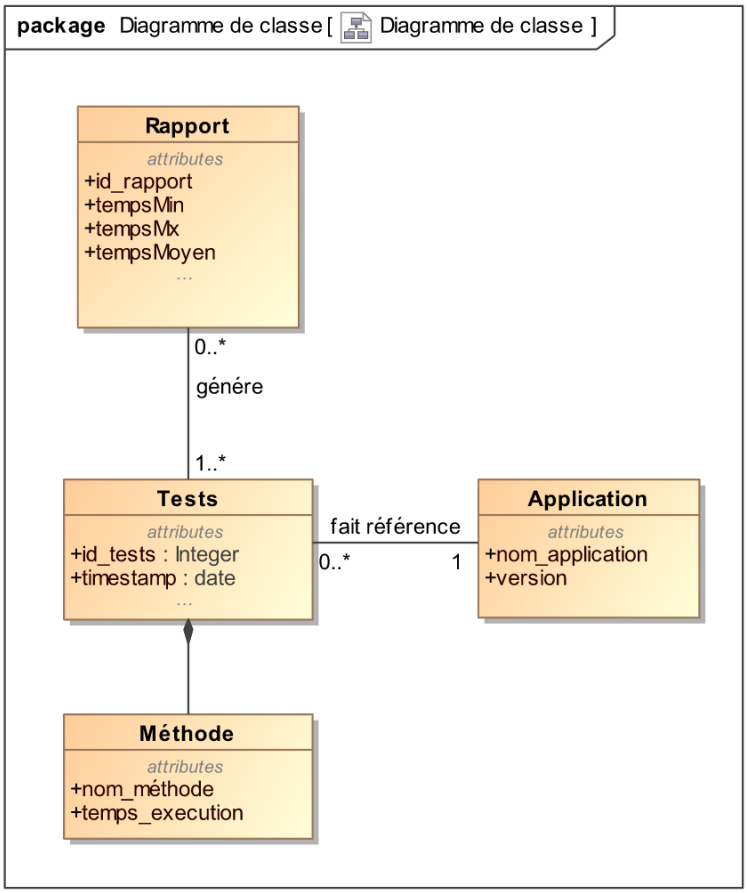

Figure 10 – Modèle conceptuel des données

Nom	Type	Description
id_rapport	int	Identifiant unique du rapport créé
tempsMax	float	Temps maximum d'exécution de la totalité des tests
tempsMin	float	Temps minimum d'exécution de la totalité des tests
tampsMoyen	float	Temps moyen d'exécution de la totalité des tests
nom_application	char	Nom de l'application
version	char	Version de l'application au moment des tests
id_tests	int	Identifiant unique du fichier de log contenant les tests réalisés
timestamp	Date	Timestamp de réalisation des tests
nom_méthode	char	Nom de la méthode testée
temps_execution	char	Temps d'exécution de la méthode testée

Figure 11 – Description des données

2.6. Stratégie d'analyse

Afin de pouvoir proposer une solution qui répond au cahier des charges, une méthodologie de gestion de projet et une stratégie d'analyse sont nécessaires. L'organisation du travail permet de mener à bien le projet.

La stratégie d'analyse choisie suit le modèle VBS (Value Breakdown Structure). Il est l'équivalent du WBS (Work Breakdown Structure). Le modèle VBS est plus adapté aux méthodes Agiles. Il permet de découper le travail en éléments qui seront présents par la suite dans le Backlog. Le niveau de granularité est suffisant pour que toutes les étapes du projet soient regroupées en plusieurs livrables. En même temps, il est possible de réaliser en priorité les composants du système avec de la valeur ajoutée.

La figure suivante montre la décomposition du projet en étapes séquentielles conformément aux étapes de conception.

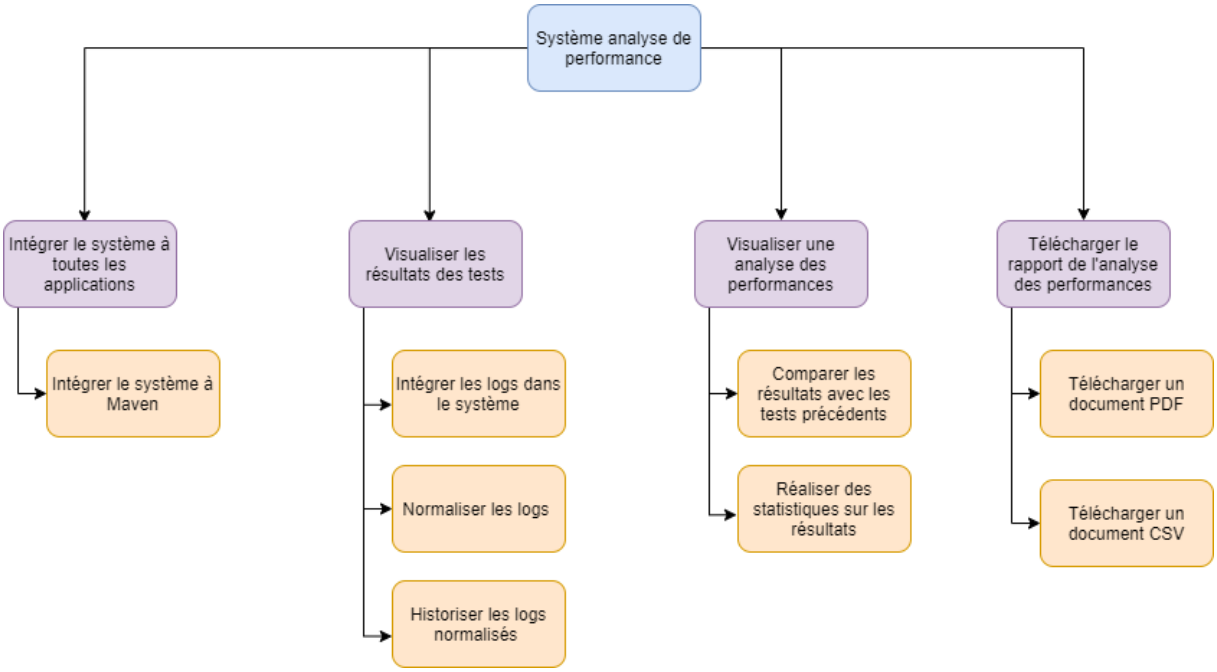

Figure 12 – Value Breakdown Structure

2.7. Gestion de projet

2.7.1. Méthode

La réalisation du système est faite en suivant la méthode Kanban. C'est une méthode pour implémenter le développement logiciel Agile et DevOps. Cette méthode permet de donner visibilité à tout moment aux collaborateurs, tout au long de la réalisation du processus. Les tâches sont représentées sur le tableau Kanban, ainsi, les participants peuvent suivre l'état de chaque tâche.

La méthode Kanban est issue de l'industrie automobile au Japon. Elle a été créée par Taiichi Ōno pour Toyota en 1950 pour d'optimiser la capacité de production de l'entreprise.
La méthode Kanban se base sur l'approche Lean, qui consiste à améliorer en continu la procédure de production.

Kanban signifie « étiquette » en japonais. La méthode se base sur un système de cartes ou d'étiquettes, qui correspondent chacune à une partie du processus. Grace à ces cartes est possible visualiser de manière explicite les tâches à réaliser, la logique de réalisation, et les tâches déjà réalisées.[5]

L'avantage de la méthode Kanban est qu'elle permet d'identifier rapidement les problématiques : si plusieurs étiquettes s'accumulent et stagnent dans la même étape, alors il y a un point bloquant qu'il faut résoudre.

L'inconvénient est que cette méthode n'est pas adaptée à tout type de projets comme ceux qui peuvent recevoir des demandes trop irrégulières. Ce type de demande ne respecte pas le même circuit de travail, ce qui rend la méthode Kanban peu efficace.

2.7.2. Langage de modélisation

Le langage de modélisation utilisé pour exprimer les différentes parties du système est UML. Le langage UML (Unified Modeling Language, ou langage de modélisation unifié) est, comme son nom l'indique, un langage de modélisation visuelle. Il a une sémantique et syntaxique spécifique. Il est utilisé pour l'architecture, la conception et la mise en œuvre de systèmes logiciels avec une structure ou comportement complexe. [24]
Ce choix a été fait car le système est composé entièrement de parties de type logiciel, donc un langage comme le SysML aurait été superflu.

2.7.3. Outils

Les outils employés pour la réalisation de l'analyse, de conception et de réalisation du système sont les suivants :

Cameo Systems Modeler

Cameo Systems Modeler est un environnement collaboratif d'ingénierie pour les Model-Based Systems Engineering (MBSE), qui fournit des outils intelligents, robustes et intuitifs pour définir, suivre et visualiser tous les aspects des systèmes modélisés. Il est conforme avec les schémas et les normes SysML et UML. [25]
Cameo est un outil prévu pour la modélisation détaillée des systèmes, afin de pouvoir visualiser son comportement dans les détails. Ainsi il permet la gestion des exigences, la gestion de la traçabilité entre différents niveaux d'abstraction, la génération des rapports et de la documentation à partir du modèle créé, l'adaptation et l'ajustement de l'outil à un domaine spécifique via un DSL.

Drow.io/ Diagrams.net

Diagrams.net (précédemment appelé draw.io) est un logiciel de création de diagrammes. Il peut être utilisé pour créer des organigrammes, créer des diagrammes UML ou SysML en ligne, concevoir le schéma de base de données ou construire des BPMN.
Diagrams.net est un outil en ligne, gratuit. Il permet de créer des schémas à partir d'une page blanche ou de se servir des nombreux *templates* disponibles parmi des modèles classiques, des *charts* ou des tableaux. Il permet également d'enrichir les différents schémas en ajoutant des images, des formes ou des arrière-plans et d'exporter les fichiers de travail en différents types de formats.
Diagrams.net est accessible sur tous les navigateurs web et il peut être téléchargé pour une utilisation offline. Le logiciel est compatible avec PC, Mac et les ordinateurs fonctionnants sous Linux. [26]

Trello

Trello est un outil de gestion de projet en ligne, inspiré par la méthode Kanban expliqué précédemment. Il consiste à créer un tableau digital de gestion de projet qui permet de séparer et répartir les tâches. Chaque tâche est présente sur le tableau sous forme de carte. Le tableau est divisé en plusieurs colonnes qui peuvent être, par exemple : "A faire", "En cours" et "Fait". D'autres colonnes peuvent être ajoutés en fonction du processus de travail. Les cartes peuvent être assignés aux participants et transférées d'une colonne à l'autre, pour indiquer l'avancement du processus. Des codes couleurs peuvent être utilisés pour distinguer les différentes propriétés des tâches. Plusieurs options sont disponibles pour chaque carte, comme des *checklists*, des dates limites et des notifications pour indiquer un changement de la tâche.
Trello propose une version de base gratuite, mais il existe aussi en version payante avec plusieurs fonctionnalités. L'outil est disponible en plusieurs langues .[19] L'accès à la version gratuite de Trello se réalise avec la création d'un compte sur le site. Une fois le compte créé, il suffit de créer et personnaliser le tableau pour l'adapter au mieux au projet.

2.8. Solution envisagée

La solution envisagée dérive des différentes phases d'analyse décrites précédemment. Elle consiste à réaliser un système qui reçoit en entrée un fichier de log d'une application du projet TMA Rationnalisée, avec l'objectif d'homogénéiser les données pour ensuite les analyser. Le système donne également la possibilité à l'utilisateur de visualiser ou télécharger le résultat de l'analyse sous forme d'un rapport.

2.8.1. Analyse boite noire

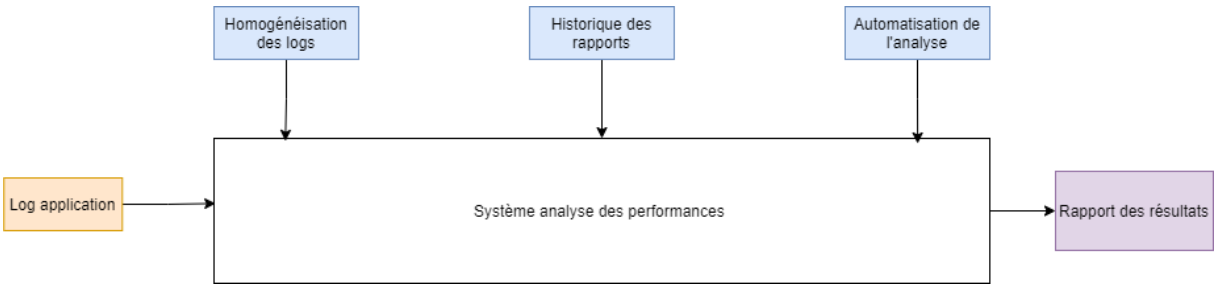

Figure 13 – Vue boite noire du système

La vue boite noire du système permet de représenter de manière générale les entrées et les sorties du système sans se soucier du fonctionnement de ce dernier. En plus sont représentés les différents facteurs externes qui ont un impact sur son fonctionnement. Dans notre cas, ces facteurs sont l'homogénéisation des données, car les fichiers de log ont une structure différente en fonction de l'application, le deuxième facteur est lié à l'historisation des rapports créés et le dernier est lié à l'automatisation du système.

2.8.2. Problématique

Afin de répondre à ce besoin et de réaliser la solution envisagée, le système devra exécuter l'évaluation de performance de manière automatisée.

Les points les plus importants sont liés à l'homogénéisation des logs qui ont une structure différente en fonction de l'application qui les produit. La structure de chaque fichier de log doit être découpée et stockée en parties identifiables par le système.
La manière dont les logs seront stockés assure la présence d'un historique, donc les logs doivent être décomposés afin d'obtenir les informations liées à l'application, à la version testée, au test réalisé ainsi que le *timestamp*.

A partir de ces informations, le système doit analyser et comparer les résultats de versions différentes ou de la même version, afin d'extraire un rapport de performances. L'analyse consiste à calculer la différence de temps de réponse des tests effectués entre deux versions de la même application, et de réaliser des statistiques de l'application au global. Afin de pouvoir réaliser cette comparaison, les données doivent être stockées dans une base de données.

Les points critiques de cette problématique sont :

- Comment réaliser l'homogénéisation de différentes structures de log
- Comment historiser les rapports
- Comment analyser leur contenu de manière automatique

3. Automatisation des tests de performance

3.1. Etat de l'art

Dans le cadre du projet TMA Rationnalisé, maintenir la qualité des applications est un point essentiel. Le domaine de la banque est particulièrement sensible à cet aspect, car offrir un service de qualité aux clients permet de surmonter la concurrence. Le système d'automatisation des tests de performance permet de valider de manière rapide les développements réalisés et maintenir une qualité de service optimale.

3.1.1. Performances

Définition

En informatique, la performance d'une application est définie comme « un ensemble d'indications chiffrées mesurant les possibilités maximales ou optimales d'un matériel, d'un logiciel, d'un système ou d'un procédé technique pour exécuter une tâche donnée ».[18]

Au niveau d'un système informatique, la performance est une notion plus vaste qui comprend les aspects suivants : les temps de réponse, la disponibilité du système, la robustesse et la capacité de montée en charge. Ces éléments permettent d'évaluer les performances d'un système de manière globale et forment un ensemble de critères pour la création des jeux de tests.

Le temps de réponse : est le temps nécessaire afin qu'une action exécutée par l'utilisateur soit exécutée pas le système informatique. La vitesse de chargement d'un site est un exemple de temps de réponse.

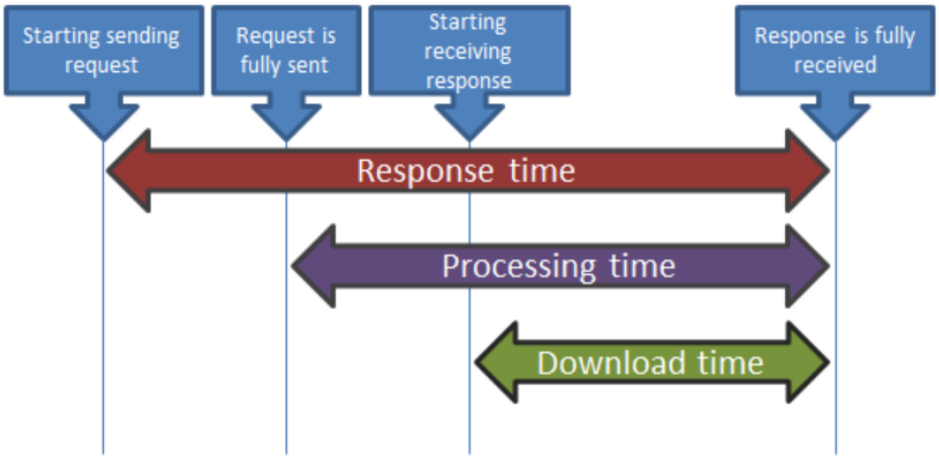

Figure 14 – Temps de réponse d'un système[27]

Plusieurs types de tests ont comme critère d'évaluation les temps de réponse, par exemple les tests d'évaluation des performances des applications web.

La disponibilité d'un composant ou d'un système informatique désigne la période pendant laquelle il est en état de fonctionnement nominal. La disponibilité est évaluée dans des conditions dégradées, sur une période donnée, pour simuler la perte d'un des composants du système.

Dans les entreprises, les applications arrivent à avoir une disponibilité jusqu'à 95%. Les systèmes qui peuvent garantir encore une meilleure disponibilité sont ceux avec une architecture à « haute disponibilité ».

La disponibilité peut concerner :

- Une application ou un service ;
- Un équipement matériel ;
- Un équipement réseau ;
- Un *cluster* applicatif ;

Une architecture à haute disponibilité permet de garantir jusqu'à 99 % de disponibilité pour ses composants, grâce à différents mécanismes spécifiques, comme représenté dans l'image ci-dessous.

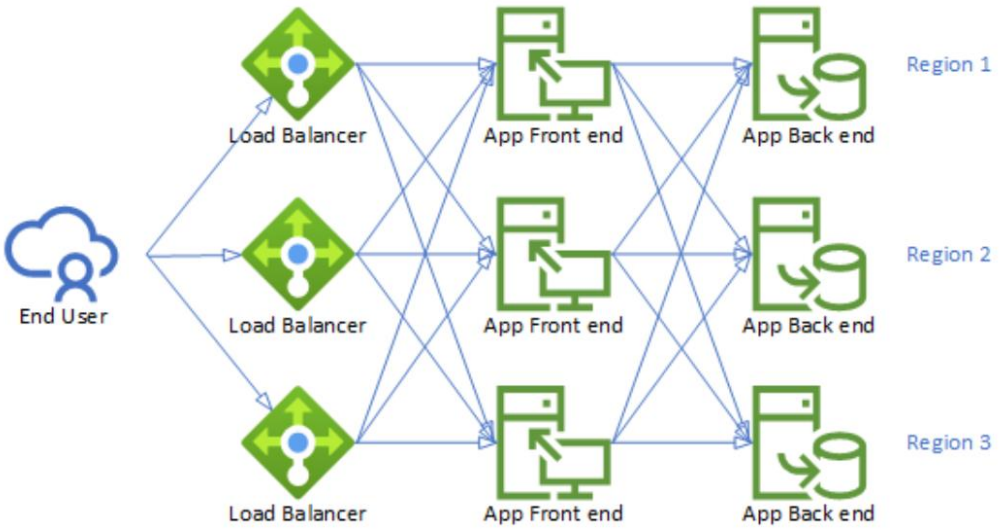

Figure 15 – Haute disponibilité d'un système [28]

La robustesse désigne la capacité d'un système à faire face à des erreurs lors de l'exécution, sans perdre ou corrompre des données ou des messages. La robustesse est un indicateur de l'intégrité des informations en situation de stress.

L'intégrité peut faire référence à des données échangées avec d'autres systèmes ou avec un composant de stockage.
La robustesse s'exprime par un indicateur, comme : le nombre maximal de message perdus est de 1 pour 100 000 messages traités ou le nombre maximal de redémarrages par mois d'une base de données est égal à deux. [11]

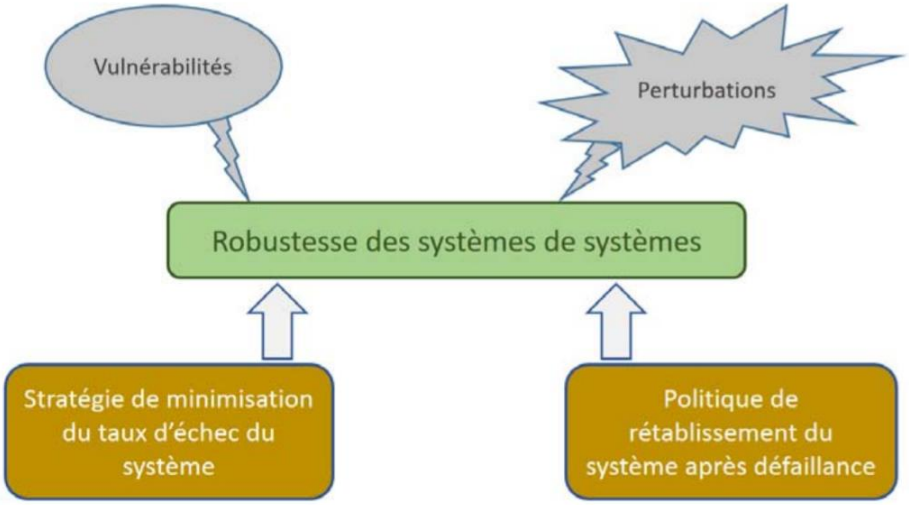

Figure 16 – Modèle d'amélioration de la robustesse d'un système[4]

La scalabilité ou capacité à monter en charge, représente l'aptitude d'un système à s'adapter au nombre croissant de demandes, tout en gardant la même qualité de service.

La capacité à monter en charge est une mesure donnée par le temps maximal de traitements pour un niveau de charge donné. Par exemple : 3 secondes est le temps maximum accepté pour afficher une page web, lors d'un accès simultané de 100 utilisateurs.

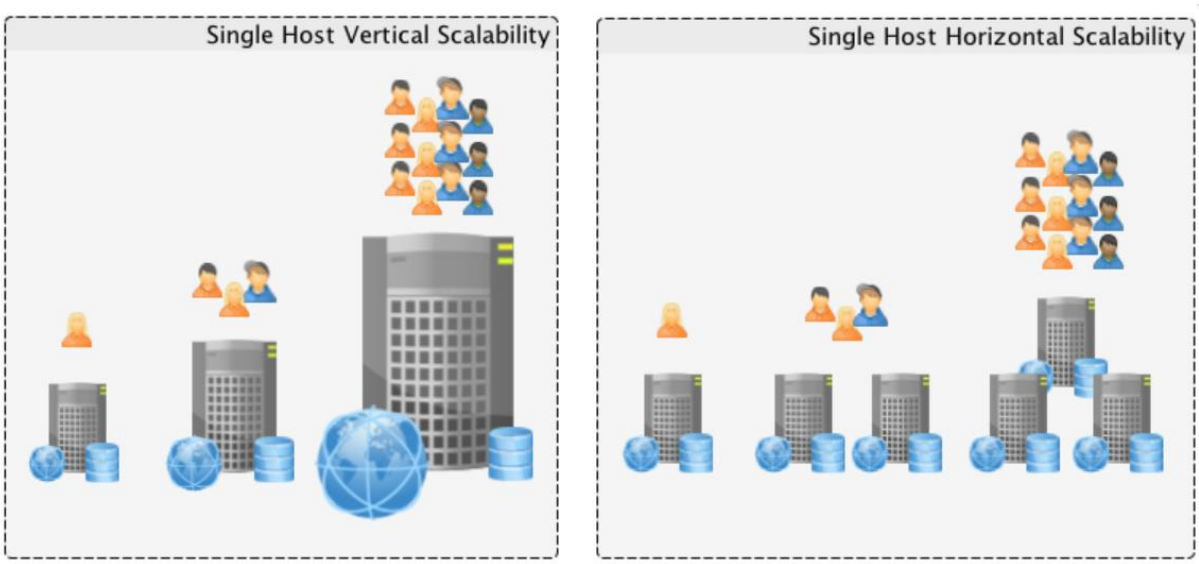

Figure 17 – Scalabilité verticale et horizontale d'un système [1]

Comme il est possible d'observer dans la Figure 17 ci-dessus, il existe deux façons de gérer la scalabilité :

- Scalabilité verticale qui consiste à augmenter les ressources des serveurs, comme le nombre de processeurs ou la mémoire.
- Scalabilité horizontale qui consiste à ajouter des serveurs supplémentaires pour augmenter les capacités et les ressources disponibles. Cette manipulation peut être faite de manière automatique ou manuelle, en fonction de l'architecture du système.

Méthodes de test

Les tests de performance ont pour objectif l'estimation de la performance d'un système informatique. Les mesures de performance d'un système sont définies au début du projet (temps de réponse maximum admissible par page, nombre d'utilisateurs simultanés…) et sont un critère d'évaluation de la qualité du système.

Afin d'évaluer les performances d'un système informatique, les tests qui peuvent être effectués sont décrits ci-dessous.

Test de charge : est un type de test qui va simuler l'utilisation progressive de l'application par un nombre d'utilisateurs jusqu'à arriver à dépasser de manière considérable le nombre d'utilisateurs cible. De cette manière est validée la capacité de l'application à répondre à une charge attendue d'utilisateurs. De plus, ce type de test permet d'identifier les points critiques de l'architecture.

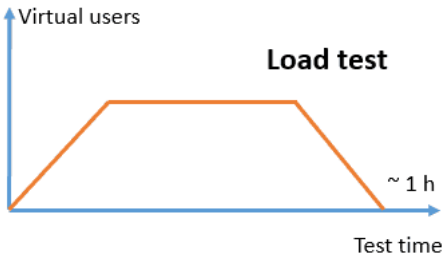

Figure 18 – Réalisation d'un test de charge[29]

Test de performance : est un type de test qui permet de mesurer les performances de l'application lors de son utilisation dans des circonstances normales, par un seul utilisateur. Les informations obtenues permettent de valider les temps de réponse de l'application, mais aussi ceux de l'environnement matériel et réseau. Pour cette raison les tests doivent s'effectuer dans un environnement similaire à celui de production.

La différence avec le type de test précédent est que celui-ci ne valide pas les performances pour la charge attendue en production, mais il vérifie plutôt les performances intrinsèques à différents niveaux de charge d'utilisateurs.[11]

Test de dégradations des transactions : il s'agit d'un test qui consiste à simuler l'activité transactionnelle d'un seul scénario fonctionnel parmi tous les scénarios du périmètre des tests, de manière à déterminer quelle est la charge limite que le système est capable de maintenir pour chaque scénario fonctionnel. De cette manière est possible d'individuer les transactions qui dégradent les performances du système.[20]

Test de stress : il s'agit d'un test qui consiste à simuler l'activité maximale attendue pour tous scénarios fonctionnels confondus, afin de voir comment le système réagit au maximum de l'activité. L'objectif est de simuler l'activité des utilisateurs en heure de pointe, et de maintenir cette charge pendant une période déterminée, afin de vérifier que le système puisse supporter cette charge.

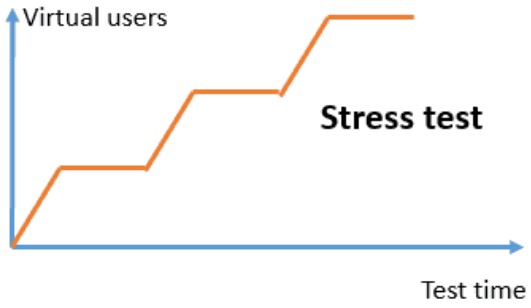

Figure 19 – Réalisation d'un test de stress [29]

Test de robustesse, d'endurance et de fiabilité : ce type de test permet de vérifier si le système informatique est capable de supporter une activité importante pendant une longue durée. Pour effectuer ce type de test il est nécessaire de simuler une charge conséquente d'utilisateurs sur une période relativement longue.

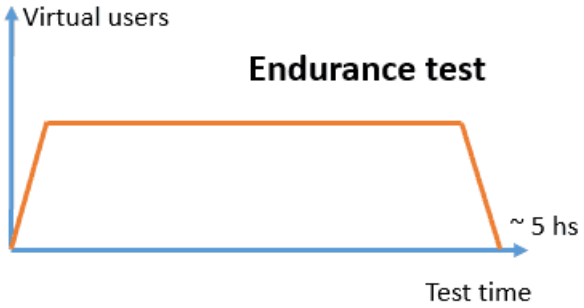

Figure 20 – Réalisation d'un test d'endurance[29]

Test de capacité, de montée en charge : est un type de test qui consiste à simuler un nombre croissant d'utilisateurs, de manière à déterminer quelle est la charge limite supportée par le système.

Test aux limites : il s'agit d'un test où le nombre d'utilisateurs simulés est nettement supérieur au nombre attendu. Similaire au test de capacité, il ne teste pas seulement l'augmentation d'un nombre d'utilisateurs simultanés, mais aussi les possibilités d'augmentation du nombre de processus métier réalisés pendant une période déterminée. [20]

Gestion de la performance dans un système informatique

Dans les domaines de l'informatique, la gestion des performances applicatives ou APM consiste à surveiller et gérer la disponibilité des logiciels. Application Performance Management est un système de gestion qui permet de détecter les problèmes de performances applicatives complexes et de les corriger afin de maintenir le niveau de service attendu. Ce système se base sur un aspect important qui est le monitoring.

Le monitoring, ou supervision d'une application, permet non seulement d'avoir une vue globale sur l'état d'une application à un instant précis, mais aussi d'avoir un historique de ses états passés. La supervision d'une application génère des indicateurs sur les performances de cette dernière et de son écosystème, comme les temps de réponse des différentes ressources du serveur.[16]

Deux ensembles de mesures de performance sont possibles. Le premier définit les performances ressenties par les utilisateurs finaux de l'application. Un exemple de performance est le temps de réponse moyen sous charge de pointe.

La charge d'une application est le volume de transactions traitées, par exemple les demandes traitées par seconde ou, dans le cas des application web, le nombre de pages demandées par seconde. Il existe de nombreux types de tests qui se basent sur le volume de transaction, comme les tests de charge ou les tests aux limites, présentés précédemment.[3]

Le second ensemble d'indicateur de performances mesure les ressources de calcul utilisées par l'application, indiquant s'il existe une capacité adéquate pour prendre en charge la demande, ainsi que les emplacements possibles d'éléments qui sont moins performants. La mesure de ces quantités établit une référence de performance empirique pour l'application en termes d'espace mémoire ou type de serveur. La ligne de base peut ensuite être utilisée pour détecter les changements de performance. Ces derniers peuvent être corrélés à des événements externes. Ces éléments peuvent être utilisés par la suite pour planifier des changements dans les environnements de l'application.[12]

Les logiciels utilisés pour la gestion des performances, permettent de surveiller et de donner des indicateurs de performance d'une application grâce à la collecte de données. Ces outils peuvent se baser sur les données de log pour évaluer le système. Cependant tous les outils ne permettent pas d'accéder directement aux logs pour identifier la source du problème.[6]

Il existe aussi la possibilité de réaliser directement l'analyse des logs afin d'évaluer la disponibilité et les performances d'une application. L'analyse des logs implique la collecte, l'évaluation et la gestion des données rapportées par divers composants. Il s'agit de la pratique de la gestion de toutes les données de log produites par les applications et l'infrastructure. L'analyse des logs peut être faite par des logiciels ou à l'aide de l'intelligence artificielle si la quantité de données est importante.[30]

La collecte et la structuration des logs est un point important qui peut améliorer de manière considérable l'efficacité de l'analyse, soit pour un programme dédié soit pour un logiciel de monitoring. Dans le paragraphe suivant seront décrites les structures des logs, qui peuvent être déterminantes pour l'analyse des informations.

3.1.2. Système des journaux ou logs

Définition

En informatique le terme « log » définit un journal de fonctionnement d'un serveur, d'un logiciel, d'une application et/ou d'un système. Dans la plupart des cas, les logs sont contenus dans un fichier texte dans lequel les évènements sont classés de façon chronologique et ligne par ligne. [31]

Contenu

La structure des logs contient les informations nécessaires à l'identification la plus précise possible de l'état de l'application. Parmi les informations les plus utilisées sont présentes : la date et l'heure de l'évènement, le niveau de log, le message de log, le nom de la fonction qui déclenche l'évènement et la *stack trace* en cas d'erreur. D'autres données comme les identifiants de l'utilisateur, le nom de l'application ou le nom du serveur peuvent s'ajouter à ces logs pour faciliter les recherches ultérieures. Cependant, les données présentes dans les fichiers log doivent être conformes à la législation du pays où l'utilisateur se trouve. En France, la CNIL s'occupe de la régularisation de la collecte et l'exploitation des données.

Objectif

Le *logging* est une pratique courante sur les systèmes applicatifs, indépendamment du langage de programmation ou du type d'application. L'objectif des logs est de permettre de tracer les évènements effectués par le système. Les logs peuvent être générés par tout type de système comme des logiciels, des équipements réseaux ou des applications.

Grâce aux informations contenues par ces fichiers, il est possible de remonter à la cause des problèmes et ainsi de corriger le dysfonctionnement de manière plus efficace. Sans les fichiers de log, il n'est pas possible de prévenir ou résoudre des problèmes.

Pour cette raison il est important de réaliser un bon paramétrage de ces fichiers afin d'avoir les données nécessaires à l'identification des problèmes lorsque ceux-ci surviennent.

Types de log

Les logs d'événements enregistrent les événements qui se produisent dans l'exécution d'un système afin de fournir des informations qui peuvent être utilisées pour comprendre l'activité du système et diagnostiquer les problèmes. Ils sont essentiels pour comprendre les activités des systèmes complexes, en particulier dans le cas d'applications avec peu d'interaction avec l'utilisateur, tels que les applications serveur.

Les logs de transactions générés par la plupart des systèmes de bases de données qui conservent un journal des transactions, ne sont pas destinés à être lisibles par l'homme. Ces logs enregistrent les modifications apportées aux données stockées pour permettre à la base de données de récupérer des pannes ou d'autres erreurs de données et de les maintenir stockées dans un état cohérent.

Les logs de message générés par des systèmes comme Internet Relay Chat (IRC), les programmes de messagerie instantanée (IM), et les jeux multi-joueurs, ont généralement la possibilité d'enregistrer automatiquement la communication textuelle, publique ou les messages de chat privé entre les utilisateurs, sous forme de journaux de messages. Les logs de messages sont souvent des fichiers de texte brut. Cependant, les clients de messagerie instantanée et VoIP (Voice over Internet Protocol) comme Skype, peuvent enregistrer les messages transités dans des fichiers HTML ou dans un format personnalisé pour faciliter la lecture ou activer le cryptage.

Les logs serveur sont des fichiers contenant des informations automatiquement créées par un serveur. Ces fichiers sont composés d'une liste d'activités que le serveur a effectuées et ne sont pas accessibles aux utilisateurs en général, mais uniquement aux administrateurs de système. Une analyse statistique du fichier de log du serveur peut être utilisée pour examiner le trafic par heure ou par jour de la semaine et ainsi prévenir de fortes activités ou identifier des fraudes. [21]

Utilisation

De nombreux systèmes d'exploitation, *frameworks* ou logiciels incluent un système de *logging*. Comme la norme Syslog qui permet à une partie du système de générer, filtrer, enregistrer et analyser les messages de log. Cela évite aux développeurs de logiciels de concevoir et créer leurs propres systèmes de log.[21]

En ce qui concerne les applications, pour bien gérer l'activité de log et créer des fichiers avec des informations exploitables, il est recommandé d'utiliser des API dédiés au *logging*. Ce type d'API est composé de trois éléments principaux :

- *Logger* : pour émettre un message généralement avec un niveau de gravité associé (les niveaux de gravité sont expliqués par la suite)
- *Formatter* : pour définir le format du message de log
- *Appender* : pour indiquer l'emplacement du log (console, fichier, base de données, email, ...)

Généralement la configuration peut être externalisée dans un fichier, ce qui rend l'utilisation de l'API plus souple et flexible.[32]

Les API de *logging* ont plusieurs inconvénients :

- Il est nécessaire de définir avec précision les messages à ajouter dans les fichiers de log
- Il est primordial de définir le niveau des messages
- L'utilisation d'une API de *logging* peut impacter de manière négative les performances d'une application

La génération des logs peut avoir un impact important sur les performances de l'application si les *loggers* sont trop présents dans les différentes fonctionnalités de l'application, sur le trafic réseau si les fichiers de log sont stockés sur un serveur distant et sur l'espace disque nécessaire à leur stockage si la quantité des logs est trop importante.

Il y'a un compromis à trouver entre la verbosité des logs et l'espace disque de stockage, car trop de logs noient les informations importantes et saturent le système, ce qui fait que leur durée de rétention soit alors plus faible, mais au contraire trop peu de logs peuvent emmener à une perte d'informations.

Les niveaux de logs permettent de pallier cette problématique. Une solution est de définir le niveau de log en fonction de l'environnement, par exemple: sur l'environnement de développement, le développeur est seul, donc les logs peuvent être plus verbeuses, alors qu'en production il peut y avoir des milliers/millions d'utilisateurs, ce qui implique une verbosité des logs réduite.[2]

Les niveaux de logs sont les suivantes :

Niveau	Usage	Exemples
TRACE	Utilisé pour le débogage en mode verbeux de l'application, souvent utilisé pour les environnements de développement.	Nouvelle valeur d'une variable Trace d'exécution d'une boucle Début et fin d'une méthode
DEBUG	Utilisé pour le débogage courant de l'application. Les informations loguées avec ce niveau ne sont pas utilisées dans un environnement de production.	Flux des données comme des requêtes SQL ou HTTP Données saisies invalides
INFO	Ces logs fournissent des informations sur le fonctionnement général de l'application et son utilisation. Ces traces à destination du métier. Les logs d'audit utilisent ce niveau de log. C'est le niveau qu'on peut trouver dans un environnement de production.	Exécution d'un batch Sauvegarde et fermeture de fichiers Connexion et déconnexion d'un utilisateur
WARN	Ces logs sont utilisés pour tracer des anomalies non bloquantes pour le système.	L'utilisation d'une méthode qui sera dépréciée avec la prochaine version.
ERROR	Ce niveau de log indique une erreur importante qui peut remettre en cause le fonctionnement de l'application.	Erreur JDBC liée à une contrainte d'intégrité Arrêt inattendu d'un batch Erreur de login

Figure 21 – Tableau des types de log [2]

3.1.3. Analyse des logs

Définition

L'analyse des logs fournit une mesure sur l'état des applications ou de l'infrastructure. Ces informations peuvent être utilisées pour améliorer les performances de l'application, résoudre des possibles problèmes ou détecter les failles de sécurité du système.

Objectif

L'analyse des logs est le processus d'examen et de compréhension des logs afin d'obtenir des informations pertinentes. Ces informations peuvent avoir différents objectifs, comme :

- Déterminer la popularité d'une page web en calculant les pages les plus visitées sur des sites web, ainsi que les horaires avec le plus de visites. Il est possible d'utiliser ces données pour mesurer la variation du public dans le temps.
- Détecter les événements suspects en repérant des événements inhabituels ou suspects. Par exemple, un utilisateur d'un service de paiement effectuant un achat d'une valeur bien supérieure à sa moyenne.
- Personnaliser l'expérience de l'utilisateur. Une autre manière d'utiliser les données des logs peut être pour effectuer une analyse commerciale sur le comportement d'un utilisateur, puis utiliser les résultats pour personnaliser son expérience sur l'application.
- Améliorer la sécurité grâce à une analyse complète des logs de sécurité permettant de détecter les tentatives d'invasion ou d'autres activités malveillantes.
- Améliorer le dépannage des problèmes grâce aux fichiers de log qui aident à localiser la cause de différents types d'erreurs.[15]

Utilisation

Souvent les règlementations des entreprises impliquent l'archivage et l'analyse des logs. Ces règlementations imposent de surveiller et analyser régulièrement les logs du système pour rechercher des erreurs, des anomalies, des activités suspectes ou non autorisées sur les différentes applications. L'analyse des logs permet de recréer la suite logique des événements qui ont produit le problème et de le résoudre efficacement. En exploitant l'analyse des logs, les problèmes sont détectés avant ou pendant qu'ils se produisent et permettent d'éviter les pertes de temps, les retards ou les coûts supplémentaires.[9]

Pour réaliser une analyse des logs efficace et obtenir des résultats pertinents, il existe trois étapes à respecter : le nettoyage, la structuration et l'analyse des informations.

Le nettoyage des données est un processus qui implique la détection et le remplacement ou la suppression d'informations inexactes, incomplètes ou non pertinentes.

La structuration des données permet d'homogénéiser les données provenant de différentes sources et qui comprend les étapes suivantes :

- Détection d'éléments communs et identification de *patterns* : pour pouvoir filtrer les messages. La détection et compréhension des modèles dans les données peuvent aider à détecter les anomalies.
- Normalisation : pour convertir les différents éléments du log, telles que les dates, au même format.
- Étiquetage et classification : pour étiqueter les éléments des logs avec des mots-clés et les classer dans des catégories.

L’analyse des informations est un processus complexe qui doit inclure les étapes suivantes :

- Analyse de corrélation : pour rassembler des informations provenant de sources et trier les messages significatifs qui se rapportent à un événement particulier. L'analyse de corrélation permet de découvrir des liens entre des données qui ne sont pas visibles dans un seul fichier de log.
- Ignorance artificielle : un processus d'apprentissage automatique pour identifier et filtrer les entrées d’informations qui ne sont pas utiles et détecter les anomalies. L'ignorance artificielle filtre les messages de routine tels que les mises à jour régulières du système, mais permet de détecter les messages nouveaux ou inhabituels et de les signaler. L'ignorance artificielle peut également identifier des événements de routine qui auraient dû se produire mais n’ont pas eu lieu.[33]

3.1.4. Méthodologie d'analyse des logs

L'outil et la réalisation de l'analyse des logs dépendent de la quantité de données. A présent, avec l'arrivée des réseaux sociaux, la quantité de données est augmentée de manière considérable et les outils d'analyse de log ont dû faire d'énormes progrès pour pouvoir les traiter. Pour des fichiers de petite taille, il est possible de réaliser une analyse en utilisant divers utilitaires de ligne de commande tels que GREP, ou des tableaux Excel, alors que pour les Big Data il est nécessaire d'utiliser des outils comme ceux de la Suite ELK.

En raison de l'arrivée des Big Data et des nombreux outils, la réalisation d'une analyse de logs peut être peu efficace si on ne respecte pas certaines étapes. Il n'existe pas une vraie méthodologie, mais des pas à suivre.

Figure 22 – Etapes de l'analyse des logs [34]

La première étape est la collecte des logs. Cette étape consiste à configurer un collecteur de logs pour rassembler tous les fichiers de log à exploiter.

La deuxième étape est la centralisation et indexation des informations. Dans cette étape les logs sont envoyés vers une plate-forme centralisée. Lorsqu'ils sont collectés à l'emplacement central, les logs sont également normalisés pour éviter toute confusion lors de la consultation et garantir l'uniformité. Avec l'indexation, les données sont facilement disponibles et consultables pour une analyse efficace des informations.

La troisième étape est la modélisation des données. Elle consiste à réaliser des analyses sur les données de manière à permettre d'identifier les informations correspondant à divers modèles et structures. Étant donné qu'il existe différents modèles d'enquête, il est impératif que l'analyse soit effectuée avec le modèle le plus approprié.

La dernière étape est la visualisation des données. Dans cette étape est affichée l'analyse des logs par le biais de rapports et de tableaux de bord. Cette étape rend les informations accessibles à tous, y compris aux personnes extérieures au service informatique. Elle permet de repérer plus facilement des tendances ou des anomalies en regardant des graphiques ou d'autres représentations visuelles des données.

Outils

Il existe plusieurs types d'outils d'analyse qui peuvent s'appliquer à une ou plusieurs étapes de cette méthodologie.

Pour la collecte de données il existe des projets comme Apache Flume, Fluentd avec FluentBit ou Logstash avec Filebeat. Certains sont plus performants que d'autres en termes de robustesse et tolérance aux pannes, comme Apache Flume, mais d'autres permettent aussi d'unifier et structurer les données comme Fluentd ou Logstash.

Pour la centralisation sont utilisés des outils comme Splunk ou Elasticsearch. Splunk en plus du stockage et de la collecte des données, se charge aussi de l'analyse de ces dernières.

Le plus connu reste Elasticsearch qui s'occupe non seulement de l'indexation et du stockage des données, mais intégré avec les autres agents comme Beats, Logstash et Kibana, il permet d'analyser et visualiser les données en temps réel.

Pour l'analyse des logs il existe des outils comme Loggly, Splunk, Fluentd. De plus des deux cités précédemment Splunk et Fluentd, Loggly aussi s'occupe de l'analyse et du monitoring des données. Lui en revanche est un fournisseur de services basé sur le Cloud, ou un SaaS (Software as a Service). A la différence de Fluentd qui est *open source*, Splunk et Loggly sont des outils payants.

Pour la visualisation sont utilisés des outils comme Grafana, Graylog et Kibana. Au contraire de ses concurrents qui sont des outils de visualisation et monitoring des logs, Grafana est un outil de visualisation des données par la réalisation des graphiques et des tableaux de bord. En ce qui concerne les outils dédiés aux logs, ils proposent en plus de la visualisation des données, le monitoring et la possibilité de créer des alertes. Ces options sont payantes pour la majorité d'entre eux.

La majorité des outils présentés peuvent être connectés entre eux pour créer le système le plus adapté à l'infrastructure et au besoin. Ils peuvent être utilisés de manière indépendante, ou intégrés avec d'autres applications spécifiques du système existant.

4. Analyse fonctionnelle

L'objectif de l'analyse fonctionnelle est de traduire le besoin du client, représenté par la matrice des exigences de haut niveau, sous la forme de fonctions intégrantes du système.

Les étapes principales qui doivent couvrir la solution envisagée sont représentées dans le diagramme ci-dessous sous la forme d'une boite blanche et elles correspondent au besoin client. Le processus commence par l'intégration des données en provenance des fichiers de log. Afin que les données soient analysées, elles doivent être stockées. Une fois que les données sont analysées, des rapports sont créés et stockés. Le système prévoit aussi la possibilité de refaire une analyse des données. Les rapports sont visibles par l'utilisateur et peuvent être exportés.

Ce diagramme permet de visualiser comment les éléments sont reliés entre eux et quelles sont les données échangées entre les différentes parties du système. [8]

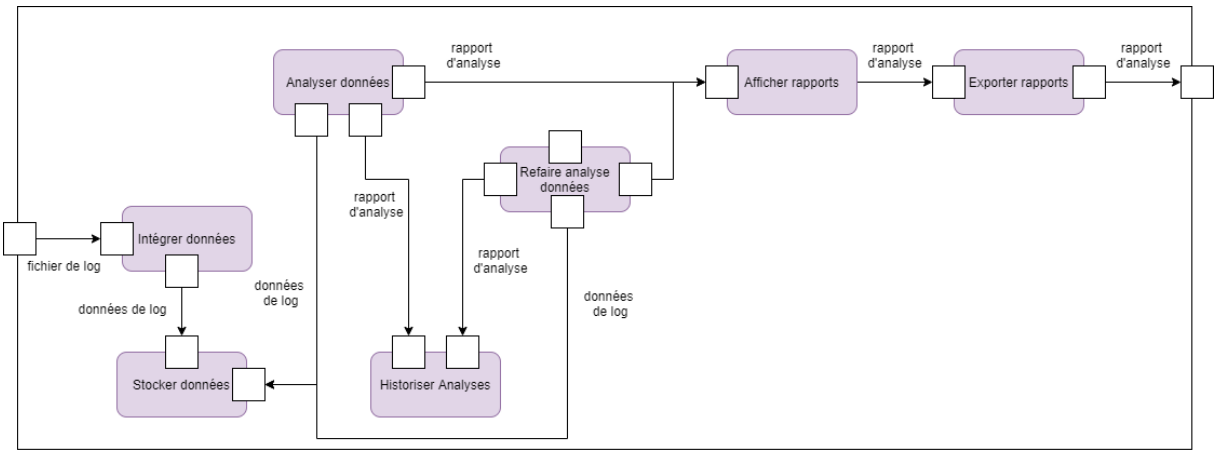

Figure 23 – Vue boite blanche du système

4.1. Analyse des fonctions

Pour procéder à l'analyse des logs, il est nécessaire de prévoir 4 étapes principales, que sont : la collecte des logs, la centralisation et indexation des logs, la modélisation des données et enfin la visualisation des résultats. Pour mettre en pratique cette méthode il est nécessaire de l'adapter de manière qu'elle respecte les fonctions représentées précédemment.

Avec le diagramme de composants qui suit, il est possible de visualiser les composants déduits et comment ils communiquent entre eux.

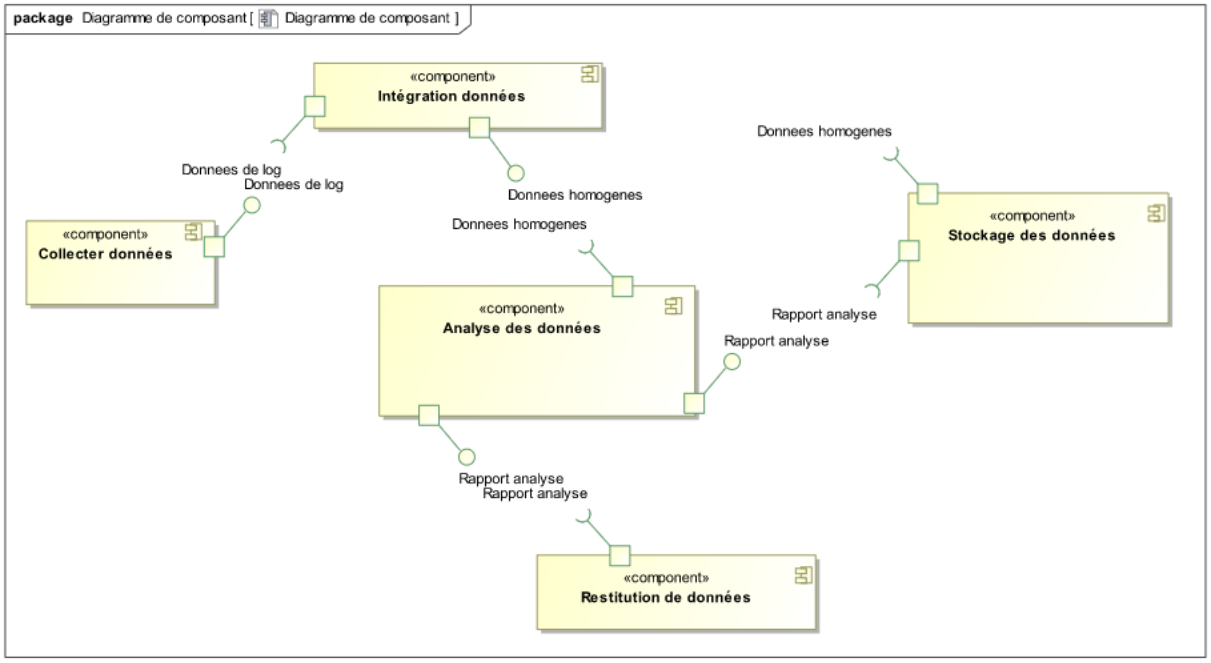

Figure 24 – Diagramme de composants

Dans notre cas, la première étape de collecte de données sera effectuée par le composant « Collecter données », la deuxième étape de centralisation et indexation des données sera couverte par les composants « Intégration données » et « Stockage données », l'étape de modélisation sera effectuée par le composant « Analyse des données » et la dernière étape de visualisation des données sera réalisée par le composant « Restitution de données ».

A partir du diagramme de composant, il est possible d'en déduire également les interfaces des différents composants. Les interfaces permettent de visualiser la relation contextuelle entre les différents éléments, et les informations que les composants attendent et reçoivent. Dans notre système, ce qui circule sont les données des fichiers de log et les rapports réalisés.

4.2. Architecture globale du système

Les composants cités précédemment doivent s'intégrer à la structure physique de l'entreprise. Pour avoir une vue globale sur le système, le diagramme de déploiement qui suit, illustre les différents composants physiques dont il y a besoin pour intégrer le système, et également les flux qui circulent entre eux.

Selon l'architecture de la solution proposée, le système est physiquement installé sur un serveur distant et accessible par le réseau de l'entreprise. Ces composants communiquent entre eux comme dans la figure ci-dessous.
Pour utiliser le système, en tant que développeur, une instance du composant « Restitution de données » est accessible via un navigateur. Le développeur y accède depuis son poste, comme illustré dans le diagramme ci-dessous. En réalité, le composant « Restitution de données » est accessible depuis tout type de navigateur, tant que l'utilisateur est connecté au réseau interne de l'entreprise.

La collecte de données se fait grâce au composant « Collecte de données » qui est installé sur le même emplacement que l'application du projet TMA Rationnalisée. Ce dernier peut être le poste du développeur ou un autre serveur distant. Cette configuration permet au système de collecter les données de toutes les applications disponibles sur le projet. La configuration de cette architecture ainsi que les choix technologiques pour la réaliser est détaillée dans le chapitre suivant.

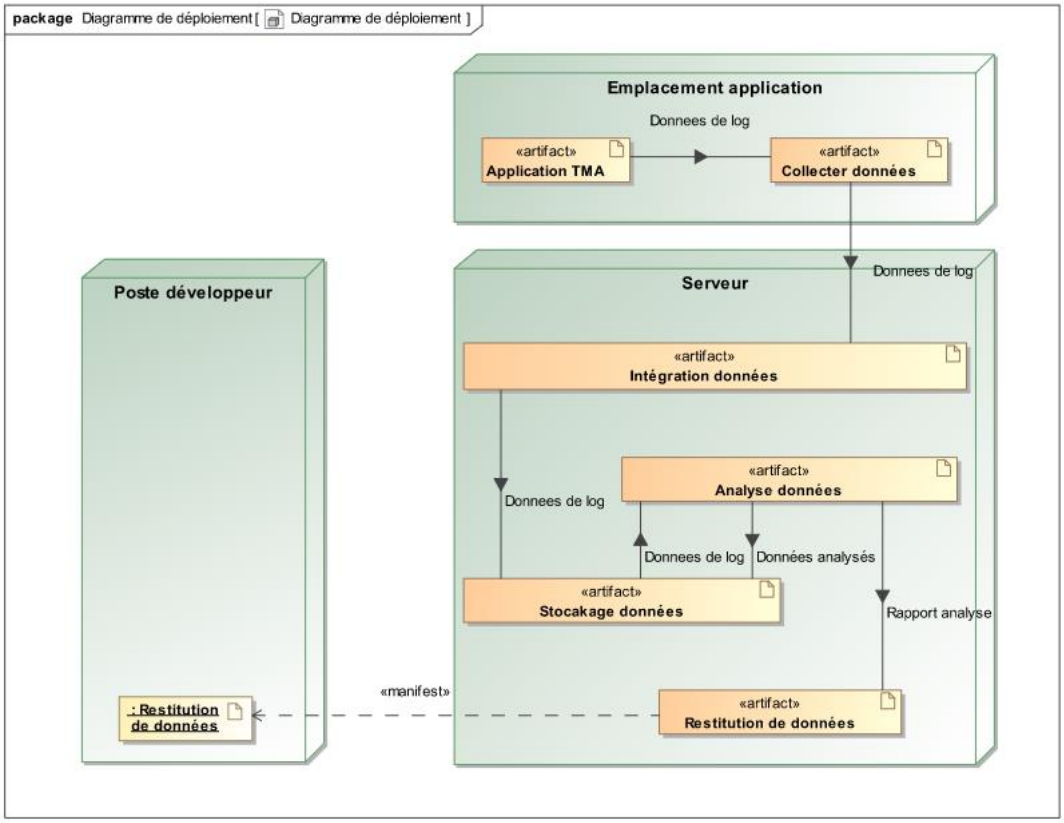

Figure 25 – Diagramme de déploiement

5. Analyse applicative

L'analyse applicative définit l'organisation logique des composants applicatifs permettant la réalisation de toutes les fonctions du système via les composants applicatifs. Dans ce chapitre est traité le cas d'utilisation principale du système avec les différentes fonctions utilisées pour chaque composant, ainsi que l'architecture globale du système.

5.1. Diagramme d'activité

Ce diagramme permet de visualiser les fonctions, le flux de données et les conditions de déclanchement de ces fonctions. Grâce à ce diagramme, il est possible de visualiser les transformations des entrées en sorties à travers des séquences contrôlées d'actions. Dans ce cas il illustre le cas nominal de « Analyser résultats ». Ce diagramme comporte trois composants et les flux de données correspondants.

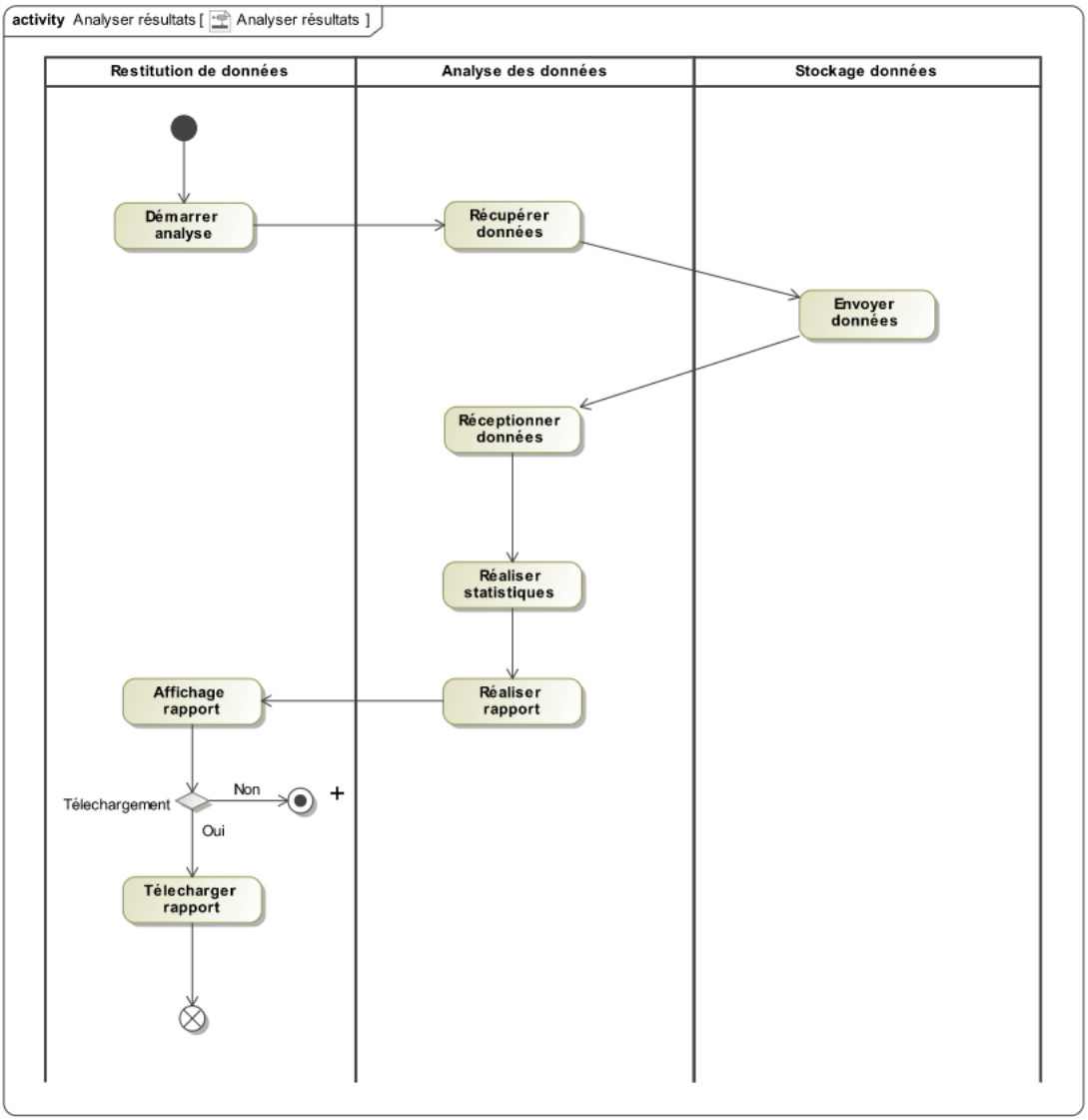

Figure 26 – Diagramme d'activité « Analyser résultats »

5.2. Diagramme de séquence

A travers le diagramme de séquence, il est possible d'exprimer de manière plus dynamique le comportement du système. Le diagramme ci-dessous illustre les étapes et les composants nécessaires à la réalisation du même cas d'utilisation que précédemment.

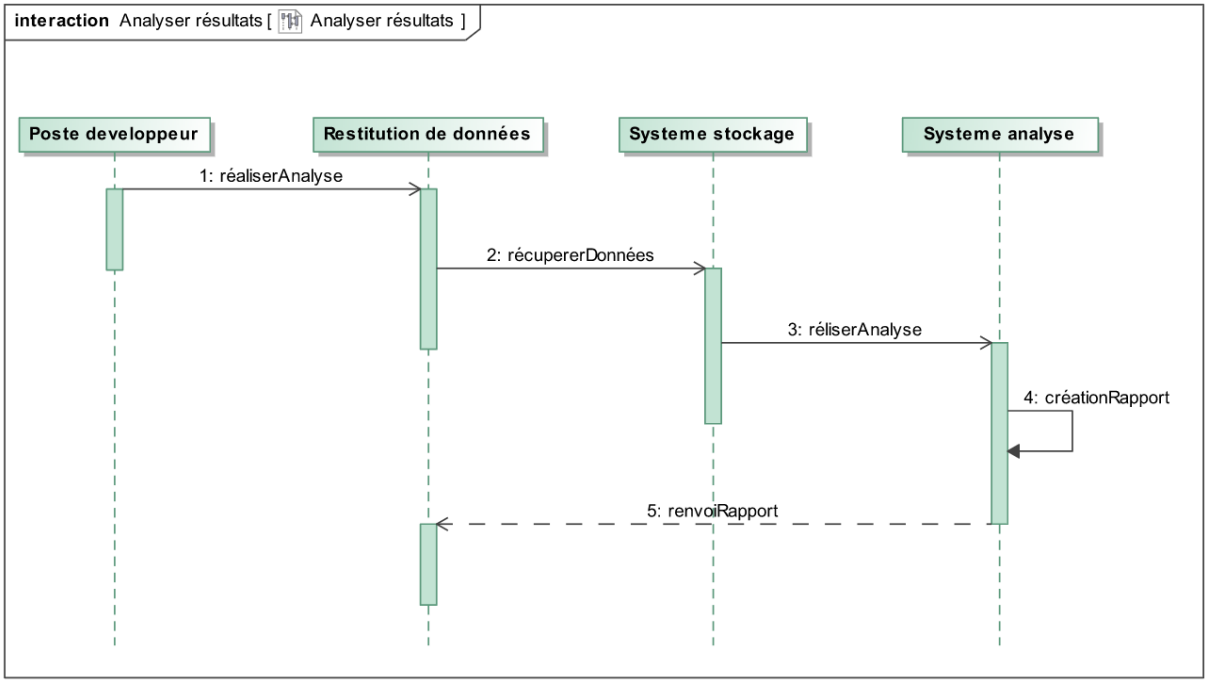

Figure 27 – Diagramme de séquence « Analyser résultats »

5.3. Architecture logicielle

L'architecture globale permet de représenter comment les différents composants sont organisés et comment ils communiquent entre eux. Dans notre cas, le composant de collecte de données envoie les différents fichiers via le protocole de communication TCP au système d'intégration de données. Le système intègre les fichiers de log reçus. Les données sont envoyées au composant de stockage via le protocole http, et lui permet d'alimenter le composant « Analyse de données » avec les informations nécessaires pour la création des rapports. Les rapports sont stockés, eux aussi, dans le composant « Stockage données ». Une instance du composant « Restitution données » est visible par l'utilisateur. Ce composant permet d'afficher ou télécharger les rapports réalisés par l'utilisateur.

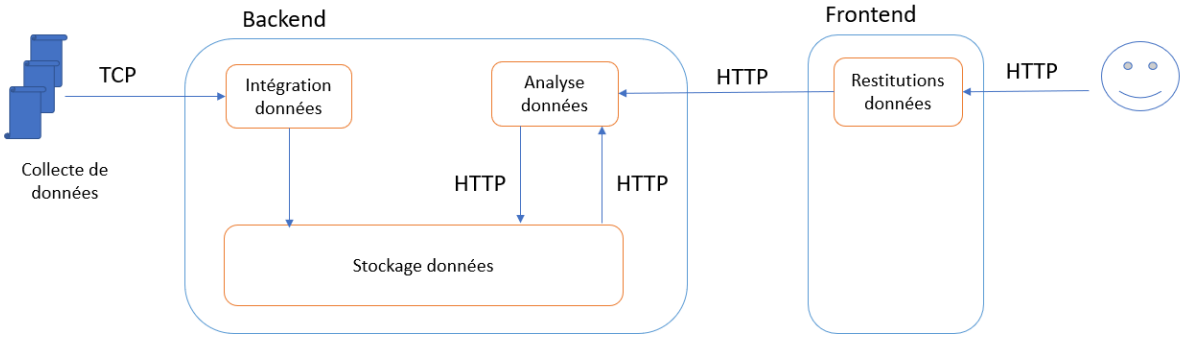

Figure 28 – Architecture globale du système

Les différents composants et les choix technologiques pour mettre en place cette architecture sont détaillés dans le chapitre suivant.

6. Mise en œuvre de la solution

Dans ce chapitre sont présentés les technologies choisies pour réaliser l'architecture du système identifié dans les étapes précédentes, ainsi que la mise en œuvre de chaque composant.

6.1. Analyse technologique

L'analyse technologique consiste à réaliser une étude de différents logiciels, langages de programmation, outils disponibles sur le marché et choisir ceux qui sont plus adaptés à la création du système.

6.2. Découpage en composants

À la vue de l'architecture globale du système présenté dans le chapitre précédent et de la méthode Kanban choisie pour la réalisation du projet, l'analyse technologique est réalisée pour chaque composant.

6.2.1. Composant pour la collecte des données

Le composant de collecte de données doit se charger de rassembler tous les fichiers de log et les transmettre au composant d'intégration. Ce composant permet au système d'obtenir les fichiers de log de toutes les applications du projet TMA Rationnalisée, indépendamment de l'emplacement de l'application.
Dans le chapitre dédié à l'état de l'art, plusieurs types d'outils de collecte ont été identifiés. Ces outils sont détaillés et comparés afin de trouver l'outil le plus adapté au besoin.

Apache Flume

Apache Flume est un outil *open source* pour collecter l'agrégation et le transport de grandes quantités de données en continu. Ces données peuvent être des fichiers de log en provenance de diverses sources. Flume est principalement conçu pour copier des données de log en continu de diverses sources vers un système de centralisation de logs. Il est scalable horizontalement, donc il est possible d'ajouter de nouveaux nœuds selon les besoins.
Apache Flume sécurise les transactions des flux de données et garantit qu'aucune donnée n'est perdue lors du processus de transmission. Il est personnalisable et prend en charge diverses sources et récepteurs comme Kafka.
Flume fournit le flux régulier de transmission de données, en adaptant la vitesse d'écriture des données si la vitesse de lecture augmente ou diminue.[35]

Filebeat

Filebeat est l'agent de transfert de données *open source* de la Suite ELK, qui permet de centraliser les logs et les fichiers. Filebeat est un outil fiable, car il permet de reprendre le transfert des données lors d'une interruption de la connexion, sans perte de données.
Lorsqu'il envoie de grands volumes de données vers un outil de centralisation, comme Elasticsearch, il utilise un protocole de régulation de flux. Pendant le traitement des données par un outil comme Logstash, Filebeat ralentit la lecture en conséquence et reprend son rythme initial quand les traitements sont effectués.[36]

FluentBit

Fluent Bit est un processeur de log et un expéditeur *open source* qui permet de collecter des données telles que les logs de différentes sources, de les enrichir avec des filtres et de les envoyer à plusieurs destinations.
FluentBit est conçu avec l'objectif d'être performant : haut débit avec une faible utilisation du processeur et de la mémoire. Il est écrit en langage C et possède une architecture prenant en charge plus de 70 extensions pour les entrées, les filtres et les sorties.
Fluent Bit est un sous-projet de la CNCF (Cloud Native Computing Foundation) sous la tutelle de Fluentd.[7]

Les trois outils présentés sont comparés en fonction de plusieurs critères. Les trois sont *open source*, mais ils sont employés dans différents cas.
Comme spécifié précédemment, Apache Flume est utilisé lorsque le transfert emploie de nombreuses données de log, comme ceux en provenance de Facebook ou Twitter et il se charge de transférer les données en continu. Le système de centralisation le plus adapté à Apache Flume est Hadoop. Apache Hadoop est un *framework* créé pour réaliser le traitement de grands volumes de données, de manière distribuée sur des clusters de serveurs. [37] Le noyau d'Apache Hadoop se compose d'une partie de stockage appelée Hadoop Distributed File System. HDFS est souvent l'agent de centralisation d'Apache Flume. Etant donné le besoin de notre système, qui n'est pas orienté Big Data, nous allons prendre en considération les deux autres outils.
Filebeat comme FluentBit sont des outils légers, c'est-à-dire qu'ils consomment peu de ressources. Ce sont des outils presque équivalents, donc le choix est fait en fonction de la simplicité d'installation des deux. Ce dernier critère est essentiel, car ce composant doit être installé sur les différents serveurs ou postes développeur. L'outil choisi est donc Filebeat.

Mise en place de la solution

Filebeat a été choisi pour la simplicité de mise en place sur les environnements Windows, afin de faciliter l'installation, notamment sur les environnements des développeurs.
La configuration de Filebeat prévoie la modification du chemin du dossier qui va recevoir les fichiers de log et l'*output* qui est l'adresse du composant d'intégration de données.

6.2.2. Composant d'intégration des données

Le composant d'intégration des données, comme son nom l'indique se charge d'intégrer les informations provenant des fichiers de log des applications du projet TMA Rationnalisée. Son objectif est d'identifier les informations nécessaires à l'analyse des données à partir de tous types de fichier de log. Même si l'ordre ou le format des données n'est pas le même, les informations doivent être identifiées.

Pour réaliser une meilleure analyse des technologies disponibles, une analyse des fichiers générés par les applications est nécessaire.

Description fichier de log application TMA

Cette étape de description des logs est importante dans la création du système, pour comprendre quels sont les points importants à respecter et effectuer les choix technologiques en fonction de certains critères.

Le seul élément fourni par les applications existantes est le fichier de log. Comme vu dans le Chapitre 2, les fichiers de log sont générés automatiquement par les applications.
L'extension de ces fichiers est « .log » ou « .txt » et le contenu est généralement exprimé sous la forme de fichiers texte ASCII.

Si un fichier de log est en cours d'utilisation par le système d'exploitation ou par un programme, il ne peut pas être ouvert, sauf si le programme avec lequel l'utilisateur essaie d'ouvrir le fichier autorise le mode en lecture seule. De même, ces fichiers ne peuvent pas être supprimés ou déplacés vers un autre répertoire lorsqu'ils sont en cours d'utilisation. Dans la plupart des applications de la TMA Rationnalisée, les programmes autorisent l'ouverture en lecture du fichier de log et même son déplacement.

Dans la plupart des cas sur les applications du projet TMA Rationnalisée, les fichiers de log sont nommés comme le *jar/war* qui les a générés, sinon ils portent le nom du composant respectif. Dans les deux cas les fichiers sont suffixés d'un *timestamp*. Cela permet de faciliter la recherche des logs. Chaque application à un dossier configurable pour stocker les fichiers de log, qui est toujours le même, indépendamment de l'emplacement de déploiement de l'application.

Plusieurs types de fichiers de log, peuvent exister sur les différentes applications, comme :

- Log de *debug* ou de traçabilité contenant les informations d'exécution de l'application sur les environnements hors production
- Log de supervision de certains composants, ne contenant que les informations de démarrage, arrêt ou erreur de ce composant.
- Log d'erreur, ne contenant que les logs avec un niveau « ERROR »
- Log de performance, ne contenant que des informations sur les temps d'exécution de certains composants.

Les types de log et les types de fichiers souhaités sont spécifiques à chaque application et ils sont configurés en fonction du besoin et de la conception de l'application. Si, par exemple l'application est relativement simple, formée d'un seul composant, alors un seul fichier de log est suffisant pour avoir les informations nécessaires. Cependant, si une application contient plusieurs composants, alors différents fichiers de log peuvent être nécessaires.

L'essentiel est d'avoir un fichier de log pour chaque composant et, si besoin, des fichiers plus spécifiques, comme le fichier de log d'erreur. Ce découpage permet d'avoir des logs contenant des informations pertinentes et facilement repérables.

Parmi les informations présentes dans un fichier de log il est possible de trouver :

- Le *timestamp* avec la date, l'heure, les minutes et secondes d'exécution
- Le niveau de log (TRACE, DEBUG, INFO, WARN, ERROR)
- Le nom du composant

```
INFO  - 25/05/2021 10:41:10 - [Service de relève] - com.cedicam.rr.
INFO  - 25/05/2021 10:41:10 - [Service de relève] - com.cedicam.rr.
ERROR - 25/05/2021 10:41:10 - [Service de relève] - com.cedicam.rr.
INFO  - 25/05/2021 10:41:10 - [Service de relève] - com.cedicam.rr.
INFO  - 25/05/2021 10:41:10 - [Service de relève] - com.cedicam.rr.
INFO  - 25/05/2021 10:41:10 - [http-apr-8080-exec-8] - com.cedicam.
INFO  - 25/05/2021 10:41:10 - [http-apr-8080-exec-8] - com.cedicam.
```

Figure 29 – Fichier de log contenant le niveau de log

- Message de log spécifique paramétré auparavant

```
La recherche de fichier dans le dossier \SICB\AID démarre
La recherche de fichier dans le dossier \SICB\AID est achevée
La recherche de fichier dans le dossier \SICB\DOL démarre
```

Figure 30 – Fichier de log contenant des messages spécifiques

- La *stack trace* en cas d'erreur

```
ERROR - 25/05/2021 10:59:52 - [localhost-startStop-2] - com.cedicam.rr.server.common.BootstrapServlet -
com.cedicam.rr.ppq.bootstrap.services.NotRunningException: Le systÃ¨me de lancement des générations PPQ
    at com.cedicam.rr.ppq.bootstrap.services.BootstrapService.stop(BootstrapService.java:130)
    at com.cedicam.rr.server.common.BootstrapServlet.destroy(BootstrapServlet.java:145)
    at org.apache.catalina.core.StandardWrapper.unload(StandardWrapper.java:1486)
    at org.apache.catalina.core.StandardWrapper.stopInternal(StandardWrapper.java:1847)
    at org.apache.catalina.util.LifecycleBase.stop(LifecycleBase.java:232)
    at org.apache.catalina.core.StandardContext.stopInternal(StandardContext.java:5675)
```

Figure 31 – Fichier de log contenant avec stack trace

- Les noms des composants pour les tests, les temps d'exécution et le résultat

```
[INFO] Running gca.caps.mt.common.model.OperationTest
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.002 s
[INFO] Running gca.caps.mt.common.model.RapportTest
[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.059 s
[INFO] Running gca.caps.mt.common.model.BanqueTest
[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.022 s
[INFO] Running gca.caps.mt.common.model.CommissionTest
[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.001 s
```

Figure 32 – Fichier de log contenant l'exécution de tests

Les informations décrites précédemment sont des exemples de ce qu'il est possible de trouver dans la majorité des fichiers des applications de la TMA Rationnalisée. Ce sera la base pour déterminer les informations qui seront intégrées dans le système et quelles seront les informations utiles pour la réalisation des rapports de performance.

Technologies disponibles

Dans la partie dédiée à l'analyse des logs, ont été identifiés plusieurs types d'outils existants qui ont été créés pour intégrer les données provenant des fichiers de log.

Fluentd

Fluentd est un collecteur de logs *open source* qui vise à unifier la collecte de logs de nombreuses sources de données et de nombreux systèmes, et à les rassembler dans une couche de journalisation unifiée.
Fluentd écrit en C et Ruby est un *hub* qui s'occupe du traitement d'un ensemble de flux. De plus, c'est un outil qui fournit les spécifications de son protocole : *Fluent forward protocol*. [7]
Fluentd peut être déployé sur les systèmes Linux et Windows et il repose sur un système de *plug-ins* décentralisé.

L'une des fonctionnalités clés des collecteurs de log est le routage des événements. Fluentd s'appuie sur des balises pour acheminer les évènements. Chaque évènement a une balise qui indique à Fluentd où il veut être acheminé. Cette méthode permet d'exprimer avec plus de facilité des configurations de routage complexes. [13]

Logstash

Logstash est un moteur *open source* de collecte et de traitement des données via *plug-in*. Il est doté de nombreux *plug-ins* qui permettent d'adapter sa configuration à l'architecture souhaitée, pour collecter, traiter et transférer les données vers un grand nombre d'outils. Logstash est compatible avec les plateformes Linux et Windows.[14]
En termes de routage des évènements, Logstash achemine toutes les données dans un seul flux, puis utilise des instructions algorithmiques « if-then » pour les envoyer à la bonne destination.
Le traitement peut être effectué par un ou plusieurs *pipelines*. Les différentes données collectées par chaque *pipelines* sont alors placées dans une file d'attente interne. La configuration par défaut de la file d'attente est telle qu'elle puisse être enregistrée en mémoire, mais il est possible de changer sa configuration pour l'agrandir et l'enregistrer sur le disque de manière permanente afin d'améliorer la fiabilité.
L'avantage de l'approche de Logstash est la simplicité : le modèle de la file d'attente dimensionnée est très simple. Logstash gère tous ses *plug-ins* sous un seul référentiel GitHub. Alors que l'utilisateur peut écrire et utiliser les siens. [13]

Logstash	**Fluentd**
Open Source	Open Source
Sur JVM	Support natif intégré sur Docker
Plug-ins actifs centralisés	Plug-ins actifs décentralisés
Achemine les évènements en fonction des conditions if-else	Achemine les évènements en utilisant l'approche tag

Figure 33 – Comparatif entre Logstash et Fluentd

Après avoir comparé les deux outils, le choix est fait en fonction du besoin client, mais aussi des compétences déjà présentes au sein de l'équipe, pour assurer la maintenabilité du système. À la vue du fait que Logstash est déjà utilisé au sein du projet TMA Rationnalisée, et que l'environnement d'utilisation est Windows avec des JVM déjà installés sur la majorité de nos environnements, le choix est Logstash. De plus, l'outil de transfert de fichiers choisi est Filebeat qui s'intègre de manière simple avec la suite ELK.

Mise en place de la solution

Le composant d'intégration de données a comme objectif principal d'homogénéiser les données qui arrivent avec un format différent. Les fichiers de log, comme spécifié précédemment, contiennent toutes les informations indispensables, comme le *timestamp* ou le nom de l'application, mais souvent ces informations ne sont pas dans le même ordre ou dans le même format, comme il est possible d'observer dans la figure ci-dessous.

```
INFO  - 25/05/2021 10:37:37 -        2021-08-16 17:41:59.885  INFO
INFO  - 25/05/2021 10:37:37 -        2021-08-16 17:41:59.895  INFO
INFO  - 25/05/2021 10:41:00 -        [INFO] ----------------------
```

Figure 34 – Comparatif des informations des fichiers de log

Il existe plusieurs types de filtres pour Logstash. Ci-dessous sont présents les filtres adaptés aux données non structurées. Cependant plusieurs types de filtres sont requis pour obtenir toutes les informations nécessaires au bon fonctionnement de l'application.

Les filtres pour les données non structurées sont « dissect » qui permet d'extraire des données à l'aide de délimitateurs et « grok » qui permet d'extraire des données en fonction d'une certaine analyse des données. « Dissect » diffère de « grok » car il n'utilise pas d'expressions régulières et est plus rapide. « Dissect » fonctionne bien lorsque les données sont répétées de manière fiable. « Grok » est un meilleur choix lorsque la structure du texte varie d'une ligne à l'autre.

Il est possible d'utiliser à la fois « dissect » et « grok » pour un cas d'utilisation hybride lorsqu'une section de la ligne est répétée de manière fiable, mais pas la ligne entière. Le filtre « dissect » peut déconstruire la section de la ligne qui se répète. Le filtre « grok » peut traiter les valeurs de champ restantes avec des champs non réguliers.[38]
Ces filtres sont disponibles sur Logstash sous la forme d'un *plug-in*, à installer en fonction du besoin. Dans ce cas spécifique, je décris le filtre « grok », pour les données qui n'ont pas de format prédéfinit.

Les filtres « grok » se basent sur les *regex*, donc la présence des champs multiples n'est pas un problème, cependant le fait d'avoir des ordres différents de certains champs peut l'être. Pour pallier ce problème il existe plusieurs solutions :

- La première est de créer plusieurs filtres « grok » pour chaque *pattern* de ligne
- Le deuxième est de mettre des caractères ou des champs optionnels

Dans notre cas, il est nécessaire d'utiliser les deux. En plus de cette souplesse, les filtres « grok » permettent de ne pas prendre en compte les lignes qui ne correspondent pas aux *patterns* établis. Ce qui permet de parcourir le fichier de log et ne classer que les lignes qui sont intéressantes. Dans ce cas, seules les informations concernant l'exécution des tests ont besoin d'être classées. Par exemple, pour filtrer les logs qui ont cette forme :

[INFO] Tests run : 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0 s - in gca.mt.common.dto.RapportDtoTest

Le filtre adapté est :

%{LOGLEVEL:loglevel}] %{WORD} %{WORD}: %{INT:nb_tests}, %{WORD}: %{INT:nb_failures}, %{WORD}: %{INT:nb_errors}, %{WORD}: %{INT:nb_skipped}, %{WORD} %{WORD}: %{INT:time_elapsed} %{WORD:seconds} - %{GREEDYDATA:methode}

Les informations seront donc stockées avec des *tags* comme *nb_tests, nb_failures, nb_errors, nb_skipped, time_elapsed* ou *methode,* qui vont permettre ensuite de réaliser des statistiques sur la méthode testée.

6.2.3. Composant pour stockage des données

En ce qui concerne le stockage des données, plusieurs types de stockage sont possibles. Le choix est orienté vers un outil de type NoSQL car, les données de type texte et les logs sont difficiles à stocker dans des bases de données relationnelles. Plusieurs outils de stockage, spécialisés pour les logs, sont décrits ci-dessous.

Splunk

Splunk est un outil qui se charge de la collecte des données et de leur indexation. Il est conçu de manière qu'il mette en corrélation les données en temps réel et qu'il les enregistre dans des archives recherchables, permettant de générer des graphiques, des rapports, des *dashboards* ou créer des alertes. Splunk est un outil payant sous licence de la multinationale du même nom. L'outil base de Splunk collecte et analyse d'importants volumes de données générées par des systèmes. Il utilise une API standard permettant d'accéder directement à ses services. De plus, différents types de produits sont proposés, en fonction du besoin, comme : Splunk User Behavior Analytics qui permet de détecter les menaces et les comportements anormaux avec le ML ou Splunk Application Monitoring qui permet de faire du monitoring afin d'améliorer la performance et la fiabilité des applications. [22]

Elasticsearch

Elasticsearch est un outil de stockage qui se base sur l'indexation de toutes les données, pour permettre ensuite de les retrouver plus facilement. C'est une base de données NoSQL, orientée Big Data, qui peut stocker et gérer un volume de données important. Il est possible de communiquer avec Elasticsearch via une API REST pour ajouter ou récupérer des données.
C'est un logiciel écrit en Java distribué sous licence Elastic, Open core, plus précisément il offre des fonctionnalités en *open source*, mais la version payante permet d'accéder à plusieurs *plug-ins*, donc plusieurs fonctionnalités.
Elasticsearch permet de stocker des données provenant de plusieurs types d'outils de centralisation de données ou de restitution de données comme Graylog, mais le plus souvent il est utilisé avec Logstash et Kibana, ce qui forme la Suite ELK.

Elasticsearch vs Splunk

Elasticsearch fait partie d'un ensemble de produits *open source* fournis. Il fait bien plus que du stockage de données, il est dédié à rendre la recherche de données non structurées plus efficace. Elasticsearch fournit également des intégrations de haute qualité dans tous types d'environnements Cloud, de sorte que la configuration d'un *cluster* de recherche sur AWS ou Azure soit simple.

Splunk est presque l'opposé d'Elasticsearch. Splunk fait partie d'une suite de produits fabriqués par la société du même nom qui se concentre sur le monitoring des informations de log. Au lieu de fournir une plate-forme *open source* que les équipes personnalisent en fonction de leurs besoins, Splunk fournit un logiciel très ciblé conçu pour fonctionner spécifiquement sur les projets qui génèrent de nombreuses données de log. Il s'intègre à la fois aux piles logicielles Cloud et sur site, ce qui le rend facile à brancher et à utiliser.

En plus des avantages financiers mentionnés ci-dessus, Elasticsearch est hautement configurable. Ce qui permet de l'adapter à presque n'importe quel environnement. Cependant la configuration d'Elasticsearch est un point important, car elle demande plus de compétences au moment de l'installation.

Le plus grand avantage de Splunk est qu'il fournit un moyen rapide d'avoir la visibilité sur les logs avec un travail minimal. Il suffit de diriger la sortie des logs vers le serveur de traitement de Splunk et se connecter à l'interface web. Cet outil, en revanche, n'est pas adapté à tous types de système, notamment à ceux qui ne génèrent pas une grande quantité de log. [17]

En considérant ces affirmations, le besoin de notre système n'est pas orienté Big Data, donc le choix pour le composant de stockage est Elasticsearch.

Mise en place de la solution

Pour mettre en place le composant de stockage de données, j'installe la version gratuite de Elasticsearch. Elle nécessite la présence de Java 8 sur le poste d'installation. Comme Filebeat et Logstash sont déjà installés, il suffit de modifier la configuration de Logstash afin qu'il transmette les données vers Elasticsearch, et inversement, configurer Elasticsearch afin de recevoir les données de Logstash.

Elasticsearch est un outil créé pour être distribuable, c'est-à-dire qui peut être reparti sur plusieurs serveurs ; une instance d'exécution de Elasticsearch est appelée nœud. Un ou plusieurs nœuds Elasticsearch peuvent s'exécuter sur une machine ou un serveur virtuel en fonction de ses ressources. Les nœuds peuvent également s'exécuter sur un *cluster* de machines lorsque l'architecture du système est prévue pour la haute disponibilité. Un *cluster* Elasticsearch peut être composé d'un ou plusieurs nœuds.

L'organisation des données stockées se base sur une indexation de tous les mots d'un document. Un document est l'équivalent des tables dans les bases de données relationnelles, mais il est structuré comme un objet JSON et chaque document possède des propriétés.

Les données stockées par Elasticsearch sont sous format JSON et il est possible d'y accéder via des requêtes http en utilisant le Query DSL. Ce dernier est composé de deux types de clauses :

- Requête feuilletée

  Elles recherchent une valeur particulière dans un champ particulier, comme les requêtes « match », « term » ou « range ». Ces requêtes peuvent être utilisées seules.

- Requête composée

  Elles peuvent contenir d'autres requêtes feuilles ou composées et sont utilisées pour combiner plusieurs requêtes de manière logique ou pour modifier leur comportement.

Elasticsearch en tant que moteur de recherche, utilise Apache Lucene pour les fonctionnalités d'indexation et de recherche de contenu sur ces documents JSON. La recherche se base sur ce qu'on appelle « Term Frequency » qui représente la présence d'un mot dans un document et la « Inverse Difference Frequency » moins un mot est commun dans le document plus il a de poids dans la recherche. Ces deux indicateurs permettent de calculer le *score* qui est un élément permettant à Lucene d'effectuer des recherches rapides.

Avec ces considérations, il faut donc paramétrer Logstash de manière qu'il transmette les données dans Elasticsearch avec une organisation préétablie. Cette organisation consiste à créer un index avec les paramètres nécessaires pour pouvoir classer les données en fonction de la méthode de test et des temps d'exécution de ces derniers.

Pour créer un index sur Elasticsearch, il faut créer une structure JSON avec les paramètres et les types nécessaires. Un exemple de cette structure est la suivante :

```
PUT /application
{
  "mappings": {
      "properties": {
        "@timestamp": {
          "type": "date"
        },
        "@version": {
          "type": "integer"
        },
        "application": {
          "type": "text"
        },
        "methode": {
          "type": "text"
        },
        "nb_tests": {
          "type": "integer"
        },
        "nb_failures": {
          "type": "integer"
        },
        "nb_errors": {
          "type": "integer"
        },
        "nb_skipped": {
          "type": "integer"
        }
        "time_elapsed": {
          "type": "float"
        }
      }
  }
}
```

Figure 35 – Exemple de création de index dans Elasticsearch

Dans ce cas l'index créé se nomme « application » et les propriétés sont *timestamp*, *version*, *methode*, *nb_tests*, *nb_failures*, *nb_errors*, *nb_skipped*, *time_elapsed*.
Ce même index est indiqué dans le fichier de configuration de Logstash, afin qu'il puisse utiliser cette syntaxe pour envoyer les données vers Elasticsearch.

Pour effectuer les opérations de création d'index dans Elasticsearch et pour visualiser les données, il est possible d'utiliser un outil comme Postman, qui permet d'envoyer des requêtes http vers un service.

Maintenant que les données sont stockées avec des noms de paramètres facilement identifiables, il est possible de passer à la phase d'analyse.

6.2.4. Composant pour l'analyse des données

Ce composant se charge de réaliser la comparaison entre deux réalisations du même test et il indique s'il y a des différences dans le temps de réponse de ce dernier. Cette comparaison, permet de réaliser un rapport qui indique si les performances de l'application sont dégradées ou améliorées. De plus, le composant se charge aussi de réaliser des statistiques des mêmes tests. Plusieurs outils qui réalisent des analyses de données sont présentés ci-dessous.

Loggly

Loggly est un outil orienté log management qui permet d'héberger et gérer les log multi-sources. Loggly est un outil édité par SonarWinds, payant qui propose plusieurs fonctionnalités. Il permet d'avoir une vue d'ensemble en temps réel de l'activité des applications dans les environnements distribués. Il utilise des protocoles ouverts, et non des agents propriétaires, pour transmettre les logs.

Étant un service payant, il est 100 % évolutif dans le Cloud, ce qui a comme avantage le fait qu'une équipe peut intervenir en permanence sur le service afin d'accélérer le dépannage ou de trouver rapidement les goulots d'étranglement et les points de blocage. En plus il peut être configuré pour traiter les logs des applications dans le Cloud. [39]
Loggly est compatible avec Linux, Windows et Mac et il se base sur une API pour la gestion des logs. Il propose également une IHM pour que l'utilisateur puisse de monitorer et gérer les logs.

Les autres outils identifiés ont été décrits précédemment. Mais malgré cette recherche, aucun de ces outils n'est compatible avec le besoin exprimé par le client. La majorité des outils permet de faire du monitoring, ou de créer des *dashboards* avec les statistiques souhaitées, mais sur une ou plusieurs données.
Pour que l'outil corresponde au besoin, il doit réaliser la comparaison entre deux données de même type, ce qui n'est pas le cas pour les outils *open source* présentés.

Donc la meilleure solution est de créer une API qui puisse récupérer les données d'Elasticsearch, réaliser les calculs sur les données, et générer les rapports.

Mise en place de la solution

Pour réaliser cette API, Spring Boot est le langage le plus adapté pour l'environnement actuel du projet TMA Rationnalisée. Cette technologie est déjà utilisée sur plusieurs applications, donc il est plus pertinent de créer une API avec la même technologie, pour des raisons de maintenabilité de cette dernière.

Cette API a trois fonctions principales qui sont :

- La récupération des données du composant de stockage
- L'analyse des données
- La réalisation des rapports

La récupération de données du composant Elasticsearch est faite grâce à un *framework* appelé « spring-data-elasticsearch ». Ce client permet d'exposer des méthodes spécifiques à l'API, qui acceptent les objets de requête en tant qu'arguments et renvoient des objets de réponse. Dans le cas de l'application d'évaluation des performances, les objets sont l'ensemble des tests réalisés sur une application TMA Rationnalisée.

L'analyse des données consiste à récupérer les objets stockés dans le composant de stockage et réaliser une comparaison entre les différents temps de réponse. Pour rappel, le besoin est de calculer les temps de réponse des tests réalisés sur la version actuelle ainsi que la version précédente de la même application, et également des indicateurs statistiques de ces données (moyenne, minimum et maximum).

A partir d'Elasticsearch sont récupérés les tests unitaires qui sont sauvegardés selon la structure décrite dans le paragraphe précédent, contenant le nom de l'application et la version indiquée. De cette manière, il est possible d'obtenir, dans un seul objet, les informations liées aux résultats de test d'une version de l'application.

Dans ce cas spécifique, l'API permet de comparer deux objets contenant le nom de la méthode testée avec le temps d'exécution respectif. Ainsi, les résultats de la différence des temps d'exécution sont stockés pour créer les rapports. Ces résultats permettent de créer des graphiques pour mieux visualiser l'avancement des tests. Les objets récupérés depuis Elasticsearch, permettent également de calculer les statistiques sur les temps de réponse.

Toutes ces données obtenues sont envoyées vers Elasticsearch pour l'historisation. Chaque rapport contient le nom de l'application et la version sur laquelle ont été réalisés les tests.

Pour réaliser des analyses sur deux versions différentes de la même application, la comparaison est faite sur deux objets contenant le même nom d'application, mais ayant des versions différentes. Si l'on souhaite réaliser des analyses sur la même version, alors sont comparés deux objets avec le même nom d'application, la même version, mais deux *timestamp* différents.

Architecture

L'architecture de l'API Spring Boot est divisée en 4 parties : Model, Services, Repository, Controller.

La partie Model se charge de décrire les objets avec leurs attributs. Les attributs doivent avoir la même structure que les documents d'Elasticsearch, de manière que l'on puisse stocker tous les éléments présents dans la base de données.
La partie Services est composée des classes qui s'occupent de la logique métier. C'est dans ces classes que le calcul des statistiques et la création des rapports ont lieu.
La partie Repository s'occupe de la persistance des données dans Elasticsearch.
La partie Controller contient les classes qui se chargent du traitement d'une requête depuis son interception jusqu'à la génération de la réponse et sa transmission. La responsabilité du contrôleur est d'appeler une ou plusieurs fonctions de la couche de service.

La figure ci-dessous présente les composants décrits précédemment et le lien avec le composant de stockage de données.

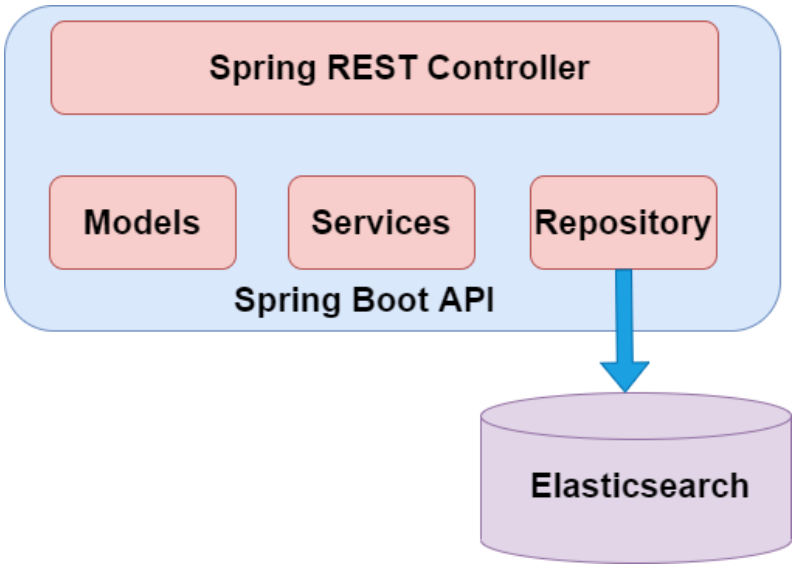

Figure 36 – Architecture de l'API

Le fonctionnement de Spring Boot se base sur des annotations qui permettent de simplifier la création de l'API. Par exemple pour créer un composant de type contrôleur, il suffit de créer la classe et de l'annoter @RestController avec un point d'accès.

Une des annotations les plus importantes de Spring Boot est *Autowired*. En ajoutant l'annotation @Autowired au constructeur, Spring s'occupera d'instancier la classe directement en injectant les dépendances décrites dans les paramètres du constructeur ; pour que chacune des classes, ayant le même service de dépendance, reçoivent la même instance de service. Ce qui permet d'utiliser le principe d'injection de dépendances de manière très efficace.

De la même manière que pour le contrôler, il existe dans Spring Boot l'annotation @Service qui permet de créer un type service.

Le cas le plus spécifique est celui du Model, qui sera annoté avec @Document. Cela est dû au fait que cette classe permet de réaliser la persistance dans Elasticsearch.
Avec le diagramme de classe ci-dessous il est possible de visualiser la relation entre les classes citées précédemment.

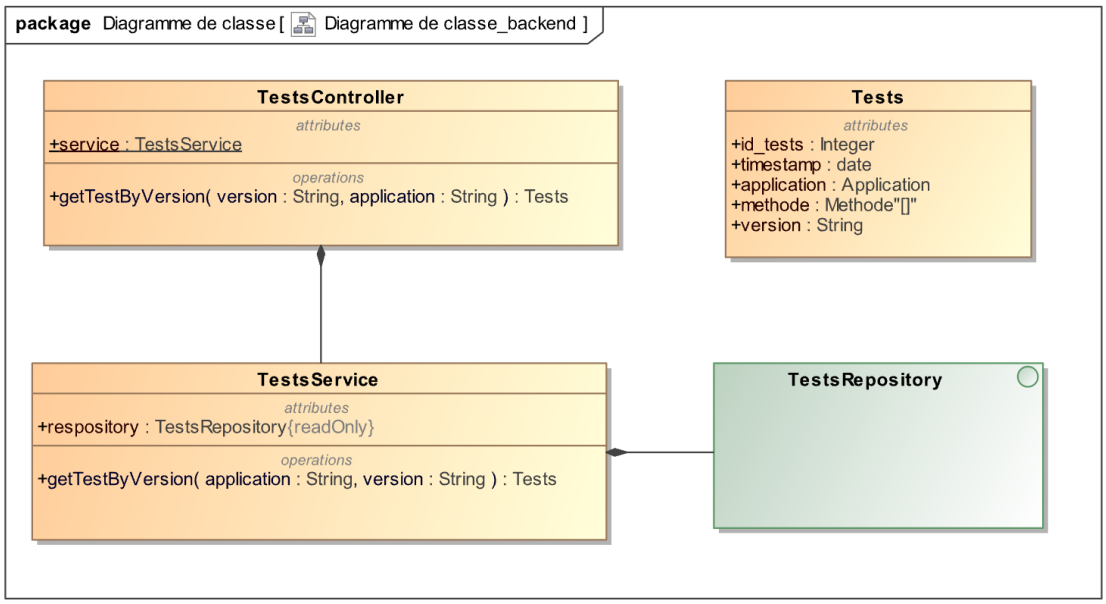

Figure 37 – Diagramme de classe du composant API

Sur ce diagramme de classe sont présentes les classes concernant l'objet « Test » qui permettent de récupérer un jeu de tests réalisé sur une certaine application à partir de son nom et sa version. Le Model de test contient le nom de l'application, la version de l'application et une liste des méthodes testées, avec le temps d'exécution.

Pour que l'API soit opérationnelle et que l'utilisateur puisse interagir avec, il est nécessaire d'intégrer un composant de restitution des données.

6.2.5. Composant pour la restitution des données

Ce composant se charge de l'affichage des rapports réalisés. Plusieurs outils qui ont été identifiés dans la partie dédiée à l'état de l'art, sont présentés ci-dessous.

Kibana

Kibana est une interface web qui permet de créer des graphiques comme des histogrammes, ou des statistiques à partir des données stockées dans Elasticsearch. Les *dashboards* de Kibana sont personnalisables et peuvent être utilisés pour différents besoins métier. De plus les rapports créés peuvent être sauvegardés ou exportés. L'interface graphique propose différents outils pour pouvoir créer des *dashboards* pour s'adapter au mieux aux besoins des projets.

Graylog2

Graylog2 est un outil *open source* de gestion de logs. Cet outil permet de collecter chaque message et l'enregistrer dans une base de données comme Elasticsearch. Doté d'une interface web, il permet de gérer et analyser vos logs. Graylog2 est découpé en deux parties : graylog2-server et graylog2-web-interface. La première est une application Java qui accepte les messages sous différents protocoles. Chaque message est analysé puis enregistré dans l'outil de stockage. Une API REST est également intégrée à l'outil et est notamment utilisée par la partie web. Celle-ci permet de gérer des utilisateurs et des *dashboards*. [40]
Graylog2 capture, stocke, permet la recherche en temps réel et l'analyse de logs par rapport à des téraoctets de données machine. Ces données peuvent être générés à partir de tout type de composant de l'infrastructure informatique et des applications. Le logiciel utilise une architecture à trois niveaux et un système stockage basé sur Elasticsearch ou MongoDB. [23]

Grafana

Portée par l'américain Grafana Labs, Grafana est une plateforme *open source* spécialisée dans le monitoring, l'analyse et la visualisation des métriques. Elle est livrée avec un serveur web permettant d'y accéder via une API. Sous licence Apache 2.02, Grafana génère des graphiques et tableaux de bord à partir de bases de données de séries temporelles, telles que Graphite ou Prometheus. Celui-ci permet de créer des tableaux de bord contenant des métriques brutes ou agrégées, provenant des bases de données, en définissant simplement des seuils d'alerte et les actions associées. Comme évoqué plus haut, Grafana, en lien avec des bases de données orientées *time series*, est adapté à l'analyse de métriques, à la différence de Kibana qui est plutôt adapté pour explorer des données de logs. [41]

Tous les outils présentés ci-dessus peuvent interagir et se connecter directement avec le composant de stockage, cependant ce qui est limitatif par rapport à l'architecture conçue est que la réalisation de l'analyse doit se faire via le composant d'analyse de données. Ces outils n'ont pas été conçus pour interagir de cette manière avec une API. De ce fait, il est nécessaire de créer une interface graphique pour l'API.

Mise en place de la solution

Cette solution permet à l'utilisateur de créer le rapport en fonction des versions de l'application qu'il souhaite, de visualiser les rapports créés et de les télécharger. De cette manière l'utilisateur décide à quel moment il souhaite réaliser une analyse de log.
Ce composant est écrit en Angular 11. Cette technologie est choisie pour s'intégrer avec le composant d'analyse de données ainsi que pour des raisons de maintenabilité, car Angular 11 est aussi une technologie qui fait partie des applications du projet TMA Rationnalisée.

Architecture

Angular est structuré avec une architecture de type MVC. Le *design pattern* Modèle-vue-contrôleur est destiné aux interfaces graphiques et est très populaire pour son utilisation dans les applications web. Le *pattern* est composé de trois types de modules ayant des responsabilités différentes : les modèles contiennent les données à afficher, les vues, la présentation de l'interface graphique et les contrôleurs, la logique concernant les actions effectuées par l'utilisateur.

Dans le cas d'Angular, l'organisation est la suivante : il y a les Models qui contiennent les classes de descriptions des Objets (comme les modèles dans Spring Boot), les Components c'est-à-dire, les classes qui se chargent de l'interaction avec l'utilisateur et les Templates qui contiennent la description des pages.[10]

Dans notre cas, la couche service est également présente pour faire le lien avec l'API. L'organisation des différents modules sont présentés dans la figure ci-dessous.

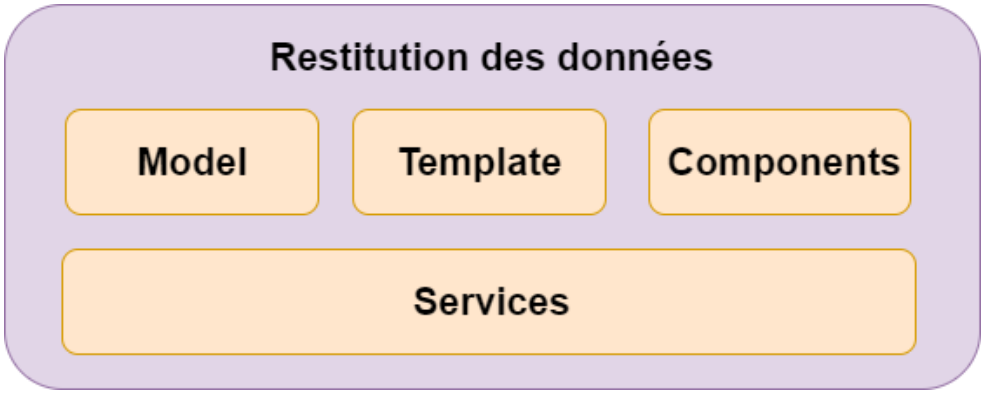

Figure 38 – Architecture composant de restitution des données

Dans notre cas les modèles ont la même structure que ceux du composant d'analyse des données. Ce sont des types utilisés par les classes du Component et des Services.

Les Components sont les classes qui vont permettre l'affichage des données à l'écran, écouter les évènements générés par les utilisateurs et effectuer les actions lancées par ces évènements. Comme pour Spring Boot, dans Angular aussi il existe des annotations pour identifier chaque composant, et dans ce cas l'annotation est @Component.

Le Template est composé des vues des Components. Ces classes sont composées du code HTML qui permet de gérer ce qui est affiché à l'écran.

Le Service se constitue de classes qui vont interagir avec l'API créée précédemment. La logique est d'instancier un objet de type Observable qui permettra d'attendre la réponse du serveur.
Le Component est inscrit à cet observable avec la méthode « subscribe() » afin d'être informé du flux d'information entre le service et l'API. Les *observables* prennent en charge la transmission de messages asynchrones entre les deux parties de l'application.

Comme il est possible de voir dans l'image ci-dessous, les services vont interagir directement avec le contrôleur de l'API, qui lui, va se charger de recevoir les évènements ou les données, d'appeler les méthodes nécessaires et ensuite de renvoyer la réponse au *frontend*.

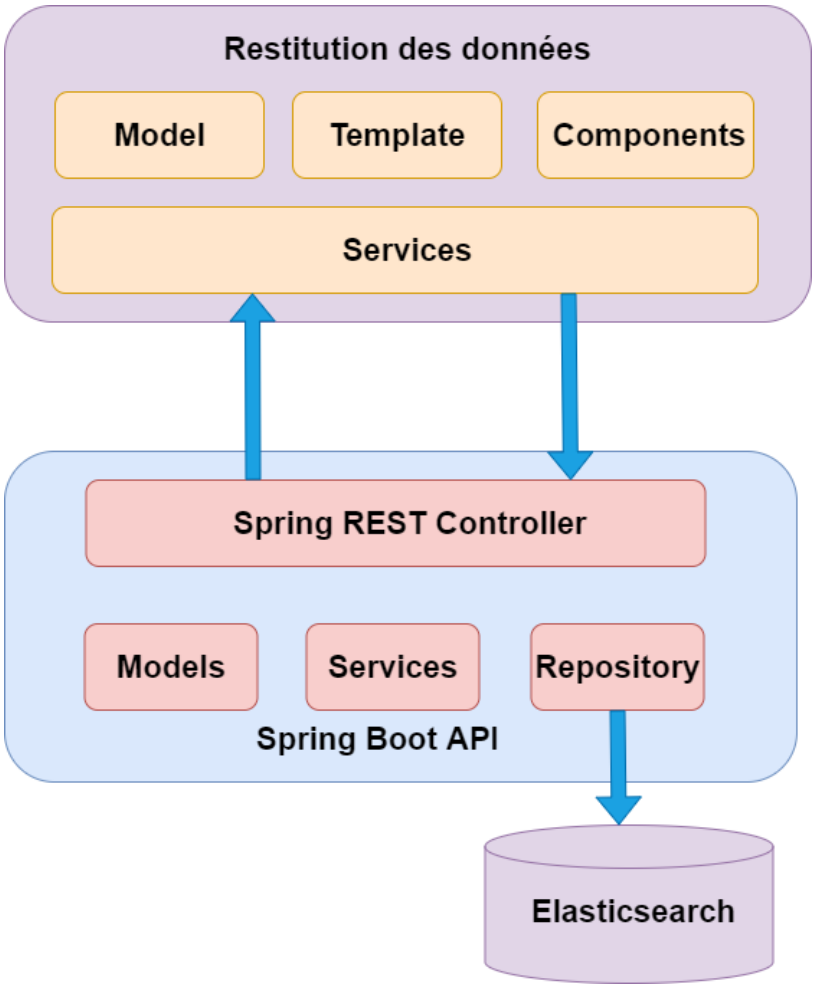

Figure 39 – Architecture API et frontend

Avec ce système, l'utilisateur dispose d'une interface qui lui permet de choisir l'application et la version sur laquelle il souhaite avoir un rapport, établir le seuil d'acceptabilité et visualiser les résultats obtenus. Le seuil d'acceptabilité permet d'établir la limite en dessous de laquelle la différence des temps de réponse des tests entraine une dégradation sévère de l'application. De plus l'utilisateur a la possibilité d'exporter les rapports réalisés.

En ce qui concerne l'affichage de l'interface utilisateur, le *framework* Bootstrap est utilisé. Bootstrap permet de créer des pages web *responsives* et facilement personnalisables. Il propose une large gamme de composants, icônes et d'autre options qui facilitent la mise en forme des pages web. De plus il est compatible avec la majorité des navigateurs ainsi qu'avec d'autres bibliothèques, comme Chart.js. Cette bibliothèque permet de créer les graphiques. Le choix de cette bibliothèque repose sur le fait qu'elle est compatible avec les autres composants du système et permet de réaliser des graphiques adaptés, comme dans l'image ci-dessous.

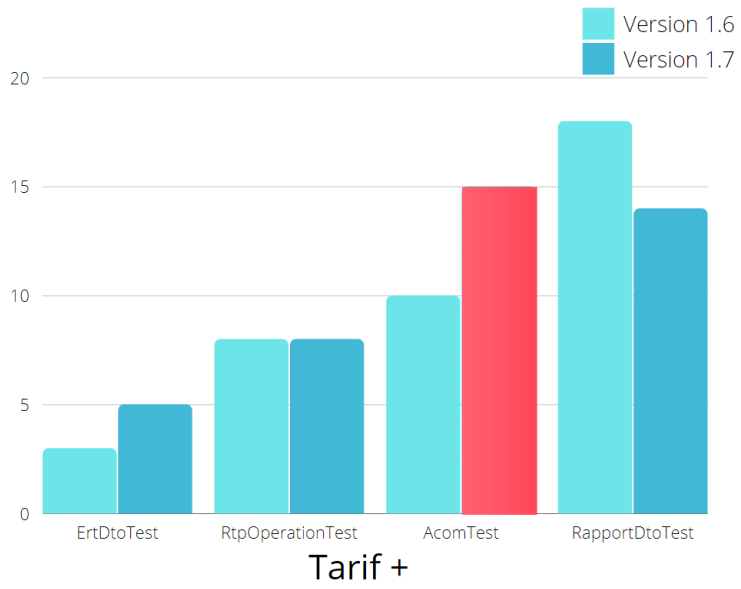

Figure 40 – Comparatif entre résultats de test

Le graphique permet d'identifier l'application sur laquelle est réalisée l'analyse, les versions comparées et les méthodes testées. La barre rouge indique que ce test a dépassé le seuil d'acceptabilité, qui est de 5 secondes pour cet exemple.

6.3. Macro-planning

Pour la réalisation de ce système, comme décrit dans la première partie du document, un tableau Kanban a été créé avec les différentes tâches à réaliser. Ces tâches correspondent aux exigences du système et ont été traitées par ordre de traitement des composants. Ce tableau a permis un suivi régulier de la part des deux tuteurs, qui sont ajoutés en tant que collaborateurs au tableau Kanban.

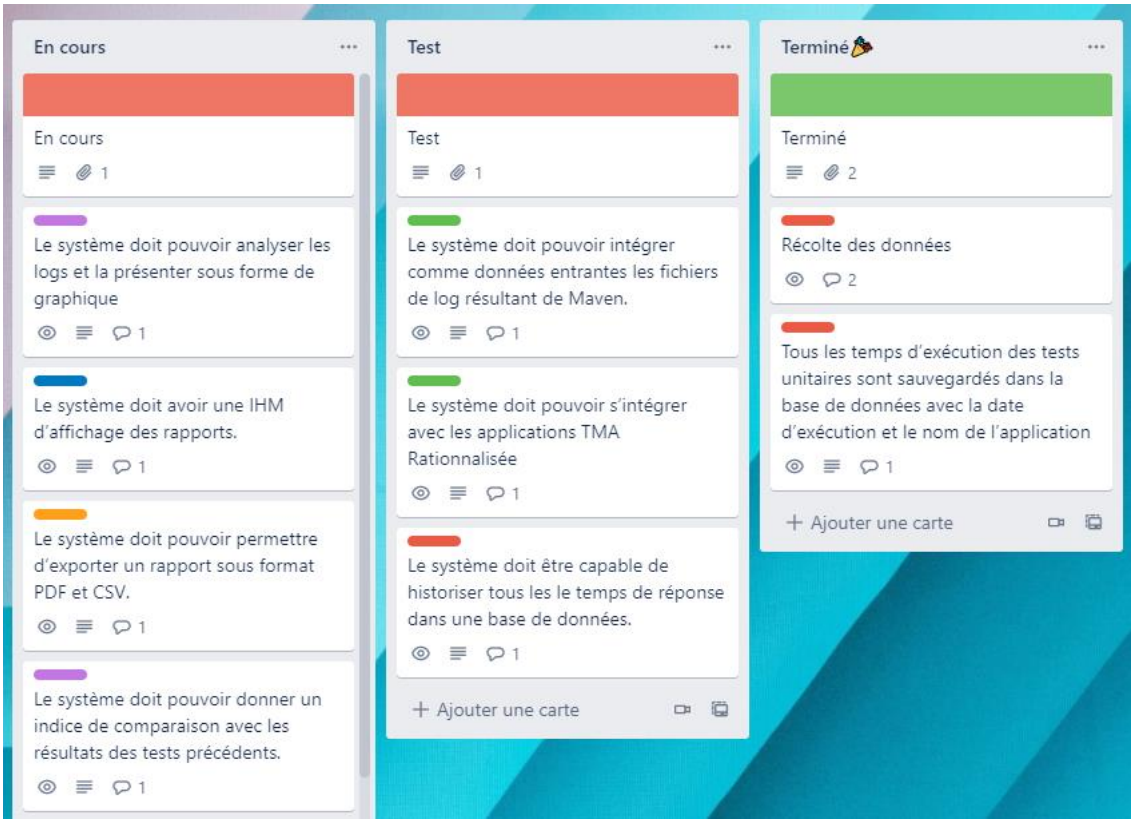

Figure 41 – Extrait du tableau Kanban

En termes de temps de réalisation du projet, un extrait du micro-planning est présenté ci-dessous.

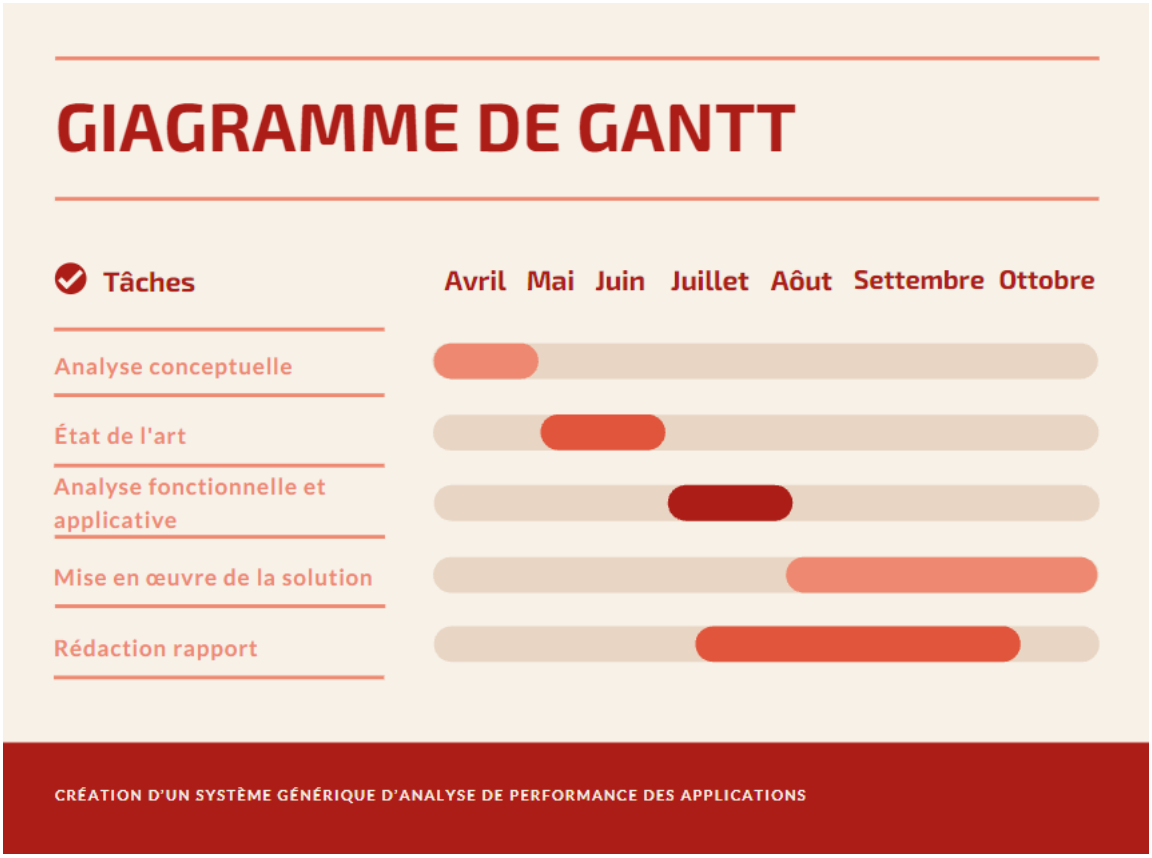

Figure 42 – Diagramme de Gantt Macro-planning

7. Evaluation de la solution

7.1. Bilan

La solution proposée répond aux besoins du client avec une architecture qui est présentée ci-dessous. Les technologies utilisées ont été choisies en fonction de l'environnement existant sur le projet TMA Rationnalisée pour des raisons de compatibilité avec les serveurs ou machines, mais aussi pour des raisons de maintenabilité du système.

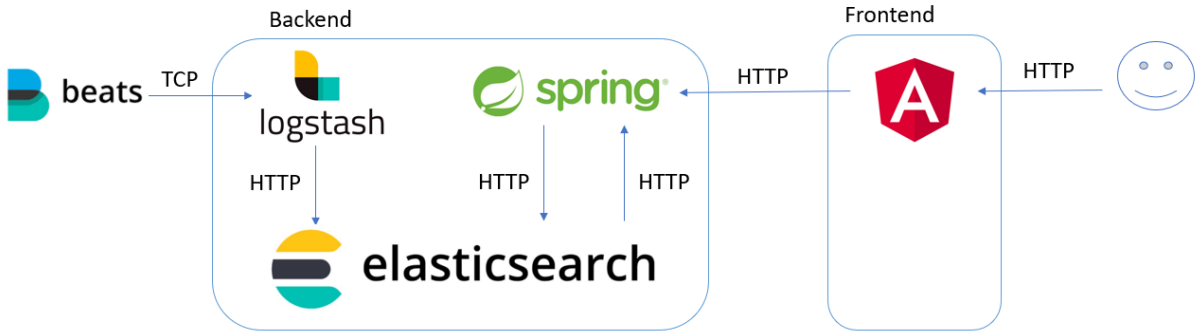

Figure 43 – Architecture applicative

La difficulté avec la création de ce système a été de trouver les bonnes technologies et de les connecter entre elles.
L'homogénéisation des logs a été un défi, car les outils existants sur le marché en version *open source*, se basent sur des filtres qui nécessitent que les données soient structurées afin que l'on puisse définir des règles de filtrage.

Dans les fichiers de log générés par les applications du projet TMA Rationnalisée, les données n'ont pas le même ordre, mais ont une structure similaire. En conséquence, il est nécessaire de créer plusieurs filtres, et Logstash se charge d'appliquer celui qui correspond à la ligne identifiée. Ces règles sont créées pour filtrer des données utiles pour l'analyse des tests, comme le nom de l'application, sa version et les méthodes testées.

Cette méthode est suffisante, mais pas optimale, car il suffit d'ajouter une application générant des logs qui ne rentrent pas dans les catégories décrites par les filtres de Logstash déjà implémentés pour que les données ne soient pas sauvegardées dans le bon format, donc elles ne seront pas exploitables. La contrainte est de créer un nouveau filtre.

En ce qui concerne le projet TMA Rationnalisée, les applications qui en font parties sont plutôt stables, et l'arrivée de nouvelles applications est rare, ce qui fait que l'ajout d'un nouveau filtre n'est pas un problème. Mais, dans un cas plus général, le problème d'homogénéisation des données reste un point à approfondir.

En ce qui concerne l'historisation des données, l'outil choisi pour le stockage de données s'adapte à tout type de données, donc il est possible de stocker des rapports et de les retrouver au besoin. Cette solution est adaptée au système, car les rapports ont une structure définie et des identifiants qui permettent de les retrouver facilement grâce au moteur de recherche de Elasticsearch.

En ce qui concerne l'automatisation du système, Filebeat est un outil qui est paramétrable afin de récupérer tous les nouveaux fichiers présents dans un répertoire. Ce qui permet d'intégrer automatiquement tous les résultats de test d'une application dans la base de données. De cette manière l'utilisateur peut réaliser une analyse de performances quand il le souhaite, à condition que les tests soient réalisés sur l'application. La seule action qu'il doit faire est de se rendre sur l'interface de l'application d'analyse des performances et de choisir les paramètres souhaités. De cette manière, il peut réaliser une analyse entre deux versions consécutives ou non.

Ce qui a été réalisé est un POC du produit pour valider l'intégration des composants. Le système a été testé avec des fichiers de log réalisés par les applications TMA Rationnalisée pour créer des filtres déjà adaptés aux fichiers des applications.

Un exemple de rapport en format PDF est présenté ci-dessous. Les informations présentes sont le nom de l'application, le numéro identifiant du rapport, le graphique représentant les différences de temps d'exécution des méthodes sur les deux versions de l'application, le seuil d'acceptation et les statistiques relatives.

Dans l'exemple, ont été analysés les versions 1.6 et 1.7 de l'application « Tarif + ». Sur le graphique sont représentés les différences de temps d'exécution de chaque méthode, entre les deux versions. Par exemple, la méthode « ErtDtoTest » n'a pas eu de variation de temps d'exécution entre les deux versions, car sa valeur est à 0. La méthode « AcomTest », à une valeur de 1.3 secondes, ce qui signifie que son temps d'exécution est fortement dégradé dans la version 1.7, par rapport à la version 1.6. De plus, sa valeur dépasse le seuil d'acceptation qui est de 1 seconde, c'est-à-dire que la variabilité de son temps d'exécution à un fort impact sur les performances de l'application. La méthode « RapportDtoTest » a une valeur négative, ce qui signifie que les temps d'exécution de cette méthode ont subi une diminution de 0.2 seconde dans la version 1.7 par rapport à la version 1.6, grâce à la nouvelle évolution réalisée.

Les statistiques sont réalisées sur les temps d'exécution des tests de la version 1.7. Dans ce cas, le temps minimum d'exécution d'une méthode est de 0.2 seconde, le temps maximum de 1.5 seconde et le temps moyen de 0.6 seconde.

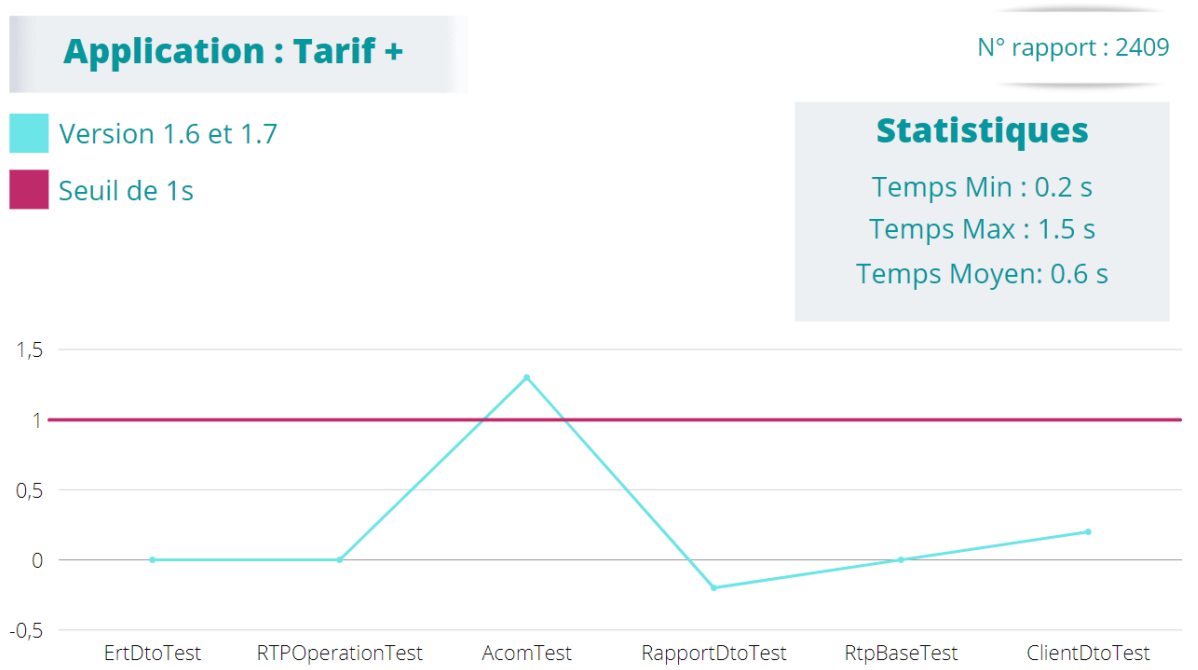

Figure 44 – Maquette de rapport en format PDF

Dans le rapport en format CSV sont présentes les mêmes informations, avec la différence que les indicateurs de différence de temps d'exécution sont présentés sous forme de liste.

	A	B	C	D	E
1	N° rapport : 2409				
2	Application : Tarif +				
3	Version 1.6 et 1.7				
4	Seuil 1s				
5					
6	**Méthodes**	**Difference temps éxecution**		**Statistiques**	
7	ErtDtoTest	0			
8	RTPOperationTest	0		Temps min	0.2
9	AcomTest	1.3		Temps max	1.5
10	RapportDtoTest	-0.2		Temps moyen	0.6
11	RtpBaseTest	0			
12	ClientDtoTest	0.2			
13					

Figure 45 – Maquette de rapport en format CSV

Le système n'a été testé que dans un environnement de développement, mais il reste la nécessité de tester le système dans un environnement de préproduction. Les tests à réaliser sont des tests fonctionnels : réaliser analyse, visualiser analyse, télécharger en format PDF, télécharger en format CSV, modifier seuil d'acceptation. De plus des tests unitaires sont également à réaliser sur les méthodes du composant d'analyse, pour valider le bon fonctionnement de chaque composant.

7.2. Perspectives

En ce qui concerne les perspectives du système d'analyse de performance des applications, il est possible d'envisager que le système puisse comparer les résultats de tests de plusieurs versions d'une même application, afin qu'ils puissent générer des graphiques d'évolution des performances de l'application sur longue durée. Ce fonctionnement permet de visualiser sur un graphique les améliorations ou dégradations de l'application, sur une période, par exemple d'un an. Le rapport réalisé peut être téléchargé comme les autres rapports.

L'affichage et les boutons peuvent être améliorés pour avoir une interface graphique plus ergonomique. Ainsi que les formats des rapports, par exemple dans le rapport en format PDF il est possible d'ajouter aux statistiques des temps d'exécution, les noms des méthodes correspondantes.
En ce qui concerne l'interface, il est possible d'envisager d'ajouter un bouton de modification du seuil d'acceptabilité.

Pour le projet ne sont pris en compte et comparés que les tests unitaires, mais il est possible d'intégrer et comparer d'autres types de tests existants sur les applications, comme les tests de performance.

Optimiser la méthode de comparaison des différentes données des tests obtenus, améliorerait l'analyse des données. Plusieurs types d'indicateurs peuvent être ajoutés aux rapports générés en fonction des besoins futurs ou des retours de l'équipe.

La sécurisation du système est un point à prendre en considération lors de la mise en production. Les outils come Filebeat, Logstash et Elasticsearch permettent d'implémenter la communication TLS, afin de chiffrer le trafic réseau. En ce qui concerne le composant d'analyse et de restitution des données, il existe des bibliothèques adaptées qui permettent de configurer la communication en https.

Conclusion

Dans ce document est présentée la conception d'un projet, à partir de l'expression du besoin, jusqu'à la réalisation. Plusieurs phases d'analyse sont présentées, en fonction de la typologie du système. Le système d'analyse de performances, est un outil créé pour être simple à utiliser et à installer, et efficace dans son fonctionnement. Pour cette raison, les outils choisis reflètent exactement ces caractéristiques.

Le choix d'utiliser des outils à intégrer comme partie du système, permet de garantir une meilleure efficacité. Dans le cas du composant de transfert des fichiers, le choix d'utiliser Filebeat est basé sur la simplicité d'installation et sur les performances de cet outil. Comme pour Elasticsearch, son moteur de recherche est ce qui fait sa puissance. Grâce à Elasticsearch, les temps de recherche de logs et le stockage des informations sont gérés de manière efficace. Ce qui est un point important à considérer dans le système réalisé.

La spécificité du système, est la méthode d'analyse et son interface. Le choix de ne pas utiliser des outils déjà créés, est une question de facilitation d'utilisation du système. Avec une interface comme Kibana, l'utilisateur aurait dû créer ses propres *dashboards* en fonction de l'application ou de l'analyse qu'il souhaite réaliser, et accéder en parallèle à l'application pour que les rapports d'analyse soient générés. Ce fonctionnement n'est pas envisageable pour un logiciel qui a comme critère la simplicité d'utilisation en premier lieu.

Bibliographie et Webographie

[1] Matthieu Anceret. 2019. Réflexion sur la scalabilité et la haute disponibilité dans le web. *Matthieu Anceret*. Retrieved August 29, 2021 from https://anceret-matthieu.fr/2019/03/r%C3%A9flexion-sur-la-scalabilit%C3%A9-et-la-haute-disponibilit%C3%A9-dans-le-web/

[2] Antoine. Bonnes pratiques de logging | Java & Moi. Retrieved August 26, 2021 from https://javaetmoi.com/2021/01/bonnes-pratiques-de-logging/

[3] Denise Dubie. 2006. Performance management from the client's point of view. *Network World*. Retrieved August 26, 2021 from https://www.networkworld.com/article/2300639/performance-management-from-the-client-s-point-of-view.html

[4] Ilyas Ed-daoui, Mhamed Itmi, Abdelkhalak El Hami, Nabil Hmina, and Tomader Mazri. 2017. Vers des systèmes de systèmes robustes. *IncertFia* 17, 2 (November 2017). DOI:https://doi.org/10.21494/ISTE.OP.2017.0187

[5] David Galiana. Qu'est-ce que la méthodologie Kanban ? Retrieved August 26, 2021 from https://www.planzone.fr/blog/quest-ce-que-la-methodologie-kanban

[6] +Bastien L. 2019. APM : tout savoir sur les outils Application Performance Management. *LeBigData.fr*. Retrieved August 27, 2021 from https://www.lebigdata.fr/apm-application-performance-management

[7] Lecarme Mathieu. Fluentd : pour une meilleure gestion de vos logs | blog Bearstech. *Bearstech | Coopérative d'ingénieurs expert en hébergement et infogérance*. Retrieved July 18, 2021 from https://bearstech.com/societe/blog/normaliser-les-logs/

[8] Yann Pollet. 2016. *Systèmes, architectures, intégration* (Ellipses ed.).

[9] sematext. 2020. What Is Log Analysis Tutorial: Logging Use Cases, Benefits & More. *Sematext*. Retrieved August 27, 2021 from https://sematext.com/blog/log-analysis/

[10] Guillaume Simard. 2018. Angular 5 (Component, Template, Modèle et Service). *Atomrace*. Retrieved September 16, 2021 from https://atomrace.com/angular-5-component-template-modele-et-service/

[11] 2011. La performance des applications. Retrieved August 29, 2021 from https://blog.axopen.com/2011/06/la-performance-des-applications/

[12] 2012. APM and MoM – Symbiotic Solution Sets. *APMdigest - Application Performance Management*. Retrieved August 26, 2021 from https://www.apmdigest.com/apm-and-mom-symbiotic-solution-sets

[13] 2015. Fluentd vs Logstash: A Comparison of Log Collectors. *Logz.io*. Retrieved July 23, 2021 from https://logz.io/blog/fluentd-logstash/

[14] 2018. A Practical Introduction to Logstash. *Elastic Blog*. Retrieved July 24, 2021 from https://www.elastic.co/fr/blog/a-practical-introduction-to-logstash

[15] 2019. What Is Log Analytics? Find Important Patterns in Your Logs - XpoLog. *XPLG Log Management | Automations, Log Analysis & Log Viewer*. Retrieved September 9, 2021 from https://www.xplg.com/what-is-log-analytics/

[16] 2020. Logging & monitoring : définitions et bonnes pratiques. Retrieved August 30, 2021 from https://www.vaadata.com/blog/fr/logging-monitoring-definitions-et-bonnes-pratiques/

[17] 2021. Elasticsearch vs Splunk: A Comparison and How to Choose. *Scalyr*. Retrieved September 14, 2021 from https://www.scalyr.com/blog/elasticsearch-vs-splunk-comparison/
[18] 2021. Performances (informatique). *Wikipédia*. Retrieved August 29, 2021 from https://fr.wikipedia.org/w/index.php?title=Performances_(informatique)&oldid=179059345
[19] 2021. Trello. *Wikipédia*. Retrieved July 25, 2021 from https://fr.wikipedia.org/w/index.php?title=Trello&oldid=181845357
[20] 2021. Test de performance. *Wikipédia*. Retrieved August 29, 2021 from https://fr.wikipedia.org/w/index.php?title=Test_de_performance&oldid=183038926
[21] 2021. Log file. *Wikipédia*. Retrieved August 30, 2021 from https://en.wikipedia.org/w/index.php?title=Log_file&oldid=1026394837
[22] 2021. Splunk. *Wikipédia*. Retrieved September 14, 2021 from https://fr.wikipedia.org/w/index.php?title=Splunk&oldid=183581237
[23] 2021. Graylog. *Wikipédia*. Retrieved September 14, 2021 from https://en.wikipedia.org/w/index.php?title=Graylog&oldid=1036129793
[24] Qu'est-ce que le langage UML (langage de modélisation unifié) ? *Lucidchart*. Retrieved September 17, 2021 from https://www.lucidchart.com/pages/fr/langage-uml
[25] Cameo Systems Modeler - CATIA - Dassault Systèmes®. Retrieved July 25, 2021 from https://www.3ds.com/products-services/catia/products/no-magic/cameo-systems-modeler/
[26] diagrams.net (draw.io) : une plateforme pour créer des diagrammes et des organigrammes. *BDM/tools*. Retrieved July 25, 2021 from https://www.blogdumoderateur.com/tools/diagrams-net-draw-io/
[27] How to analyze a load test report? Part 4: Response time. *On Web Load Testing*. Retrieved August 29, 2021 from https://www.loadtestingtool.com/blog/wapt-usage/how-to-analyze-load-test-report-response-time/
[28] When (if ever) is High Availability "good enough"? Do I always need disaster recovery or business continuity plans? :: Systems Flow, Inc. Retrieved August 29, 2021 from http://www.sysflow.com/blog/high-availability/
[29] Types of Performance Tests. *Abstracta Software Testing Services*. Retrieved August 31, 2021 from https://abstracta.us/blog/performance-testing/why-performance-testing-is-necessary/attachment/why-perf-testing-is-nec-chart/
[30] What is Log Analysis? Uses Cases For Digital Performance. *AppDynamics*. Retrieved August 27, 2021 from https://www.appdynamics.com/topics/what-is-log-analysis/index.html
[31] Qu'est-ce qu'un fichier LOG et comment l'utiliser ? Retrieved August 30, 2021 from https://www.1min30.com/dictionnaire-du-web/fichier-log-definition
[32] Développons en Java - Le logging. Retrieved August 30, 2021 from https://www.jmdoudoux.fr/java/dej/chap-logging.htm
[33] What is Log Analysis? Use Cases, Best Practices, and More | Digital Guardian. Retrieved August 27, 2021 from https://digitalguardian.com/blog/what-log-analysis-use-cases-best-practices-and-more
[34] Log analysis…an effective methodology. *NTT*. Retrieved August 27, 2021 from https://hello.global.ntt/en-us/insights/blog/log-analysis-an-effective-methodology

[35] Apache Flume | Architecture | Fonctionnement et avantages d'Apache Flume. Retrieved September 10, 2021 from https://fr.education-wiki.com/fr.education-wiki.com/8903343-apache-flume
[36] Filebeat : Analyse de logs légère et Elasticsearch. *Elastic*. Retrieved September 10, 2021 from https://www.elastic.co/fr/beats/filebeat
[37] Apache Hadoop. Retrieved September 10, 2021 from https://hadoop.apache.org/
[38] Plugin de filtre Grok | Référence Logstash [7.14] | Élastique. Retrieved September 13, 2021 from https://www.elastic.co/guide/en/logstash/current/plugins-filters-grok.html
[39] Loggly de SolarWinds. *Orsenna*. Retrieved July 24, 2021 from https://www.orsenna.fr/loggly/
[40] Présentation – Graylog. Retrieved September 14, 2021 from https://graylog.fr/presentation/
[41] Grafana (gratuit) : la data visualisation open source orientée observabilité. Retrieved September 15, 2021 from https://www.journaldunet.fr/web-tech/guide-de-l-entreprise-digitale/1443882-grafana-gratuit-la-data-visualisation-du-monitoring-it-open-sourc130921/

Liste de abréviations

Intitulé	Description
API	Application Programming Interface
APM	Application Performance Management
ASCII	American Standard Code for Information Interchange
AWS	Amazon Web Services
BPCE	Banque Populaire et caisse d'Epargne
BI	Business Intelligence
BNP	Banque nationale de Paris
BPMN	Business Process Model and Notation
CAPS	Crédit Agricole Payment Services
CNCF	Cloud Native Computing Foundation
CNIL	Commission Nationale de l'Informatique et des Libertés
CSE	Centre de Services Espagne
CSV	Comma-Separated Values
ELK	Elasticsearch Logstash Kibana
ESN	Entreprise au Service du Numérique
HDFS	Hadoop Distributed File System
HLR	High Level Requirements
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
IBM	International Business Machines Corporation
IHM	Interface Homme Machine
IRC	Internet Relay Chat
JDBC	Java Database Connectivity
JSON	JavaScript Objet Notation
JVM	Java Virtual Machine
MBSE	Model-Based Systems Engineering
MCD	Modèle Conceptuel de Données
ML	Machine Learning
MVC	Modèle Vue Contrôleur
MVS	Multiple Virtual Storage
NT	Nouvelles Technologies
OBC	Odier Bungener et Courvoisier
PDF	Portable Document Format
POC	Proof Of Concept
REST	REpresentational State Transfer

SI	Service Informatique
SIG	Système d'Informations de Gestion
SQL	Structured Query Language
STIE	Scientifique Technique Industrielle et Embarquée
SysML	Systems Modeling Language
TCP	Transmission Control Protocol
TLS	Toulouse
TMA	Tierce Maintenance Applicative
UML	Unified Modeling Language
VBS	Value Breakdown Structure
VoIP	Voice over Internet Protocol
WBS	Work Breakdown Structure

Liste des figures

Annexe

Annexe 1 - Cas d'utilisation « Visualiser résultats »

Cas d'utilisation :	Visualiser résultats
Acteurs :	Développeurs des applications
Scénario nominal détaillé :	Le développeur visualise l'analyse de des performances d'une l'application.
Conditions préalables :	Le développeur doit avoir installé le composant de collecte du système sur l'application qu'il gère. Le développeur doit être connecté au réseau interne à l'entreprise. Le développeur doit avoir réalisé l'analyse des performances d'une application.
Postconditions :	/
Flux Autre(s) :	/
Spécifications supplémentaires :	L'application doit être paramétrée dans le système.

Figure 1 –Cas d'utilisation « Visualiser résultats »

Annexe 2 - Cas d'utilisation « Relancer analyse »

Cas d'utilisation :	Relancer analyse
Acteurs :	Développeurs des applications
Scénario nominal détaillé :	Le développeur re lance la réalisation d'une analyse des performances de l'application.
Conditions préalables :	Le développeur doit avoir installé le composant de collecte du système sur l'application qu'il gère. Le développeur doit être connecté au réseau interne de l'entreprise. Le développeur doit avoir réalisé au préalable l'analyse d'une application.
Postconditions :	/
Flux Autre(s) :	/
Spécifications supplémentaires :	L'application doit être paramétrée dans le système.

Figure 2 –Cas d'utilisation « Relancer analyse »

Annexe 3 - Cas d'utilisation « Télécharger résultats »

Cas d'utilisation :	Télécharger résultats
Acteurs :	Développeurs des applications
Scénario nominal détaillé :	Le développeur télécharge l'analyse des performances d'une application. L'utilisateur peut choisir entre le téléchargement de l'analyse en format PDF ou en format CSV
Conditions préalables :	Le développeur doit avoir installé le composant de collecte du système sur l'application qu'il gère. Le développeur doit être connecté au réseau interne à l'entreprise. Le développeur doit avoir réalisé au préalable l'analyse d'une application.
Postconditions :	/
Flux Autre(s) :	/
Spécifications supplémentaires :	L'application doit être paramétrée dans le système.

Figure 3 –Cas d'utilisation « Télécharger résultats »

Création d'un système générique d'analyse de performance des applications
Mémoire d'Ingénieur CNAM, Toulouse 2021

Résumé

La qualité d'un logiciel est un critère essentiel sur lequel les clients sont intransigeants. Ce processus est composé de plusieurs phases qui permettent de mettre en place des actions qui vont faciliter la réalisation d'un produit qualitatif. Pour cette raison la réalisation des tests de performance sur les applications est un point important. Cette étape peut être difficile et longue à réaliser. C'est pourquoi il est nécessaire de créer un système qui permet de déterminer rapidement et efficacement les performances des applications. Le système décrit dans le document est conçu pour qu'il puisse analyser les résultats de test de toute application qui respecte des critères de log déterminés et la présence de tests unitaires. Les rapports générés par ce système permettent au développeur d'avoir une vue globale sur l'état de l'application en termes de performances.

Mots clés : test unitaire, performance applicative, Gestion des Performances Applicatives, logs, automatisation, statistiques.

Abstract

The quality of a software is an essential criterion on which customers are intransigent. This process is composed of several phases that allow us to set up actions that will facilitate the realization of a qualitative product. For this reason, the performance testing of the applications is an important point. This step can be difficult and time consuming. That's why is necessary to create a system that allows to determine quickly and efficiently the performance of applications. The system described in the document is designed so that it can analyze the test results of any application that meets certain log criteria and the presence of unit tests. The reports generated by this system allow the developer to have a global view on the state of the application in terms of performance.

Keywords : unit testing, application performance, Application Performance Management, logs, automation, statistics.