Université IBN TOFAIL
Faculté de Sciences-Kenitra
Département Informatique

Rapport De Projet De Find'Etude

pour l'obtention du diplôme :

Master Big Data & Cloud Computing

REFONTE DE L'APPLICATION MONOLITHIQUE DE GESTION DES INCIDENTS VERS UNE ARCHITECTURE MICROSERVICES

Réalisé Par :
Roudane Rachid
Effectue à :
Office Chérifien des Phosphates - El Jadida

Encadrent Pédagogique par :
Dr. Jaafar Abouchabaka

Encadrent à l'OCP par :
M. Youssef Chriai

Soutenu le ····· devant le jury compose de :

Dr. Jaafar Abouchabaka ..... Chef de departement informatique

2018-2019

Remerciement

Mes vifs remerciements vont à mon encadrant de stage M. Jaafar Abouchabaka qui, par son expérience a su toujours répondre à toutes mes interrogations. Sans oublier ces remarques pertinentes, sa disponibilité, son encouragement et ses précieux conseils.

J'adresse également mes sincères remerciements à mon responsable de stage M. Youssef Chriai pour son encouragement, sa participation au cheminement de ce rapport, son aide et ses pertinentes directives.

Enfin, je voudrais exprimer ma profonde reconnaissance à toute personne qui a contribué à l'aboutissement de ce projet. Ce travail n'aurait pas pu voir jour sans l'aide précieuse de tout le personnel du groupe OCP, particulièrement à la direction des systèmes informatiques qui m'ont soutenu durant toute ma période de stage.

Table des matières

Table des figures

Introduction

Mon stage de fin d'étude effectué au sein de GROUP OCP durant environ quatre mois s'inscrit dans le cadre d'un projet de recherche & développement et veille technologique portant sur les tendances et concepts technologiques Big Data & Microservices .

Les Big Data sont des méthodes et des technologies pour des environnements évolutifs, pour l'intégration, le stockage et l'analyse des données multi-structurées (structurées, semi structurées et non structurées).

Le terme "Microservice" a connu une émergeance au cours des dernières années pour décrire une un style d'architecture bien particulier. Cette approche consiste à développer une seule application en un ensemble de petits services, isolés, autonomes et indépendamment déployés.

L'objectif de ce stage a consisté à monter en compétence sur les technologies Big Data et microservices afin d'élaborer une présentation commerciale des ces technologies, puis de faire des tests techniques des composants de l'écosystème Hadoop (framework Java libre, destiné à faciliter la création d'applications distribuées et scalables) et refonte de l'application monolithique de gestion des incidents vers une architecture microservices.

1 - Présentation de l’organisme d’accueil

1.1 Présentation de l'organisme d'accueil

1.1.1 Historique

Le groupe OCP est un acteur de référence incontournable sur le marché international, il est domicilié au Maroc. OCP S.A est le leader mondial des exportations de phosphates et produits dérivés, avec des activités couvrant l'ensemble de la chaîne de valeur, allant de l'extraction de la roche de phosphate à la transformation chimique en acide phosphorique et différents engrais.

Mondialement, le Maroc détient les réservoirs les plus importants de phosphate dans son sous-sol, il est de l'ordre de 51,8 Milliard de tonnes de minerai, ce qui représente 75 % des réserves mondiales. Ses premières traces ont été découvertes en 1912, dans les régions des OULAD ABDOUN, zone de Khouribga. Son gisement est une superposition de couches de différentes teneurs situées à 120 Km du Sud-est de Casablanca.

Le phosphate provient de la décomposition des fossiles des animaux de mers qui ont vécu, il y a plusieurs millions d'années du fait que les mers et les océans recouvraient une grande partie des continents actuels.

La création de l'OCP fut en 1920 et l'exploitation n'a commencé qu'en 1921 dans la région de Oued-Zem, depuis lors les besoins continus de l'agriculture mondiale en phosphate ont fait de l'office une entreprise qui jusqu'à nos jours n'a cessé de grandir et pour se maintenir sur le plan de la concurrence par rapport aux autres pays producteurs de phosphate et dérives, elle se modernise, se développe continuellement et s'affirme comme le LEADER du marché mondial des phosphates.

1.1.2 Status Juriqique de l'OCP

L'OCP a été constitué sous la forme d'un organisme semi-publique sous contrôle de L'ETAT. Il fonctionne ainsi comme une société dont le seul actionnaire est l'état Marocain, appliquant les méthodes de gestion privée, dynamique, souple et efficace vu le caractère de ses activités industrielles et commerciales, il est dirigé par un Directeur General nomme par DAHIR, le contrôle est exercé par un conseil d'administration préside par le premier ministre sur le plan fiscal, elle est inscrite sur le registre de commerce soumis à la même obligation que d'autres entreprises privées.

Sur le plan financier ; entièrement indépendante de celle de l'ETAT. Toutes les années, le Groupe, participe au budget de l'Etat par le versement des dividendes. La gestion du personnel est régie par le statut du mineur du 1er janvier 1973. Ce statut a été élaboré en conformité avec le DAHIR n° 16007 du 24 décembre 1960 sur le statut des entreprises minières au MAROC.

Les structures actuelles ont été modifiées par le document 716 du 1/1/1971 signe par le Directeur General du Groupe OCP est d'environ 22677 dont 725 ingénieurs.

1.1.3 Sites miniers

Le Groupe est présent dans cinq zones géographiques du pays dont trois sites d'exploitation minière Khouribga, Benguerir/Youssoufia, Boucraa/Laayoune et deux sites de transformation chimiques : Safi et Jorf Lasfar.

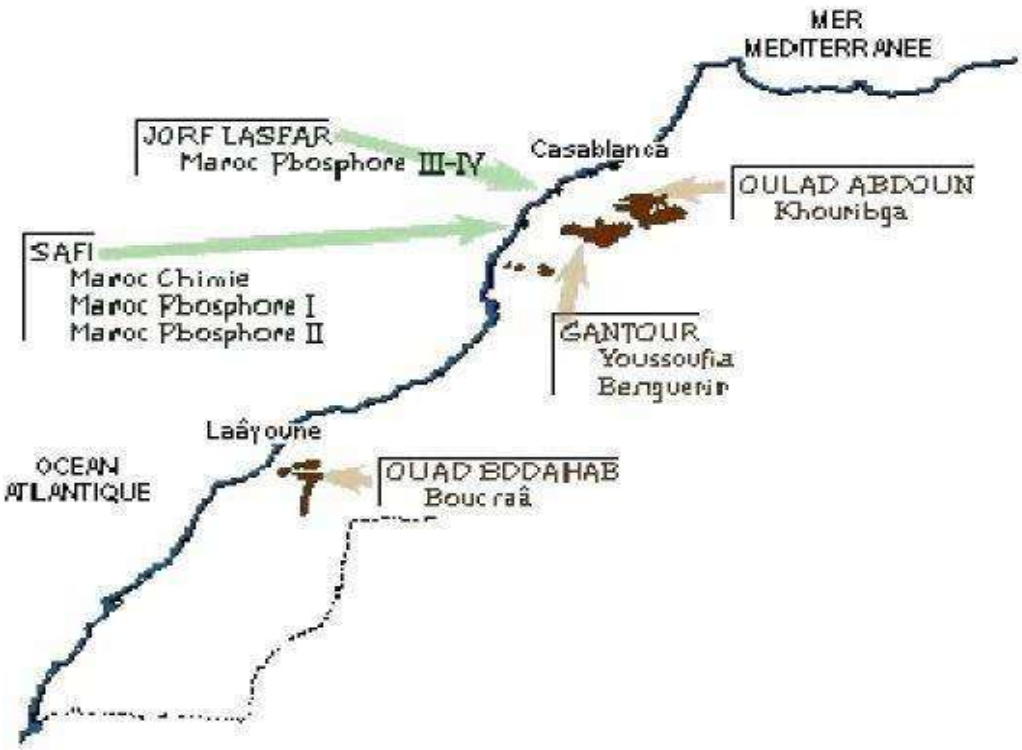


Figure 1:Carte géographique des sites miniers distribués sur le royaume

L'implantation géographique a connu plusieurs mines telles qu'Ouled Abdoun, Gantour, Boucraa… etc.

OULED ABDOUN :

C'est la plus importante mine de la production du groupe OCP, située dans la zone de Khouribga, ses réserves en phosphates sont estimées a plus de 35 milliards de mètre cubes. Sa capacité de production est de 19 millions de tonnes par an.

GANTOUR :

La Direction des Exploitations Minières de GANTOUR a pour mission l'extraction, le traitement et la livraison du phosphate a partir du gisement de GANTOUR, ce gisement sur 125 Km d'Est à l'ouest etsur 20Km du Nord au Sud. Il couvre une superficie de 2500Km2. Il existe deux centres qui sont en exploitation : le centre de YOUSSOUFIA (depuis 1939) et le centre de BENGURIR (depuis 1980).

OUED EDDAHAB (BOUCRAA-LAAYOUNE) :

Situe a 50 Km au Sud-est de la ville de LAAYOUNE, les réserves des gisements des phosphates d'OUED EDDAHAB sont estimées a 1,13 milliards de mètres cubes. Le gisement de Boucraa est en exploitation, compte tenu de ses réserves et de sa teneur.

1.1.4 Missions

- ✓ **L'extraction :** est une opération qui se fait en découverte, elle consiste à extraire le phosphate de la terre et s'exécute en 4 phases : Forage, sautage, décapage et défruitage.
- ✓ **Le traitement :** c'est une opération qui se fait après extraction et a pour but l'enrichissement du minerai en améliorant sa teneur.

- ✓ **Le transport :** une fois le phosphate extrait puis traite, il est transporté vers le port de CASA, SAFI ou EL JADIDA à destination des différents pays importateurs.
- ✓ **La vente :** le phosphate est vendu, selon des demandes des clients, soit brut soit après valorisation (transforme en engrais, acides phosphoriques ou acides sulfuriques), aux industries chimiques.

1.1.5 Impact Economiques

Le Groupe OCP constitue un vecteur de développement régional et national important :

- ✓ 1er exportateur mondial de phosphate,
- ✓ 1er exportateur mondial d'acide phosphorique,
- ✓ 1er exportateur mondial de P2O5 sous toutes formes.
- ✓ Sa contribution dans le PIB est de 2 à 3% et dans les exportations marocaines en valeur de 18% à 20%.

1.1.6 Filiales

En 1975, le groupe OCP a été institué et comprend en plus de l'OCP les sociétés suivantes :

- ✓ **CERPHOS :** Centre d'études et de recherche des phosphates minéraux, sa mission est d'organiser et exécuter toute activité d'analyse, d'étude et de recherche scientifique et technique.
- ✓ **FERTIMA :** Société marocaine des fertilisants, son but est de commercialiser les engrais à l'intérieur du pays en provenance des chimiques du groupe OCP.
- ✓ **MARPHOCEAN :** Il est spécialisé dans le transport maritime de l'acide phosphorique, les engrais et d'autres produits chimiques.
- ✓ **SMESI :** Société marocaine d'études spéciales et industrielles, ses activités principales sont l'étude et la réalisation des installations industrielles (manutention, stockage…).
- ✓ **SOTREG :** Société de transport régional, est chargée du transport du personnel du groupe OCP.
- ✓ **PHOSBUCRAÄ :** Sa mission est l'extraction, le traitement et l'expédition du phosphate.
- ✓ **STAR :** Société de transport et d'affrètement réunis des navires et service annexes.

1.1.7 Structure et Organigramme

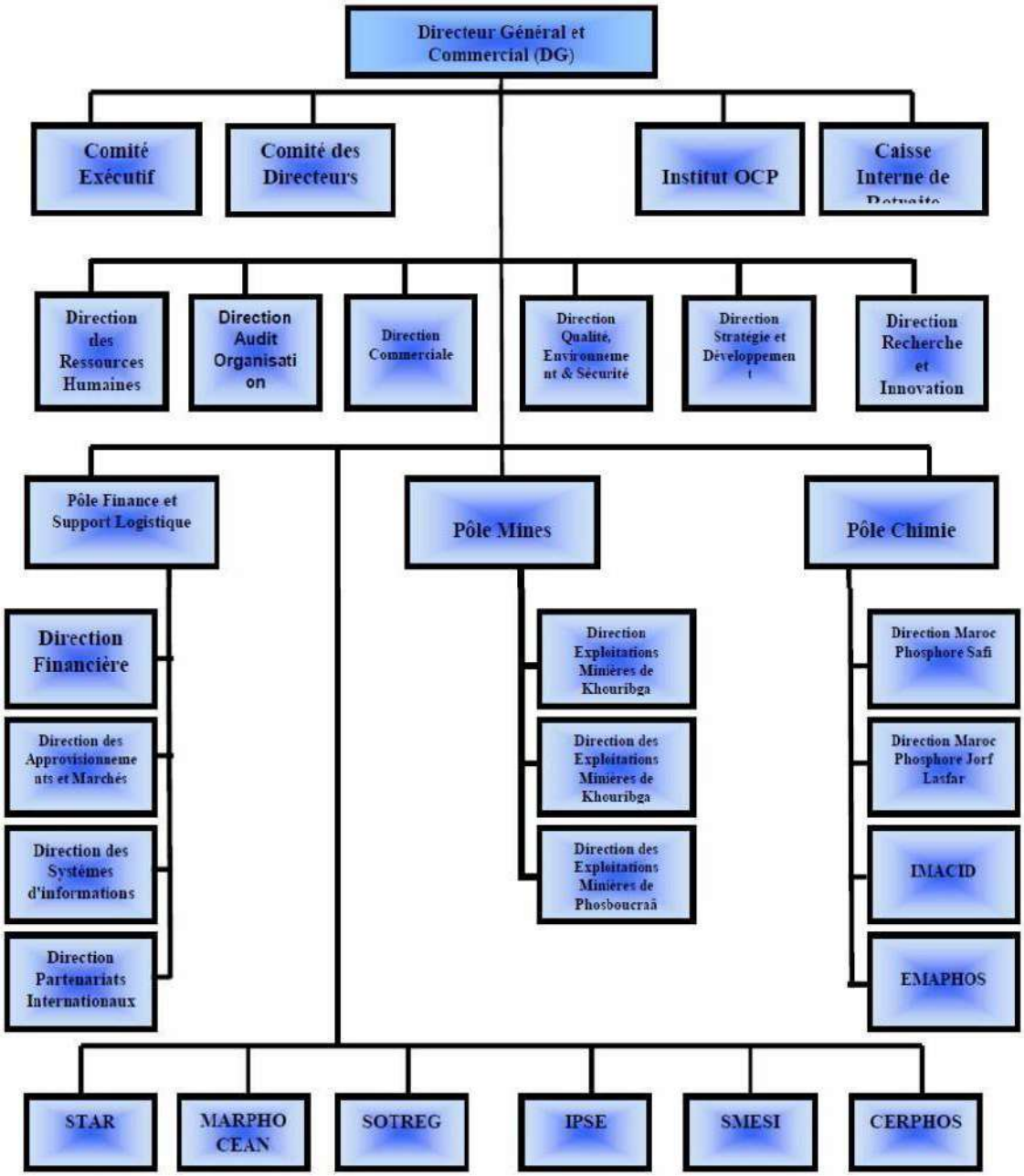


Figure 2: Organigramme de l'OCP

1.2 Contexte du projet

1.2.1 Présentation de l'application monolithique du gestion des incidents

Application monolithique du gestion des incidents est un outil assurant la gestion des incidents informatique .

Application web qui est une application monolithique, développer en un seul bloc et par la même technologie, permet d'invoquer des services web en utilisant des classes qui sont fortement couplées ce qui rend difficile sa maintenabilité et son évolution.

Le développement d' application web gestion des incidents est subdivisé par ordre de priorité : des fonctionnalités qui sont en production, d'autres en développement et d'autres en attente pour des itérations ultérieures. D'un autre côté, les cycles de changement sont répétitifs. Dans ces situations

l'architecture monolithique a prouvé des limites en termes de flexibilité et dynamisme. Ce style de structure reste un frein face à la construction et la modification d' application . Pour ceci group OCP prévoit de migrer cet outil afin de pallier ces problèmes.

1.2.2 Limites et critiques de l'existant

Après chaque itération l'outilgestion des incidents , subit des améliorations. Des fonctionnalités s'ajoutent pour s'aligner plus au besoin du client. Le projet a commencé depuis des années, et selon son plan d'évolution, il continuera à évoluer encore pour quelques années. Ceci a généré plusieurs défis.

En effet,la taille du projet n'a cessé d'augmenter pour devenir une application monolithique gigantesque, difficile à gérer et à comprendre. Même le respect des bonnes pratiques et les efforts fournis pour maintenir un code modulaire et évolutif n'a pas pu éliminer la complexité de ce projet. Avec une application qui comporte des milliers des lignes de code et un grand nombre de classes plusieurs problèmes se présentent.

En premier lieu, faire évoluer l'équipe est devenu de plus en plus coûteux. L'ajout d'un nouveau développeur au projet implique le sacrifice de plusieurs jours et semaines pour comprendre le code existant et afin qu'il soit capable de le modifier ou d'ajouter d'autres fonctionnalités.

En second lieu, la modification de quelques lignes au niveau de l'application entraîne le redéploiement, le test (les tests unitaires, les tests de régression, les tests IHM) et la révision de la qualité de code de toute l'application. La répétition manuelle de toute la chaîne de déploiement après chaque modification rend le travail de l'équipe de test plus coûteux en termes d'homme/jours.

En troisième lieu, la haute disponibilité, la rapidité de traitement et la fiabilité sont des priorités pour les fonctionnalités de base de l'outilgestion des incidents . Or, la réplication de toute l'application afin de garantir ces exigences pour quelques fonctionnalités est très coûteuse en termes de ressources matérielles.

Par ailleurs, un autre problème qui s'impose, depuis toujours, concerne l'agilité au niveau d'équipe qui est constituée traditionnellement de deux sous équipes : une équipe de développement et une équipe opérationnelle. L'équipe de développement collecte les exigences métier, rédige le code, exécute et teste le code dans un environnement isolé. Cette équipe n'est pas souvent préoccupée par l'impact de leur code sur la production. L'équipe opérationnelle exploite le code fourni par l'équipe de développement et elle est plutôt concentrée sur la stabilisation des services et de l'architecture ainsi que sur la performance des instances actuelles.

Cette organisation traditionnelle entre les équipes alourdit énormément la mise en production. En effet, il y a de nouvelles fonctionnalités qui augmente le temps de la mise en production. Potentiellement, cela implique l'augmentation du temps d'indisponibilité pour l'application, du stress pour les équipes et bien entendu de l'argent perdu.

1.2.3 Impact de l'architecture monolithique

L'architecture monolithique est l'une des plus répandues dans les projets informatiques. Elle consiste à développer des applications entières en une seule pièce. Ces applications produisent de bons résultats pour les petits projets voire les moyens. Cependant après des mois de développement elle limite l'intégration des ouvertures, l'agrégation de nouveaux besoins et l'innovation technologique. Le tableau 1.1 récapitule les principales limites de l'architecture monolithique.

Limites	Impact
Limites technologiques	- Le choix des technologies est décidé avant que le développement de l'application commence. - Une seule technologie est utilisée pour le développement de l'application.
Mise à l'échelle coûteuse	- La mise à l'échelle d'une partie qui a besoin de plus de ressources requiert toute l'application.
Modification du code coûteuse	- Une modification dans une petite partie de l'application requiert un rebuild et redéploiement de toute l'application.
Coût des tests élevé	- Durée des tests automatiques et des builds est longue.
Tolérance aux pannes limitée	- Un dysfonctionnement sur une partie du système a un impacte sur toute l'application.
Agilité limitée	- Agilité réduite de l'équipe et fréquence de livraison limitée à cause du couplage fort entre les composants. - Effort important de montée en compétences pour un nouvel arrivant sur des composants fortement couplés. - Nécessité de l'intervention de plusieurs personnes pour chaque modification.

Figure 3:Inconvénients de l'architecture monolithique

2 - Généralités sur la Technologie BIG DATA

2.1 Présentation

Après le très en vogue "cloud computing", un nouveau concept émerge dans le secteur informatique, celui du "Big data". A l'origine du concept de "Big data" se trouve l'explosion du volume de données informatiques, conséquence de la flambée de l'usage d'Internet, au travers des réseaux sociaux, des appareils mobiles, des objets connectés, etc.
Selon le CXP, les Big Data désignent des méthodes et des technologies (pas seulement des outils) pour des environnements évolutifs (augmentation du volume de données, augmentation du nombre d'utilisateurs, augmentation de la complexité des analyses, disponibilité rapide des données) pour l'intégration, le stockage et l'analyse des données multi-structurées (structurées, semi structurées et non structurées).

2.2 Caractéristiques des Big Data

Le Big Data se caractérise par la problématique des 3V :

- ✓ **Volume :** c'est le poids des données à collecter. Confrontées à des contraintes de stockage, les entreprises doivent aussi gérer le tsunami des réseaux sociaux. La montée en puissance des réseaux sociaux a accentué cette production de données.
- ✓ **Variété :** l'origine variée des sources de données qui sont générées. premièrement, l'époque où les entreprises s'appuyaient uniquement sur les informations qu'elles détenaient dans leurs archives et leurs ordinateurs est révolue. De plus en plus de données nécessaires à une parfaite compréhension du marché sont produites par des tiers. Deuxièmement, les sources se sont multipliées : banques de données, sites, blogs, réseaux sociaux, terminaux connectés comme les smartphones, puces RFID, capteurs, caméras... Les appareils produisant et transmettant des données par l'Internet ou d'autres réseaux sont partout, y compris dans des boîtiers aux apparences anodines comme les compteurs électriques.
- ✓ **Vélocité :** la vitesse à laquelle les données sont traitées simultanément. à l'ère d'internet et de l'information quasi instantanée, les prises de décision doivent être rapides pour que l'entreprise ne soit pas dépassée par ses concurrents. La vélocité va du batch au temps réel.Vélocité, la vitesse à laquelle les données sont traitées simultanément. à l'ère d'internet et de l'information quasi instantanée, les prises de décision doivent être rapides pour que l'entreprise ne soit pas dépassée par ses concurrents. La vélocité va du batch au temps réel.

A ces « 3V », les uns rajoutent la **visualisation** des données qui permet d'analyser les tendances ; les autres rajoutent la **variabilité** pour exprimer le fait que l'on ne sait pas prévoir l'évolution des types de données.

2.3 Les Big Data en chiffres

Après le siècle du pétrole, nous entrons dans l'ère de la donnée. Les chiffres ci-dessous permettent de présenter la quantité de données générées jusqu'ici et la croissance dans les prochaines années :

- ✓ **12 zettaoctets** de données ont été créés dans le monde en **2011.**
- ✓ **118 milliards** d'emails sont envoyés **chaque jour**.
- ✓ **235 téraoctets de données** ont été collectés par The Library of Congress en avril **2011.**
- ✓ **30 fois** plus de données seront générées d'ici 2030.
- ✓ Le télescope "**Square kilometers away**" produira plus **d'1 téraoctet de données** par minute en 2024.
- ✓ **Twitter** génère **7 téraoctets** de données par jour.
- ✓ **Facebook** génère **10 téraoctets** de données par jour.
- ✓ **Facebook** traite **50 milliards** de photos.
- ✓ **30 milliards** de contenus sont échangés chaque mois sur **Facebook.**

Nous pouvons bien constater que nous nageons dans un océan de donnée où le niveau de la mer augmente rapidement.

2.4 Intérêts des Big Data

L'utilisation des Big Data pourrait impacter fortement le monde de l'entreprise et ce de façon méliorative, ainsi les entreprises pourront :

- ✓ Améliorer la prise de décision.
- ✓ Réduire les coûts d'infrastructures informatiques via l'utilisation des serveurs standards et des logiciels open source.
- ✓ Développer la réactivité et l'interactivité à l'égard des clients.
- ✓ Améliorer les performances opérationnelles.

Tout ce ci orientera les entreprises vers une économie centrée sur la donnée.

2.5 Enjeux des Big Data

Le Big Data apparaît comme le challenge technologique des années 2010-2020. Dépassant les domaines techniques et informatiques, le Big Data suscite un vif intérêt auprès des politiciens, des scientifiques et des entreprises. Les enjeux du Big Data touchent plusieurs secteurs d'activités.

2.5.1 Enjeux techniques

Les enjeux techniques s'articulent autour de l'intégration, le stockage, l'analyse, l'archivage, l'organisation et la protection des données.

2.5.2 Enjeux économiques

D'après le cabinet de conseil dans le marketing IDC, « le marché du Big Data représentera 24 milliards de dollars e 2016, avec une part de stockage estimée à 1/3 de ce montant » (Saulem, Enjeux économiques du Big Data à l'échelle mondiale - Le big data.htm, 2013). Il va sans dire que la « donnée » est le nouvel or noir du siècle présent, les spécialistes s'accordent déjà sur le fait que le Big Data sera l'arme économique de demain pour les entreprises et se présentera comme un levier qui fera la différence.

Les entreprises collectent de plus en plus d'information en relation avec leurs activités (production, stockage, logistique, ventes, clients, fournisseurs, partenaires, etc), toutes ces informations peuvent être stockées et exploitées pour stimuler leur croissance. Les Big Data permettent :

- ✓ D'améliorer les stratégies marketing et commerciale.
- ✓ D'améliorer et entretenir la relation client.
- ✓ De fidéliser la clientèle.
- ✓ De gagner de nouvelles parts de marché.
- ✓ De réduire les coûts logistiques.
- ✓ De favoriser la veille concurrentielle

Le client est un acteur majeur dans ce contexte. Jusqu'à présent, la vente consistait à se demander « J'ai un produit, à qui vais-je pouvoir le vendre? ». A l'ère du Big Data, nous devons changer le

paradigme pour dire « J'ai un client, de quoi a-t-il besoin aujourd'hui ? ». En connaissant mieux son public, à travers ses achats, ses activités sur Internet, son environnement, les commerçants peuvent améliorer l'expérience-client, exploiter la recommandation, imaginer le marketing prédictif (le marketing prédictif regroupe les techniques de traitement et de modélisation des comportements clients qui permettent d'anticiper leurs actions futures à partir du comportement présent).

2.6 Limites des Big Data

Si le terrain de jeu du Big Data est loin d'être restreint, il n'est pas sans limites. Elles tiennent, en premier lieu, à la nature des données et aux traitements envisagés, et lorsqu'il est question des données personnelles, la vigilance est nécessaire. Dans certains pays, le traitement des données à caractère personnel est régi par des dispositions particulières, ce qui n'est pas le cas dans la majorité des pays.

Nous sommes arrivés à un point où la protection des données personnelles, portée à la défense des libertés fondamentales de l'individu, est en train de devenir un argument économique. L'enjeu étant dorénavant, d'élaborer le cadre normatif le plus attractif pour le développement de l'économie numérique et des échanges de données, ceci nécessite d'être vigilent dans un contexte de forte concurrence entre les puissances économiques.

L'autre préoccupation provient de la sécurité : une faille minuscule peut menacer des quantités de données considérables. Si les utilisateurs perdent confiance dans l'utilisation de leurs informations, c'est donc tout l'édifice du big data qui risque de s'écrouler. Pour éviter cela, par exemple en Europe, la commission européenne a présenté, en début 2012, un règlement qui vise à protéger davantage les utilisateurs. Ce texte devrait être voté en 2014 pour une application en 2016, il obligera les entreprises à demander le consentement explicite de l'utilisateur avant de collecter ses données. (Haas, 2013).

2.7 Paysage technologique des Big Data

Après la présentation des Big Data, il est aussi important de voir le paysage technologique qui constitue cette technologie. Les données quelque soit leur structure passent par plusieurs étapes avant que leur valeur ne soit perceptible. Ci-dessus, nous avons un aperçu global de différentes technologies présentes dans le paysage Big Data.

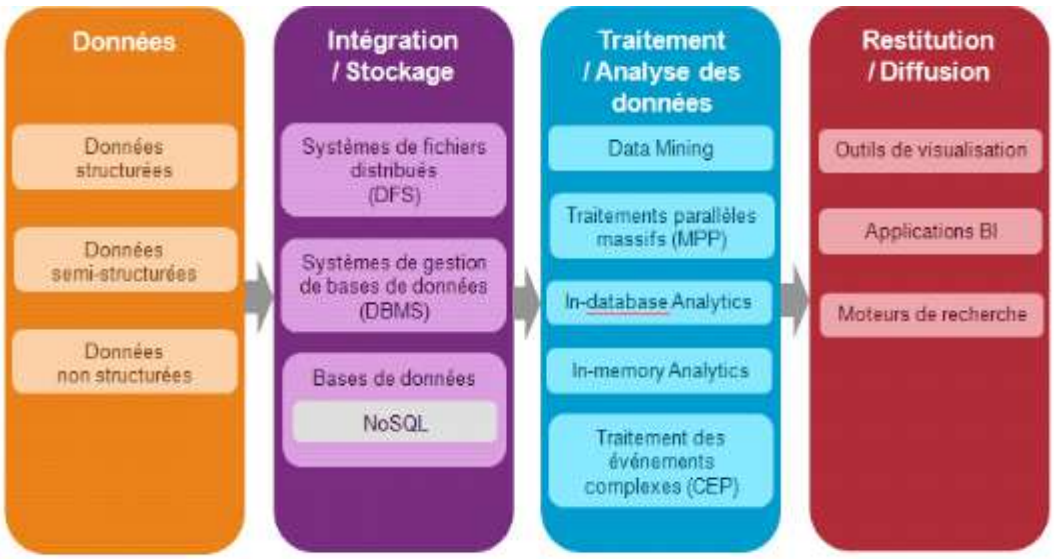


Figure 4:Paysage technologique Big Data, (Bermond, 2013)

Le Big Data repose sur plusieurs technologies, qui sont utilisées pour exploiter les gigantesques masses de données.

2.7.1 Intégration

Dans le contexte Big Data, l'intégration des données s'est étendue à des données non structurées (données des capteurs, journaux Web, réseaux sociaux, documents). Hadoop utilise le scripting via MapReduce ; Sqoop et Flume participent également à l'intégration des donnés non structurées. Ainsi, certains outils d'intégration comprenant un adaptateur Big Data existe déjà sur le marché ; c'est le cas de Talend Enterprise Data Integration - Big Data Edition. Pour intégrer des gros volumes de données issus de briques fondatrices du système d'information des entreprises (ERP, CRM, Supply Chain (gestion de la chaîne logistique)), les ETL, les EAI, les EII sont toujours utilisés.

2.7.2 Stockage

Le premier élément structurant dans le contexte Big Data est le socle de stockage des données. Anciennement, la solution était les DatawareHouse (entrepôts de données), qui ont évolué pour supporter de plus grandes quantités de données et faire porter par le stockage, une capacité de traitement étendue. Les solutions de DatawareHouse ont toutes en commun un modèle de données profondément structuré (schéma, base de données, tables, types, vues, etc) et un langage de requête SQL.
Le Big Data vient rompre cette approche ; l'approche du Big Data consiste en 2 grands principes.

Premièrement, le principe de la scalabilité (horizontale) des clusters de traitement. Puis deuxièmement, on peut s'affranchir de certaines contraintes inhérentes aux bases de données relationnelles traditionnelles et qui ne sont pas forcément nécessaires pour le Big Data. C'est le cas de l'ACIDité (Atomicité, Cohérence, Isolation et Durabilité). (*Mathieu Millet, 2013*).

Pour mettre en oeuvre cette approche avec une infrastructure simple, scalable (mot utilisé pour indiquer à quel point un système matériel ou logiciel parvient à répondre à une demande grandissante de la part des utilisateurs, il traduit aussi la capacité de montée en charge), du matériel à bas coût, le framework Hadoop est utilisé pour la gestion du cluster, l'organisation et la manière de développer. La solution la plus emblématique de cette approche est Hadoop et son écosystème.

2.7.3 Analyse

C'est bien de pouvoir stocker les données, mais faut-il pouvoir également les rechercher, les retrouver et les exploiter : c'est l'analyse des données. Elle est naturellement l'autre volet majeur du paysage technologique du Big Data. En la matière, une technologie qui s'impose, Hadoop. Ce projet applicatif fait l'unanimité même s'il est loin d'être stable et mature dans ses développements. Hadoop utilise MapReduce pour le traitement distribué.

MapReduce est un patron d'architecture de développement informatique introduit par Google qui permet de réaliser des calculs parallèles de données volumineuses (supérieures à 1 téraoctet). Les calculs peuvent être distribués sur plusieurs machines ce qui permet de répartir les charges de travail et d'ajuster facilement le nombre de serveurs suivant les besoins.

Plusieurs implémentations de MapReduce existent dont les plus connues sont l'implémentation réalisée par Google nommée Google MapReduce et l'implémentation Apache MapReduce. L'implémentation de Google est propriétaire alors qu'Apache MapReduce est open source.

Apache MapReduce est un des deux composants majeurs du framework open source Apache Hadoop qui permet la gestion des Big Data.

2.7.4 Restitution

Le Big data met aussi l'accent sur l'importance de restituer efficacement les résultats d'analyse et d'accroître l'interactivité entre utilisateurs et données. Ainsi, des produits comme QlikView, Tableau (de Tableau Software), PowerView, SpotFire proposent des visualisations graphiques innovantes.

2.8 Ecosystème Hadoop

Pour certains, l'avenir appartiendra à ceux qui seront capable d'analyser les vastes volumes de données qu'ils ont collectés.

En écologie, un écosystème est l'ensemble formé par une association ou communauté d'êtres vivants et son environnement biologique, géologique, édaphique, hydrologique, climatique, etc. l'écosystème Hadoop est l'ensemble des projets Apache ou non, liés à Hadoop et qui sont appelés à se cohabiter.

Hadoop désigne un framework Java libre ou un environnement d'exécution distribuée, performant et scalable, dont la vocation est de traiter des volumes des données considérables. Il est le socle d'un vaste écosystème constitué d'autres projets spécialisés dans un domaine particulier parmi lesquels on compte les entrepôts de données, le suivi applicatif (monitorring) ou la persistance de données.

Le schéma ci-après présente les différents éléments de l'écosystème Hadoop en fonction du type d'opération.

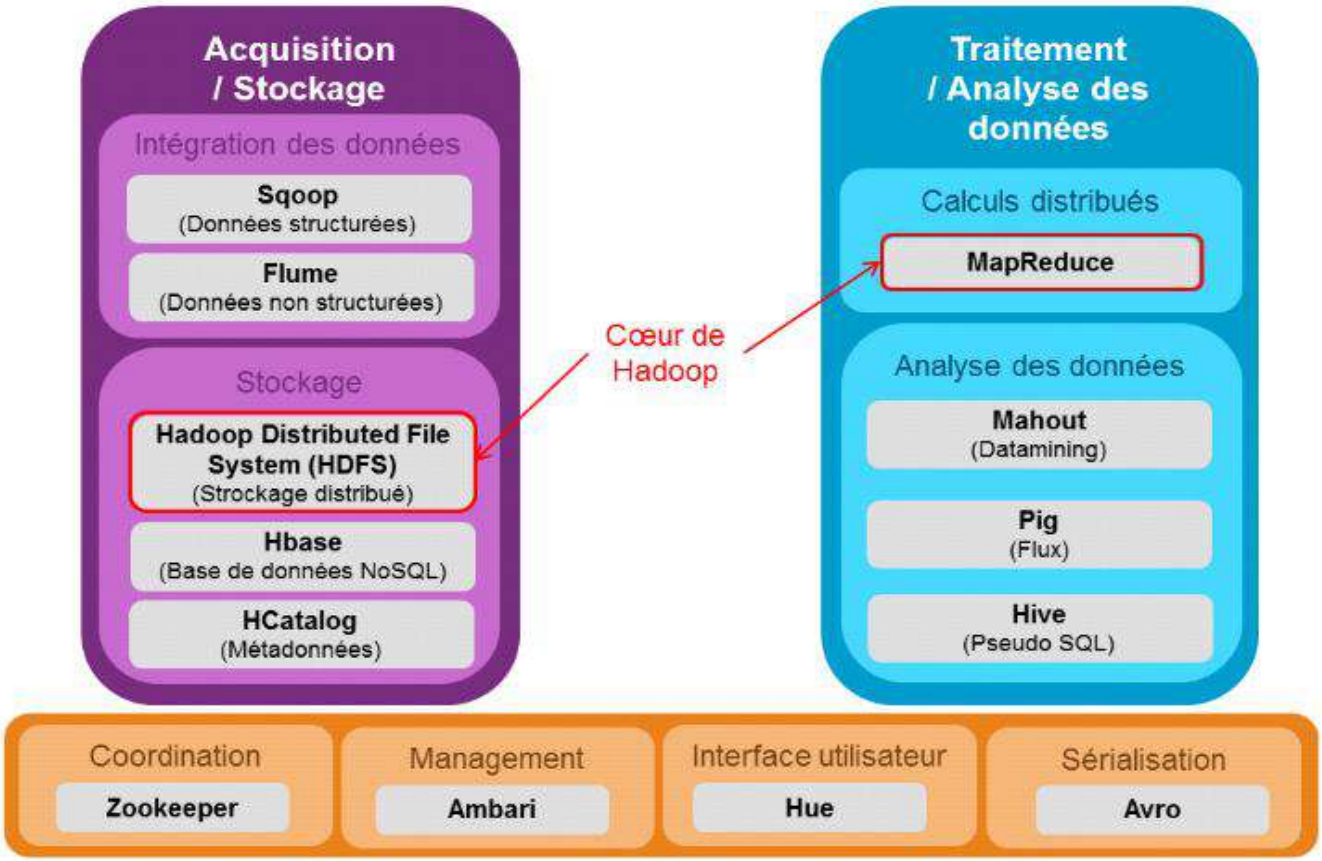


Figure 5: Ecosystème Hadoop

2.9 Hadoop kernel

Hadoop Kernel est le coeur de l'écosystème Hadoop. Ce framework est actuellement le plus utilisé pour faire du Big Data. Hadoop est écrit en Java et a été crée par Doug Cuttting et Michael Cafarella en 2005, (*Le framework Apache Hadoop*). Le noyau est composé de 2 composants. Hadoop présente l'avantage d'être issu de la communauté open source, de ce fait un porte un message exprimant une opportunité économique. En revanche, il affiche une complexité qui est loin de rendre accessible au commun des DSI.

Nous continuerons par une présentation de quelques grands concepts d'Hadoop.

2.9.1 Architecture Hadoop

Le schéma ci-dessous présente l'architecture distribué dans le contexte Hadoop.

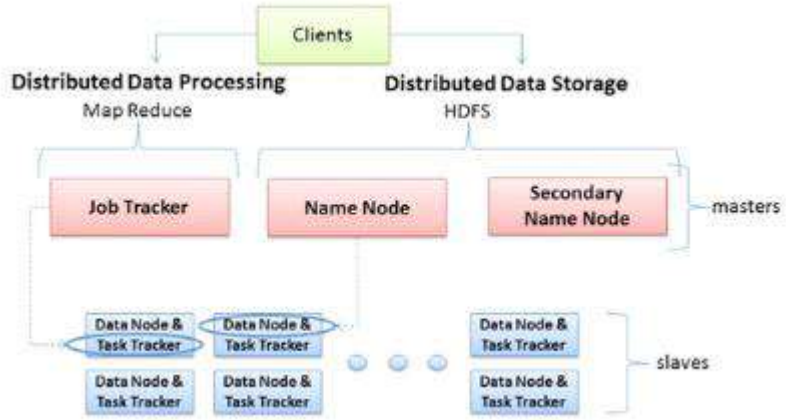


Figure 6:Architecture Hadoop avec les principaux rôles des machines, (Le framework Apache Hadoop)

Il est primordial de savoir qu'une architecture Hadoop est basée sur le principe maître/esclave, représentant les deux principaux rôles des machines. Les sous rôles relatifs au système de fichiers et à l'exécution des tâches distribuées sont associés à chaque machine de l'architecture.

Les machines maîtres ont trois principaux rôles qui leur sont associées :

- ✓ **JobTracker :** c'est le rôle qui permet à la machine maître de lancer des tâches distribuées, en coordonnant les esclaves. Il planifie les exécutions, gère l'état des machines esclaves et agrège les résultats des calculs.
- ✓ **NameNode :** ce rôle assure la répartition des données sur les machines esclaves et la gestion de l'espace de nom du cluster. La machine qui joue ce rôle contient des métadonnées qui lui permettent de savoir sur quelle machine chaque fichier est hébergé.
- ✓ **SecondaryNameNode :** ce rôle intervient pour la redondance du NameNode. Normalement, il doit être assuré par une autre machine physique autre que le NameNode car il permet en cas de panne de ce dernier, d'assurer la continuité de fonctionnement du cluster.

Deux rôles sont associés aux machines esclaves :

- ✓ **TaskTracker :** ce rôle permet à un esclave d'exécuter une tâche MapReduce sur les données qu'elle héberge. Le TaskTracker est piloté par JobTracker d'une machine maître qui lui envoie la tâche à exécuter.
- ✓ **DataNode :** dans le cluster, c'est une machine qui héberge une partie des données. Les noeuds de données sont généralement répliqués dans le cadre d'une architecture Hadoop dans l'optique d'assurer la haute disponibilité des données.

Lorsqu'un client veut accéder aux données ou exécuter une tâche distribuée, il fait appel à la machine maître qui joue le rôle de JobTracker et de Namenode.

Maintenant que nous avons vu globalement l'articulation d'une architecture Hadoop, nous allons voir deux principaux concepts inhérents aux différents rôles que nous avons présentes.

2.9.2 HDFS

HDFS est le système de fichiers Java, permettant de gérer le stockage des données sur des machines d'une architecture Hadoop. Il s'appuie sur le système de fichier natif de l'OS (unix) pour présenter un système de stockage unifié reposant sur un ensemble de disques et de systèmes de fichiers.

La consistance des données réside sur la redondance ; une donnée est stockée sur au moins n volumes différents.

Le NameNode rendrait le cluster inaccessible s'il venait à tomber en panne, il représente le SPOF (maillon faible) du cluster Hadoop. Actuellement, la version 2.0 introduit le failover automatisé (capacité d'un équipement à basculer automatiquement vers un équipement alternatif, en cas de panne). Bien qu'il y ait plusieurs NameNodes, la promotion d'un NameNode se fait manuellement sur la version 1.0.

Dans un cluster les données sont découpées et distribuées en blocks selon la taille unitaire de stockage (généralement 64 ou 128 Mo) et le facteur de réplication (nombre de copie d'une donnée, qui est de 3 par défaut).

Un principe important de HDFS est que les fichiers sont de type « write-one » ; ce ci est lié au fait que lors des opérations analytiques, la lecture des données est beaucoup plus utilisée que l'écriture.

2.9.3 MapReduce

Mapreduce qui est le deuxième composant du noyau Hadoop permet d'effectuer des traitements distribués sur les noeuds du cluster. Il décompose un job (unité de traitement mettant en oeuvre un jeu de données en entrée, un programme MapReduce (packagé dans un JAR (Java Archive : fichier d'archive, utilisé pour distribuer un ensemble de classes **Java**)) et des éléments de configuration) en un ensemble de tâche plus petites qui vont produire chacune un sous ensemble du résultat final ; ce au moyen de la fonction **Map**.

L'ensemble des résultats intermédiaires est traité (par agrégation, filtrage), ce au moyen de la fonction **Reduce**.

Le schéma ci-dessous présente le processus d'un traitement MapReduce.

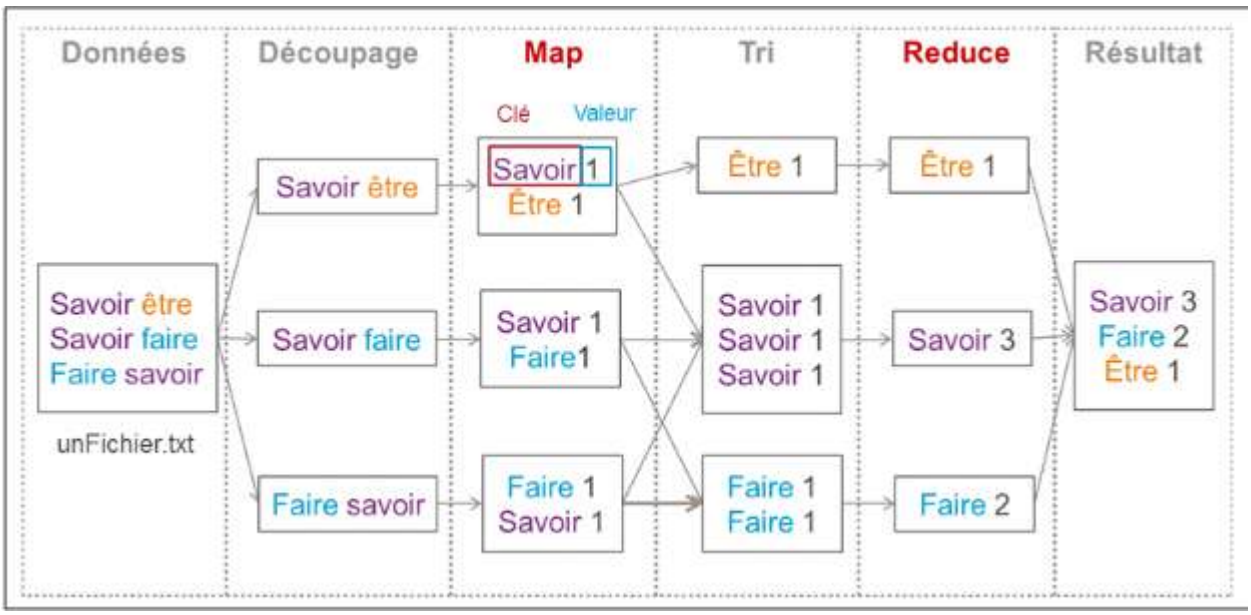


Figure 7:Processus d'un traitement MapReduce, (Bermond, 2013)

Le MapReduce présenté sur le schéma permet de trouver le nombre d'occurrence des mots d'un fichier nommé ici « unFichier.txt ».

Durant la phase de « Découpage », les lignes du fichier sont découpées en blocs.

Puis lors de la phase « Map », des clés sont créées avec une valeur associée. Dans cet exemple une clé est un mot et la valeur est 1 pour signifier que le mot est présent une fois.

Lors du « Tri », toutes les clés identiques sont regroupées, ici ce sont tous les mots identiques.
Ensuite lors de la phase « Reduce » un traitement est réalisé sur toutes les valeurs d'une même clé. Dans cet exemple on additionne les valeurs ce qui permet d'obtenir le nombre d'occurrence des mots.

2.9.4 Modes d'utilisation

Hadoop peut être sous trois modes différents :

- ✓ **Mode Standalone :** ce mode à pour objectif de tester le fonctionnement d'une tâche MapReduce. Ici, la tâche est exécutée sur le poste client dans la seule machine virtuelle Java (JVM), pas besoin d'une configuration particulière car c'est la mode de fonctionnement de base de Hadoop.
- ✓ **Mode Pseudo distributed :** ce mode permettra de tester l'exécution d'une tâche MapReduce sur une seule machine tout en simulant le fonctionnement d'un cluster Hadoop. Je job est exécuté sur la machine et les opérations de stockage et de traitement du job seront gérées par des processus Java différents. L'objectif de ce mode est de tester le bon fonctionnement d'un job sans besoin de mobiliser toutes les ressources du cluster.
- ✓ **Mode Fully distributed :** c'est le mode réel d'exécution d'Hadoop. Il permet de mobiliser le système de fichier distribué et les jobs MapReduce sur un ensemble de machines ; ceci nécessite de disposer de plusieurs postes pour héberger les données et exécuter les tâches.

Dans la suite, d'autres composants qui entrent dans l'écosystème Hahoop sont présentés.

2.10Composants Apache Hadoop

2.10.1 Hbase

Hbase est un système de gestion de bases de données non relationnelles distribuées, écrit en Java, disposant d'un stockage structuré pour les grandes tables. C'est une base de données NoSQL, orientée colonnes. Utilisé conjointement avec HDFS, ce dernier facilite la distribution des données de Hbase sur plusieurs noeuds. Contrairement à HDFS, HBase permet de gérer les accès aléatoires read/write pour des applications de type temps réel.

2.10.2 HaCatalog

HCatalog permet l'interopérabilité d'un cluster de données Hadoop avec d'autres systèmes (Hive, Pig, ...). C'est un service de gestion de tables et de schéma Hadoop. Il permet :

- ✓ D'attaquer les données HDFS via des schémas de type tables de données en lecture/écriture.
- ✓ D'opérer sur des données issues de MapReduce, Pig ou Hive.

2.10.3 Hive

Hive est un outil de requêtage des données, il permet l'exécution de requêtes SQL sur le cluster Hadoop en vue d'analyser et d'agréger les données. Le langage utilisé par Hive est nommé HiveQL. C'est un langage de visualisation uniquement, raison pour laquelle seules les instructions de type « Select » sont supportées pour la manipulation des données.

Hive proposent des fonctions prédéfinies (calcul de la somme, du maximum, de la moyenne), il permet également à l'utilisateur de définir ses propres fonctions qui peuvent être de 3 types :

- ✓ UDF (User Defined Function) : qui prennent une ligne en entrée et retournent une ligne en sortie. Exemple : mettre une chaîne de caractère en minuscule et inversement
- ✓ UDAF (User Defined Aggregate Function) : qui prennent plusieurs lignes en entrée et retournent une ligne en sortie. Exemple : somme, moyenne, max....
- ✓ UDTF (User Defined Table Function) : qui prennent une ligne en entrée et retournent plusieurs lignes en sortie. Exemple : découper une chaîne de caractère en plusieurs mots.

Hive utilise un connecteur jdbc/odbc, ce qui permet de le connecter à des outils de création de rapport comme QlikView.

2.10.4 Pig

Pig est une brique qui permet le requêtage des données Hadoop à partir d'un langage de script (langage qui interprète le code ligne par ligne au lieu de faire une compilation). Pig est basé sur un langage de haut niveau appelé PigLatin. Il transforme étape par étape des flux de données en exécutant des programmes MapReduce successivement ou en utilisant des méthodes prédéfinies du type calcul de la moyenne, de la valeur minimale, ou en permettant à l'utilisateur de définir ses propres méthode appelées User Defined Functions (UDF).

2.10.5 Sqoop

Sqoop est une brique pour l'intégration des données. Il permet le transfert des données entre un cluster et une base de données relationnelles.

2.10.6 Flume

Flume permet la collecte et l'agrégation des fichiers logs, destinés à être stockés et traités par Hadoop. Il s'interface directement avec HDFS au moyen d'une API native.

2.10.7 Oozie

Oozie est utilisé pour gérer et coordonner les tâches de traitement de données à destination de Hadoop. Il supporte des jobs Mapreduce, Pig, Hive, Sqoop, etc.

2.10.8 Zookeeper

Zookeeper est une solution de gestion de cluster Hadoop. Il permet de coordonner les tâches des services d'un cluster Hadoop. Il fournit au composants Hadoop les fonctionnalités de distribution.

2.10.9 Ambari

Ambari est une solution de supervision et d'administration de clusters Hadoop. Il propose un tableau de bord qui permet de visualiser rapidement l'état d'un cluster. Ambari inclut un système de gestion de configuration permettant de déployer des services d'Hadoop ou de son écosystème sur des clusters de machines. Il ne se limite pas à Hadoop mais permet de gérer également tous les outils de l'écosystème.

2.10.10 Mahout

Mahout est un projet de la fondation Apache visant à créer des implémentations d'algorithmes d'apprentissage automatique et de datamining.

2.10.11 Avro

Avro est un format utilisé pour la sérialisation des données.
Le caractère open source de Hadoop a permis à des entreprises de développer leur propre distribution en ajoutant des spécificités.

2.11 Solutions Big Data sur le marché

Sur le marché, on retrouve une panoplie de solutions, chacune avec ses particularités.

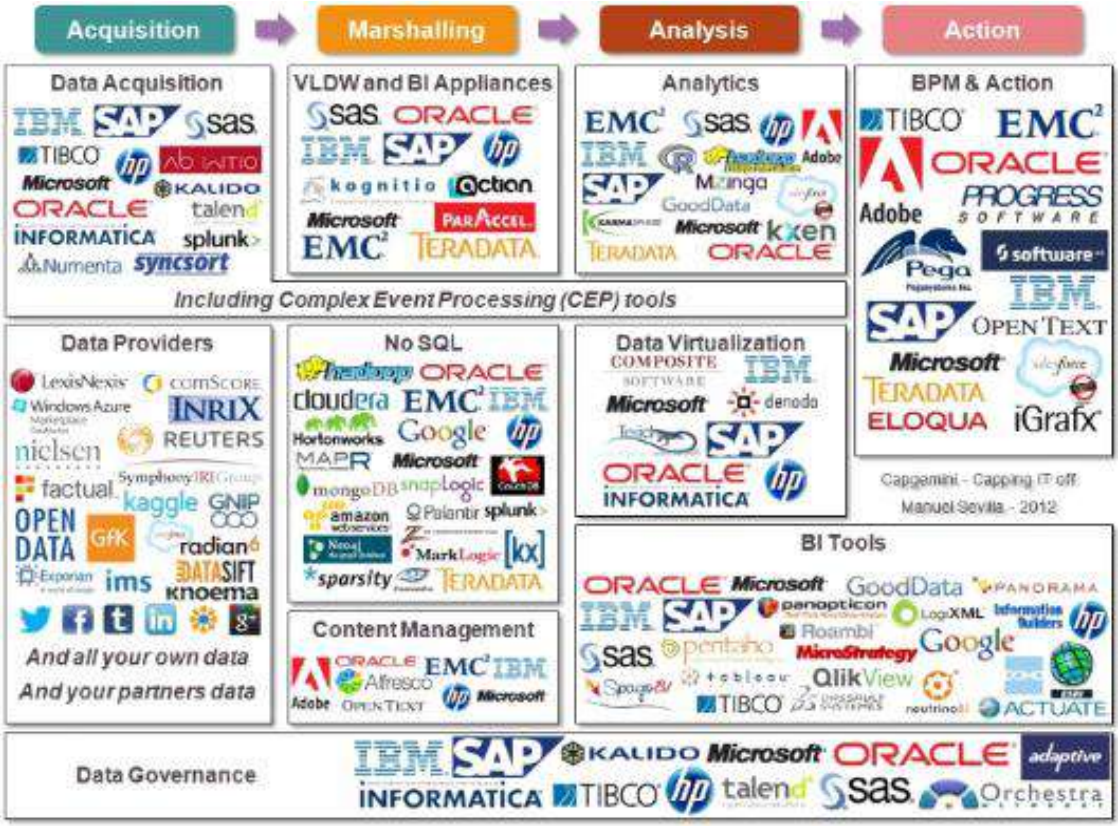


Figure 8:Solutions Big Data sur le marché, (Bermond, 2013)

Parmi cette panoplie, trois se distinguent par le développement d'une distribution Hadoop :

- ✓ **Cloudera :** c'est le leader, ce qui lui donne une légitimité avec un nombre de clients supérieur à celui de ses concurrents. Le fait de disposer du créateur du framework Hadoop dans ces rangs est un grand avantage.
- ✓ **MapR :** cette distribution offre une solution un peu éloignée d'Apache Hadoop car elle intègre sa propre vision de MapReduce et HDFS. Elle vient juste après Cloudera.
- ✓ **Hortonworks :** cette distribution est l'unique plateforme entièrement Hadoop. Sa stratégie est de se baser sur les versions stables de Hadoop plutôt que sur les dernières versions.

2.12 Futur du Big Data

Les technologies du Big Data s'inscrivent dans une évolution continue compte tenu du fait qu'elles sont jeunes et pas encore stables, ce qui leur vaut la réticence des certaines entreprises. Actuellement, le virage technologique est d'ores et déjà annoncé. Le Big Data s'impose tout doucement, mais certains aspects ne sont pas encore à la hauteur des attentes, certaines pistes sont à explorer profondément avant l'intégration dans les systèmes d'information :

- ✓ **La sécurité :** elle est encore balbutiante malgré quelques initiatives comme Apache Knox (système qui fournit un point unique d'authentification et d'accès pour les services Apache Hadoop dans un cluster. Le but est de simplifier la sécurité Hadoop pour les utilisateurs (qui ont accès aux données du cluster et exécutent des jobs) et les opérateurs (qui contrôlent les accès et de gèrent le cluster).
- ✓ **L'intégration avec le système d'information (SI) :** une plate forme Hadoop isolée et non intégrée au système d'information ne sera plus possible dans le futur (en tout cas certains besoins exigeront une interaction plus grande). Cette intégration entraînera une

modification des processus et par conséquent des besoins de formation des ressources humaines.

- ✓ **Les ressources compétentes :** actuellement les compétences ne sont pas encore assez poussées dans le domaine
- ✓ **Protection de la vie privée :** la manipulation à grande échelle de des données pose aussi le problème de la vie privée. Trouver l'équilibre entre le respect de son intimité et les bénéfices tirés du big data n'est pas simple. Les utilisateurs des réseaux sociaux ignorent souvent que leurs données privées sont accessibles par un grand public, beaucoup reste à faire afin de garantir la protection des utilisateurs.

« L'avenir appartiendra à ceux qui seront capable d'analyser les vastes volumes de données qu'ils ont collectés ».

3 - Le NoSQL, une nouvelle approche de stockage et de manipulation de données.

3.1 Introduction

Avant de présenter le concept NoSQL comme une nouvelle approche de stockage et de manipulation de données, commençons par dire ce que le NoSQL n'est pas. Ce n'est pas du SQL et ce n'est pas non plus du relationnel. Comme son nom l'indique, ce n'est pas un remplacent des SGBDR, mais il vient en complément de ces derniers. Le NoSQL est conçu pour les magasins de données distribuées et requérant des exigences de performance à grande échelle. Pensez à Facebook avec ses 800 millions d'utilisateurs selon Citizen Adhésions en 2010 ou Twitter, qui à eux seuls stockent des térabits de données chaque jour. Depuis quelques années, de nouvelles approches de stockage et de gestion des données sont apparues, et permettent de se libérer (se détacher ou se démarquer) de certaines contraintes, en particulier de scalabilité, inhérentes au paradigme relationnel. Regroupées derrière le vocable NoSQL, ces technologies auraient très bien pu être nommées "NoRel" comme le suggère l'inventeur du terme NoSQL, Carl Strozzi. Plus que des technologies de remplacement intégral des outils relationnels, le NoSQL correspond le plus souvent à des approches qui complètent les outils existants, pour en combler les faiblesses. Ainsi, on observe de plus en plus souvent la traduction "Not Only SQL" au lieu de "NoSQL"[1]. Ces technologies, portées par des acteurs majeurs du web comme Google, Facebook, Twitter ont rapidement acquis une légitimité réelle dans le domaine de la manipulation de volumétries de données très importantes et ont rapidement gagné tant en maturité qu'en notoriété. Le NoSQL est un type de magasin de données, c'est une manière de stocker et de récupérer des données de façon rapide, un peu comme une base de données relationnelle, sauf qu'il n'est pas basé sur une relation mathématique entre les tables comme une base de données relationnelle traditionnelle le fait. Un magasin de données NoSQL peut être beaucoup plus rapide que les bases de données relationnelles, mais n'a généralement pas la robustesse que l'on connait aux bases de données classiques. La croissance des grands du Web comme Facebook et Google a conduit à l'élaboration du NoSQL comme un moyen de briser les contraintes de vitesse que fournissent des bases de données relationnelles. On parle donc de la mouvance NoSQL comme étant celle des bases de données de prochaine génération portant pour la plupart sur le fait d'être non-relationnelle, distribuée, open-source et horizontalement évolutive. Le mouvement a commencé au début de 2009 et se développe rapidement . Ce nouveau concept repose sur les caractéristiques tels que: la liberté du schéma de données, un support de réplication facile, une API simple, finalement une gestion cohérente, une capacité de gestion d'énorme quantité de données et plus encore.

3.2 La technologie NoSQL

NoSQL ou « Not Only SQL » est un mouvement très récent (2009), qui concerne les bases de données. L'idée du mouvement est simple : proposer des alternatives aux bases de données relationnelles pour coller aux nouvelles tendances et architectures du moment, notamment le Cloud Computing [W14]. Les axes principaux du NoSQL sont une haute disponibilité et un partitionnement horizontal des données, au détriment de la consistance. Alors que les bases de données relationnelles actuelles sont basées sur les propriétés ACID (Atomicité, Consistance, Isolation et Durabilité). NoSQL signifie "Not Only SQL", littéralement "pas seulement SQL". Ce terme désigne l'ensemble des bases de données qui s'opposent à la notion relationnelle des SGBDR. La définition, "pas seulement SQL", apporte un début de réponse à la question "Est ce que le NoSQL va tuer les bases relationnelles?".

En effet, NoSQL ne vient pas remplacer les BD relationnelles mais proposer une alternative ou compléter les fonctionnalités des SGBDR pour donner des solutions plus intéressantes dans certains contextes. Le NoSQL regroupe de nombreuses bases de données, récentes pour la plupart, qui se différencient du modèle SQL par une logique de représentation de données non relationnelle. Leurs principaux avantages sont leurs performances et leur capacité à traiter de très grands volumes de données. En revanche, dans les projets, il ne faut pas opposer ces deux approches mais bien souvent les faire cohabiter ! Cette technologie (le NoSQL) ne vise finalement pas à remplacer les SGBD traditionnels mais plutôt à les compléter en déportant une partie de la charge des traitements et de stockage de données vers des serveurs-tiers (dans les architectures web classiques).

3.3 Les concepts forts du NoSQL

Les bases de données NoSQL reposent essentiellement sur plusieurs aspects qui font leurs forces et justifient leur usage aujourd'hui des géants du web. Il s'agit principalement du partitionnement horizontal des données, du fait que ces bases sont sans schéma et donc une grande flexibilité du schéma de données.

3.3.1 Scalabilité maîtrisée à travers le partitionnement horizontal

Ces bases de données proposent une nouvelle représentation de l'information. En s'affranchissant des contraintes ACID du modèle SQL, elles ont le très gros avantage de fournir une architecture technique où il suffit de rajouter des serveurs pour gagner en performance sans trop se poser de questions. Cette technique consiste lorsque la charge des traitements ou des données devient très importante au niveau d'un ou de plusieurs serveurs à ajouter un ou plusieurs serveurs qui se partagent les données et les traitements : c'est ce que nous avons évoqué plus haut sous les termes de scaling de données et de scaling de traitements [W4]. Cette approche de stockage permet d'avoir des bases de données performantes et une disponibilité que les SGBD classiques ne peuvent égaler même en multipliant les serveurs miroirs. Un SGBDR pour répondre aux exigences de performance face aux gros volumes de données, doit se retourner vers du matériel de plus en plus rapide et à l'ajout de mémoire. Le NoSQL, pour sa part, pour gérer la "montée en charge" se réfère à la répartition de la charge sur les systèmes de Cloud Computing. Il s'agit la de composant de NoSQL qui fait d'elle une solution peu coûteuse pour les grands ensembles de données.

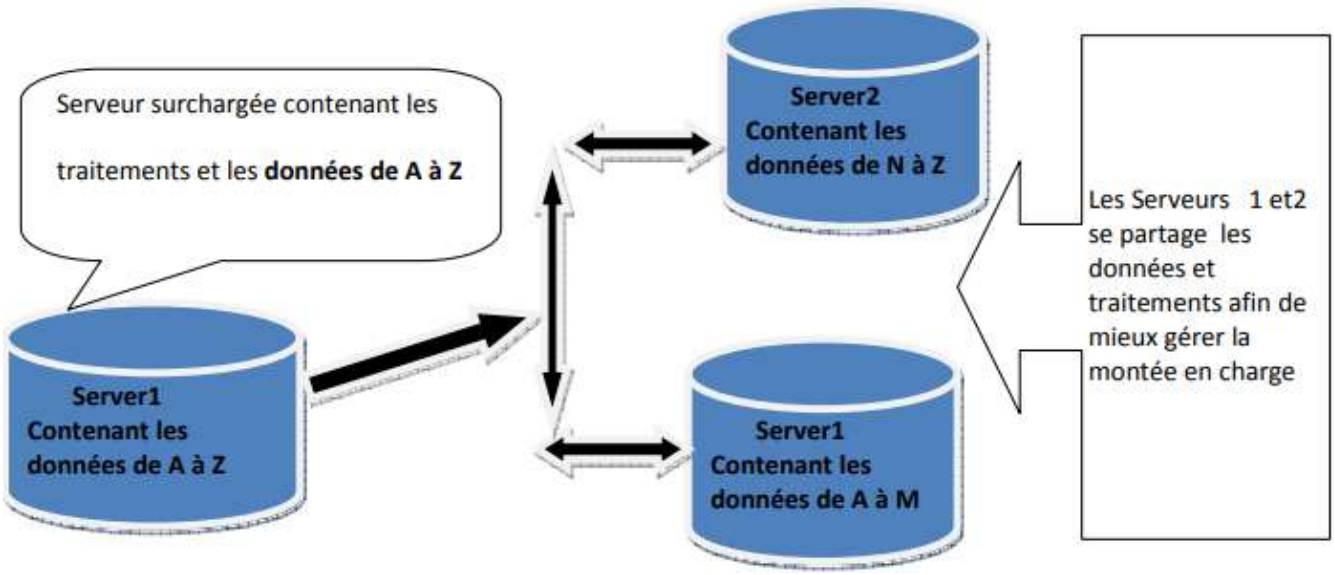


Figure 9:Partitionnement Horizontal de données

Bien que le réflexe premier soit de penser aux performances des bases NoSQL sur des gros volumes de données ou des données faiblement structurées, certaines utilisations peuvent se justifier dans

des environnements de plus faible volume ou de plus faible charge. Par exemple, les bases orientées colonne faciliteront l'évolution du "schéma" des données et donc vous orienteront vers une maintenance plus intelligente, plus agile et donc plus durable de vos applications. Dans l'approche orienté colonne, les colonnes ne sont pas connues et déclarées d'avance comme dans les bases de données relationnelles. Elles sont crées dynamiquement pendant l'insertion des données dans la base. De plus les champs de valeur NULL n'existent pas car la déclaration d'une colonne ne concerne pas toutes les lignes d'une table comme c'est le cas dans le monde du relationnel. Avec le NoSQL, chaque ligne ou entrée d'une table (respectivement famille de colonne en NoSQL) a ses propres colonnes. Dans une base de données NoSQL, il n'y a pas de schéma fixe. Toutes les données susceptibles d'être lues ou modifiées au même moment sont en général regroupées au sein d'une même famille de colonnes. Une des conséquences immédiate de cette évolutivité des schémas de données est la gestion des données faiblement structurées.

Meilleur gestion des données faiblement structurées

A part être une nouvelle technologie, le NoSQL permet de stocker les informations d'une manière qui colle mieux à leur représentation. Exemple :

- ✓ **les bases de données orientées document** : s'adaptent au stockage des données non planes (type profil utilisateur).
- ✓ **les bases de données orientées colonne :** s'adaptent très bien au stockage de listes (messages, posts, commentaires, etc...).
- ✓ **les bases de données orientées graphe** : permettent de mieux gérer des relations multiples entre les objets (comme pour les relations dans les réseaux sociaux).

La première rupture est inhérente au modèle de données non relationnel qui ne dispose pas deméthodes effi caces permettant d'effectuer des opérations de jointure. Dans le modèle relationnel, la pratique courante est d'utiliser des données sous forme normalisée et de procéder à des jointures entre les tables. Sur le sujet de la normalisation, les théoriciens de la mouvance NoSQL prônent une approche pragmatique guidé par l'utilisation qui est faite des données :

- ✓ Dans certains cas les données seront embarquées dans la donnée parente (la manière de procéder dépend du paradigme utilisé). C'est notamment le cas lorsque le volume de données est faible et que celle-ci est peu mise à jour. Il se pose alors la problématique de la mise à jour des données.
- ✓ Dans les autres cas, on stockera simplement les identifiants permettant d'accéder aux données comme on le ferait dans le modèle relationnel. La différence réside dans le fait que c'est le plus souvent à l'application de procéder par la suite aux jointures.

L'engouement pour les bases NoSQL est la conclusion logique de l'ère Web 2.0. Les données stockées sont aujourd'hui beaucoup plus importantes qu'elles n'ont pu l'être avant, et les besoins ont changé. On a aujourd'hui besoin de pouvoir stocker ou lire très rapidement des millions de données (prenez l'exemple de Facebook par exemple) mais aussi, avec le Cloud, d'avoir des systèmes "scalable" ou élastiques. Pour répondre à cette demande, de nombreux projets de bases NoSQL sont apparus. On distingue deux publications qui ont servi de point de départ à certains de ces projets :

- ✓ BigTable de Google sur lequel est basé Hadoop/HBase ;
- ✓ Dynamo d'Amazon sur lequel repose Cassandra initié par Facebook.

On dénombre actuellement une douzaine de solutions existantes, selon différents types de bases de données à savoir :

- ✓ Les bases clé-valeur, comme Redis, Riak, Voldemort
- ✓ Les bases documentaires, comme Mongo DB, CouchDB, Terra store
- ✓ Les bases orientées comme cassandra, Amazon, simple DB, Google Bigtable, Hbase.

- ✓ Les bases orientées graphe, comme Neo4j, Orient DB [1,21]

3.4 Les Enjeux des bases NoSQL

Le premier besoin fondamental auquel répond NoSQL est la performance. En effet, ces dernières années, les géants du Web comme Google et Amazon ont vu leurs besoins en termes de charge et de volumétrie de données croître de façon exponentielle. Et c'est pour répondre à ces besoins que ces solutions ont vu le jour.

Les architectes de ces organisations ont procédé à des compromis sur le caractère ACID des SGBDR. Ces intelligents compromis sur la notion du relationnel ont permis de dégager les SGBDR de leurs freins à la scalabilité horizontale et à l'évolutivité. Par la suite des entreprises comme Facebook, Twitter ou encore LinkedIn ont migré une partie de leurs données sur des bases NoSQL. Cette adoption croissante des bases NoSQL conduit à une multiplication et une amélioration des offres Open Source des moteurs. Par ailleurs, l'idée est que les deux technologies (relationnel et non relationnel) peuvent coexister, chacun ayant sa place. Le mouvement NoSQL a été à la une ces dernières années, car la plupart des leaders du Web 2.0 l'ont adopté. Des sociétés comme Facebook, Twitter, Digg, Amazon, LinkedIn et Google l'utilisent déjà avec beaucoup de succès. Mais, seulement il reste, pour cette nouvelle approche de stockage et de manipulation des données, de tenir le pari des enjeux comme la gestion des données de plus en plus complexes et l'augmentation de plus en plus croissante des données ainsi que leur distribution à travers le monde. S'agissant **du Stockage de données**, le stockage mondial des données numériques est mesuré en exaoctets; un exaoctet étant égal à un milliard de giga-octets (Go) de données. Selon Internet.com, la quantité de données stockées en 2006 était en augmentation de 161 exaoctets [9]. Seulement 4 ans plus tard, en 2010, la quantité de données stockées était de près de 1.000 exaoctets ce qui représente une augmentation de plus de 500% [9]. En d'autres termes, il y a beaucoup de données stockées dans le monde et cela va continuer à croître. Par rapport aux **données interconnectées**, les données continuent d'être de plus en plus plus partagées et donc distribuées. Les principaux systèmes sont conçus pour être reliés entre eux et proposent aux utilisateurs des hyperliens, des blogs, des wiki et le partage des documents. S'agissant des structures de données, **NoSQL** peut gérer des structures de données hiérarchiques imbriquées facilement. Pour accomplir la même chose en SQL, vous avez besoin de plusieurs tables relationnelles avec toutes sortes de clés. En outre, il existe une relation entre la performance et la complexité des données. Les performances peuvent se dégrader dans un SGBDR traditionnel lorsque nous stockons des quantités massives de données requises dans les applications de réseautage social et de web sémantique. Nous l'avons vu, les SGDB relationnels disposent d'atouts indéniables expliquant leur popularité et leur large utilisation, mais aussi des limites importantes qui ont conduit à de nouvelles approches. Tout comme les SGBD relationnels ne sont pas la réponse unique aux problèmes de base de données, on ne doit pas considérer qu'un seul autre outil sera en mesure d'apporter une solution à caractère universel. Il existe dans la mouvance NoSQL, une diversité importante d'approches classées en quatre grandes catégories à savoir : le paradigme clé-valeur, orienté colonnes, orienté documents et orienté graphes.

3.5 Les types de bases de données NoSQL

Il existe différents types de bases de données NoSQL spécifiques à différents besoins, en voici quelques uns avec le nom des solutions ou moteurs associés :

- ✓ Orientées colonnes : HBase, Hyper table, Cassandra et BigTable
- ✓ Orientées graphes (Euler) : Neo4J
- ✓ Orientées clé-valeur : Voldemort , Dynamite et Riak
- ✓ Orientées document : CouchDB

Passons à présent à la description des différents types de base de données NoSQL.

3.5.1 Base de données Orientée Clé- valeur

Cette structure est très adaptée à la gestion des caches ou pour fournir un accès rapide aux informations. Elle fonctionne comme un grand tableau associatif et retourne une valeur dont elle ne connaît pas la structure. Il s'agit de la catégorie de bases de données la plus basique [1,13]. Dans ce modèle, chaque objet est identifié par une clé unique qui constitue la seule manière de le requêter.

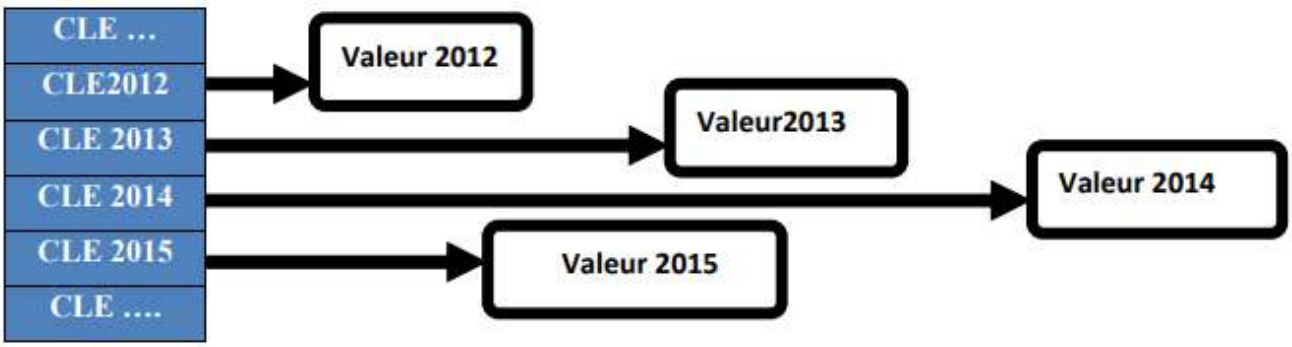


Figure 10:Illustration d'une Base de données Orientées Clef- valeur

La structure de l'objet est libre et le plus souvent laissé à la charge du développeur de l'applicatio n (par exemple XML, JSON, ...), la base ne gérant généralement que des chaines d'octets [1]. Ce modèle peut être assimilé à une hashmap distribuée. Les données sont donc simplement représentées par un couple clé-valeur. La valeur peut être une simple chaine de caractères ou un objet sérialisé [W14]. Cette absence de structure ou de typage a un impact important sur le requêtage. En effet, toute l'intelligence portée auparavant par les requêtes SQL devra être portée par l'applicatif qui interroge la BD. Néanmoins, la communication avec la BD se résumera aux opérations PUT, GET et DELETE. Les solutions les plus connues sont Redis, Riak et Voldemort créé par LinkedIn.

3.5.2 Base de données Orientée Document

Ce modèle ajoute au modèle Clé-valeur, l'association d'une valeur à structure non plane, c'est-à-dire qui nécessiterait un ensemble de jointures en logique relationnelle. Les bases de données documentaires sont constituées de collections de documents. Un document est composé de champs et des valeurs associées, ces dernières pouvant être requêtées. Par ailleurs, les valeurs peuvent être, soit d'un type simple (entier, chaine de caractère, date…), soit elles-mêmes composées de plusieurs Couples clé-valeur. Bien que les documents soient structurés, ces bases sont dites "schemaless". A ce titre, il n'est pas nécessaire de définir au préalable les champs utilisés dans un document. Les documents peuvent être très hétérogènes au sein de labase. Le stockage structuré des documents leur confère des fonctionnalités dont ne disposent pas les bases clés-valeurs simples dont la plus évidente est la capacité à effectuer des requêtes sur le contenu des objets. Ce modèle se base sur le paradigme clé-valeur.

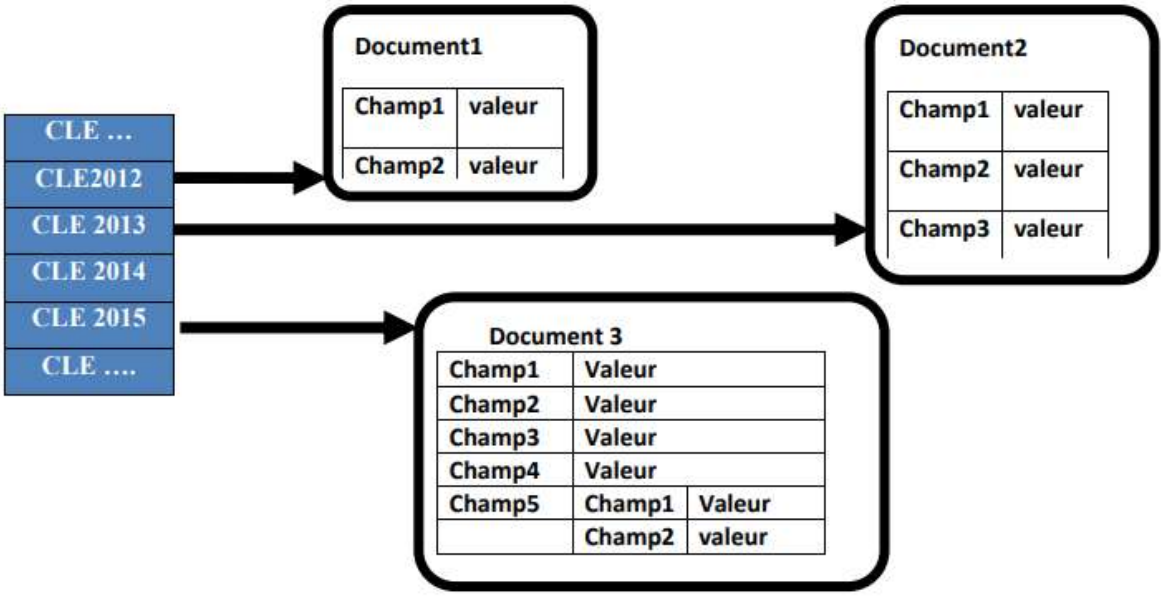


Figure 11:Illustration d'une Base de données Orientées Document

La valeur, dans ce cas, est un document de type JSON ou XML par exemple . L'avantage est de pouvoir récupérer, via une seule clé, un ensemble d'informations structurées de manière hiérarchique. La même opération dans le monde relationnel impliquerait plusieurs jointures. Pour ce modèle, les implémentations les plus populaires sont CouchDB d'Apache, RavenDB (destiné aux plateformes .NET /Windows avec la possibilité d'interrogation via LINQ) et Mongo DB.

3.5.3 Base de données Orientée Colonne

Ce modèle ressemble à première vue à une table dans un SGBDR à la différence qu'avec une BD NoSQL orientée colonne, le nombre de colonnes est dynamique. En effet, dans une table relationnelle, le nombre de colonnes est fixé dés la création du schéma de la table et ce nombre reste le même pour tous les enregistrements dans cette table. Par contre, avec ce modèle, le nombre de colonnes peut varier d'un enregistrement à un autre ce qui évite de retrouver des colonnes ayant des valeurs NULL [W14]. Comme solutions, on retrouve principalement HBase (implémentation Open Source du modèle Bigtable publié par Google) ainsi que Cassandra (projet Apache qui respecte l'architecture distribuée de Dynamo d'Amazon et le modèle Bigtable de Google). Les bases de données orientées colonnes sont résolument les plus complexes à appréhender parmi la mouvance NoSQL. Bien que l'on obtienne au final un schéma relati- vement proche des bases documentaires, l'organisation sous-jacente des données permet à ces bases d'exceller dans les traitements d'analyse de données et dans les traitements massifs (notamment via des opérations de type MapReduce). Bien que l'on trouve des variations en fonction de la base considérée, les concepts essentiels sont les suivants :

- ✓ **La colonne:** il s'agit de l'entité de base représentant un champ de donnée. Chaque colonne est définie par un couple clé-valeur. Une colonne contenant d'autres colonnes est nommée super colonne .

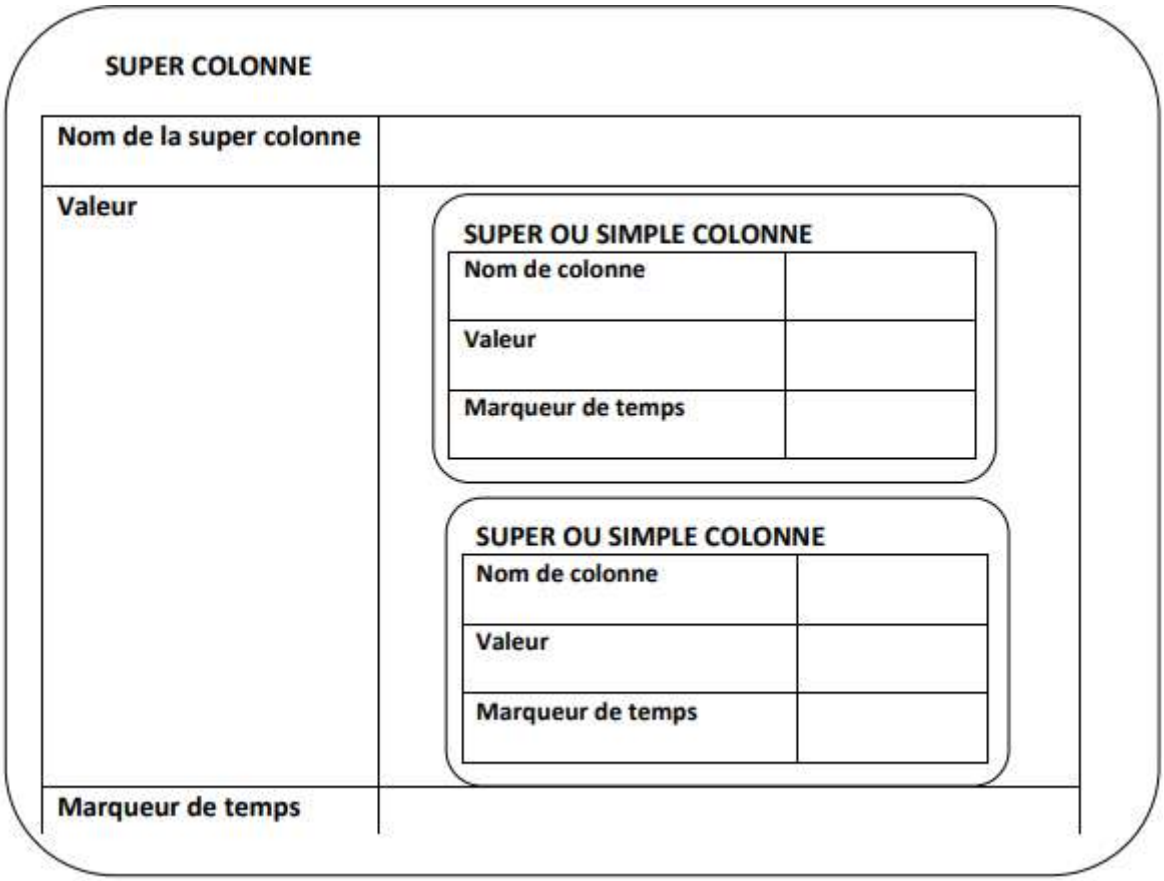


Figure 12:Illustration d'une colonne dans une BDNOC

- ✓ Famille de colonnes : il s'agit d'un conteneur permettant de regrouper plusieurs colonnes ou super colonnes. Les colonnes sont regroupées par ligne et chaque ligne est identifiée par un identifiant unique. Elles sont généralement assimilées aux tables dans le modèle relationnel et sont identifiées par un nom unique. Les super colonnes situées dans les familles de colonnes sont souvent utilisées comme les lignes d'une table de jointure dans le modèle relationnel. Comparativement au modèle relationnel, les bases orientées colonnes offrent plus de flexibilité. Il est en effet possible d'ajouter une colonne ou super colonne à n'importe quelle ligne d'une famille de colonnes à tout instant. Il s'agit d'une autre évolution du modèle Clé valeur, il permet de disposer d'un très grand nombre de valeurs sur une même ligne, permettant ainsi de stocker les relations de type one-to-many. Contrairement au système Clé valeur, celui-ci permet d'effectuer des requêtes par clé.

3.5.4 Base de Données Orientée Graphe

Ce modèle de représentation des données se base sur la théorie des graphes. Il s'appuie sur la notion de nœuds, de relations et de propriétés qui leur sont rattachées. Ce modèle facilite la représentation du monde réel, ce qui le rend adapté au traitement des données des réseaux sociaux. La principale solution est Neo4J. Ce modèle permet la modélisation, le stockage et la manipulation de données complexes liées par des relations non-triviales ou variables. Ce modèle s'appuie principalement sur deux concepts : d'une part l'utilisation d'un moteur de stockage pour les objets (qui se présentent sous la forme d'une base documenta ire, chaque entité de cette base étant nommée nœud). D'autre part, à ce modèle, vient s'ajouter un mécanis me permettant de décrire les arcs (relations entre les objets), ceux-ci étant orientés et disposant des propriétés (nom, date, …).

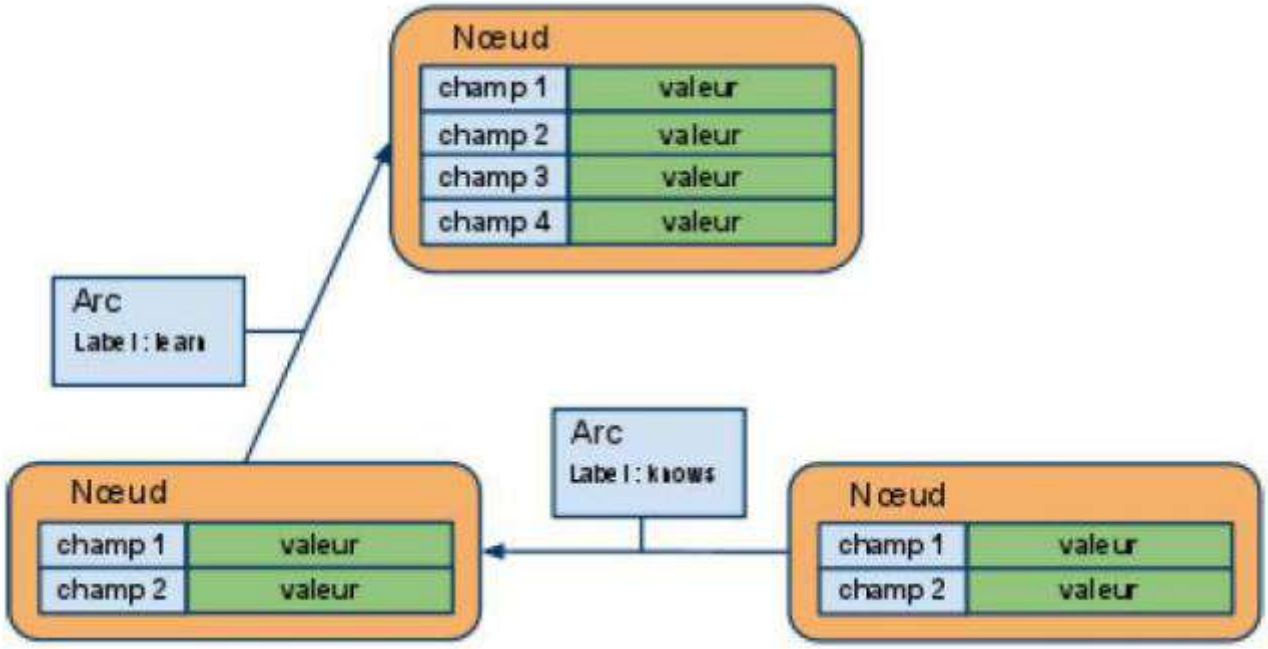


Figure 13:Illustration d'une Base de données orientée Graphe

Par essence ces bases de données sont nettement plus efficaces que leur concurrent relationnel pour traiter les problématiques liées aux réseaux (cartographie, relations entre personnes). En effet, lorsqu'on utilise le modèle relationnel, cela nécessite un grand nombre d'opérations complexes pour obtenir des résultats. Pour le moment, le NoSQL est un mouvement grandissant, bien qu'il soit utilisé par de grandes firmes qui sont à l'origine du mouvement. Il n'y a pas de solutions qui se démarquent vraiment du lot, il n'y a que des solutions adaptées aux besoins. On remarquera aussi qu'il manque vraiment de standards, comme JDBC et SQL le sont pour les SGBDR, mais cela s'explique sans doute par la jeunesse du mouvement et l'hétérogénéité des bases de données. Dans la section suivante, nous allons nous intéresser sur le mode d'interrogation des bases de données de type NoSQL après qu'elles aient été présentées selon différentes typologies.

Le requêtage NoSQL

Dans le monde du NoSQL il n'y a pas de langage standard comme SQL l'est dans le monde des bases de données relationnelles. L'interrogation des bases de données NoSQL se fait au niveau applicatif à travers principalement la technique dite de « **MapReduce** ». MapReduce est une technique de programmation distribuée très utilisée dans le milieu NoSQL et qui vise à produire des requêtes distribuées. Cette techniq ue se décompose en deux grandes étapes :

Etape de mapping : Chaque item d'une liste d'objets clé-valeur est passé à la fonction map qui va retourner un nouvel élément clé-valeur. Exemples de fonction map. A chaque couple (UserId, User), on assigne le couple (Role, User). A l'issue de l'étape de mapping, on obtient une liste contenant les utilisateurs groupés par rôle. A un couple (UserId, User), on assigne le couple (UserId, User) uniquement si l'email de l'utilisateur se termine par ".fr"

Etape de Reduce : La fonction reduce est appelée sur le résultat de l'étape de mapping et permet d'appliquer une opération sur la liste .Exemples de fonction reduce :

- ✓ Moyenne des valeurs contenues dans la liste
- ✓ Comptabilisation des différentes clés de la liste
- ✓ Comptabilisation du nombre d'entrées par clé dans la liste

L'étape de mapping peut être parallélisée en traitant l'application sur différents nœuds du système po ur chaque couple clé-valeur. L'étape de réduction n'est pas parallélisée et ne peut être exécutée avant la fin de l'étape de mapping. Les bases de données NoSQL proposent diverses implémentations de la techni que MapReduce permettant le plus souvent de développer les méthodes **map** et **reduce** en JavaScript ou en Java.

3.6 Les défis majeurs du NoSQL

Les organisations qui ont d'énormes besoins de stockage de données sont sérieusement à la recherche du NoSQL. Apparemment, le concept ne reçoit pas que beaucoup de sympathie auprès des petites organisations. Dans une enquête menée par Information Week, 44% des professionnels en activité de l'informatique n'ont pas entendu parler de NoSQL. En outre, seulement 1% des répondants ont indiqué que NoSQL est une partie de leur orientation stratégique. De toute évidence, NoSQL a sa place dans notre monde connecté, mais devra continuer à évoluer pour obtenir l'appel de masse que beaucoup pensent qu'elle pourrait avoir. La technologie NoSQL ne cesse de faire parler d'elle et semble avoir le vent en poupe. Attrayante, la barrière d'entrée pour un nouveau développement est d'ailleurs assez peu élevée pour tout développeur ayant bien compris les sous-jacents de la solution retenue. Néanmoins, il est essentiel de garder à l'esprit que NoSQL apporte une réponse à des besoins bien spécifiques. Dit autrement, il est nécessaire d'avoir identifier au préalable la nécessité d'utiliser cette technologie dans vos services et pas uniquement avec une motivation : « et si on a autant de succès demain que Twitter ».Cependant, n'oubliez pas que, bien que robuste, cette technologie reste jeune et doit par conséquent encore évoluer : en se dotant par exemple de solutions ORM éprouvées, en gommant l'absence d'un langage de requêtage commun et capitaliser sur l'utilisation, comme nos chers SGBDR l'ont fait sur les 20 dernières années[W1]. Les bases de données NoSQL ont suscité beaucoup d'enthousiasme, mais il y'a de nombreux obstacles à surmonter avant de pouvoir faire appel aux principaux acteurs de l'industrie des bases de données. Voici quelques uns des principaux défis.

3.6.1 Maturité

Les outils permettant l'exploitation de ce type de bases de données sont encore très peu matures tout au moins pour la plupart. Si l'on se place au niveau des outils de développements, ils n'intègrent pas encore des Framework ou plugin compatibles au NoSQL. Pour ce qui est des outils d'administration (outils de sauvegarde, monitoring, …), quand ils existent, ils sont spécifiques à des moteurs NoSQL particuliers .C'est le cas de Google qui a mis sur pieds son propre outil d'administration de bases de données orientées colonne. Aussi, les bases de données de type NoSQL étant des technologies encore très récentes, il n'y a pas encore de normes qui permettent de définir une architecture type pour tel ou tel type de base de données, ni de syntaxe particulière ; bien que le mouvement tende vers une convergence pour des requêtes basées sur le langage JavaScript . Il manque vraiment des standards, comme JDBC et SQL le sont pour les SGBDR, mais cela s'explique sans doute par la jeunesse du mouvement et l'hétérogénéité des types de bases.

3.6.2 Assistance et maintenance

Les entreprises veulent avoir l'assurance que si un système de clés échoue, les fournisseurs de la base NoSQL seront en mesure d'offrir un soutien rapide et compétent .Tous les fournisseurs de SGBDR font de grands efforts pour fournir un niveau élevé de soutien aux entreprises. En revanche, la plus part des systèmes NoSQL sont des projets open source, et bien qu'il existe généralement une ou plusieurs entreprises qui offrent un soutien pour chaque base de données NoSQL, ces entreprises sont souvent des startups sans portée mondiale et n'ont pas les ressources de soutien, ou la crédibilité d'un Oracle, de Microsoft, ou IBM.

3.6.3 Outils d'analyse et administration

Les bases de données NoSQL offrent peu d'installations pour des requêtes ad-hoc et d'analyse de données. Même une simple requête nécessite une expertise significative de programmation. Couramment utilisés, les outils de BI ne fournissent pas de connectivité aux bases de données NoSQL. Certains de secours sont fournis par l'émergence de solutions telles que le HIVE ou PIG, qui peuvent fournir un accès plus facile aux données détenues dans les clusters Hadoop et peut-être à terme, d'autres bases de données NoSQL. Quest Software a développé un produit dénommé « Toad » pour les bases de données qui peut fournir des capacités de requête ad-hoc à une variété de bases de données NoSQL. L'objectif de la conception des bases de données NoSQL était de fournir une solution zéro-administration, mais la réalité actuelle est bien loin de cet objectif [W17]. Pour administrer une base NoSQL aujourd'hui, il faut beaucoup d'habileté non seulement pour l'installer et beaucoup d'efforts pour la maintenir.

3.6.4 Expertise

Il y a littéralement des millions de développeurs à travers le monde, et dans chaque secteur d'activité, qui sont familiers avec les concepts de programmation et de SGBDR. En revanche, presque tous les développeurs NoSQL sont encore dans un mode d'apprentissage. Cette situation se penchera naturellement au fil du temps, mais pour l'instant, c'est beaucoup plus facile de trouver des programmeurs expérimentés SGBDR ou les administrateurs que par un expert NoSQL.

3.7 base de données MongoBD :

MongoDB est une base de données NoSQL orientée documents. Comme nous le verrons, l'ensemble du système tourne autour de cette gestion de documents, y compris le langage d'interrogation, ce qui en fait son point fort. Nous allons nous attaquer dès maintenant à la mise en place d'un serveur Mongo et comment intégrer vos données dans cet environnement.

Figure 14:Logo de base de données MongoDB

3.7.1 L' architecture de ReplicaSet

Nous avons pu voir comment interroger des données, mais nous n'avons pas encore attaqué le problème de la tolérance aux pannes. MongoDB utilise pour cela une architecture basée sur le principe « maître/esclave .

Le serveur primaire *"Primary",* à qui toutes les requêtes sont envoyées (lecture/écriture), va s'occuper de gérer la cohérence des données. Ainsi, lors d'une mise à jour, une réplication est effectuée sur les serveurs secondaires "*Secondary*".

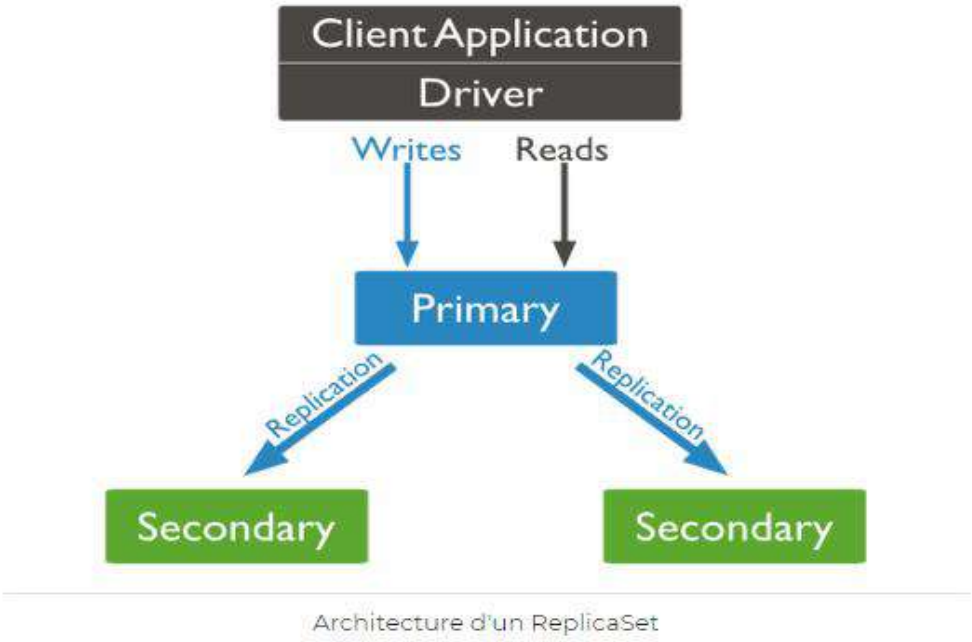


Figure 15:Architecture d'un ReplocaSet

Toutefois, si l'on reste sur une architecture de la sorte, nous ne sommes pas tolérant aux pannes puisque le serveur primaire reste un point critique. Pour cela, nous allons rajouter un arbitre "*arbiter*", dont la tâche est de vérifier l'état du réseau. Ainsi, dès que le primary tombe, un système de vote va permettre d'élire le secondary qui remplacera le serveur tombé en panne. Nous sommes donc tranquilles : cette architecture tient tant qu'au moins un serveur tourne. C'est ce que l'on appelle un **ReplicaSet**.

Un *ReplicaSet* doit contenir au minimum 3 serveurs (1 Primary et 2 Secondary) pour garantir un minimum de tolérance aux pannes. Un ReplicaSet peut contenir jusqu'à 50 serveurs ; toutefois lorsqu'un vote a lieu pour élire le nouveau Primary, au maximum 7 peuvent participer à cette élection (les premiers ayant répondu à l'Arbiter).

3.8 MongoDB et Hadoop

De nombreuses organisations exploitent ensemble les capacités de Hadoop et de MongoDB pour créer des applications Big Data complètes:

- ✓ MongoDB alimente l'application opérationnelle en ligne et en temps réel, au service des processus métier et des utilisateurs finaux, en exposant les modèles d'analyse créés par Hadoop aux processus opérationnels.
- ✓ Hadoop utilise les données de MongoDB et les mélange avec des données d'autres sources pour générer des analyses sophistiquées et des modèles d'apprentissage automatique. Les résultats sont transférés à MongoDB pour servir des processus opérationnels plus intelligents et sensibles au contexte - c'est-à-dire proposer des offres plus pertinentes, identifier plus rapidement les fraudes, mieux prédire les taux d'échec des processus de fabrication.

3.8.1 Comment les organisations utilisent MongoDB avec Hadoop

Les utilisateurs doivent mettre les résultats analytiques de Hadoop à la disposition de leurs applications opérationnelles en ligne. Ces applications ont des demandes d'accès spécifiques auxquelles HDFS ne peut satisfaire, notamment:

- ✓ Réactivité des requêtes en latence milliseconde.
- ✓ Accès aléatoire à des sous-ensembles de données indexés.
- ✓ Prise en charge en temps réel de requêtes ad hoc et d'agrégations expressives sur les données, rendant les applications en ligne plus intelligentes et contextuelles.
- ✓ Mise à jour en temps réel de données qui changent rapidement lorsque les utilisateurs interagissent avec des applications en ligne, sans avoir à réécrire l'intégralité du jeu de données. La gestion des analyses en temps réel depuis Hadoop jusqu'aux applications et aux utilisateurs en ligne nécessite l'intégration d'une couche de base de données opérationnelle hautement évolutive et extrêmement flexible.
- ✓ Le tableau suivant fournit des exemples de clients utilisant MongoDB avec Hadoop pour alimenter des applications Big Data.

	MongoDB	Hadoop
Ebay	Gestion des données utilisateur et des métadonnées pour le catalogue de produits	Analyse utilisateur pour recherche personnalisée et recommandations
Orbitz	Gestion des données et des prix de l'hôtel	Segmentation des hôtels pour prendre en charge la recherche de bâtiments
Pearson	Identité de l'étudiant et contrôle d'accès. Gestion du contenu du matériel de cours	Analyse des étudiants pour créer des programmes d'apprentissage adaptatifs
Foursquare	Données utilisateur, enregistrements, critiques, gestion du contenu des sites	Analyse utilisateur, segmentation et personnalisation

Figure 16:Tableau des société utilisé MongoDB & Hadoop

3.9 Bilan du chapitre

Les bases de données NoSQL ont souvent vu le jour en tant que projets internes des grands sites de commerce, ou de recherche. Ainsi, BigTable, dont le développement à commencé en 2004, gère la plupart des données de Google, une référence presque absolue en termes de volume et de performances. Plusieurs grands sites utilisent ces outils particuliers : Cassandra chez Facebook et Digg, Dynamo chez Amazon, PNUTS chez Yahoo, etc. Et l'orientation actuelle vers le Cloud Computing ne fait que renforcer l'intérêt pour ces approches, comme en témoignent les services de stockage de la plateforme Azure de Microsoft .Dans la foulée de ces solutions propriétaires, de nombreux projets libres ont vu le jour. Même si une grande diversité règne, plusieurs caractéristiques fréquentes semblent émerger : absence de schéma, partitionnement horizontal sur

un grand nombre de nœuds, dé-normalisation, réplication automatique, versioning des données, etc. Bien sûr, tout cela vient battre en brèche un éventail de principes érigé en dogme depuis plus de trente ans. Comme nous l'avons vu, la plupart des SGBD de la mouvance NoSQL ont donc été construits en faisant fi des contraintes ACID, quitte à ne pas proposer des fonctionnalités transactionnelles. Le groupe NoSQL, vise en premier lieu à démontrer qu'il existe désormais des solutions alternatives utilisées au sein de systèmes exigeants. L'objectif est évidemment d'obtenir une meilleure disponibilité des données, au moyen de capacités de partitionnement étendues mais au prix d'un relâchement des contraintes des propriétés ACID. Toutefois, avant de décider de l'adoption du NoSQL, il convient de se rassurer que l'environnement d'immersion lui est favorable. Afin d'éviter des surprises désagréables, il convient de répondre d'abord aux questions ci-après :

NoSQL : pour quels usages ?

Nous avons vu que NoSQL était une technologie qui avait été principalement développée dans un contexte où la volumétrie des données rendait indispensable l'utilisation des systèmes distribués pour assurer leur stockage et leur traitement. Il s'agit finalement du cas le plus évident qui pourrait décider de l'utilisation d'une technologie NoSQL dans un projet. La plupart des moteurs NoSQL apportent par ailleurs une flexibilité plus importante que leurs concurrents relationnels en ce qui concerne la gestion de données hétérogènes. Cela entraîne une simplification importante des modélisations qui, lorsqu'elles sont bien menées, peuvent aboutir à des gains considérables en termes de performances pour une application. Le plus souvent, cela peut aussi s'avérer bénéfique en termes de maintenabilité sur l'application. Dans tous les cas, une analyse poussée de l'appli cation doit être menée. Cette analyse doit permettre de répondre à plusieurs questions essentielles. L'autre question qu'il faut se poser avant de faire le choix du NoSQL est de savoir s'il existe un souci de conception de son application.

Existe t-il un problème fondamental de « design » dans son application ?

Pour une application existante et avant toute chose, il est important de s'assurer par un audit de l'application que celle-ci ne présente pas de défaut de conception entraînant des problèmes de performance importants ou limitant sa capacité à évoluer. Les technologies NoSQL gagnent du terrain, notamment auprès des développeurs qui sont toujours très enclins à vouloir tester de nouvelles solutions. Mais il se peut que votre application ne soit pas une bonne candidate pour être implémentée sous NoSQL. La plupart du temps, les applications ayant une forte composante transactionnelle ne sont pas de bonnes candidates à une implémentation au dessus d'une base NoSQL, bien qu'il existe des contre-exemples.

Quand doit-on intégrer des technologies NoSQL dans une application?

Si votre application souffre de sous-performances qui ne sont pas liées à une erreur de conception de la base de données ; c'est que celle-ci est une bonne candidate à un passage en totalité sous une base NoSQL. Alors il peut être possible d'optimiser les choses en plaçant uniquement certaines parties de l'application sous NoSQL. L'application résultante utilisera donc NoSQL uniquement pour les parties qui le requièrent. Cette approche est le plus souvent choisie pour une application existante, pour d'évidentes raisons de coût. Rappelons toujours que NoSQL ne signifie pas NoSQL mais bien « Not Only SQL ». Dans un certain sens, le mélange des technologies pour obtenir le meilleur de chacune est une approche fondamentalement encouragée par les tenants du NoSQL. Ce choix ne peut intervenir que lorsqu'on a répondu à l'ensemble des précédentes questions. Outre les caractéristiques purement techniques de la solution retenue qui seront évidemment déterminantes, il est conseillé de prêter une attention importante au temps de montée en compétence des équipes de développement.

4 - L’architecture microservices

4.1 Introduction

Afin d'atteindre les objectifs de notre projet, l'étude des concepts théoriques qui lui sont relatifs et des différents moyens mis à notre disposition est une étape primordiale.

Dans ce chapitre, nous nous intéresserons aux concepts de base liés à notre travail. Nous décrirons, tout d'abord, l'architecture microservices. Puis, nous projetterons, les concepts liés à ce style architectural. A savoir, l'approche de la conception pilotée par le domaine, ou encore en anglais Domain Driven Design (DDD), le développement à base de composants et la polyglotte de persistances.

4.2 Microservices

Le terme "Microservice" a connu une émergeance au cours des dernières années pour décrire une un style d'architecture bien particulier. Cette approche consiste à développer une seule application en un ensemble de petits services, isolés, autonomes et indépendamment déployés.

Ces services peuvent communiquer ensemble afin de fournir les fonctionnalités nécessaires. Les microservices sont, dans la plus part du temps, implémentés et pilotés par des équipes de petite taille avec suffisamment d'autonomie. Chacune peut changer l'impléméntation de chaque microservice, ajouter ou supprimer des fonctionnalités de ce service avec un impact minimal sur les autres microservices.Ce style d'architecture présente plusieurs avantages comme l'hétérogénéité technologique, la résistance contre l'échec, la scalabilité sur mesure, la facilité de déploiement, l'alignement organisationnel, la réutilisabilité.

4.3 Caractéristiques de l'architecture microservices

D'après Martin FOWLER , l'architecture microservices possède neuf principales caractéristiques qu'il est essentiel d'appliquer durant la conception et le développement d'une application en microservices.

4.3.1 La division en composants via les services

Cette caractéristique est héritée de l'architecture à base de composants. Les microservices sont indépendamment développés, testés et déployés. Un changement dans un service ne nécessite que son déploiement et n'affecte pas l'intégrité du système. Les services permettent d'éviter le couplage fort entre les différents composants en utilisant des mécanismes d'appel distants explicites.

4.3.2 L'organisation autour des capacités métiers

La division d'une application en microservices est très différente de la décomposition classique qui est souvent basée sur les couches techniques. Chaque microservice est autonome vis à vis de la fonctionnalité qu'il réalise puisqu'il possède son propre code, sa propre interface et gère ses propres données.

4.3.3 Un produit, pas un projet

Le but de l'utilisation des microservices est de livrer rapidement un morceau de logiciel qui est alors considéré comme terminé. Dans la vision microservices, une équipe est responsable d'un produit durant tout son cycle de vie. Elle est entièrement responsable du logiciel en production.

4.3.4 Une gouvernance décentralisée

En effet, il est difficile de trouver une seule technologie permettant de résoudre tous les problèmes d'une façon efficace. D'où, il est préférable d'utiliser le bon outil au bon moment. Avec l'architectures microservices, nous pouvons utiliser pour chaque service le langage d'implémentation et la plateforme technologique les plus adéquats pour accomplir le besoin.

4.3.5 Gestion de données décentralisée

L'architecture en microservices admet l'utilisation de plusieurs bases de données. Dans le cadre d'une application monolithique, nous n'avons qu'une seule base données logique pour les entités persistantes alors que le cadre d'une applcation en microservices, chaque service a son propre modèle conceptuel et gère sa propre base de données.

4.3.6 Les extrémités intelligentes et les canaux stupides

Plusieurs entreprises s'investissent dans les canaux de communication intelligents entre les services, alors qu'avec les microservices, l'utilisation de communications stupides est favorisée. Ces communication non intelligentes ne font que transmettre les messages, alors que le microservice s'en charge du reste. L'intercommunication entre les microservices via des protocoles ouverts est privilégiée et beaucoup d'autres interagissent les uns avec les autres via des appels REST ou à travers des systèmes de file d'attente.

4.3.7 Automatisation de l'infrastructure

Les techniques d'automatisation de l'infrastructure ont connu une évolution considérable ces dernières années. L'évolution du cloud a réduit la complexité opérationnelle de la construction, du déploiement et de l'exploitation de microservices. Les entreprises, qui ont migré vers l'architecture microservices, ont gagné de l'expérience dans la livraison continue et l'intégration continue et elles utilisent maintenant des outils d'automatisation de l'infrastructure.

4.3.8 Conception pour l'échec

L'un des atouts majeur des microservices est qu'il sont conçus pour être tolérants aux pannes. Dans une application en microservices, si un service échoue, les autres services ne sont pas affectés et adaptent leurs fonctionnements selon l'état du système dans lequel ils évoluent.

4.3.9 Une conception évolutive

L'un des éléments déterminants dans l'architecture en microservices, est la notion d'indépendance et d'évolutivité. En général, l'évolution d'une application consiste à l'ajout de nouvelles fonctionnalités qui se traduit par la création de nouveaux microservices et ou par la mise à jour des services existants qui implique seulement la mise à jour et le redéploiement du microservice concerné.

4.4 Les concepts liés à l'architecture microservices

Dans cette section, nous exposerons les concepts liés à l'architecture microservices à savoir la conception pilotée par le dommaine, le développement à base de composants ainsi que le polygotte de persistance.

4.4.1 La conception pilotée par le domaine

La conception pilotée par le domaine, dite en anglais domain driven design (DDD), est une approche de développement qui favorise la création des systèmes informatiques autour des compétences métiers pour combler l'écart entre la réalité de l'entreprise et le code. En pratique, DDD encapsule la logique métier complexe en des modèles métiers.

Le maintien d'un modèle dans un état pur est difficile quand il s'étend sur l'intégralité de l'entreprise, donc c'est préférable de tracer des limites à travers le pattern « Bounded Context » pour les définir. D'après Martin Fowler [5] le contexte borné est un pattern central dans la DDD. Il est au centre de la section de conception stratégique. Ces relations entre les contextes bornés sont généralement décrites par une carte de contexte. La carte de contexte est un document partagé entre ceux qui travaillent sur le projet. Elle met en évidence les différents contextes bornés et leurs liaisons. Cette carte est illustrée par la figure 2.1 :

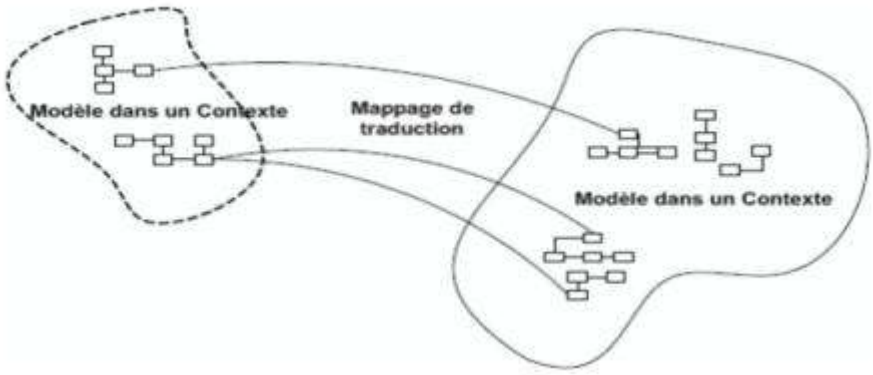


Figure 17:Carte de contexte

4.4.2 Développement à base de composants

Le développement à base de composant est une branche de l'ingénierie logicielle qui met l'accent sur la séparation des préoccupations à l'égard des vastes fonctionnalités disponibles à travers un système de logiciel. C'est une approche basée sur la réutilisation et la redéfinition.

Cependant, un système est un ensemble de préoccupations fonctionnelles et extrafonctionnelles. Les préoccupations fonctionnelles sont les fonctionnalités métiers que le système doit assurer, alors que les préoccupations extra-fonctionnelles sont des services dont le système a besoin pour effectuer ses fonctionnalités métiers.

Cette situation augmente la complexité, empêche la réutilisation et gêne l'évolution des systèmes. La séparation avancée des préoccupations permet de séparer les parties extra-fonctionnelles des parties fonctionnelles d'un système. L'objectif escompté étant d'offrir une meilleure réutilisation, faciliter la maintenance, réduire la complexité des systèmes et augmenter leur évolutivité.

4.4.3 Persistance polyglotte

Le terme persistance polyglotte (Polyglot Persistence en anglais) a été introduit par Scott Leberknight. Il tire son nom de la programmation polyglotte, qui favorise la coexistence de différents langages dans une même application, en tirant profit des forces de chacun d'entre eux.

L'un des atouts majeurs de l'architecture microservices, est que chaque microservice peut être écrit dans un langage de programmation qui lui est propre donnant plus de rapidité et de performance. En raison de l'isolement et de l'indépendance des microservices, les services individuels peuvent être polyglottes en termes de langage de programmation.

4.5 L'architecture de l'application en microservices

La figure suivant présente les composants primordiaux pour mettre en place une architecture microservices. Elle est composée généralement de :

- ✓ **Un serveur de configuration** encore appelé "Config Server" qui permet de centraliser les fichiers des configurations de chaque microservice dans un simple dépôt Git. Ceci permet d'avoir une configuration partagée et évolutive indépendamment des applications. Au démarrage, chaque microservice récupère ses propriétés et sa configuration auprès du serveur de configuration.
- ✓ **Un service d'enregistrement** "Service Registration" qui contient la liste de toutes les instances disponibles des microservices. Donc, aprés avoir récupéré leurs configurations, les microservices s'enregistrent dans le serveur d'enregistrement. Cela rend la découverte des microservices plus facile.
- ✓ **Une passerelle appelée encore "API Gateway",** qui présente le point d'entrée au système. Elle encapsule l'architecture du système interne et fournit des API adaptées pour chaque type de client. l'API Gateway encapusle un composant très important qui est l'équilibreur de charge, appelé "Load Balancer". Il gére le routage et répartion de la charge entre les instances des microservices disponibles. Pour avoir la liste des instances disponibles, le load balancer consulte le serveur d'enregistrement.

- ✓ **Un disjoncteur de circuit**, "Circuit Breaker", ce composant est primordiale dans une architecture microservices. Il garantit une caractéristique très importante qui est la conception pour l'échec, que nous avons évoqué dans le chapitre 2. Le disjoncteur de circuit permet d'éviter l'échec de tout le système en cas de défaillance d'un microservice.

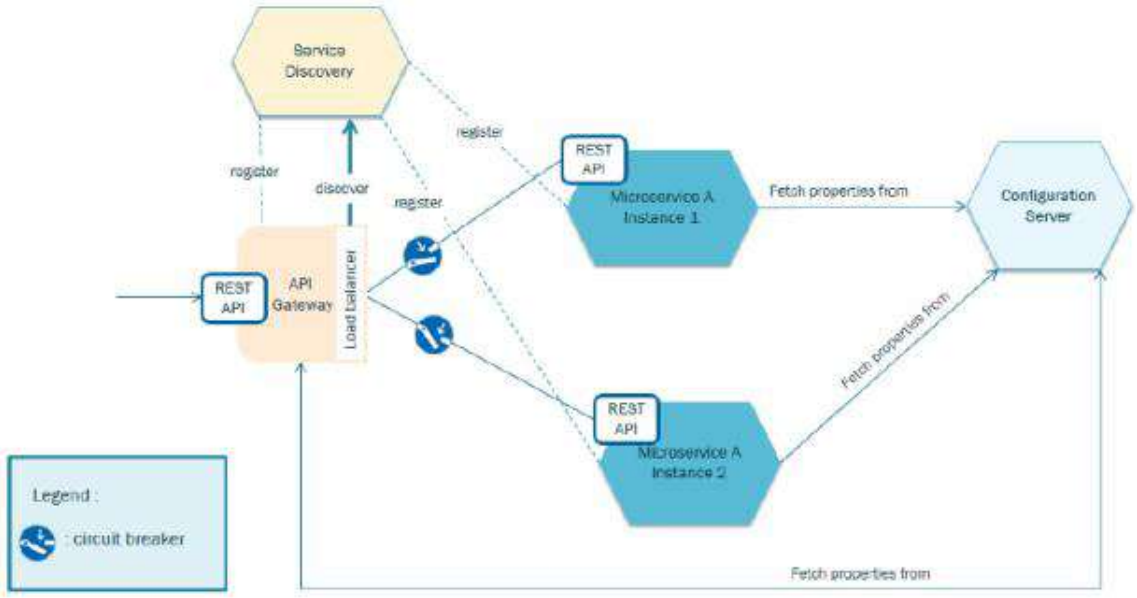


Figure 18:Les composants d'une architecture microservices

4.6 Les Edge Microservices

Les **Edge Microservices** sont des Microservices spécialisés dans l'orchestration des Microservices centraux responsables de la logique de l'application.

Les Edge Microservices les plus populaires sont ceux publiés et utilisés par Netflix pour sa plateforme. Ils sont en grande partie regroupés sous Spring Cloud et disposent de fonctionnalités pour fonctionner nativement ensemble.

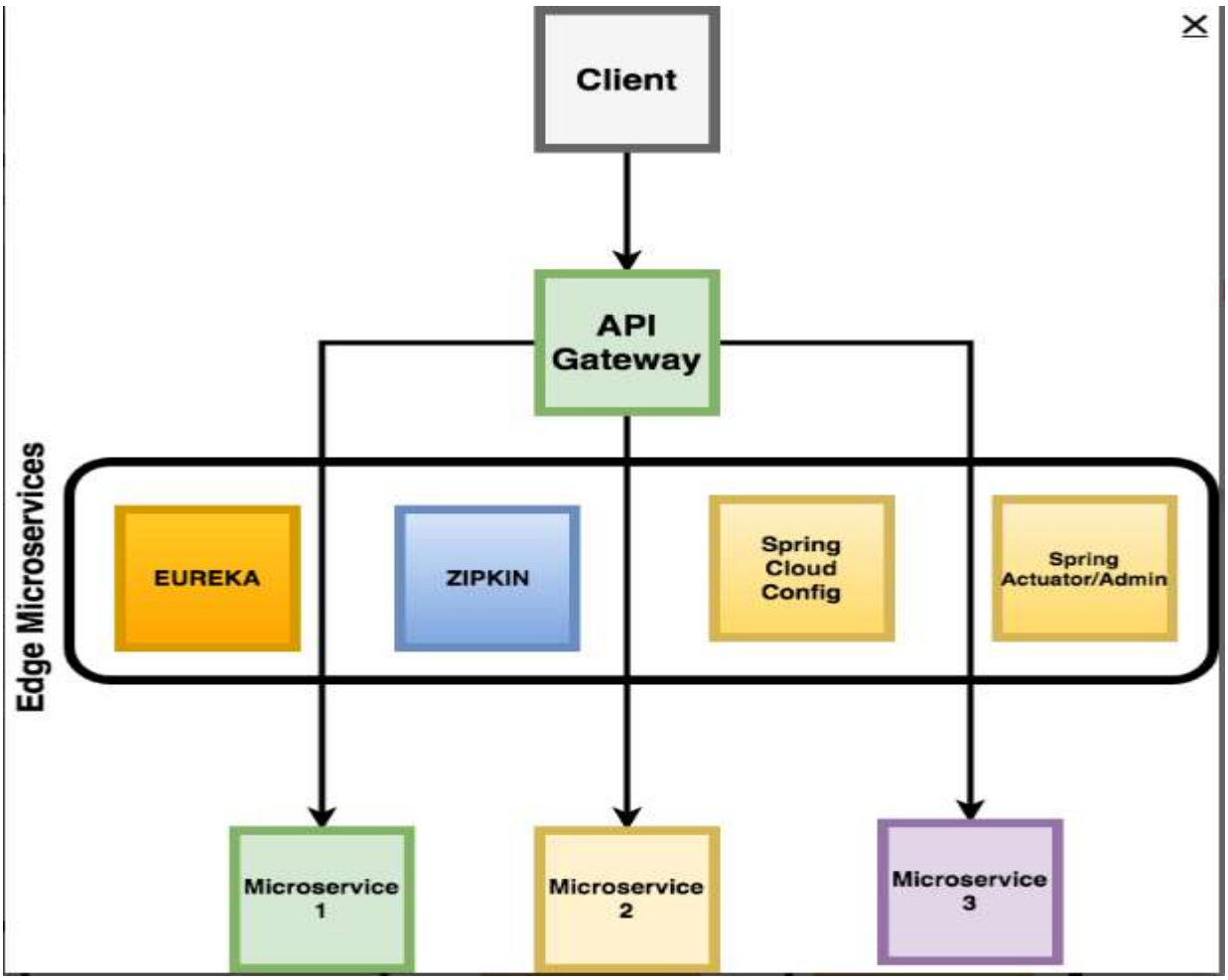


Figure 19:Edge Microservices

4.6.1 Spring Boot

Spring Boot est un framework avancée qui simplifie le démarrage et le développement de nouvelles applications Java EE. Les configurations sont atténuées avec Spring Boot, qui soutient des conteneurs embarqués. Cela permet à des applications web de s'exécuter indépendemment et sans déploiement sur le serveur web.

4.6.2 Spring Cloud

Spring Cloud, basé sur Spring Boot, offre une boite à outils permettant de créer un système d'applications distribuées rapidement.

4.6.3 Spring Data MongoDB

Pour une uilisation simplifiée de la base de données NOSQL MongoDB nous avons opté pour le framework Spring Data MongoDB pour l'implémentation de la couche dao du microservice journalisation, qui propose une certaine unification pour les accès aux bases de données NOSQL.

Spring Data MongoDB est une implémentation qui fournit une couche d'abstraction au client Java fourni par Mongo et offre les fonctionnalités nécessaires en s'intégrant avec le framework Spring.

4.6.4 Spring Cloud Config

Spring Cloud Config permet de centraliser les configurations du système distribué. Cela dans le but de rendre plus aisé la maintenance des microservices.

4.6.5 Spring Eureka Service

Eureka, faisant partie du projet Netflix Open Source Software, est une application permettant la localisation d'instances de services. Elle se caractérise par une partie serveur et une partie cliente. Ces services doivent être créés en tant que clients Eureka, ayant pour objectif de se connecter et s'enregistrer sur un serveur Eureka.

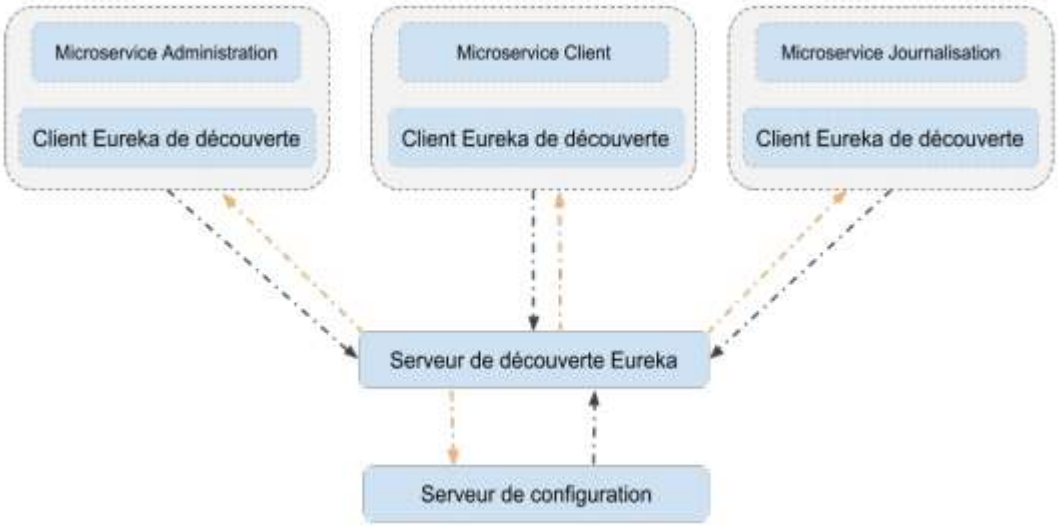


Figure 20:Enregistrement des microservices

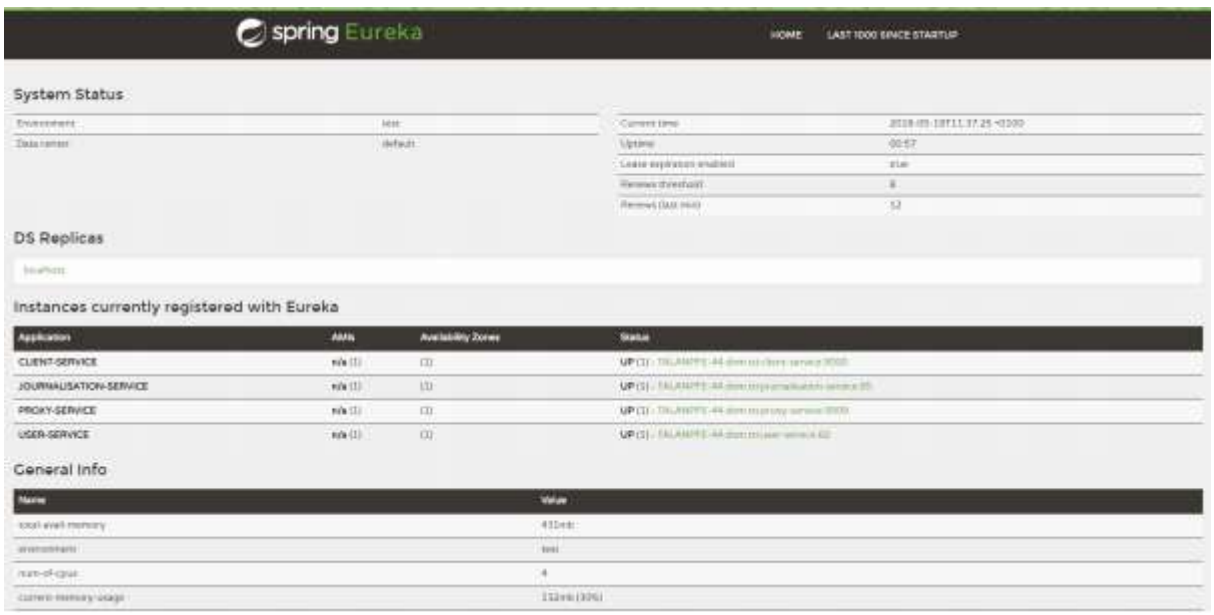


Figure 21:Dashboard Eureka

4.6.6 Zuul Proxy Service

Zuul, faisant partie de la stack Netflix OSS, joue le rôle de point d'entrée à notre système. Il se place donc en entrée de l'architecture et permet de réaliser des opérations sur les requête ainsi que sur leurs retour.

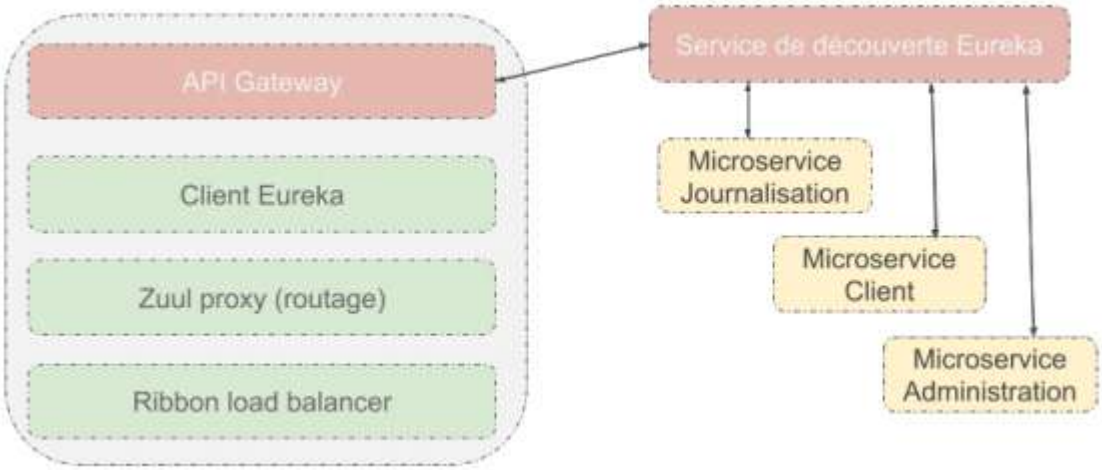


Figure 22:Fonctionnement de l'API Gateway Zuul

De chaque microservice, nous avons plusieurs instances dupliquées, si l'instance appelée par le proxy est défaillante, alors il appelle une autre instance inscrite dans l'annuaire de service Eureka. Dans le but d'assurer la tolérance aux pannes, l'utilisation du Circuit Breaker, implémenté par l'outil Hystrix, est primordiale. La figure 7.5 ci-dessous illustre la tolérance aux pannes pour le microservice Administation.

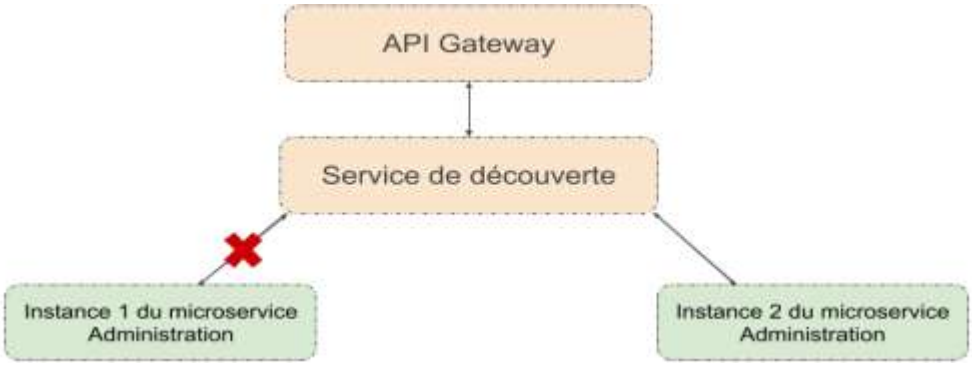


Figure 23:Tolérance aux pannes au niveau de l'API Gateway

4.6.7 Angular 5

Angular 5 est l'un des frameworks Javascript les plus avancés pour le web, créé par Google. Il est le résultat d'amélioration d'Angular 4 et Angular 2, développé avec TypeScript. Il offre des performances améliorées, une modularité amplifié, un respect des nouveaux standards du web et un code plus expressif.

5 - Partie pratique et Implémentation

5.1 Réalisation d'un cluster hadoop

Pour notre application, nous utilisons le langage Java pour développer les Mappers et les Reducers. Les traitements intermédiaires (comme le tri par exemple) sont effectués automatiquement par Hadoop. La première tâche consiste à lire les données à partir d'une Base de données MongoDB et les découper pour que chaque datanode puisse travailler sur une partie du données. Puisqu'on s'intéresse aux calcule le nombre des incidents pour chaque années. La clef dans notre cas est :le champ date de création. Quant à notre opération MAP, elle sera elle aussi très simple : on va simplement parcourir le fragment qui nous est fourni et, pour chacun des incidents, générer le couple clef/valeur : (Champ_date_création ; 1). La valeur indique ici l'occurrence pour cette clef - puisqu'on a croisé le champ date de création une fois, on donne la valeur « 1 ». Une fois notre opération MAP effectuée (de manière distribuée), Hadoop groupera (shuffle) tous les couples par clef commune. Il nous reste à créer notre opération REDUCE, qui sera appelée pour chacun des groupes/clef distincte. Dans notre cas, elle va simplement consister à additionner toutes les valeurs liées à la clef spécifiée : Une fois l'opération REDUCE effectuée, on obtiendra donc une valeur unique pour chaque clef distincte.

5.1.1 Architecture du cluster mise en place

Le schéma ci-dessous présente l'architecture du cluster hadoop que j'ai mis en place dans le cadre de ce mon travail.

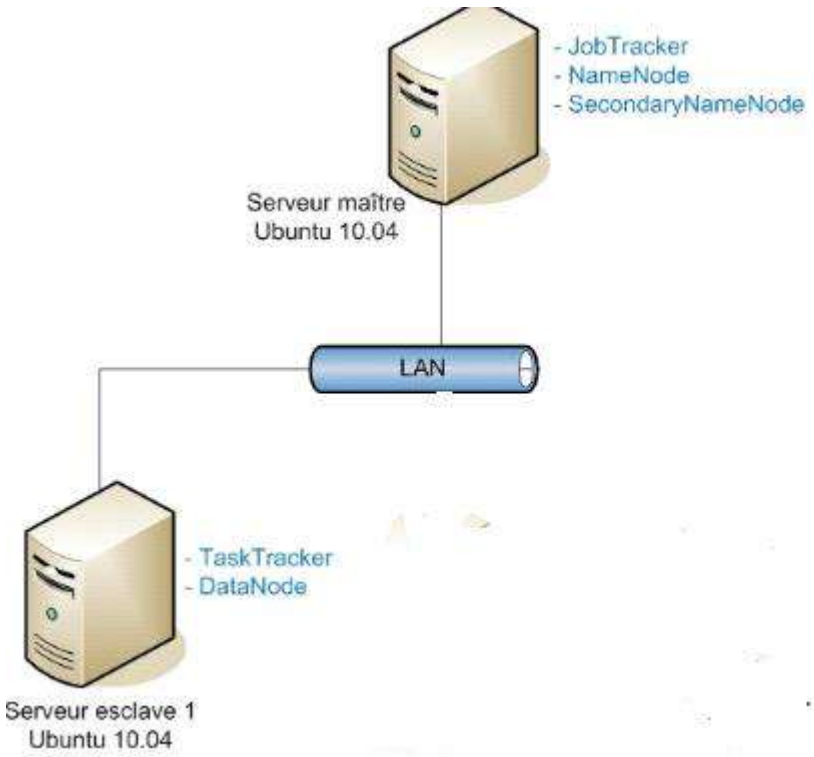


Figure 24:Architecture du cluster Hadoop mis en place

Ce cluster est constitué de postes standards équipés de système d'exploitation Ubuntu (version 18.04). Cette architecture est hébergée dans un environnement virtuel, ce qui nous a permis de tester la virtualisation d'un cluster Hadoop, solution incontournable pour faire du Big Data sur le cloud. Ce schéma présente les différentes machines (maître et esclave) du cluster et les rôles qui leurs sont associés dans le cadre d'une architecture Hadoop.

5.1.2 Démarrage de cluster Hadoop

Dans le **master** On va déplacer vers le Dossier **/usr/local/hadoop/sbin** et lancer le script **start-all.sh**

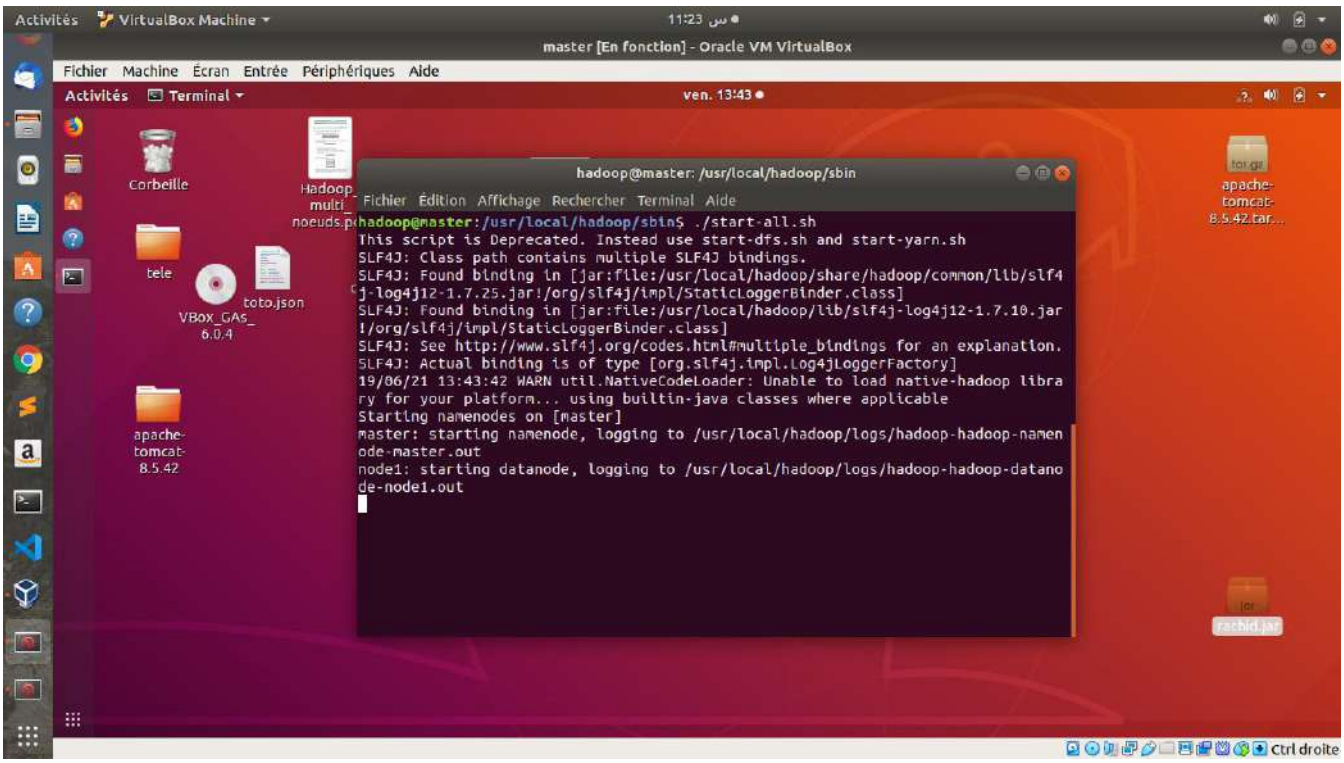
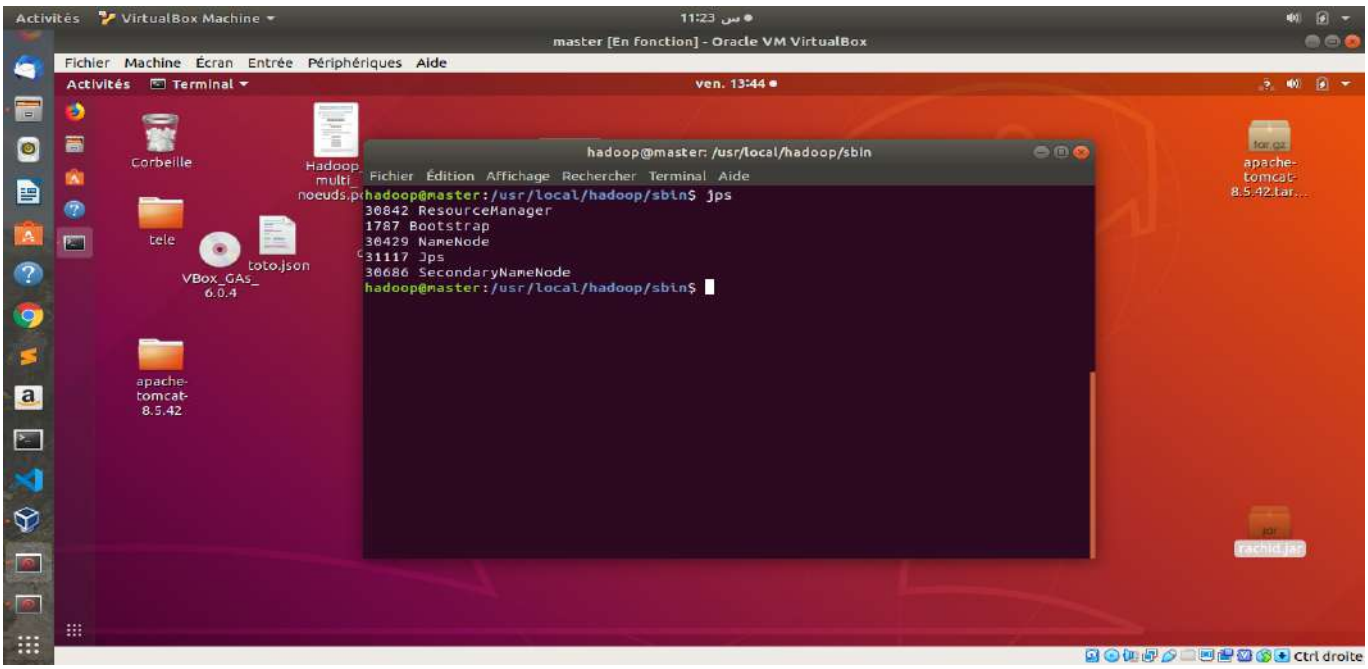


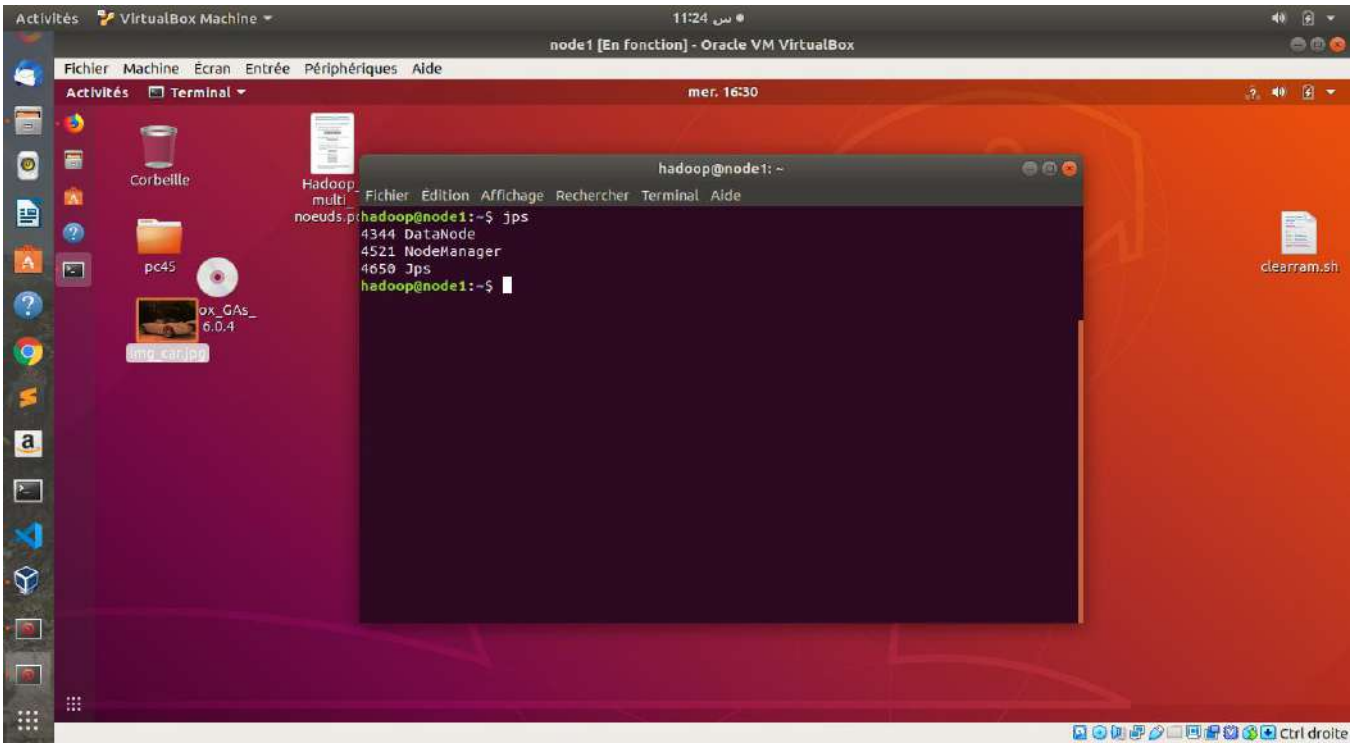
Figure 25:Démarrage Hadoop

Maintenant,on va taper la commande **jps** pour voir les processus activer :

- ✓ Machine Master:



- Machine slave :



Exécution de notre fonction **Map** et **Raduce** par la commande hadoop jar rachid.jar ocp.sta1.Main :

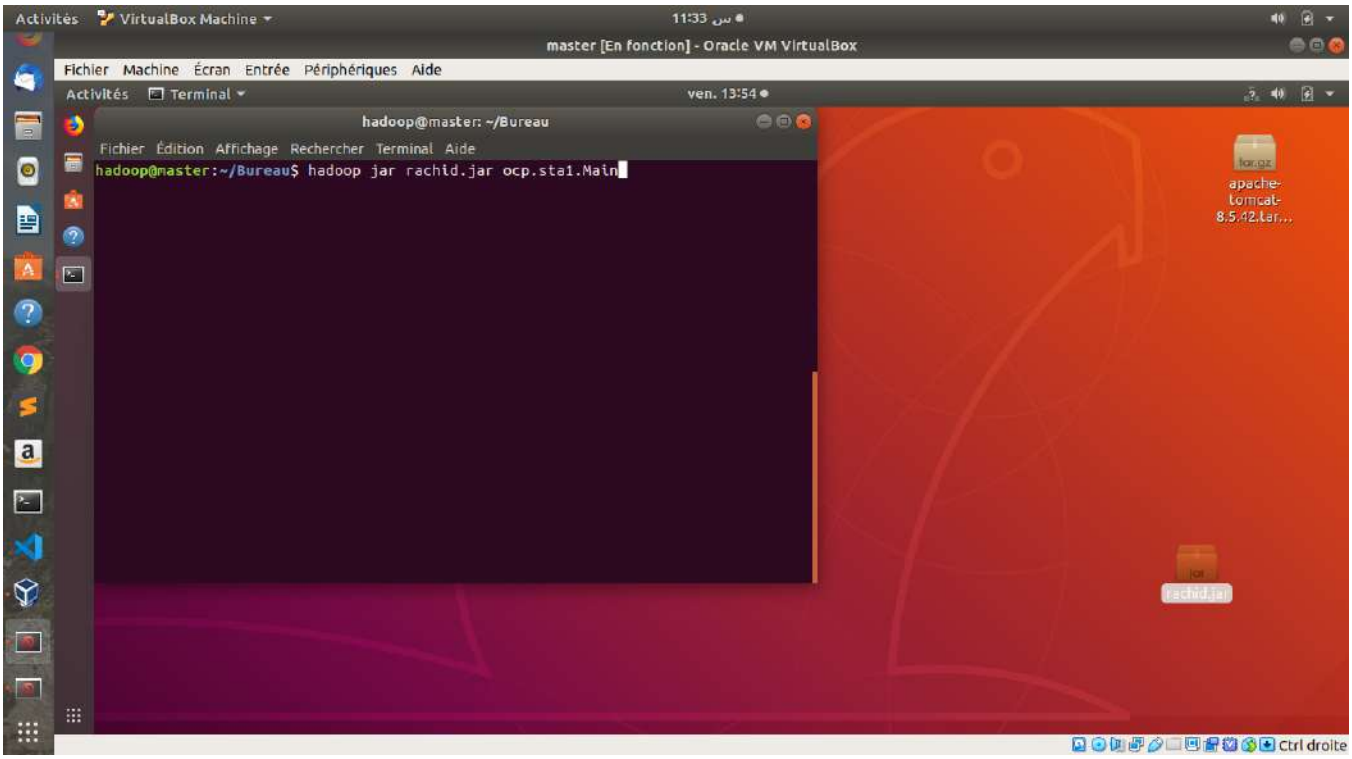
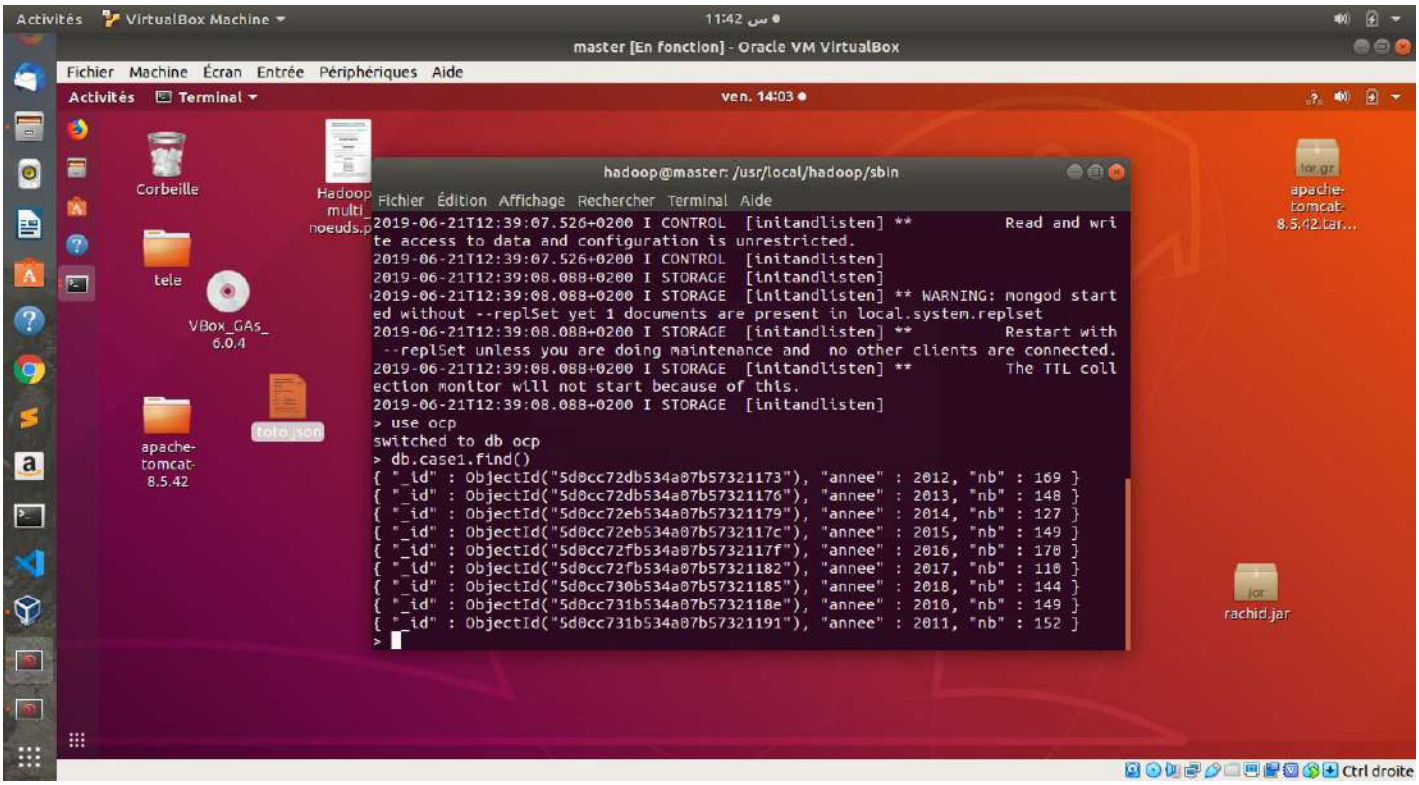


Figure 26:lancer un programme MapReduce

Les résultats de cette opération sont stockés dans la base de données **MongoDB**, pour l'utiliser dans notre application :



5.2 Application Web de gestion des incidents base sur la ARCHITECTURE MICROSERVICES.

5.2.1 L'architecture de l'application

- ✓ **Un serveur de configuration** encore appelé "Config Server" qui permet de centraliser les fichiers des configurations de chaque microservice dans un simple dépôt Git. Ceci permet d'avoir une configuration partagée et évolutive indépendamment des applications. Au démarrage, chaque microservice récupère ses propriétés et sa configuration auprès du serveur de configuration.
- ✓ **Un service d'enregistrement** "Service Registration" qui contient la liste de toutes les instances disponibles des microservices. Donc, aprés avoir récupéré leurs configurations, les microservices s'enregistrent dans le serveur d'enregistrement. Cela rend la découverte des microservices plus facile.
- ✓ **Une passerelle** appelée encore "API Gateway", qui présente le point d'entrée au système. Elle encapsule l'architecture du système interne et fournit des API adaptées pour chaque type de client. l'API Gateway encapusle un composant très important qui est l'équilibreur de charge, appelé "Load Balancer". Il gére le routage et répartion de la charge entre les instances des microservices disponibles. Pour avoir la liste des instances disponibles, le load balancer consulte le serveur d'enregistrement.

5.2.1 Diagramme de cas utilisation

Diagramme associé à l'administrateur :

Le diagramme ci-dessous présente le diagramme de cas d'utilisation décrivant les différentes fonctionnalités de l'admin :

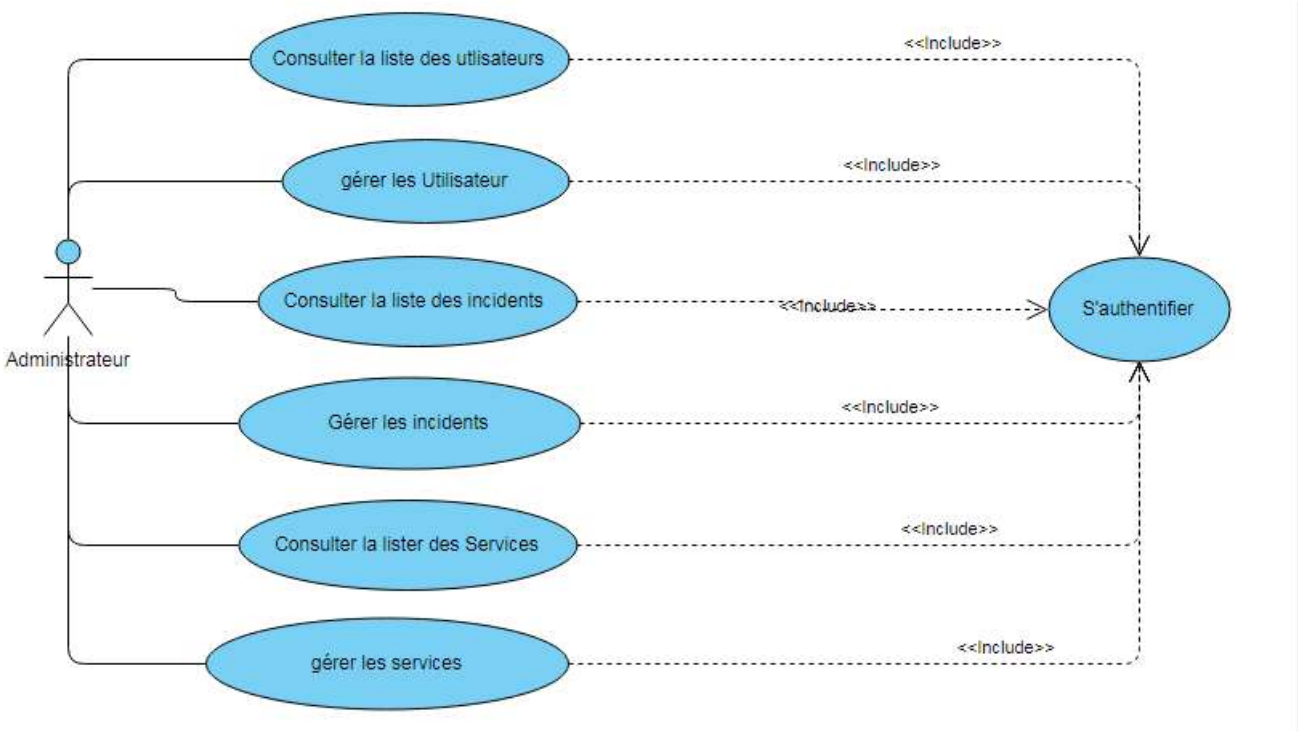


Figure 27:Diagramme de cas d'utilisation gestion des administrateurs

Diagramme associé à l'agent :

Le diagramme ci-dessous présente le diagramme de cas d'utilisation décrivant les différentes fonctionnalités de l'agent.

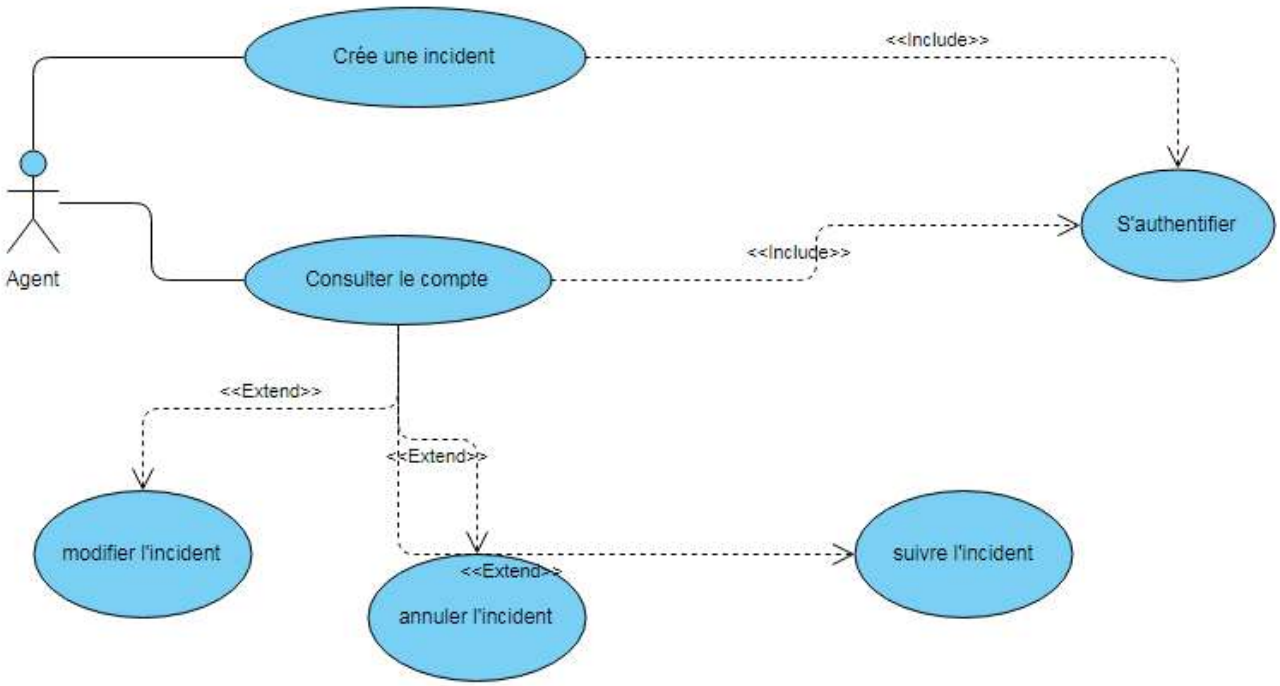


Figure 28:Diagramme de cas d'utilisation gestion des agents

Diagramme associé au technicien :

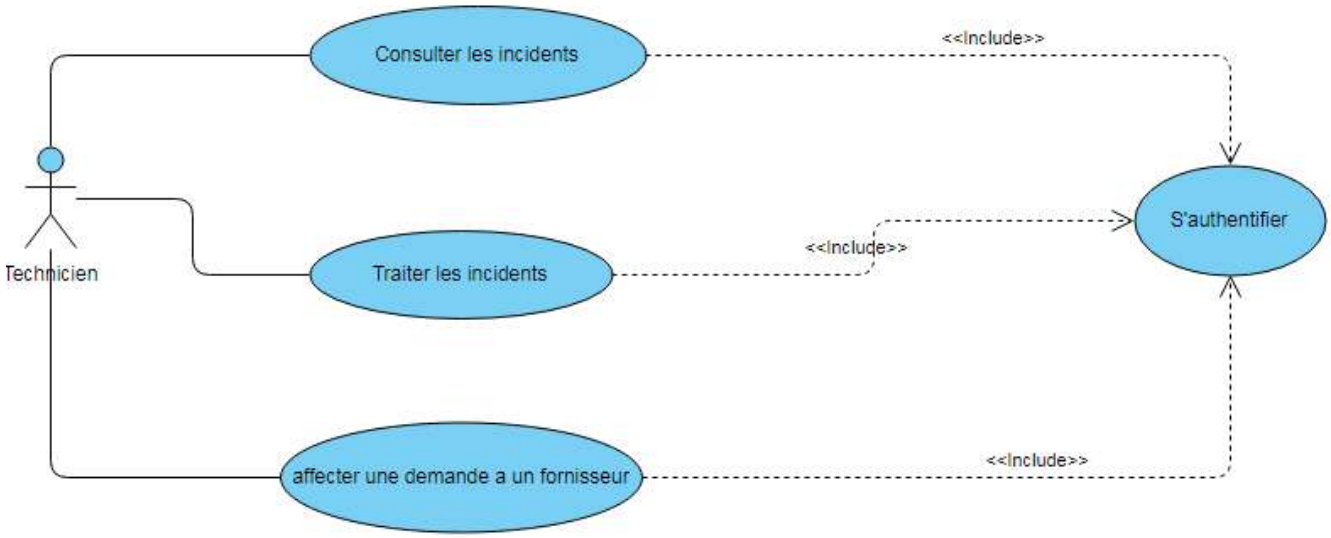


Figure 29:Diagramme de cas d'utilisation gestion de technicien

5.2.1 Diagramme de classes :

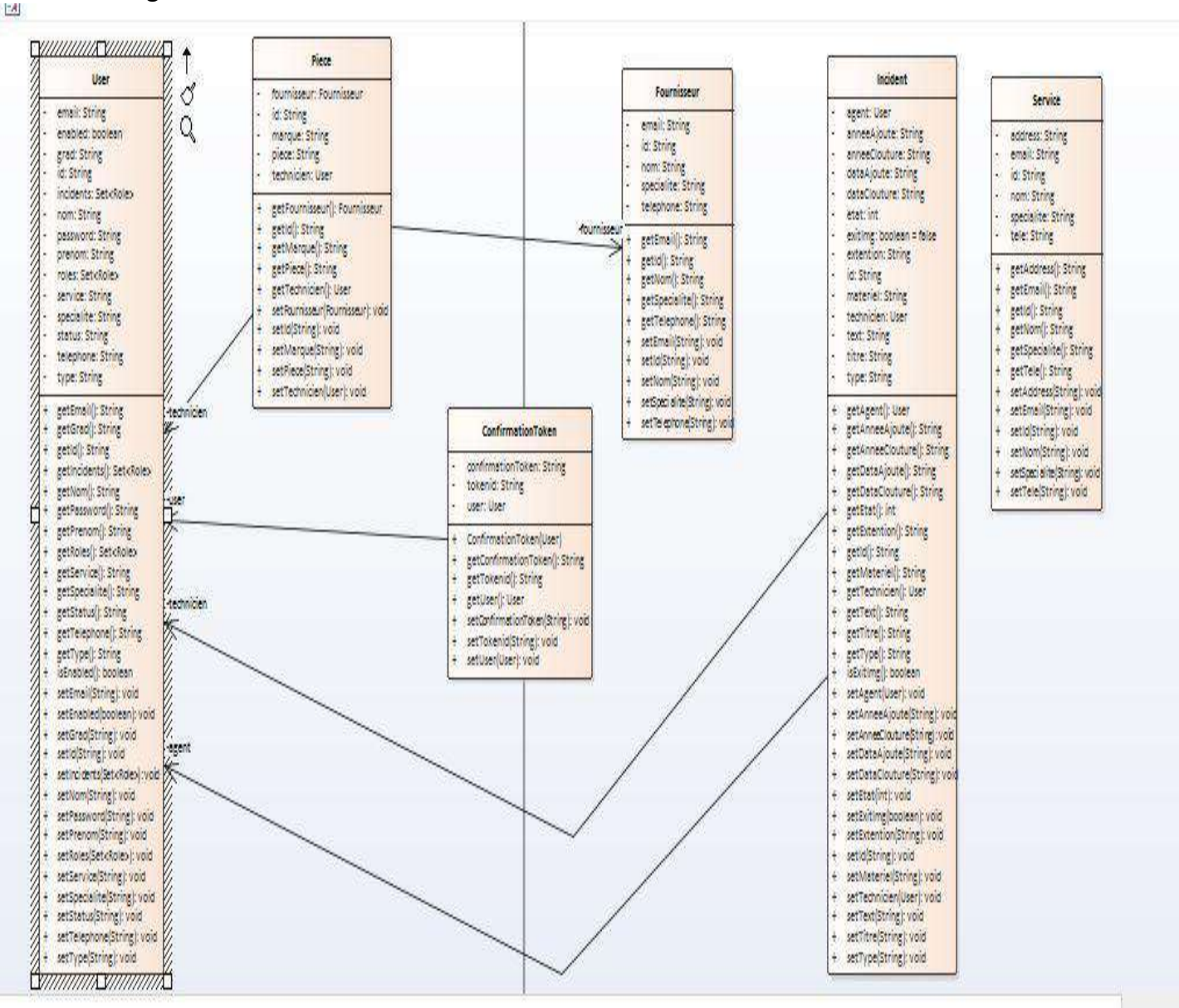


Figure 30:Diagramme de classes

5.2.2 Réalisation :

5.2.2.1 Espace Administrateur :

Une fois authentifié, Administrateur peut accéder à son espace.

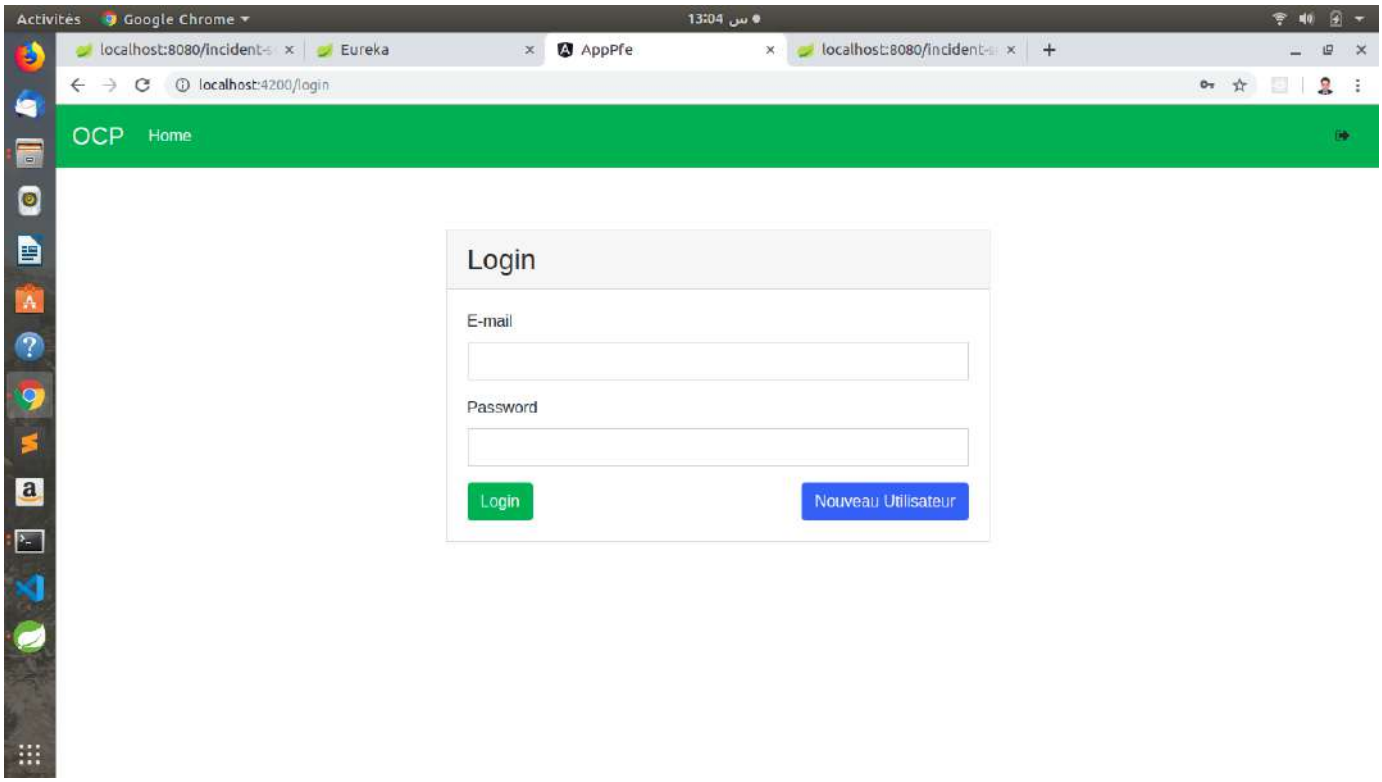


Figure 31:Interface d'authentification

L'administrateur peut accéder à son espace. Il peut consulter la liste des incidents , comme il peut soit modifier ou supprimer un incident et affiché les graphe , il peut gérer les incidents , Il peut consulter la liste des techniciens et les agents , comme il peut soit modifier ou supprimer un agent ou technicien via la liste affichée, Il peut consulter la liste des services.

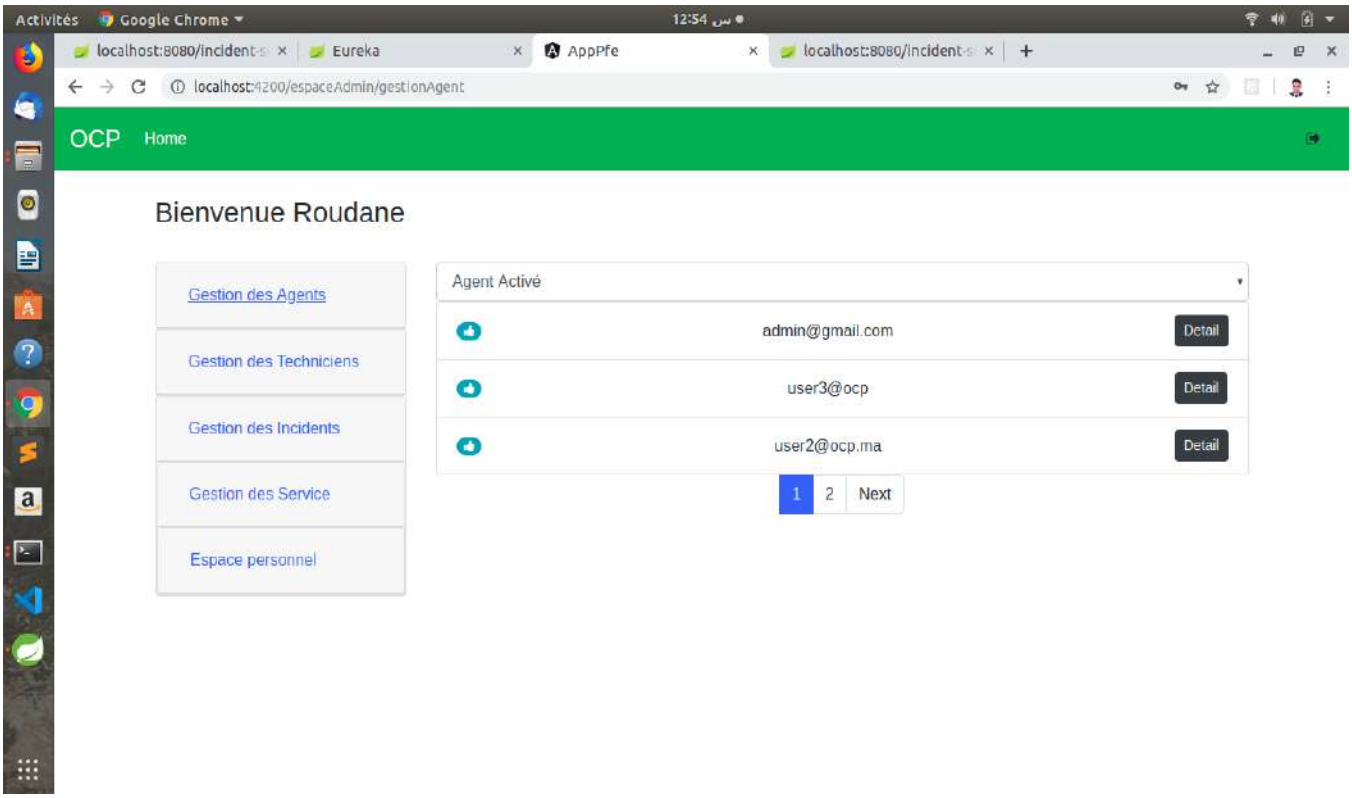


Figure 32:Liste des agents

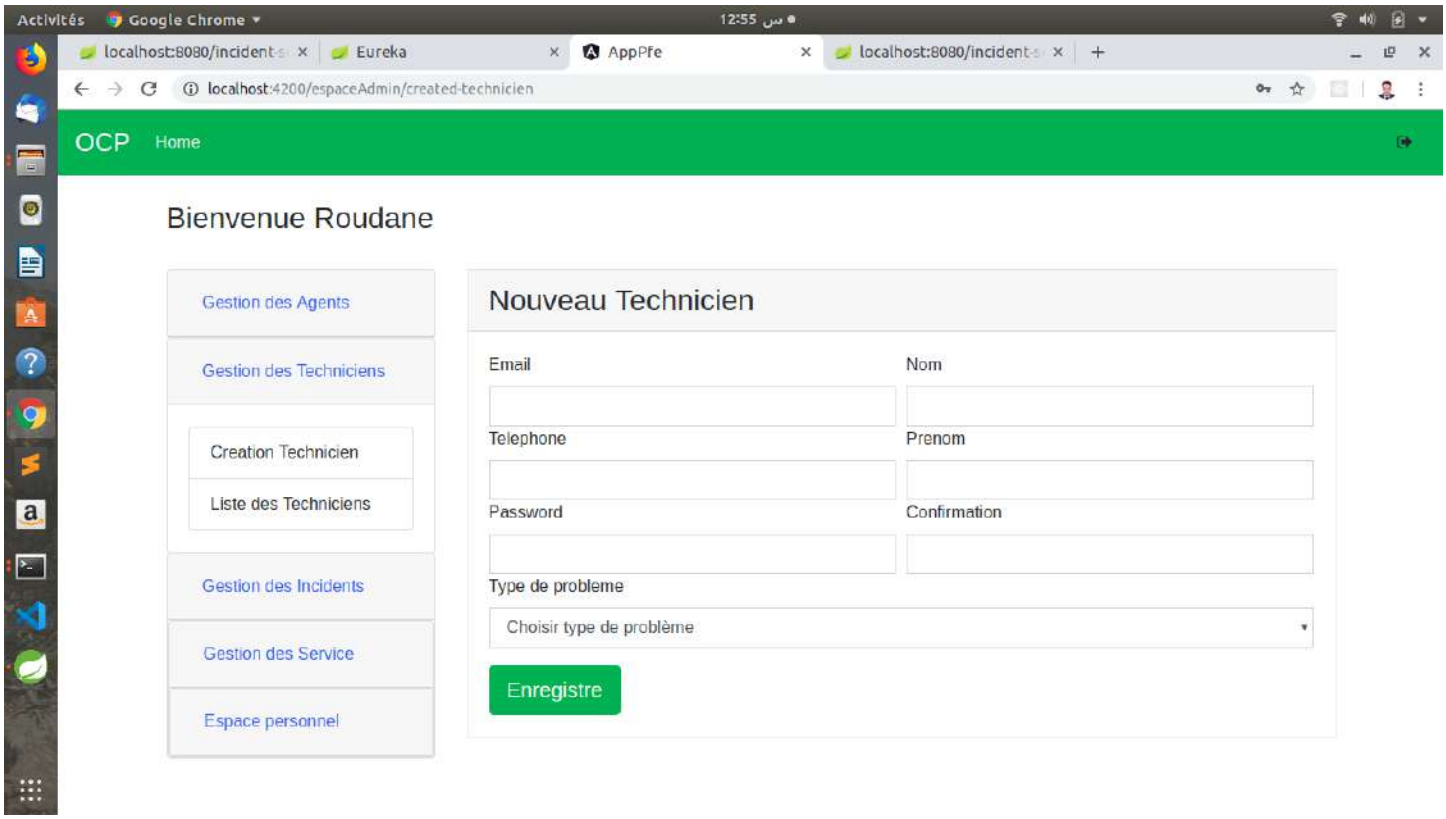


Figure 33:crée d'un nouveau technicien

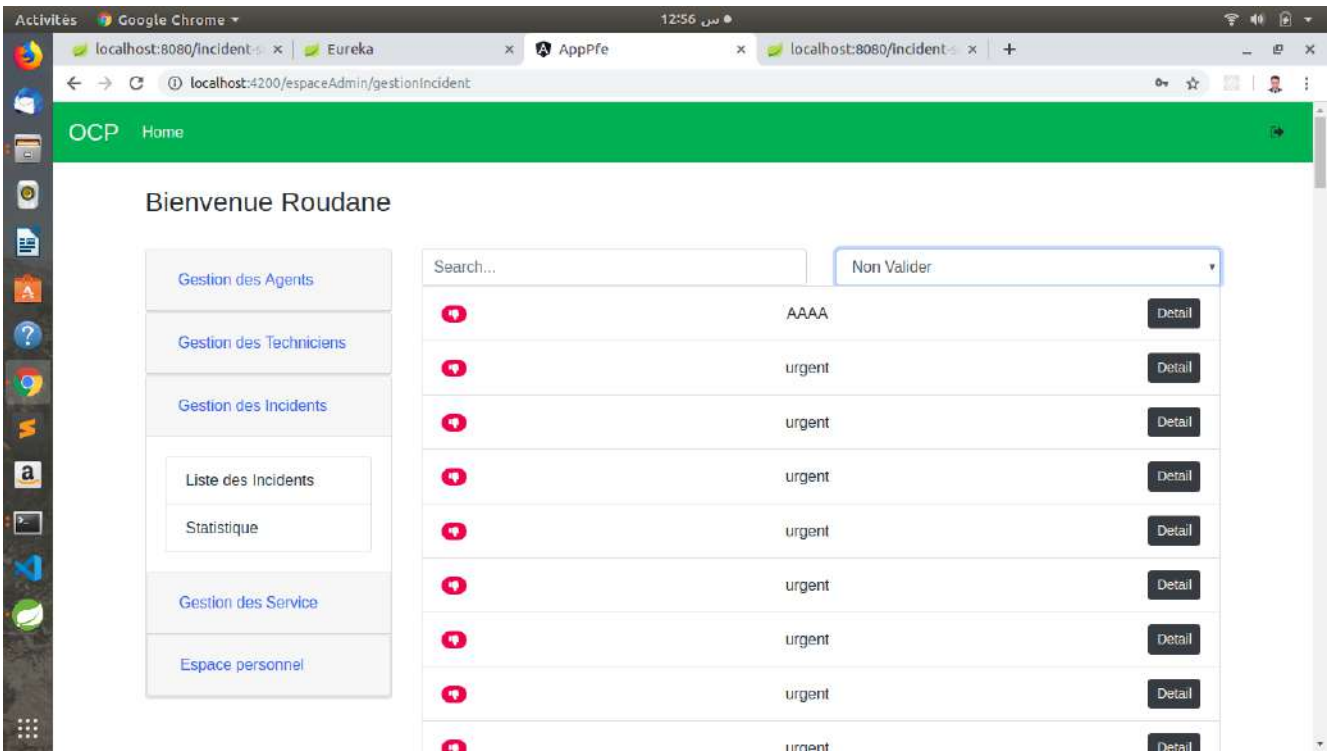


Figure 34:Liste des incidents admin

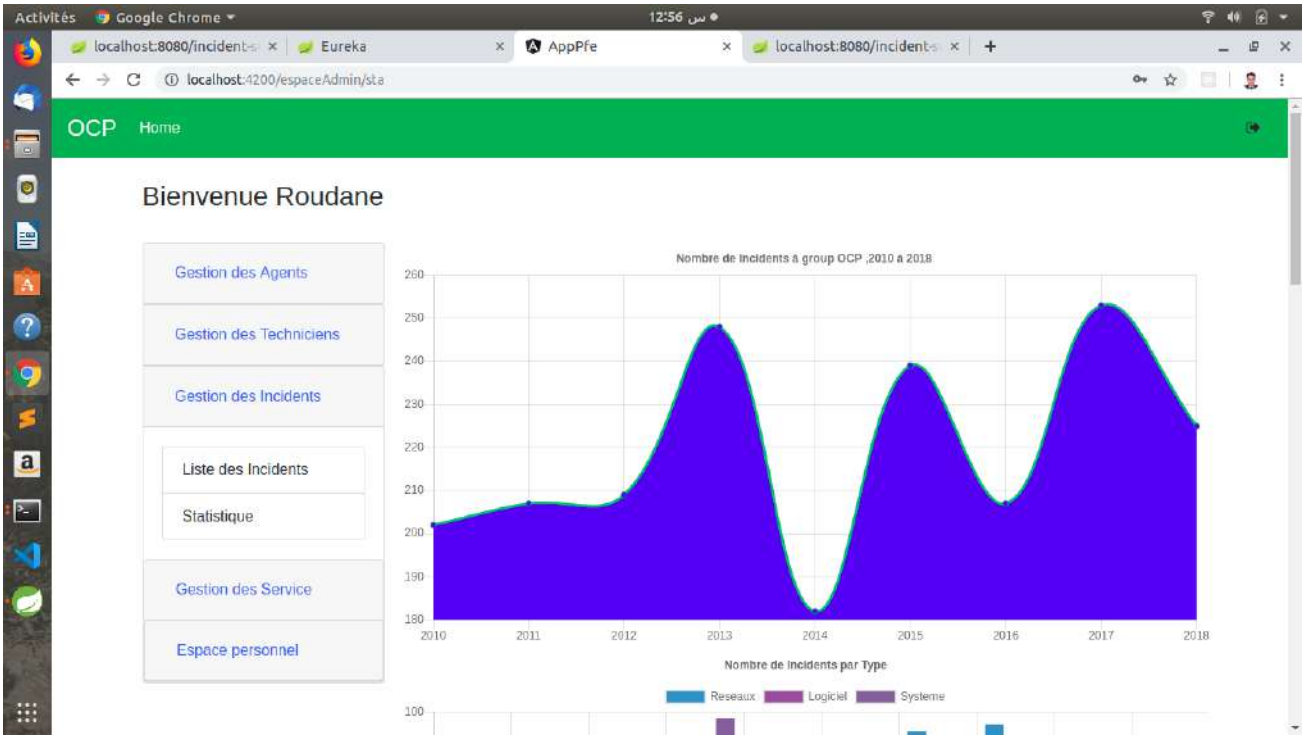


Figure 35:graphe présente le nombre des incidents par année

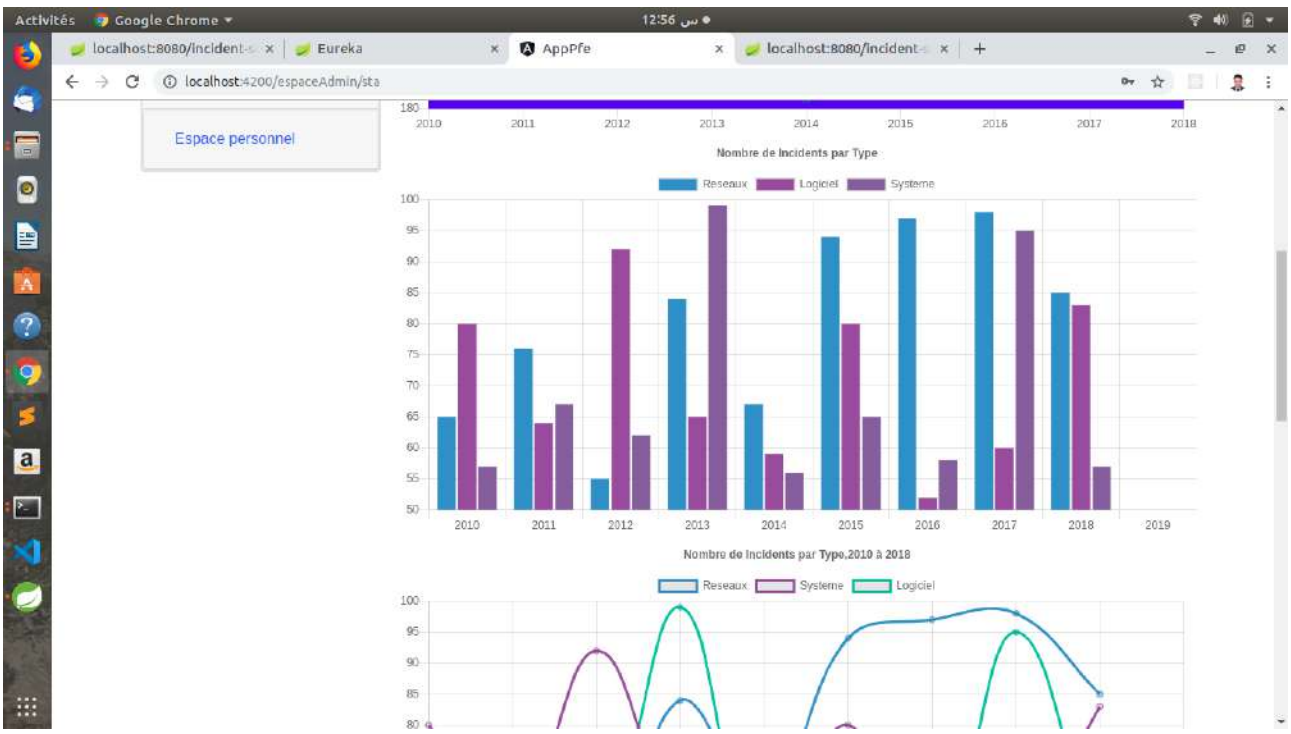


Figure 36:graphe de barre présente le nombre des incidents par année

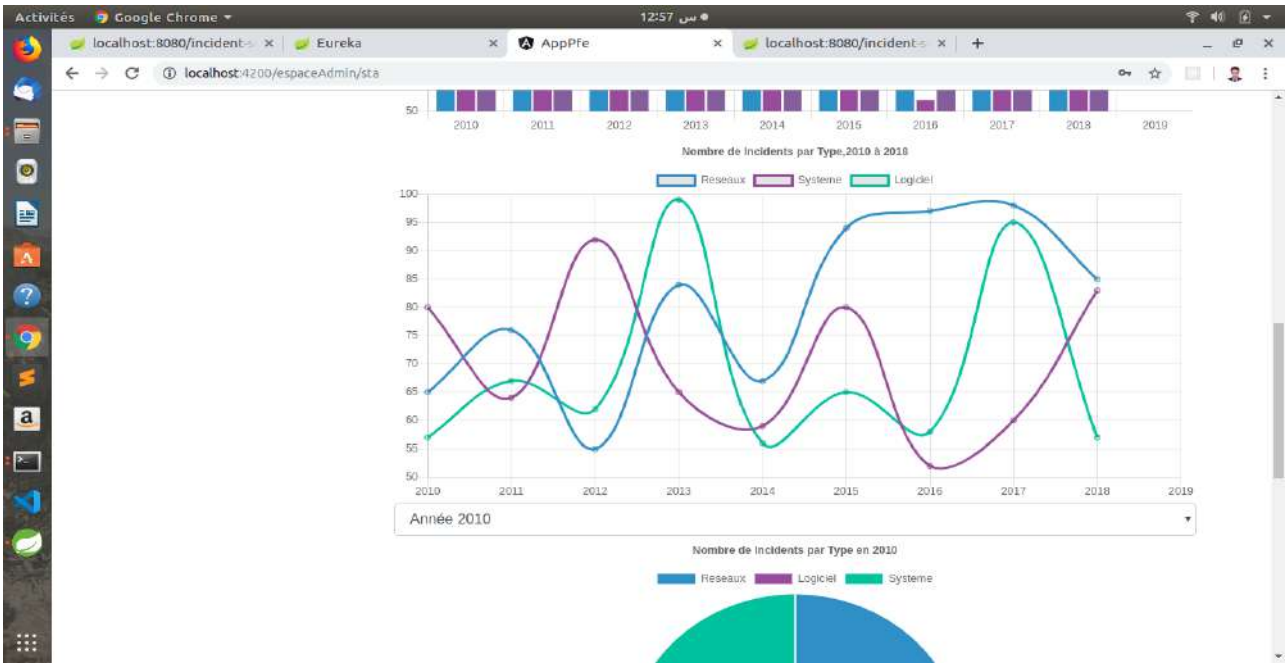


Figure 37:graphe de ligne présente le nombre des incidents par type selon chaque année

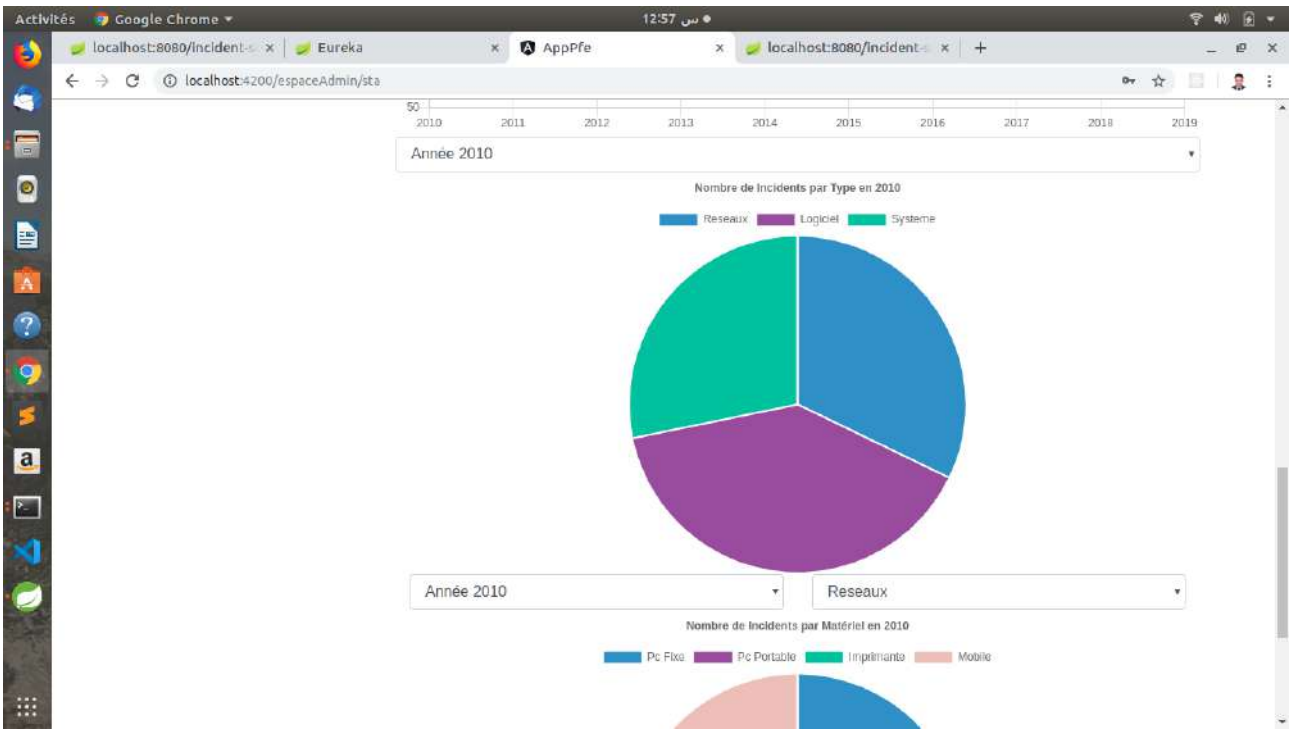


Figure 38:graphe circulaire présente le nombre des incidents par année et par type

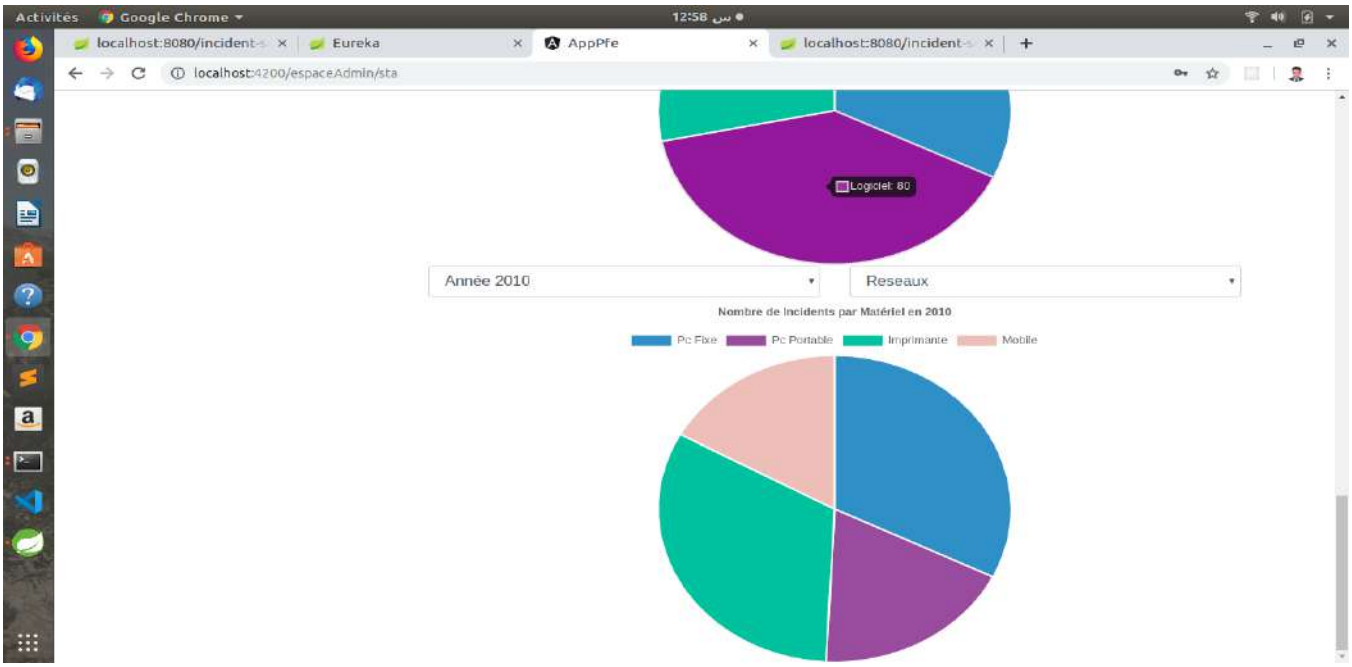


Figure 39:graphe circulaire présente le nombre des incidents par année et par Materiel

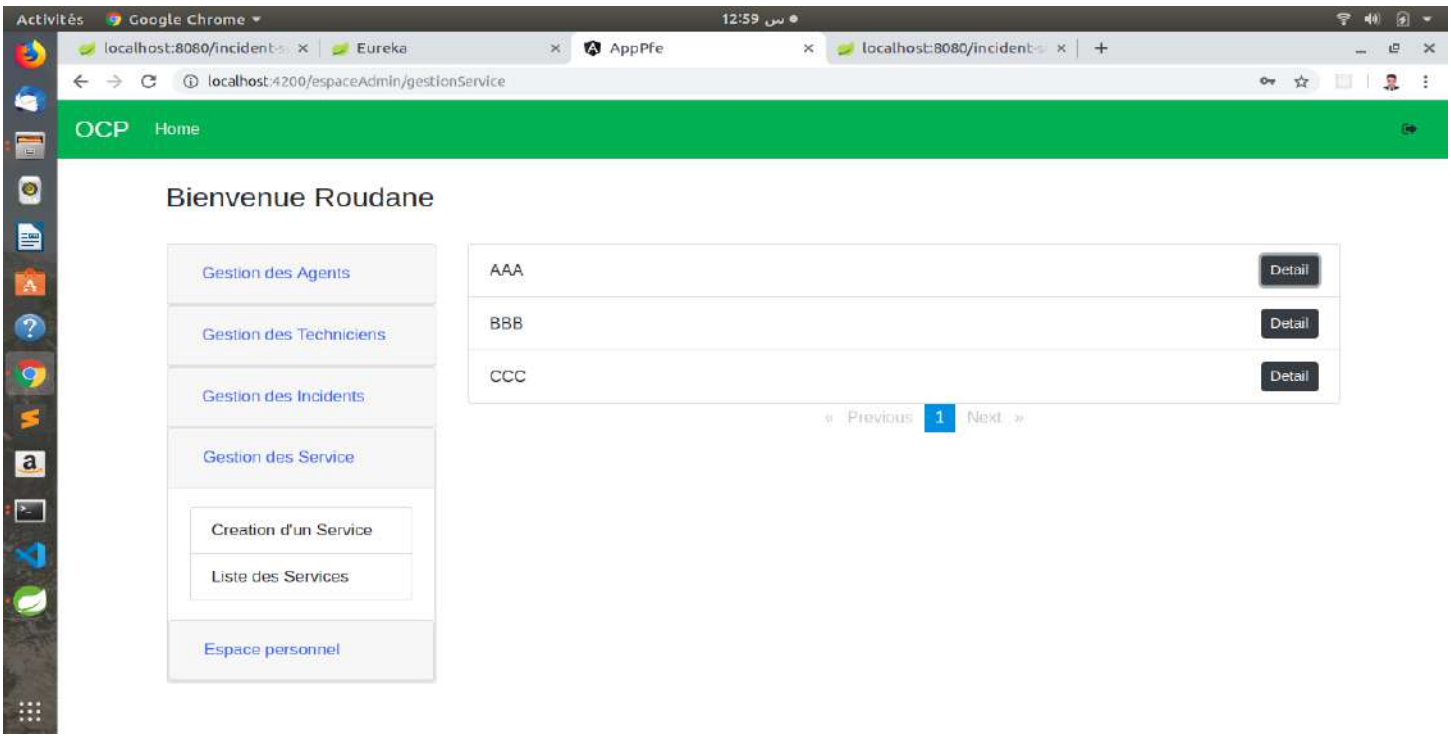


Figure 40:Liste des incidents

5.2.2.2 Espace Technicien

Une fois authentifié, Technicienpeut accéder à son espace.

Technicien peut accéder à son espace. Il peut consulter la liste des incidents , comme il peut clôture une incident

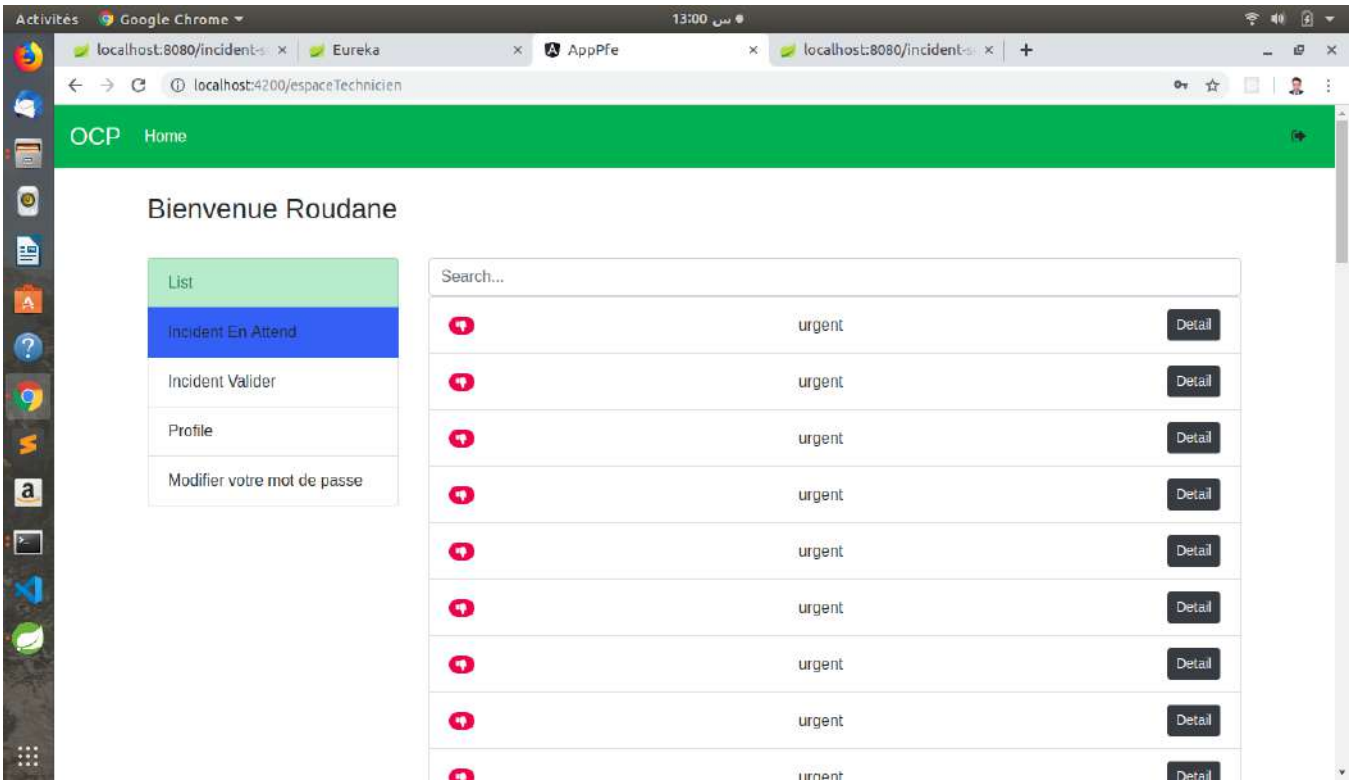


Figure 41:Liste des incidents Technicien

5.2.2.3 Espace Agent

L'agent peut accéder à son espace. Il peut consulter les incidents , comme il peut soit crée ou modifier ou supprimer un incident , il peut détail un incident.

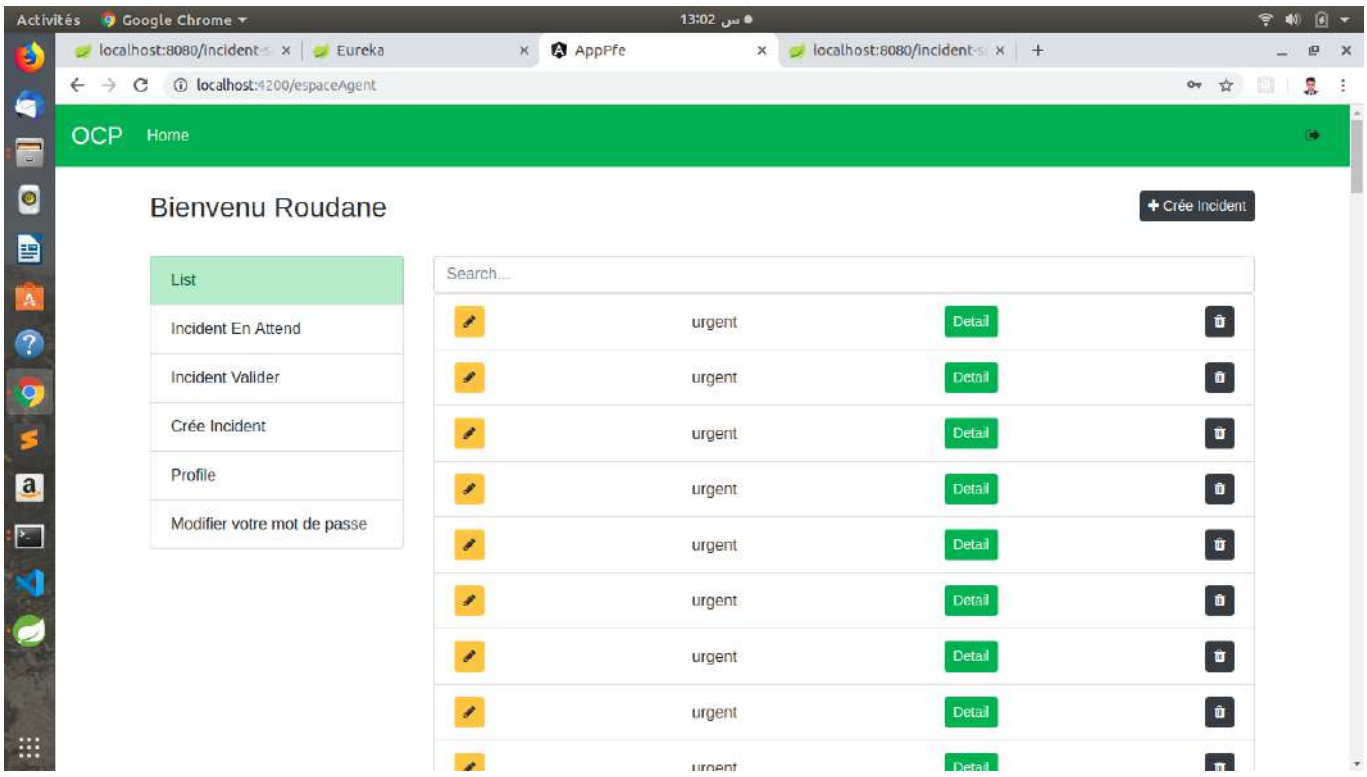


Figure 42:Liste des incidents Agent

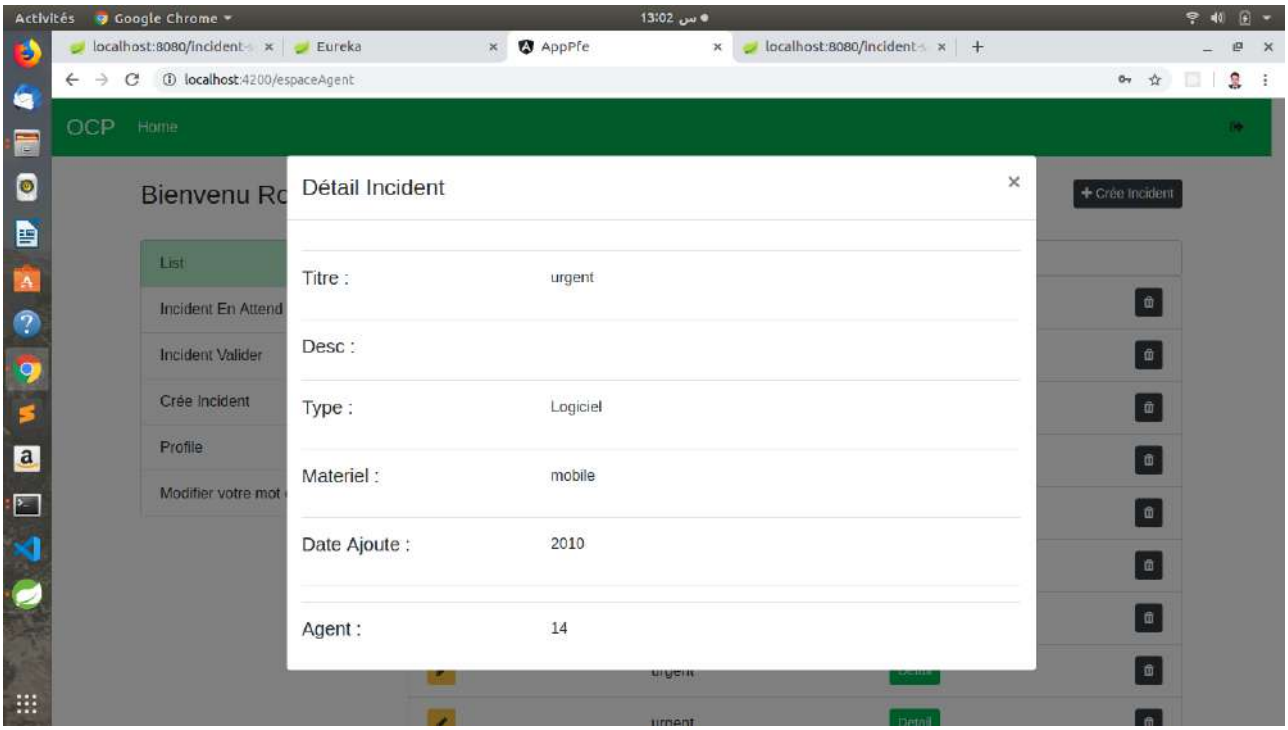


Figure 43:détail d'un incident

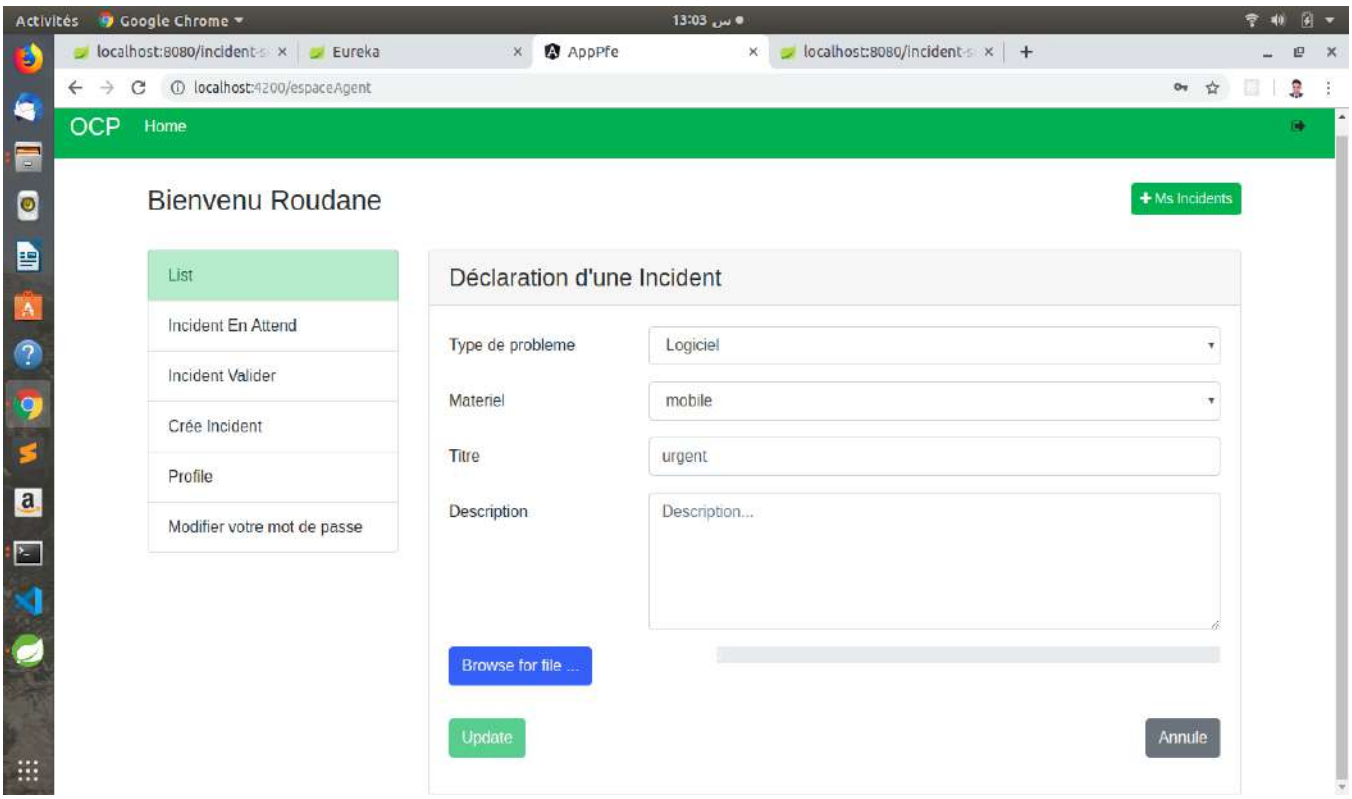


Figure 44:modifié un incident

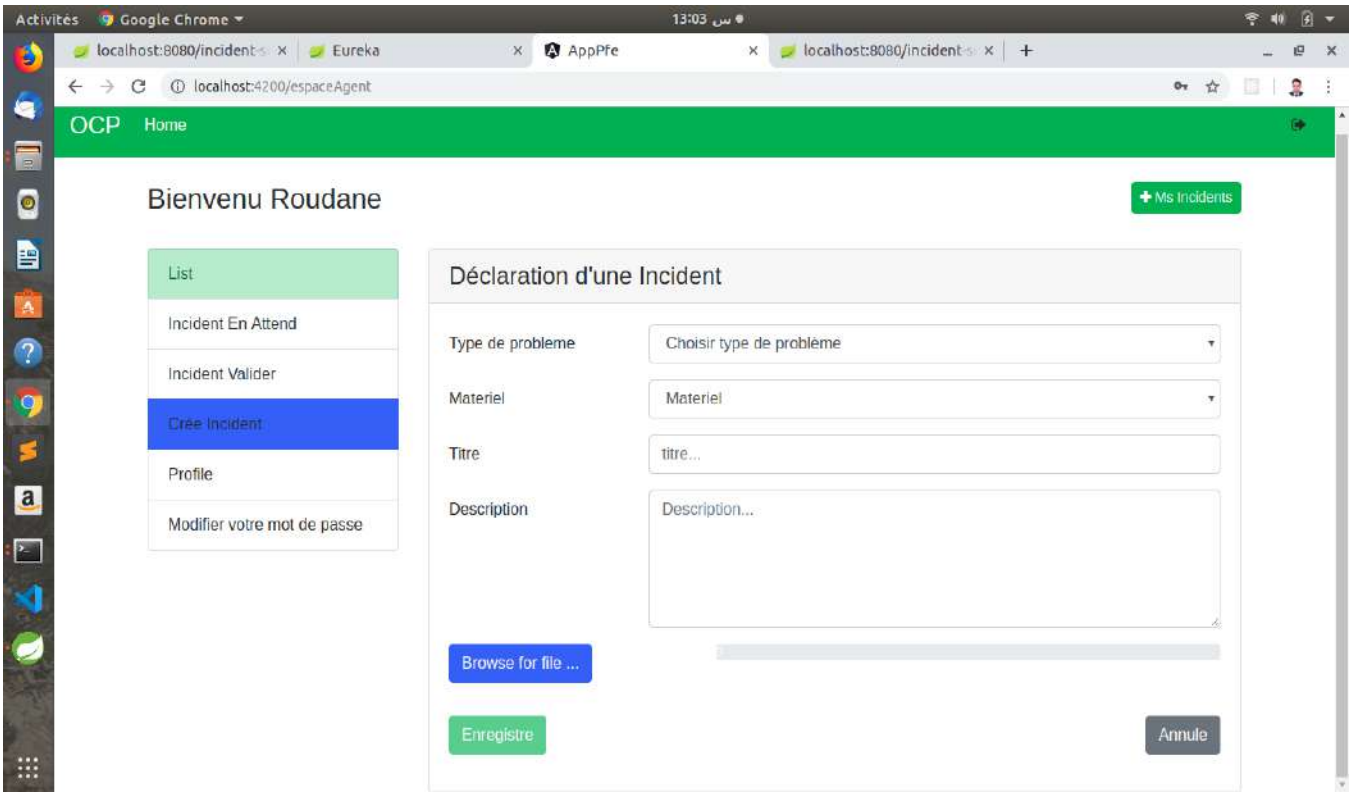


Figure 45: crée un incident

Conclusion générale

L'informatique à la base était le traitement de grande quantité de données, mais ce qui a changé avec la venue du Big Data, c'est la formalisation de l'évolution des volumes, de la vitesse et de la variété des données, qui crée de la valeur ajoutée.

Jusqu'ici, il n'existe pas encore sur le marché un logiciel Big Data prêt à l'emploi que l'on puisse installer dans une entreprise. Le Big Data est avant tout une démarche stratégique, il faut se poser la question : comment est-ce qu'à partir des données que j'ai ou que je peux collecter dans mon entreprise, je peux créer de la valeur ? Autrement dit, qu'est ce que je peux créer à partir de mes données. C'est ça la stratégie, donner de la valeur à vos données, transformer les données en euros ou en dollars. C'est grâce à cette stratégie que le Big Data fonctionnera à l'intérieur d'une entreprise.

L'écosystème Hadoop est constitué de plusieurs briques, qui pour certaines sont similaires en terme de fonctionnalités. Les tests techniques effectués m'ont permit de comprendre que toutes les briques ne sont pas forcément nécessaires pour faire fonctionner le cluster, celui-ci est modulable ; il revient à l'utilisateur de choisir les briques qui lui sont utiles en fonction de ses besoins. L'installation des briques séparément est possible mais il est plus judicieux d'utiliser un gestionnaire de cluster tel que Cloudera Manager pour centraliser toutes les tâches de déploiement, d'administration et de supervision du cluster.

Déployer une infrastructure Big Data & Microservices demande de mettre les mains dans le cambouis et de programmer. L'on peut faire ce qu'il veut des données, à condition de rentrer dans le code pour obtenir ce qu'il souhaite.

Le bilan de ce stage est dans l'ensemble positif, les principaux buts du projet étant accomplis. La plus grande partie du travail qui m'a été demandée a été réalisée.

Pour conclure, je dirais que ce stage m'a été d'un apport indéniable en matière de connaissances acquises sur les technologies Big Data & Microservices . C'est une expérience enrichissante qui m'a permit de comprendre les enjeux d'un projet de Recherche & Développement et de veille technologique, fort de plus dans un domaine porteur ; pour capitaliser davantage sur cette expérience, je continue à faire de la veille technologique sur concepts et technologies Big Data.

Bibliographie

- ✓ https://harry-wanki.developpez.com/tutoriels/mongodb/debuter-mongodb-introduction-base-donnees-nosql/
- ✓ https://docs.mongodb.com/manual/
- ✓ https://docs.mongodb.com/bi-connector/current/
- ✓ https://www.lebigdata.fr/hadoop
- ✓ https://docs.mongodb.com/ecosystem/tutorial/getting-started-with-hadoop/
- ✓ https://docs.spring.io/spring-boot/docs/current-SNAPSHOT/reference/htmlsingle/
- ✓ https://docs.spring.io/spring-data/jpa/docs/current/reference/html/
- ✓ https://docs.pivotal.io/spring-cloud-services/1-5/common/index.html
- ✓ https://stackabuse.com/spring-cloud-routing-with-zuul-and-gateway/
- ✓ https://cloud.spring.io/spring-cloud-netflix/reference/html/
- ✓ https://spring.io/projects/spring-cloud-config
- ✓ https://angular.io/docs/