École des Professions et des sciences de l'Informatique
91 rue Nationale
59800 Lille

Pictime Groupe
61 Avenue de l'Harmonie,
59262Sainghin-en-Mélantois

EPSI LILLE – 5ÈME ANNÉE

E-commerce : quels sont les impacts de l'utilisation de microservices et de base de données noSQL ?

Alexis DUBUS

Tuteur Entreprise : Cédric BERTHE
Tuteur EPSI : Didier NAKACHE

Promotion 2019-2020, soutenu en juillet 2020

Remerciements

Je voudrais dans un premier temps remercier, mes tuteurs de mémoire, Didier Nakache et Cédric Berthe pour leur patience et leurs judicieux conseils, qui ont contribués à alimenter ma réflexion sur de nombreuses problématiques.

Je remercie également toute les équipes de développement de Pictime Retail ou j'ai travaillé pendant deux ans et tout particulièrement Tommy Magron, Adrien Specq et David Vergison, pour leurs conseils et leurs soutiens.

Ainsi que mes parents et grands-parents, pour leurs encouragements et leurs soutiens constant.

Je remercie également toute l'équipe pédagogique de l'EPSI Lille et les intervenants professionnels, pour avoir assuré la partie théorique de cette année.

ATTESTATION DE NON-PLAGIAT (A inclure dans le mémoire professionnel)

Je soussigné(e)

Nom : DUBUS……. Prénom : ALEXIS…… ……

Auteur du mémoire professionnel pour le Titre de niveau 7 RNCP – Expert en informatique et Système d'information

(Titre du mémoire) E-commerce : impacts de l'utilisation de microservices et de base de données noSQL ?

……

Déclare :

- Que ce mémoire est un document original, fruit d'un travail personnel
- Avoir obtenu les autorisations nécessaires pour la reproduction d'images, d'extraits, de tableaux, figures ou graphiques
- Ne pas avoir contrefait, falsifié, copié tout ou partie de l'œuvre d'autrui afin de la faire passer pour mienne ;

Atteste :

- Que toutes les sources d'information utilisées pour ce travail de réflexion et de rédaction sont référencées de manière exhaustive et claire dans la bibliographie/webographie de mon mémoire professionnel
- Etre informé(e) et conscient(e) que le fait de ne pas citer une source ou de ne pas la citer clairement et complètement est constitutif de plagiat, et que le plagiat est considéré comme une faute grave et sanctionné par la Loi.

Date et Signature :

Fait le 28/06/2020…… A Linselles……

Signature : 

Sommaire

Résumé

Dans ce monde hyper connecté, les entreprises du e-commerce font face à une compétition féroce. Ces entreprises cherchent à avoir le dernier produit à la mode avec les meilleurs outils disponibles afin de récupérer des clients et des parts de marché. D'autre part, les habitudes des consommateurs évoluent constamment et il est impossible de savoir quel mode de consommation sera préférée dans le futur. Les entreprises doivent donc devenir plus flexibles.

Dans ce contexte, certaines entreprises ont pris les devants il y a quelques années en améliorant leur système d'information afin de pouvoir soutenir au mieux l'entreprise. Ceci est un changement total avec l'ancienne habitude considérant le système d'information comme un poids mort de l'entreprise, couteux et qu'il est nécessaire d'entretenir sans avoir de contrepartie.

Pour ce faire, l'entreprise va définir des grands objectifs stratégiques, pour lesquels la direction du système d'information va s'efforcer d'apporter des solutions. Un tel système d'information est alors capable de répondre rapidement et efficacement aux besoins de l'entreprise.

De tels changements du système d'information sont malheureusement bien souvent impossibles avec des logiciels monolithiques. Ceux-ci sont trop gros, trop complexes… Et ne peuvent pas être modifiés dans un court laps de temps afin de donner un avantage significatif pour l'entreprise, quand cela est fonctionnellement possible.

Les entreprises du e-commerce ont aussi une grande profusion de produits à vendre, car leurs clients leur demandent cette diversité. Cela peut nécessiter des centaines de milliers d'entrées en base de données, ce qui pose de gros problèmes de volumétrie et de rapidité. D'autre part, les bases de données, dites SQL et les logiciels monolithiques, ne savent pas bien gérer les pics de demandes très élevées du secteur du e-commerce. Cela est dût à leur conception originelle basée sur un seul serveur.

Afin de répondre à ces problématiques, certaines personnes ont créé les microservices. Ce sont de petits logiciels extrêmement spécialisés, qui en communiquant ensemble, vont pouvoir créer une architecture. Cette dernière, si elle est bien conçue, peut répondre aux besoins des entreprises là où les logiciels monolithiques n'ont pas réussi.

Quant aux problèmes liés aux bases de données SQL, les bases de données NoSQL peuvent répondre à certains d'entre eux, notamment en ayant la possibilité de répondre rapidement à l'augmentation de la demande.

Les microservices et les bases de données NoSQL sont donc deux nouvelles technologies qui peuvent tout à fait répondre aux besoins des entreprises du secteur e-commerce. Leur utilisation commune peut même créer de nouvelles fonctionnalités et ouvrir des possibilités inexplorées pour les entreprises.

Cependant, ces deux technologies ne sont pas encore extensivement éprouvées et leur seule utilisation ne suffit pas à mettre en place un système d'information efficient. Cela nécessite aussi de la préparation et une vision sur le long terme. Car tout comme la planification d'une ville, la gestion d'un système d'information se fait sur le long terme et nécessite de multiples compétences.

Au travers de ce mémoire, nous allons tenter d'apporter une réponse à la thèse suivante : E-commerce : quels sont les impacts de l'utilisation de microservices et de base de données noSQL ?

Abstract

In this hyper-connected world, e-commerce companies face fierce competition. They are looking for the latest fashionable product and the best tools available in order to gain customers and market share. On the other hand, consumer habits are in constant change and it is impossible to know which mode of consumption will be preferred in the future. Companies therefore need to become more flexible and adaptable.

In this context, some companies took the lead a few years ago by improving their information system, in order to better support themselves. This is a complete change from the old habit of considering the information system a dead weight for the busines, because it's expensive and it's necessary to maintain it without having any counterpart.

In order to do this, the company will define major strategic objectives, for which the management of the information system will strive to provide solutions. Such an information system is then able to respond quickly and efficiently to the needs of the business.

Unfortunately, such information system changes are often impossible with monolithic software. These are too big, too complex ... and they cannot be changed in a short period of time to give a significant advantage to the company, when this is functionally possible.

E-commerce companies also have a great profusion of goods, because their customers ask them this diversity. This can require hundreds of thousands of database entries, which poses big volumes and speed problems. Moreover, SQL databases and monolithic software, do not know how to handle the very high demand peaks in the e-commerce sector. This is due to their original design based on a single server.

In order to respond to these issues, some peoples have created microservices. Microservices are small and extremely specialized software. These software, by communicating together, will form an architecture. This new kind of architecture, called microservice architecture, if well designed, can meet the needs of businesses where monolithic software has not succeeded.

As for problems with SQL databases, NoSQL databases can respond to some of them, because NoSQL databases have the hability to scale in order to respond quickly to the increased demand.

NoSQL microservices and databases are therefore two new technologies that can fully meet the needs of businesses in the e-commerce sector. Their common use can even create new functionalities and open up unexplored possibilities for companies.

However, these two technologies are not yet fully tested and their use alone is not enough to set up an efficient information system. It also requires preparation and a long-term vision. Because just like city planning, managing an information system is long-term and requires multiple skills.

Through this thesis, we will try to provide an answer to the following thesis: E-commerce: what is the impacts of the use of microservices and noSQL database ?

1 Introduction

Tout comme Netflix et Amazon, grands noms de la vente en ligne, le secteur du e-commerce se développe. Cependant, comme l'a remarqué Amazon, les outils génériques du e-commerce ne sont pas toujours pertinents dans ce secteur en constant changement.

Pour ce faire, Netflix et Amazon ont créé des nouveaux services utilisant entre autres les bases de données NoSQL et les microservices. Ces derniers sont cependant rarement utilisés ensemble en entreprise. En effet, les sites d'e-commerce sont bien souvent des sites gérés par des CRMs avec des bases de données relationnelles.

Cependant, le regroupement de bases de données NoSQL et de microservices pourrait être une grande source d'innovation et l'opportunité de nouvelles fonctionnalités pour les entreprises du e-commerce.

En effet, ce secteur sans cesse a la recherche de nouvelle fonctionnalité gagnerait fortement à investir dans ses technologies, car cela permet de mettre en place rapidement les changements demandés par les clients et les fournisseurs. La capacité de répondre rapidement à ses changements étant une caractéristiques vitale dans ce secteur, bien plus que les autres.

Il y a par exemple les changements induit par les décisions de justice ou par des autorités administratives. Sachant que l'impossibilité de changer les processus mis en cause peut aussi causer des millions d'euros d'amendes en cas de non-respect de la loi, comme pour le nouveau Règlement Général sur la Protection des données.

Tout ceci illustre donc a quel point les entreprises du e-commerce ont besoin de flexibilité dans le monde actuel.

Afin de répondre à la question de la thèse, nous verrons donc premièrement les besoins du secteur du e-commerce, besoins auxquels il n'est pas toujours possible de répondre avec les outils habituels.

Deuxièmement, nous allons décrire de quelle façon les microservices et les bases de données NoSQL peuvent répondre à ces besoins et quelles sont les implications de ces choix.

Troisièmement, nous verrons qu'il faut néanmoins rester critique et vigilant, car elles ne peuvent répondre à tous les problèmes.

Quatrièmement, nous analyserons comment avoir une gouvernance flexible et éclairée de l'architecture, le tout afin d'avoir une vision claire du futur et des besoins clients.

Et en conclusion, mettre en pratique l'utilisation de microservices et de bases de données noSQL dans un contexte e-commerce.

2 Eléments de contexte : quelques définitions

Afin de répondre aux besoins sans cesse en expansion des outils informatique de vente en ligne et de leurs utilisateurs, de nouvelles technologies ont vu le jour au cours des dernières années. On y trouve notamment les bases NoSQL et les microservices.

2.1 Le secteur du e-commerce

Dans ce chapitre, nous verrons les spécificités du secteur du e-commerce, qui est un secteur en pleine croissance et très concurrentiel.

2.1.1 Des besoins très spécifiques

Le secteur du e-commerce est très spécifique. Il est connu pour être très concurrentiel et avoir de nombreux intervenants, avec les fournisseurs, les grossistes, les prestataires, les vendeurs de la grande distribution… En France, lors de l'année 2018, le secteur du e-commerce a fait un chiffre d'affaires de 92,6 milliards d'€ [1]. Cela représente une augmentation de 13,4 % par rapport à 2017 [1]. Le nombre d'acheteurs ne cesse de croître avec plus d'un million de nouveaux acheteurs par rapport à l'année dernière. On y retrouve tous les âges avec bien évidemment les moins de 35 ans, très présent sur mobile, mais aussi les plus de 65 ans dont plus de 80% achètent désormais sur Internet. À la suite du confinement, cet état de fait s'est accéléré en Avril 2020 avec en autre une augmentation de 45% des ventes en ligne pour l'enseigne Carrefour [2].

De ce fait, l'information se transmettant de plus en plus vite, il est nécessaire pour les professionnels du secteur de rester à niveau afin de répondre aux besoins de ses consommateurs. En effet, qui a décidé de ne pas acheter sur un site internet, car il était trop lent, ou peu sécurisé ? Les exigences des clients se faisant de plus en pressantes, il faut des sites aux catalogues variés, qui changent au fil du temps, avec le meilleur rapport qualité/prix… Dans ce secteur, une interruption de service est gravissime, car cela peut représenter des millions d'euros de chiffres d'affaires et une image entachée pour longtemps.

D'autre part, la pression internationale se fait sentir, notamment avec la présence grandissante d'Amazon dans de nombreux domaines de la vente en ligne. Il y a également des périodes de l'année pendant lesquelles il y a une forte activité, par exemple pendant les soldes, les fêtes de fin d'année, les blacks friday… Cela représente une période de demandes accrues ou certains professionnels font plus de la moitié de leur chiffre d'affaires annuel. Ceci cause cependant des problèmes comme nous allons le voir par la suite.

2.1.2 Des solutions standards coûteuses et parfois inefficaces

Pendant les périodes avec beaucoup d'activités et afin de maintenir un site e-commerce disponible de façon constante pour tous les utilisateurs, il y a plusieurs alternatives. La première, et la plus courante, consiste à augmenter momentanément la capacité du/des serveurs hébergeant le site internet ainsi que la base de données. En effet, un site internet avec plus de ressources peut supporter plus d'utilisateurs. Cependant, on atteint vite les limites logicielles du site internet. On appelle ce mode de fonctionnement la scalabilité verticale. Cela signifie aussi que dans le cas d'une entreprise qui détient ses serveurs, sans réutilisation du matériel, il y a une perte d'argent, car on doit racheter un nouveau serveur et l'ancien n'est plus utilisé.

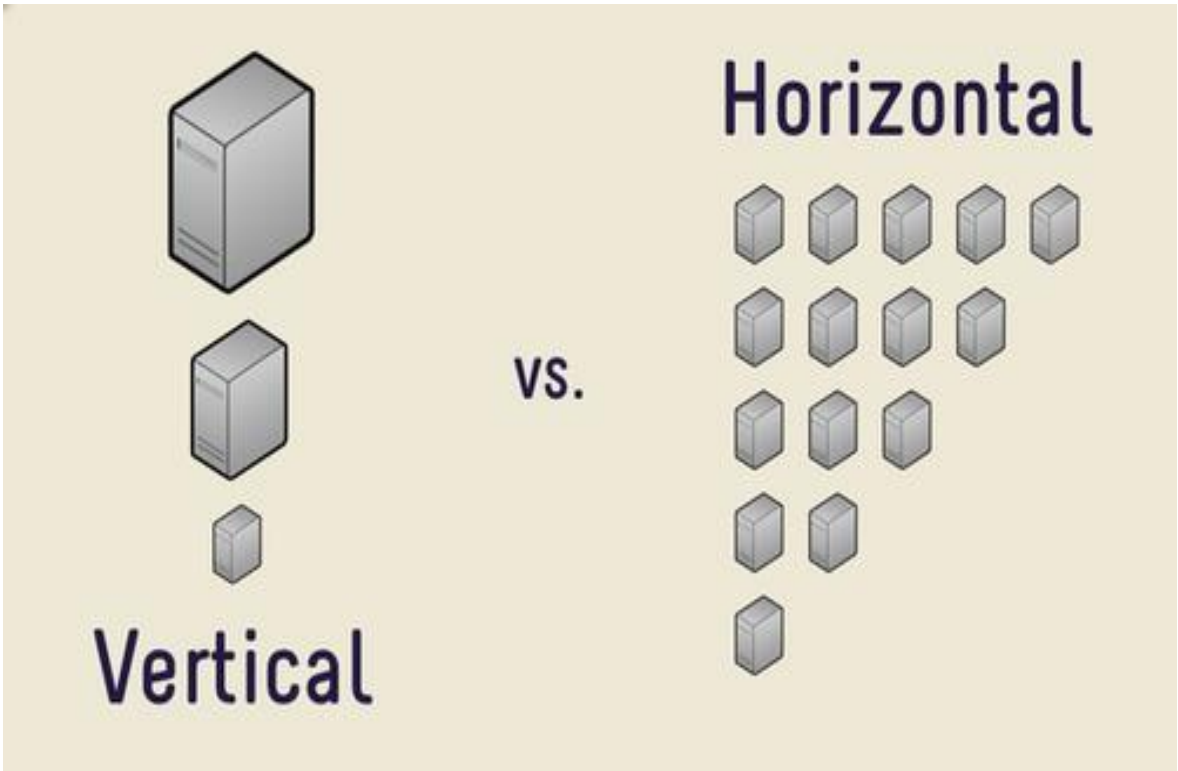

Figure 1 : explication des différents modes de scalabilité, site de Supinfo : https://www.supinfo.com

La seconde alternative consiste à mettre en place un système de cache des données. Ces systèmes de cache sont très divers et permettent de servir à l'utilisateur les informations du site internet à moindre coût pour le serveur et le tout de façon beaucoup plus rapide. Mais en faisant cela, on n'envoie plus forcément d'informations cohérentes aux utilisateurs. Dans le cadre du e-commerce, cela peut être particulièrement impactant. Par exemple, la page d'un produit est mise en cache avec le stock du produit lors de la mise en cache. De ce fait, la page peut afficher que le produit est en stock alors qu'il ne l'est pas, ou inversement.

Ainsi, lors du paiement, il y aura une vérification des stocks et l'utilisateur sera notifié qu'il ne pourra pas acheter le produit. Ou il n'y a pas de vérification du stock et la commande sera validée comme s'il y avait du stock ! Ce qui va causer des problèmes logistiques pour l'entreprise qui va devoir trouver en urgence du stock pour ce produit ou annuler partiellement la commande. Cela créée donc de la frustration pour l'utilisateur, qui risque de ne pas revenir sur ce site incapable de donner des informations cohérentes. En cas de problème fréquent de ce type, l'image de marque de l'entreprise propriétaire peut être fortement dégradée. Enfin, pour finir, cela génère évidemment un manque à gagner pour l'entreprise derrière le site internet.

Cet exemple du cache et de la gestion des stocks et un exemple parmi tant d'autres. Le cache est un outil très utile, mais qui doit être utilisé sur des données relativement stables dans le temps avec un temps de vie limité. Mais la seule utilisation du cache dans le cadre d'un pic d'activités démontre un manque de compréhension de son intérêt. Il est alors nécessaire de purger ce cache de façon régulière afin de supprimer les données incorrectes. Action qui n'est pas possible lors des périodes de pic d'activités. Le temps de recréer le cache lors d'un pic de charge rendrait le site extrêmement lent, voire même indisponible.

Encore actuellement, la plupart des sites de e-commerce dits grand public, basés sur des CMS de vente ou non, sont encore assez monolithiques. Une architecture monolithique n'est cependant pas du tout adaptée au contexte actuel de changement fréquent des usages. En effet, les composants d'une architecture dites monolithique sont interconnectés et interdépendants plutôt qu'associer de manière flexible comme dans le cas des programmes modulaires, tels que les microservices. On dit qu'il y a un fort couplage des composants. De ce fait, chaque composant étant fortement lié aux autres, il est difficile de modifier un élément sans impacter tout le reste. Cela signifie que tout changement est coûteux en temps et en argent.

Il est bien souvent difficile de savoir ou doit être situé le domaine métier d'une application. Sur une application en couches, cela est assez difficile, mais pour une application monolithique, cela l'est encore plus, car on ne peut pas vraiment avoir une vision globale et nette de l'ensemble.

Aussi, un site monolithique n'est généralement pas conçu pour avoir plusieurs instances de lui-même, ce qui le rend particulièrement vulnérable aux pics de charge ou aux attaques par déni de service. Ainsi, cela fait de ce type de site un point de défaillance unique, ou SPOF en anglais. Cela est un point auquel il faut remédier, car on l'a vu, le e-commerce a une part de plus en plus importante dans l'économie et l'indisponibilité du site peut causer des dizaines de milliers d'euros de perte de chiffre d'affaires.

Afin de permettre aux divers logiciels une bonne communication, il faut assurer l'interopérabilité. Celle-ci est malheureusement bien souvent absente des sites internet. En effet, bien qu'ils utilisent tous des standards internationaux tels que les langages et protocoles HTML, HTTP et CSS, de nombreuses fonctionnalités dépendent encore du navigateur et de l'ordinateur.

Cette interopérabilité pose aussi des problèmes quand il s'agit de faire des tests d'intégration. En effet, il est nécessaire de prendre des centaines de facteurs, tel que la taille de l'écran, le logiciel, la connexion de l'utilisateur… Cela implique le besoin de centaines de tests différents. Il est préférable, dans un souci de qualité, de tester le plus grand cas de figures possible qui ont un intérêt.

Quant aux bases de données relationnelles, elles fonctionnent relativement bien quand il y a quelques dizaines de milliers de lignes. Mais au-delà, il est nécessaire de faire des optimisations, tel que des indexations sur la base de données pour améliorer les temps de réponse sur les tables les plus fréquemment utilisées.

Cependant, dans le cadre du e-commerce, cela ne suffit généralement pas, car il y a souvent pour une enseigne des dizaines de milliers de références de produits, chacune ayant parfois des centaines de déclinaisons. Cela cause encore une fois des problèmes de performance auxquelles les bases de données relationnelles peinent a répondre.

Les bases de données relationnelles sont aussi basées sur une structure relativement monolithique. Il y a une instance unique de la base de données, qui est seule responsable de la donnée. Cela simplifie donc la gestion de cette dernière. En effet, les bases de données relationnelles ne sont pas conçues dans le but d'être dupliquées comme des bases de données non-relationnelles. Cela causerait des problèmes de performance, car il faut répercuter constamment les requêtes entre les instances.

Ce système basé sur une instance unique signifie qu'en cas d'indisponibilité de la base de données, alors le site e-commerce, l'ERP… Bref, tous les services qui en dépendent seront indisponibles. Cela créera donc un deuxième point de défaillance unique dans l'architecture. Dans une vision e-commerce des choses, il faut y remédier.

Pour résumer, le secteur est donc en pleine expansion avec un important nombre d'acteurs. Avant la crise du Covid-19 ce secteur gagnait des clients de façon très rapide cet état de fait continue de s'accentuer avec les modifications des habitudes de consommation [1] [2]. Cependant, les clients se montrent de plus en plus exigeants et les sites internet actuels ne permettent pas toujours de répondre parfaitement à leurs attentes.

De ce fait, dans ce secteur à la concurrence multiple, il est primordial de maintenir au mieux la qualité de service afin de pouvoir garder ses clients, qui auraient tôt fait de se détourner vers la concurrence. Nous verrons par la suite en quoi l'utilisation de microservices et de bases de données noSQL va permettre d'améliorer les plates-formes de ventes en ligne.

2.2 Outils et termes annexes

À mesure que le monde informatique se diversifie, de nouveaux outils apparaissent et avec eux, de nouveaux termes. L'un des premiers termes à apparaître et celui de pattern informatique. Il désigne dans le monde informatique un modèle, une structure, une organisation répétitive à laquelle il est possible de conférer des propriétés caractéristiques.

Avec la conceptualisation des patterns sont venus les anti-patterns, que l'on souhaite éviter. Ce sont des erreurs de conception courantes des logiciels, liées à l'absence ou la mauvaise utilisation des patterns.

L'un des patterns et anti-pattern le plus commun est la création de librairie, ou bibliothèque logicielle. Leur utilisation peut être très diverse : calcul, statistiques, affichage… Elles ont toute comme objectif de mettre des choses en commun afin de ne pas réinventer ce qui existe déjà. Leur création est un anti-pattern dans les cas ou il n'est pas pertinent de créer des librairies.

Cette volonté de mise en regroupement se retrouve aussi dans les clusters. Aussi appelés des grappes de serveurs, c'est un groupe de serveurs indépendants qui fonctionne comme un seul et même système. Cela permet de réduire les coûts, car monter en gamme un serveur s'avère vite coûteux. D'autre part, les clusters améliorent la sécurité, car si l'un des serveurs tombe en panne, les autres peuvent compenser.

Dans l'informatique, justement, les clusters sont très utilisés pour maintenir la haute disponibilité. Celle-ci permet de garantir la bonne organisation des applications ou services constamment. Pour ce faire, on s'assure de la répartition des charges et la tolérance aux pannes.

Afin d'avoir une répartition des tâches efficaces, un load balancer est nécessaire, qui va diviser les appels venant de l'extérieur et les répartir entre les différents serveurs et applicatifs présents de façon équitable.

Ces appels seront par exemple ensuite envoyés aux frontweb, qui sont des serveurs applicatifs dédiés à l'affichage des informations. Celui-ci peut avoir plusieurs instances afin de gérer plus de demandes de la part des utilisateurs.

De plus, la gestion de la demande et de l'attribution des moyens informatique est géré par un orchestrateur. On peut citer parmi eux Kubernetes, Red Hat Ansible Automation Platform et Slat. En informatique, l'orchestration décrit un processus automatique de coordination, gestion et organisation de logiciels. L'orchestration permet entre autre de créer de multiples instances d'un même logiciel. Cela permet de diviser la charge entre les différentes instances. Mais l'orchestration décrit aussi le processus automatique d'organisation, de coordination, et de gestion de systèmes informatiques complexes, de middleware et de services. Cela est devenu une approche DevOps très fréquente dans le monde informatique.

Le terme DevOps définit un mouvement apparu en Belgique en 2007 [3]. Celui-ci est aussi une pratique visant à unifier le développement des logiciels et l'administration des infrastructures informatiques. Celui-ci vise notamment à promouvoir l'automatisation, notamment du suivi, des tests, déploiement nécessaire au projet.

Ce terme tend à se répandre dans les différents systèmes d'information a travers le monde. Ceux-ci sont des ensembles organisés de ressources informatiques. Leur but est de collecter, stocker, traiter et distribuer de l'information afin de répondre à des tâches précises.

L'exécution de ces tâches est faite par des logiciels. Certains de ces logiciels sont créés grâce à des Framework, qui sont des ensembles cohérents de composants logiciels structurels afin d'établir les fondations de ces logiciels.

Ces derniers peuvent utiliser divers moyens de communication, comme par exemple l'architecture REST (representational state transfer). Celui-ci est un style d'architecture logicielle dont le but est de définir un ensemble de contraintes à utiliser afin de pouvoir créer des services web efficaces. REST est l'un moyen de communication utilisé sur Internet par les microservices.

Parmi les autres moyens de communication sur Internet, il y a, entre autre, RMI (Remote method invocation), ORB (Object Request Broker)…

Ces moyens de communication peuvent être utilisés pour du reporting. Le reporting consiste en la génération de rapports sur les résultats et les activités d'une organisation. L'objectif étant d'en sortir des conclusions afin de pouvoir établir des actions, si nécessaire.

Le reporting est facilité par l'interopérabilité des systèmes. En effet, il est plus simple d'obtenir des informations cohérentes et concises quand il n'y a pas d'obstacle à leur communication.

Dans le monde informatique basé sur cette communication, il est nécessaire d'éviter les attaques informatique, ou du moins de les ralentir et d'en limiter l'impact. Il s'agit là de l'objectif de la défense en profondeur, terme à l'origine militaire qui consiste à exploiter plusieurs techniques de sécurité afin de réduire les risques dans le SI. Cela signifie de sécuriser tous les sous-ensembles du SI.

Dans les SI, on peut parfois avoir une certaine résistance au changement celle-ci est définit généralement de la façon suivante : « *Attitude consciente ou inconsciente d'un individu ou d'une personne morale qui l'incite à refuser toute modification/évolution de son état actuel. Elle reflète la peur de l'inconnu et le risque de se placer dans un état de dissonance cognitive. Motivée le plus souvent par la crainte de la perte d'acquis, la résistance au changement, si elle est généralisée au sein de la population cible, peut constituer un frein à l'introduction de nouveaux produits.* » Définition de la résistance au changement selon e-marketing.fr [4].

2.3 Du SQL au NoSql

Les bases de données NoSQL signifient littéralement en français pas uniquement langage de données structurées. Celles-ci sont un type de base de données qui n'appliquent pas le même paradigme de stockage de données que les bases de données SQL dites relationnelles. Une base de données SQL est constituée de tables, eux-mêmes faites de colonnes définies avec certaines propriétés (identité, unicité, nullabilité ...). De plus, dans une base de données SQL, les données sont dites normalisées. La normalisation est un processus qui permet d'optimiser le modèle logique présent en base afin de le rendre non redondant. Ce processus conduit à la fragmentation des données dans plusieurs tables. On dit aussi que les bases de données SQL répondent aux propriétés ACID [5] (atomicité, cohérence, isolation et durabilité). Ceux-ci sont un ensemble de propriétés garantissant la fiabilité de l'exécution de la transaction informatique.

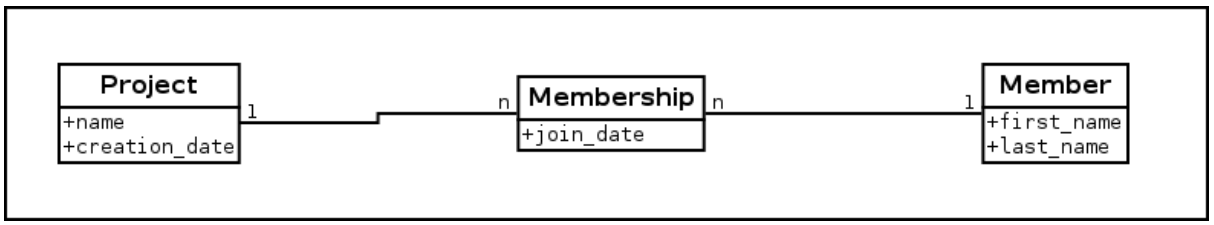

Figure 2 : Exemple de structure de données au sein d'une base de données relationnelle

Au contraire, dans une base de données non relationnelle, la donnée n'est pas organisée avec un système de tables et de colonnes et peut être dénormalisée [6]. L'objectif de la dénormalisation est d'améliorer les performances de la BDD sur les tables dénormalisées, en implémentant les jointures plutôt qu'en les calculant. De ce fait, les bases de données NoSQL ne répondent pas aux propriétés ACID exposées précédemment. Elles répondent cependant au théorème CDP [7], ou théorème de Bewer. Celui-ci stipule qu'il est impossible sur un système informatique de calcul distribué de garantir de manière synchrone les trois contraintes suivantes :

- la cohérence des données. Cela signifie que tous les nœuds du système informatique reçoivent exactement les mêmes données de manière synchrone ;
- la disponibilité, qui garantit que toutes les requêtes recevront une réponse,
- la tolérance au partitionnement : aucune panne moins importante qu'une coupure totale du réseau ne doit pas empêcher le système de répondre correctement. Cela signifie donc qu'en cas de morcellement en sous-réseaux de machine physique, chacun d'entre-eux doit pouvoir continuer à fonctionner de manière autonome.

De ce fait, les bases de données NoSQL englobent aussi les bases SQL, qui répondent aux contraintes de disponibilité et cohérence. L'un des principaux avantage des bases de données NoSQL est la rapidité de lecture, qui est un atout important dans le cadre du commerce en ligne. Mais, tout comme les microservices, cela permet aussi de choisir une base de données dans le but de spécialiser cette dernière pour une utilisation bien spécifique.

L'un des grands avantages des bases de données NoSQL et des microservices, consiste en leur capacité à mettre en place une forme d'élasticité. S'il est soudainement nécessaire d'avoir quatre instances d'un microservices au lieu de deux parce qu'il y a un soudain pic de charges, l'orchestrateur va automatiquement en créer deux nouveaux. De même, quand le pic est passé, en fonction de la puissance nécessaire, l'orchestrateur arrêtera le nombre nécessaire d'instance.

Idem pour les bases de données NoSQL qui sont particulièrement pertinentes dans le cas de gros besoins de stockage en base de données. Cela permet de répondre aux problèmes exposés précédemment. Mais cela permet aussi l'utilisation de toutes nouvelles fonctionnalités proposées par les bases de données NoSQL et les microservices.

2.4 Du Monolithe aux microservices

Au tout début de l'informatique, de nombreux développements ont été qualifiés de code spaghetti. On doit ce nom au fait que les éléments constituant les logiciels étaient extrêmement liés entre eux, on appelle cela le couplage. Le fait de vouloir modifier un élément dans un tel ensemble revenait à trouver et modifier un spaghetti spécifique dans un plat, le tout sans bouger les autres. Au final, les systèmes étaient monolithiques, extrêmement peu lisibles… Il y avait également beaucoup de codes dupliqués car l'utilisation de librairie annexe étant encore peu répandue. Bien que le code spaghetti existe encore de nos jours, il se fait heureusement de plus en plus rare.

Puis est apparu en 1978 l'un des tous premiers patterns, le modèle MVC [8], pour Modèle Vue Contrôleur. Celui-ci est assez simple, très populaire dans les applications web et permet de développer plus facilement. Il s'agit d'une des premières répartitions des tâches dans les logiciels informatique. Cette répartition des tâches et extrêmement importantes. En effet, si l'on veut pouvoir être capable de comprendre un programme et de le modifier, il est nécessaire de bien diviser les tâches pour que chaque composant fasse son travail au mieux.

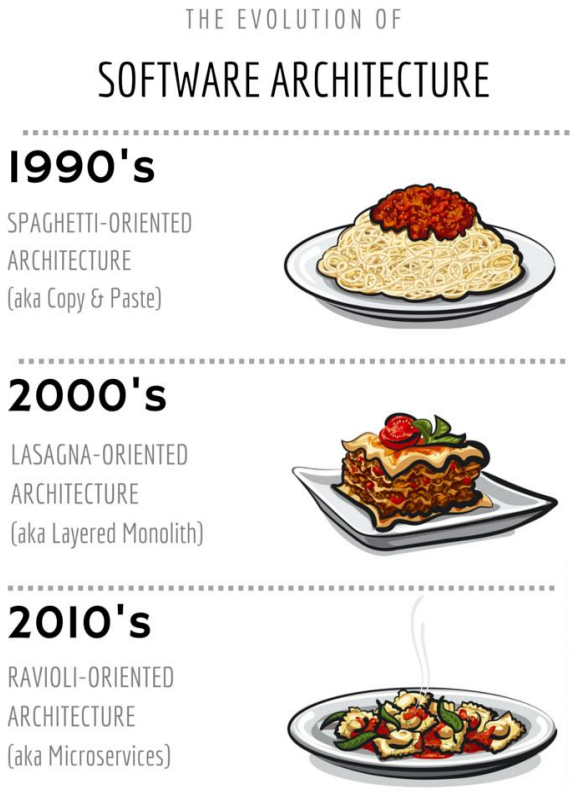

Figure 3 : L'évolution de l'architecture logicielle, site de Benoit Ediard : https://medium.com/@benorama/the-evolution-of-software-architecture-bd6ea674c477 [9]

Les logiciels conçus selon le pattern MVC [8] et les modèles n-tiers sont souvent qualifiés de logiciel avec une architecture en mille-feuilles. Chaque couche ayant un ensemble de tâches. Cela simplifie le développement et l'évolution future, cependant, chaque couche doit être compatible avec les couches adjacentes. Le couplage est réduit, mais toujours présent, quant à l'interopérabilité elle est ainsi améliorée, mais incomplète. On arrive aussi à un autre niveau de spécialisation des tâches, qui se verra accentué avec les microservices.

L'approche par microservices reprend aussi le principe de division des tâches, qui a été entre autre, appliquée par la suite de protocoles internet TCP/IP. En effet, il est plus simple de gérer plusieurs petits problèmes autonomes qu'un ensemble de problèmes entremêlés.

Les microservices, quant à eux, sont basés sur une technique de développement logiciel qui permet de structurer l'ensemble d'une architecture comme un groupement de services faiblement couplés. Chaque microservices remplit une tâche spécifique, ce qui permet une spécialisation très poussée. Chaque microservices est donc indépendant des autres et il est possible d'avoir une ou plusieurs instances d'un service.

De ce fait, pour communiquer, les microservices utilisent des API indépendantes du langage de programmation. Cela permet donc d'avoir des microservices simples à créer et modifier. De plus, grâce à des outils d'orchestration, la création de nouvelles instances de microservices afin de pouvoir répondre automatiquement à la demande se fait de façon automatique.

3 Exploitation fonctionnelle des micro-services combinés aux bases de données noSQL dans le cadre du e-commerce

Les microservices et les bases de données NoSQL étant de nouveaux outils, ils ont donc de nouvelles implications fonctionnelles. Ce qui nécessite de faire des choix afin d'avoir les outils les plus adaptés aux besoins. Cela donnera plusieurs avantages fonctionnels et permettra de nouvelles fonctionnalités. Cependant cela ne gère pas tous les problèmes des SI et il y aura plusieurs éléments de méthodologie à appliquer pour utiliser correctement ces nouveaux outils.

3.1 Implications fonctionnelles

3.1.1 Choix des technologies

Comme vue précédemment, il est possible de créer des combinaisons avec le bon Framework ou langage ainsi que la bonne base de données. Le tout dans le but de créer un microservice qui réponde au mieux à une fonctionnalité spécifique.

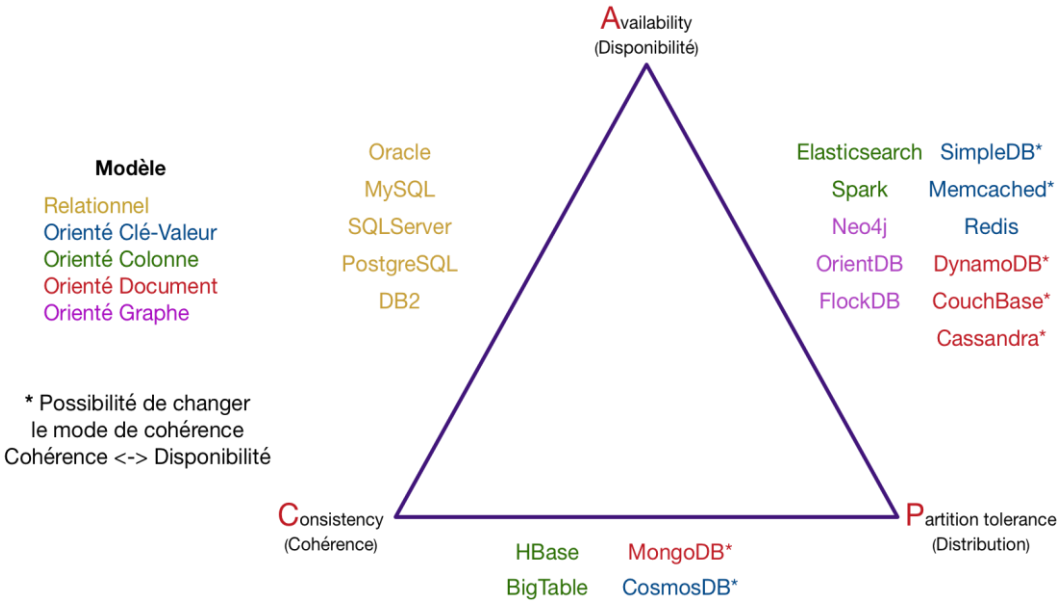

Figure 4 : Application du théorème CDP par rapport aux bases de données présente sur le marché, site d'OpenClassrooms : https://openclassrooms.com

Dans le but de choisir la bonne base de données NoSQL, on a besoin d'utiliser le théorème CDP [10] et de connaître les grandes familles de bases de données non relationnelles. Il est nécessaire aussi de savoir quel langage ou Framework est le plus pertinent d'utilisation. Cela dépend évidemment des compétences que l'entreprise a dans le domaine. Néanmoins, l'application et la mise en application étant rapide, il y aura peu de temps perdu et beaucoup de choses d'apprises.

De ce fait, comparons les différents avantages des couples de propriétés. Premièrement, un système cohérent et disponible est pertinent pour une application qui nécessite des données consistantes et disponibles, mais qui n'a pas besoin de modifier le volume de données disponible. La plupart des bases de données relationnelles traditionnelles sont des systèmes avec ses propriétés.

Deuxièmement, un système disponible et tolérant au partitionnement permet d'être certain que la base de données sera toujours disponible pour répondre et gérera les problèmes de volumétrie. Cependant, il n'est pas possible de garantir la consistance des données. Les bases de données ElasticSearch, Redis et DynamoDB sont des exemples.

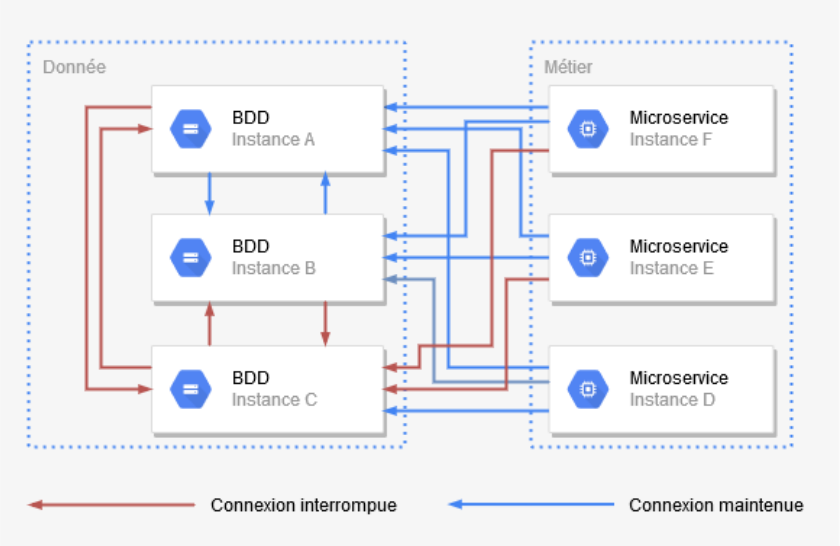

Figure 5 Schéma expliquant l'impact sur les données d'une interruption partielle du réseau

De ce fait, si on prend l'exemple d'une base de données NoSQL dont les instances sont réparties sur trois serveurs, que l'on nommera A, B et C. Cette base de données est basée sur une gestion paire à paire des données. Chaque instance a les données et réplique les changements sur les autres instances. Il y a aussi un microservice ayant trois instances répartis de la même façon, que l'on nommera D, E et F. Si une coupure réseau empêche les instances E et F du microservice d'accéder à l'instance C de la base et que celle-ci ne peut communiquer avec les autres instances de base de données, alors, seule l'instance D peut communiquer avec C. Cela signifie donc que si l'instance D effectue des actions d'écriture sur C, alors il y aura un conflit de données entre les instances de base de données. En effet, les bases de données de ce type ont un système de synchronisation des données. Cependant cela ne peut fonctionner en l'absence de connexion réseau. Afin de gérer cela, les bases de données NoSQL ont un système de gestion des conflits qui se déclenchera dès que la connexion réseau sera rétablie.

Troisièmement, un système consistant avec une tolérance au partitionnement permet d'avoir des données cohérentes, une adaptabilité au niveau du volume de données et une résilience au niveau réseau. Cependant, toutes les requêtes n'auront pas forcément de réponse et de ce fait, la disponibilité n'est pas assurée. Les bases de données BigTable, MongoDb [11] et HBase sont des exemples.

Toutefois, il reste encore à faire le choix du type de format de la base de données. En effet, les formats diffèrent en fonction des fonctionnalités recherchées.

Il y a tout d'abord le format de données orienté colonne. Chaque donnée y est stockée dans une colonne, avec un seul type de données et un ou plusieurs identifiants, ce qui permet une compression des données optimale. Comme on peut voir dans la figure suivante, les entrées de chaque colonne sont liées entre elles par les identifiants. Finalement l'ensemble sera utilisé pour être agrégé en plusieurs lignes, chaque ligne étant l'accumulation des données, regroupées par identifiants. Cela permet de savoir que l'ordinateur PC Asus coute 1200 euros et a un Intel I7 comme processeur. Il est donc extrêmement simple de rajouter une colonne ou de la supprimer, car cela n'impacte en rien les autres. Ceci permet un changement rapide du volume de données. Cependant, la modification du format d'une colonne nécessite parfois de faire des modifications sur des millions d'entrées de cette colonne, ce qui est long et couteux. Ce format représente donc un outil parfait pour faire du reporting, car ce type d'outil a besoin de faire des analyses sur des grandes lignes de données avec des volumes très importants.

Nom	Id
PC Toshiba	1
PC Asus	2
PC Lenovo	3
PC Dell	4

Prix	Id
1000	1
400	3
1200	2
1300	4

Processeur	Id
Intel I5	1
Intel I7	2,4
Intel i3	3

Figure 6 Exemple de format de base de données orienté colonne

Un autre format de base de données qui est utile pour stocker des grands volumes est le format clé-valeur. Ce type de format peut stocker n'importe quoi : des objets, des valeurs numériques, du texte… Chaque entrée sera identifiée par un identifiant et une valeur. Sa simplicité permet aussi une scalabilité très poussée et des requêtes extrêmement rapides.

Certaines bases de données orientées clé-valeur sont utilisées pour la mise en cache, tel que Redis. Ces dernières peuvent être paramétrées pour stocker leurs données en RAM, permettant des requêtes presque instantanées. L'une des utilisations les plus fréquentes est la mise en cache de tout ou partie des éléments les plus utilisés des sites internet, tel que la page d'accueil, ou les pages des produits les plus vues.

Les bases de données orientées document [12] [11] au contraire, sont légèrement plus complexes et se basent sur le fait que chaque document ait une structure commune, mais néanmoins variable. Par exemple, chaque produit a toujours, en autre, un nom et un prix, cependant, ils peuvent aussi avoir d'autres informations qu'il n'est pas possible de déterminer à l'avance. Ainsi, une ramette de papier aura des informations sur la couleur du papier et sa quantité, tandis qu'une boite d'œufs aura des informations sur la date de péremption et la date de ponte. Ce format est donc utile quand il n'est pas possible de prévoir à l'avance tous les cas. Ce qui arrive très souvent dans le domaine du e-commerce à cause des produits extrêmement variés. Cela permet donc d'avoir une flexibilité très poussée. Ainsi, le fait qu'une entreprise se reconvertisse ne signifie pas qu'il faille forcément repenser l'ensemble de la base de données.


```
{
    "id": "1",
    "name": "Boite de 100 clous",
    "gtin": "1234",
    "price": {
        "value": "1.99",
        "currency": "EURO"
    },
    "targetCountry": [
        "FR",
        "BE",
        "LUX"
    ]
}
```

Figure 7 Exemple de document stocké en base de données

Le dernier type de format est celui du format orienté graphique [13] [14]. Celui-ci est le plus complexe. Il est utilisé afin de stocker des relations entre des entités. Les entités peuvent être aussi bien des entreprises, des personnes ou des concepts. Ce format est pertinent dans le cas de données qui est fortement interconnecté. D'après le Modèle de Graphe Attribué [13], un graphe est modélisé grâce à trois blocs de base :

- le nœud, ou sommet, qui représente une entité et ses données,
- la relation où arrête, qui a une orientation entre deux nœuds ainsi qu'un type,
- la propriété ou attribut, porté par un attribut ou une relation.

Du fait de sa spécificité, ce type de format est relativement peu utilisé actuellement. Il est pertinent pour gérer des arborescences, par exemple avec des catégories et sous-catégories, ou encore des graphes sociaux en vue de publicité ciblé.

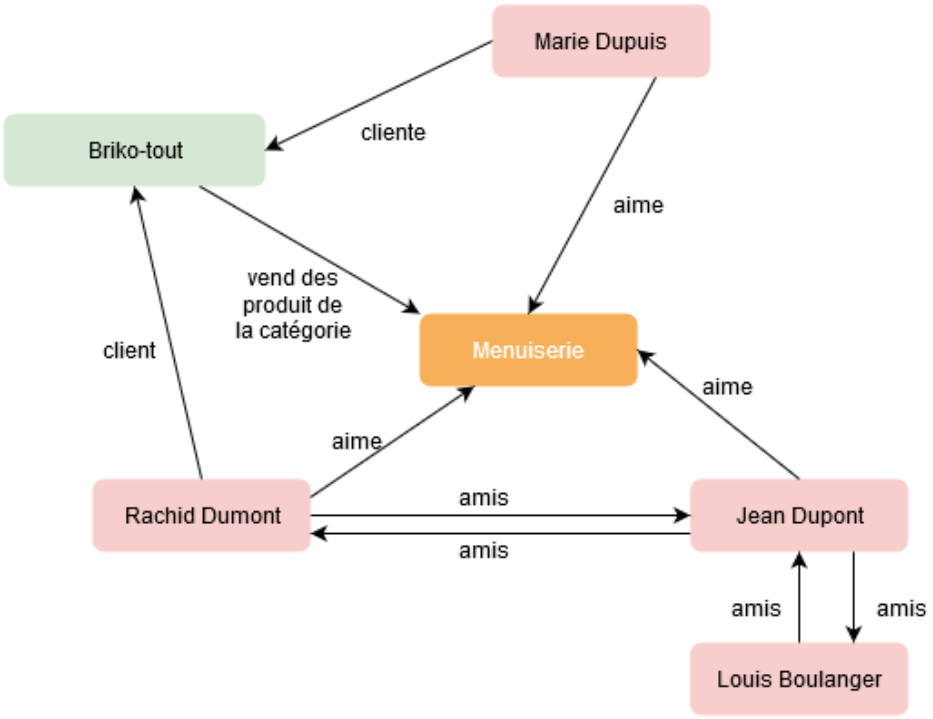

Figure 8 Schéma explicatif de l'application des bases de données de type graphique

Comme on peut le voir dans le schéma précédent, il y a deux personnes qui aiment la menuiserie et qui sont clients de Briko-tout, enseigne qui vend des produits de menuiserie. Cette enseigne souhaite agrandir sa clientèle. Or, on voit aussi une relation d'amitié entre l'un des clients et Jean Dupont qui lui aussi apprécie la menuiserie. Ce serait donc probablement un client potentiel pour Briko-tout, qui pourrait lui proposer des publicités ciblées. De même, au vu de sa relation avec Jean Dupont, on pourrait aussi supposer que Louis Boulanger aurait les mêmes centres d'intérêt.

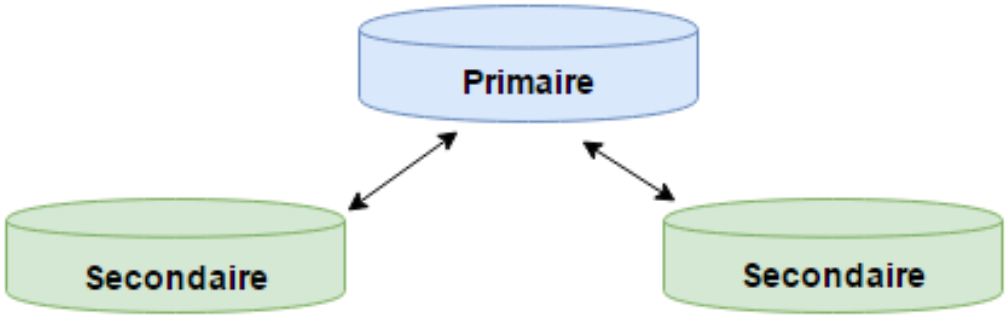

Figure 9 Replicat set de base de données MongoDB, site de Supinfo : https://www.supinfo.com

Les bases de données NoSQL ont aussi plusieurs modes de sauvegarde des données. La plus fréquente est le mode maitre-esclave. C'est l'exemple de la figure précédente, ou l'on voit un replicat set, c'est-à-dire un groupe d'instances, appelées des nœuds, qui maintiennent un même ensemble de données. Le nœud primaire est ici le maître et les nœuds secondaires sont les esclaves. Le nœud primaire est le seul à recevoir les requêtes d'insertion, modification et suppression, qu'il effectue puis les transmet aux nœuds secondaires si la requête en question est correcte. Cependant, l'ensemble des nœuds peut servir pour faire des requêtes de lecture en base de données, ce qui permet d'équilibrer la charge.

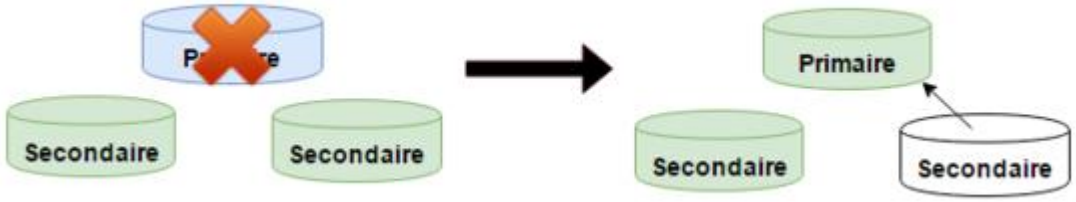

Figure 10 Election d'un nœud primaire en cas d'indisponibilité du nœud primaire, site de Supinfo : https://www.supinfo.com

Ainsi, pour maintenir la haute disponibilité des données, il y a toujours au moins deux nœuds secondaires qui peuvent prendre le relais du nœud principal si celui-ci venait à avoir un problème. Tout ceci se fait de façon automatique et par la suite, le contenu du nouveau nœud primaire sera dupliqué dans un nouveau nœud secondaire.

Une autre mode de sauvegarde est la sauvegarde pair-a-pair. Comme le fameux réseau de transfert de fichier, dans celui-ci également, chaque instance de la base de données à un duplicat des données, et effectue les opérations en parallèle des autres. Quand une modification est effectuée, toutes les autres instances sont notifiées et jouent à leur tour la requête liée. Si jamais deux requêtes rentrent en conflit, comme par exemple la modification d'un produit par deux personnes, un système de gestion des conflits se déclenche automatiquement.

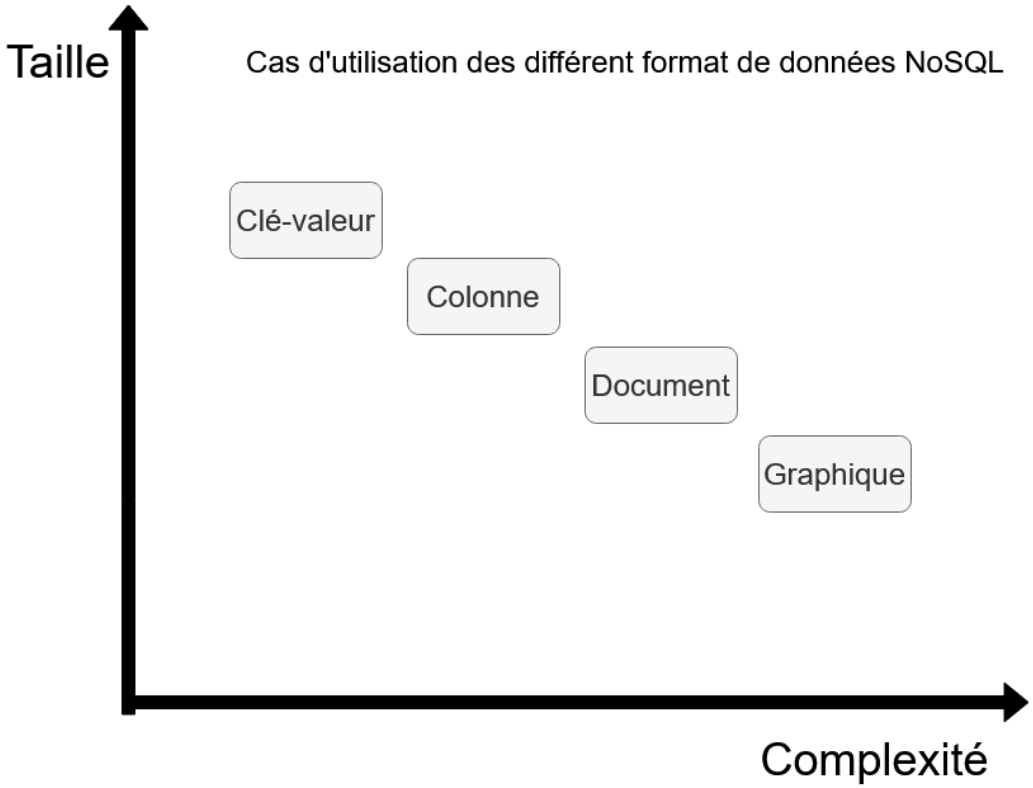

Figure 11, graphique représentant les cas d'utilisation des différents formats de base de données NoSQL

On peut donc conclure que les bases de données NoSQL sont extrêmement diverses, tant du point de vue des fonctionnalités, du mode de sauvegarde, que des formats disponibles. Cela permet donc d'avoir des outils parfaitement dédiés à certaines situations, tel que l'analyse des données client pour améliorer la pertinence des publicités. En fonction des besoins, il sera ainsi possible de trouver précisément la ou les bases de données adaptées. En effet, il est par exemple possible de stocker des données magasins dans MongoDB et de les mettre en cache dans REDIS.

3.1.2 Bonnes pratiques

Les bonnes pratiques sont vitales afin de garder le système d'information opérationnel. Celles-ci peuvent dépendre des entreprises et de leur contexte.

Il est nécessaire de mettre en place avec la ou les équipes concernées un standard. Pour cela, il faut faire des ateliers où l'on rappelle ce qui fait un bon microservice, qui utilisera ce standard. Celui-ci doit avoir un champ d'application réduit, au-delà de quoi ce ne serait plus un microservice. Pour ce faire, un microservice doit avoir une logique métier délimité dans le cadre de l'entreprise.

Evidemment, il ne faut que celui-ci soit trop simpliste, car cela risque de causer des problèmes de performance. Par exemple, il peut être pertinent que le microservice de gestion des produits gère aussi les différentes déclinaisons de ces produits. Toutefois, la notion de produit est assez vaste, et si l'entreprise a un entrepôt central, il peut paraître pertinent de gérer les stocks dans le microservice des produits. Par contre, on prend le risque de devoir modifier le microservice par la suite car cela fait un grand domaine métier.

Pour citer Ben Christensen de Netflix à propos du système informatique dans son ensemble, « *Il faut que ce soit un système cohérent composé de plusieurs petits éléments avec des cycles de vies autonomes qui finissent par formés un tout* » [15].

Le standard va servir à garder une cohérence technique dans l'ensemble des applicatifs. La première chose à laquelle il faudra penser est comment vont communiquer les applications entre elles ? Il est pertinent d'en avoir peu afin de ne pas se perdre dans les différents types de flux de données.

Ce standard doit également inclure les technologies et les langages qui formeront le socle du projet. Cela est nécessaire, car tout comme les flux de données, plus il y en aura, plus cela sera difficile de réunir les compétences nécessaires en cas de changement. Le géant américain de la vidéo en ligne Netflix a par exemple décidé que le standard pour toutes ses bases de données seraient Cassandra. Netflix a fait ce choix, car Cassandra s'accorde particulièrement à ses besoins de latence, de scalabilité et parce que ses équipes ont les compétences nécessaires dans le domaine.

Il est aussi absolument nécessaire de mettre en place une surveillance de l'ensemble des applications. En effet, sans celle-ci, il est impossible d'être au courant de l'état du SI. Cette surveillance devra se faire aussi grâce à un standard d'implémentation commun. L'objectif est de pouvoir visualiser l'ensemble des données afin de créer des statistiques.

Pour ce faire, des outils tel que ELK, qui combine ElasticSearch, Logstach et Kibana permettent le stockage et l'analyse des journaux d'appels, appelés aussi logs d'erreur. Cela permet par exemple, de remarquer une baisse ou augmentation des erreurs à la suite d'une mise en production récente. Cette surveillance permet aussi de s'assurer de la solidité et de la sécurité des logiciels déployés.

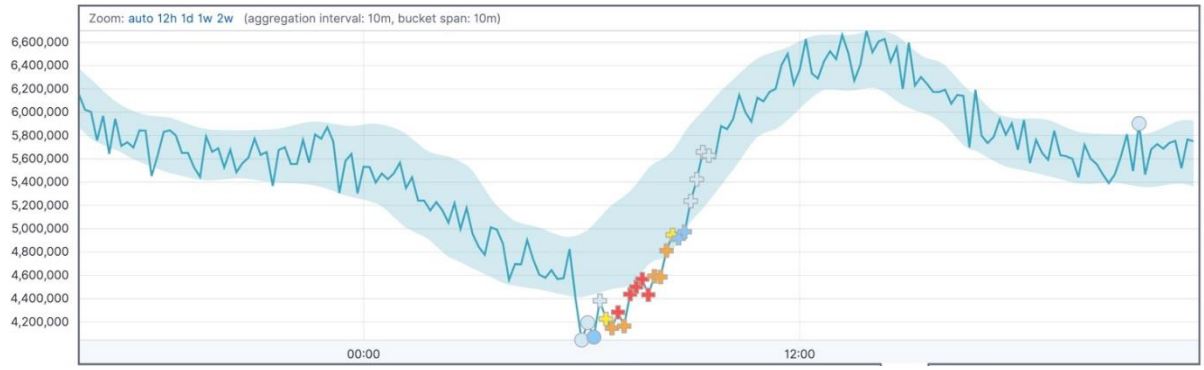

Figure 12, imprime-écran d'un graphique d'analyse ElasticSearch ayant détecter une anomalie dans le trafic, source Elastic : https://www.elastic.co/guide/en/kibana/current/xpack-ml-anomalies.html

Comme on peut le voir sur la figure précédente, il y a une anomalie du nombre de clics entre 7h et 9h. Le système d'alerte a remarqué au vu des appels précédent cette anomalie et a pu ainsi prévenir les équipes.

Ainsi, celles-ci ont pu intervenir dans les plus brefs délais. Ses anomalies ne sont pas forcément des choses négatives. Bien souvent, à la suite d'une interview télévisée par exemple, les visites d'un site peuvent augmenter significativement. Cela permet donc aussi d'analyser l'impact d'un point de vu marketing et aussi afin de s'assurer de la bonne santé du système d'information.

Les microservices étant interconnectés entre eux mais aussi avec le reste du SI, il est nécessaire de s'assurer que l'ensemble réponde correctement. Par exemple, si le standard d'intégration/communication des données est HTTP/REST, il faut s'assurer que les codes de retours sont bien ceux attendus. Il sera entre autres nécessaires de s'assurer que les logiciels ne confondent pas les erreurs 4XX et 5XX.

Comme tout logiciel, les microservices peuvent donc avoir des erreurs. Ou le réseau peut créer des erreurs, car la bande passante n'est pas infinie et la latence n'est pas nulle. Surtout dans le cadre du e-commerce qui connaît souvent des périodes particulièrement chargées. Pour remédier a ceci, en fonction des besoins de l'entreprise, le standard peut inclure des patterns, comme par exemple le design pattern circuit breaker.

3.1.3 Maintenir la cohérence métier

Dans le cadre de la création de microservices, il peut être complexe de garder la cohérence métier. Pour ce faire, afin d'éviter de devoir faire des modifications inutiles, il est nécessaire de déterminer un contexte métier bien délimité. En ce sens, le plus simple est de faire, en équipe, des schémas des grands contextes qui seront nécessaire [15]. Si ceux-ci forment des intersections, peut-être faut-il définir un contexte métier intermédiaire, lié à d'autres. L'objectif final est d'avoir un schéma avec des séparations nettes regroupant les différents contextes métiers. En fonction de la pertinence et de la faisabilité, certains seront regroupés ou non.

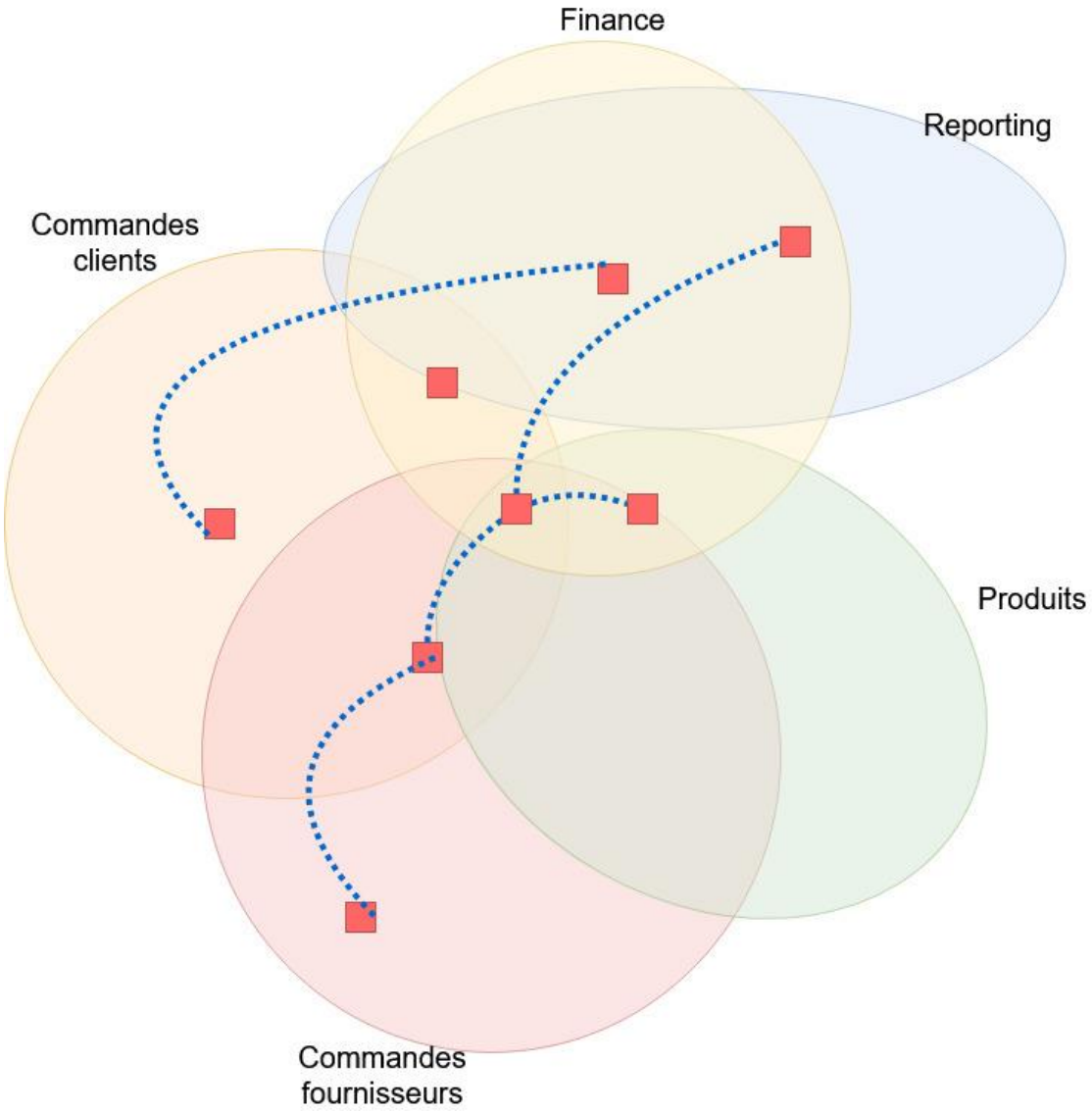

Figure 13, schéma de Venn des différents domaines métier que gère un SI fictif

On voit sur le schéma de Venn précédent que tous les domaines s'entrecroisent. Il y a par exemple le reporting avec les commandes clients et les finances, car le reporting analyse les commandes clients dans le but de donné des rapports pour le domaine financier.

Les produits, les commandes clients et fournisseurs s'entrecroisent aussi. Cela fait apparaître une notion de stock des produits, ou encore d'offre produits. Car ici, les produits seront achetés par les clients, mais il faut aussi les commander. Et cela est géré par le domaine financier.

De ce fait, le schéma précédent nous informe qu'il faut définir de façon plus fine les grands domaines métiers. Pour cela, il est nécessaire de faire d'autres schémas, et de noter les interactions entre les domaines métiers. Chaque contexte métier aura une description, ainsi que ses objectifs, besoins ainsi que les interactions nécessaires afin de les remplir.

Ses contextes métiers seront les bases de réflexion pour les microservices à venir. Ayant été défini précisément à l'avance, chaque contexte métier précis aura une description précise. Cela inclut aussi :

- les besoins de chaque contexte métier,
- les modes d'interactions entre les contextes,
- les fréquences d'interactions,
- la technologie et le schéma de sauvegarde le plus pertinent au vu des besoins et des compétences,
- le niveau de criticité,
- les inconnus au moment de la création,
- toutes autres informations pertinentes au vu du contexte métier.

Toutes ses informations sont nécessaires afin de pouvoir avoir une vision globale du chantier que forment les changements à venir. Cela permet aussi d'avoir une gestion de projet efficace, car certains éléments métiers dépendent les uns des autres. Par exemple, il n'est pas possible d'afficher des produits s'il n'y a pas de système de gestion de produits.

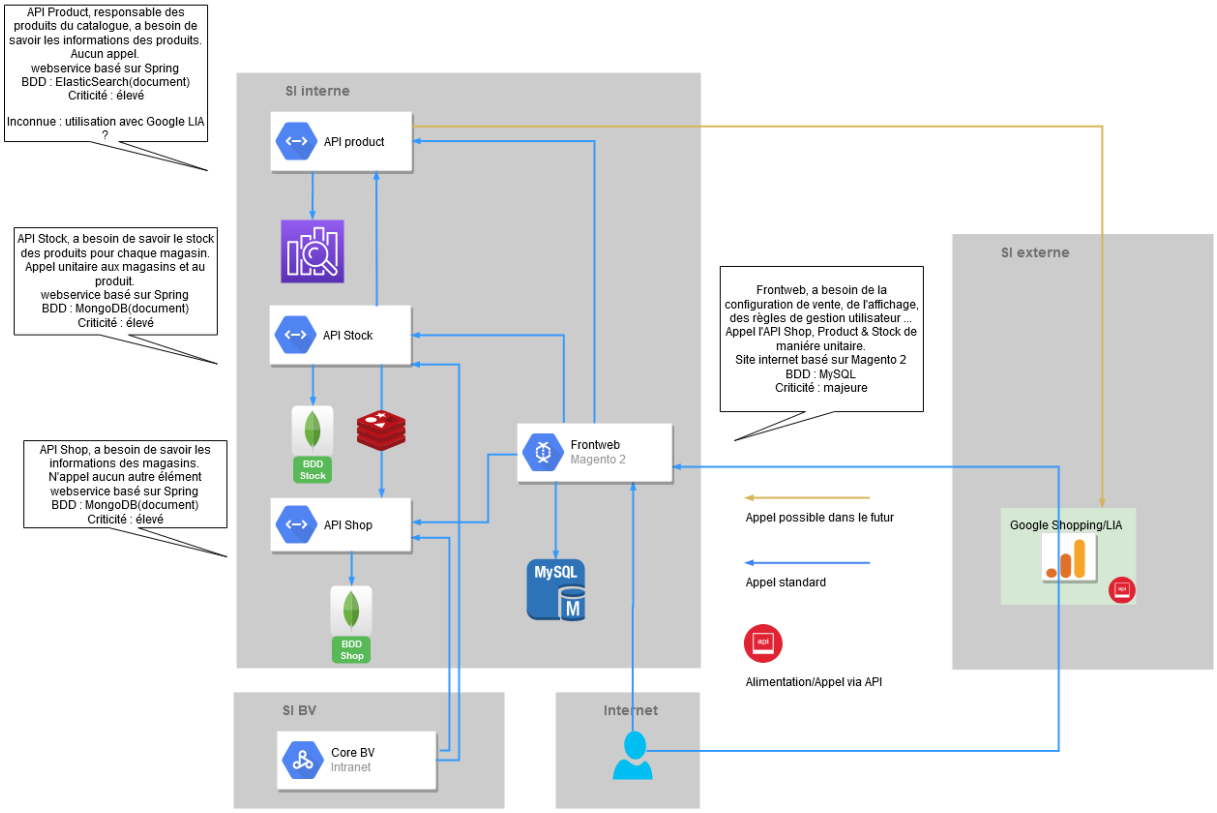

Figure 14, schéma simplifié des contextes métier appliqué avec une partie des éléments du SI nécessaire à l'affichage et la vente des produits

Suite à ce schéma, on remarque de nombreux liens entre les grands domaines métiers. Ses liens forment les interactions entre chaque domaine. Ils sont nécessaires afin de garder une séparation nette entre ces derniers. Lors de la création du schéma, il est nécessaire de s'interroger sur le standard de communication nécessaire. Ce questionnement est particulièrement important, car comme vue précédemment dans les bonnes pratiques, la standardisation est nécessaire dans le but d'avoir une bonne compréhension globale du SI.

Ses standards peuvent être des librairies préexistantes, mais dans ce cas leur implémentation doit être en cohérence avec les bonnes pratiques.

En observant le schéma de Venn, on remarque aussi que certaines entités métiers, comme les produits devront être présent dans plusieurs domaines métiers. En effet, les commandes fournisseur ont besoin des produits pour savoir quels produits commander, mais le système d'envoi aussi. Cependant, les informations demandées ne sont pas les mêmes, et seront probablement modifiées. On va par exemple faire une marge de 5 % sur les chaises, ce qui évidemment augmente leur prix de vente au consommateur final. Et de ce fait, chacune de ses entités métier, aussi appelées modèles, sont dépendantes des contextes métiers.

Ainsi, pour garder une cohérence métier maximale, il ne faut absolument pas dupliquer de code du microservices de gestion des produits dans celui de gestion des commandes. Et il ne faut pas non plus en faire une librairie commune aux deux microservices. Car en faisant cela, on créerait des couplages forts entre la dépendance et les microservices. Tout le travail de séparation des tâches et de division des domaines métier aurait été fait en vain.

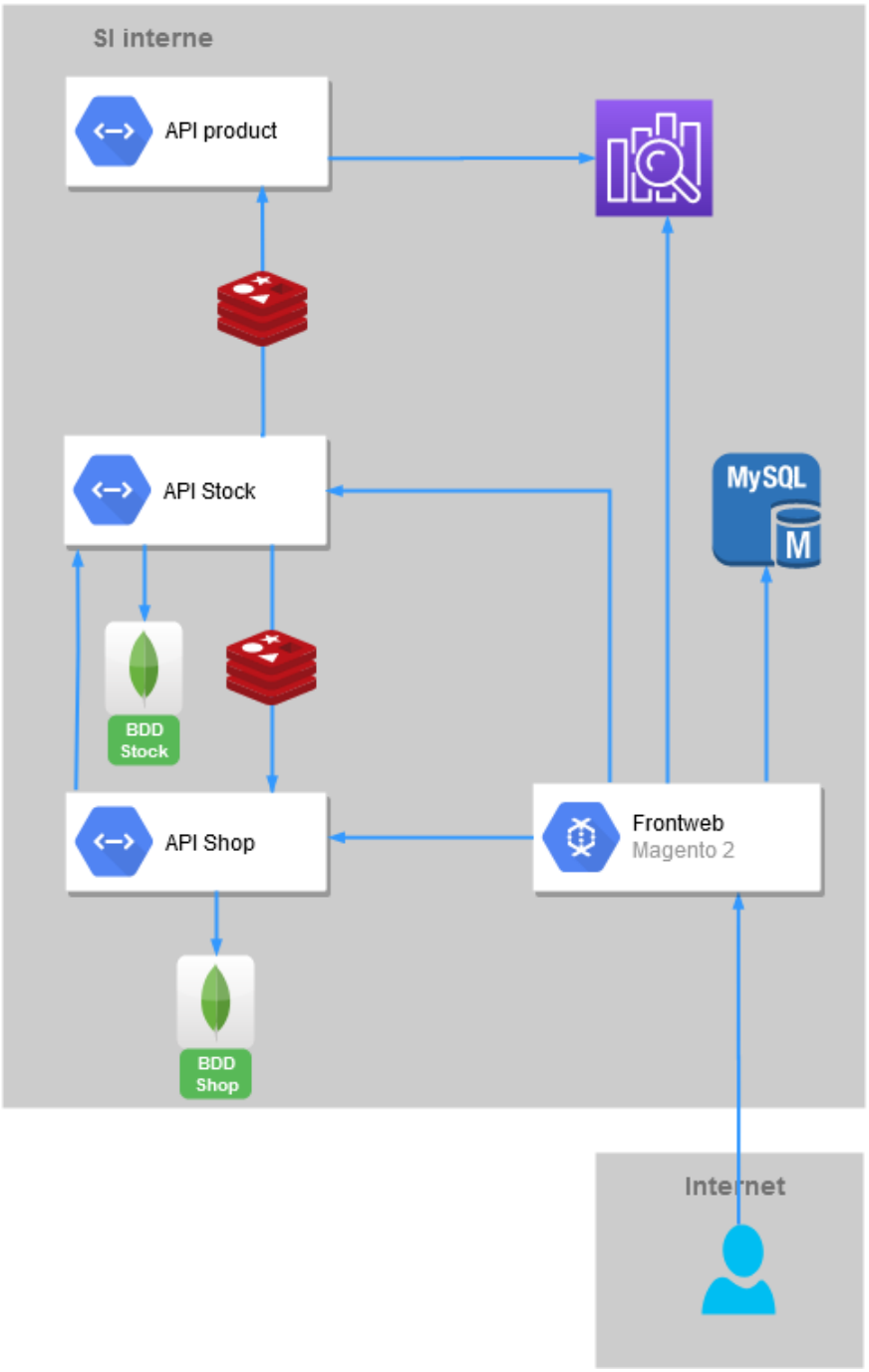

Figure 15, schéma simplifié des contextes métier appliqué montrant deux anti-pattern et une incohérence

Ainsi, la conception de ce schéma permet de remarquer des incohérences et des anti-patterns. Ses anti-patterns doivent absolument être éliminés dès que possible afin de réduire les risques par la suite.

En effet, si un anti-pattern est introduit, il peut devenir très difficile de s'en défaire ultérieurement. Comme décrit dans la figure précédente, on remarque un anti-pattern entre L'API Stock et l'API Produit. Les deux éléments sont interdépendant. Ce qui signifie que si l'un des deux est indisponible, les deux le sont. De plus, cela introduit un risque de boucle infini. Ce type de défaut de conception doit absolument être évité.

De plus, le frontweb appelle la base de données ElasticSearch des produits. Or cela vient en conflit avec le principe de la division des tâches. Ce n'est pas au frontweb de gérer les produits, car il y a un microservice dédié. De ce fait, une ressource n'est pas correctement utilisée, la base de données ElasticSearch, tandis qu'une autre ressource, le microservice des produits n'est pas utilisé alors qu'il devrait l'être. Enfin, si jamais le fontweb et le microservice des produits faisait des actions de modifications des données en base, il pourrait y avoir des conflits et cela est aussi clairement en contradiction avec le principe de séparation des tâches.

Enfin, il y a un anti-pattern, aussi appeler anti-pattern du marteau doré au niveau de l'appel entre le microservice des produits et des stocks. Ici, les données des produits peuvent changer fréquemment, contrairement aux données magasin. De ce fait, il n'est pas pertinent d'utiliser le système de mise en cache de redis.

3.2 Avantages fonctionnels

3.2.1 Des technologies hétérogènes pour des outils pertinents

Une architecture orientée microservices permet de choisir quelle est la technologie la plus pertinente pour chaque situation. Il n'est plus nécessaire d'avoir une base de données gigantesque ou les besoins de chacune des fonctionnalités ne sont que peu représenté.

De ce fait, il est possible de mettre en place des couples gagnants en combinant le bon langage avec la bonne base de données. Par exemple, plutôt que d'avoir un site qui doit exécuter de longues requêtes SQL sur des dizaines de tables, il est maintenant possible d'avoir un microservice dédié à la recherche, avec une base de données elasticsearch. Tous deux étant particulièrement efficaces dans le domaine de la recherche.

Ce qui permet d'améliorer l'efficience du système d'information en général et de manière rapide. La rapidité de développement des microservices est en effet un de leur avantage principal. Le domaine métier et les fonctionnalités couvertes étant très réduites, le temps de développement en est impacté de la même façon.

Le développement peut ainsi être effectué par des équipes plus petites et plus efficaces [16] et il est plus facilement possible de tester de nouvelles technologies, car si jamais cela n'est pas pertinent, le coût final reste faible.

3.2.2 Scalabilité

La scalabilité est la capacité d'un système à s'adapter aux dimensions du problème qu'il a besoin de traiter. Les logiciels monolithiques ne sont pas bien adaptés à cela. En effet, quand cela est possible, il y a toujours des coûts inutiles et des limitations. L'ensemble d'un système monolithique fonctionne alors aussi bien que la partie la moins efficace ou la plus surchargé de ses composants. Ce qui implique que pour un composant problématique, il va être nécessaire de mettre à l'échelle tout le reste du site internet.

Quant aux bases de données relationnelles, comme on a pu le voir précédemment, elles ne sont pas conçues dans un objectif de scalabilité. Les bases de données NoSQL, quant à elles, sont connues pour leur grande facilité d'être scalable, car elles ont été créées dans cet objectif. Cela est grâce à la tolérance au partitionnement du théorème CDP.

De ce fait, si lors d'une pandémie, l'utilisation du e-commerce pour augmente fortement pour ses courses. Alors, le nombre de client, de vues et de commandes d'un site de e-commerce est multiplié par dix. De ce fait, le système d'orchestration analysera une montée en charge. Alors, le nombre d'instance de microservices associés sera augmentés jusqu'à atteindre un nombre suffisant pour maintenir la qualité de service. De même, l'espace disque, RAM, puissance processeur et instance de base de données non relationnelle seront augmentés de manière proportionnelle. Si jamais le nombre de clients diminue, alors l'orchestrateur réduira de manière conséquente les ressources allouées tout en étant suffisante pour l'utilisation du site.

Ainsi, le site e-commerce pourra continuer de fonctionner correctement et au passage aura permis de démontrer la résilience du système d'information. Elle aura permis d'augmenter fortement le chiffre d'affaire de l'entreprise et améliorer son image auprès des clients en pleine pandémie.

3.2.3 Résilience du SI

La résilience désigne la capacité du SI à continuer de fonctionner en cas de problème. Grâce à la séparation des applications, le système d'application a de plus grandes capacités de résilience. En effet, si toutes les instances d'un microservice sont soudainement indisponibles et qu'il n'est pas possible de créer de nouvelles instances viables, le SI continue de fonctionner de manière dégradée. Ce qui n'est pas possible dans une application monolithique ou soit tout fonctionne, soit rien ne fonctionne. Ceci permet donc de créer des systèmes d'information performante et robuste.

Il en va de même pour les bases de données NoSQL. Grâce à la tolérance au partitionnement du théorème CDP, si un élément du réseau ou si un serveur avec un reliquat de la base est indisponible, il est toujours possible d'accéder aux données. Comme décrit dans le théorème CDP, seul une coupure de l'ensemble du système peut empêcher l'ensemble de fonctionner. Cela signifie qu'il faut soit un piratage particulièrement bien effectué, soit une défaillance totale du réseau informatique pour mettre à l'arrêt l'ensemble.

3.2.4 Amélioration de la sécurité

Au-delà de la résilience, vient aussi la question de la sécurité. Le chiffre d'affaires des entreprises du e-commerce repose sur la confiance de leurs clients. Cette confiance peut être ébranlée par des piratages informatiques. De plus, une attaque informatique bien organisée peut paralyser tout un SI et donc mettre en péril l'activité de l'entreprise.

Pour palier à cela, il convient de cloisonner les logiciels, autant d'un point de vue système, réseau, que métier. En séparant les logiciels, cela fait d'autant plus de cibles à atteindre pour les pirates s'ils veulent prendre contrôle de l'ensemble du SI. Chaque logiciel doit bien évidemment être sécurisé indépendamment en fonction de sa criticité et de son contexte métier.

3.2.5 Déploiement simplifié

Comme vu précédemment, le monde du e-commerce est en plein changement. Avec les changements sociétaux et les différentes crises, notamment sanitaires, les clients demandent de nouvelles fonctionnalités plus fréquemment. Les microservices sont particulièrement adapté à cela. Les changements y sont rapides, facilement testables et peuvent être faits sans aucune interruption de services pour cause de maintenance.

Ceci, associé à une méthodologie DevOps fait que les déploiements et mise en production deviennent une sorte de non-événement. En effet, l'architecture est maintenant beaucoup plus résiliente, les changements mineurs et les erreurs rarissimes, car ses changements ont pu être testés bien plus facilement. Et si un changement déclenche une erreur, celle-ci sera bien souvent minime, car le changement l'était. De plus, il est toujours possible de rétablir l'ancienne en quelques clics le temps de corriger le problème.

3.2.6 Simplification de l'intégration des données

Dans le développement orienté objet, qui est maintenant présent partout dans les sites e-commerces, il est nécessaire de faire le lien entre la donnée en base et leur représentation, les objets. Cela est coûteux en temps de développement. Avec une base de données NoSQL, les données peuvent représenter directement leur équivalent métier. Cela permet aussi de visualiser les objets directement en base sans faire de complexes requêtes SQL.

3.2.7 Simplification de l'organisation

Tout comme la gestion d'une application, l'organisation de large équipe peut être une tâche très ardue. En effet, plus il y a de personnes, plus il y a de cerveaux, mais plus il devient complexe de se coordonner et de s'entendre. C'est pourquoi de nombreuses entreprises préfère avoir des petites équipes qui travaillent sur des petits projets. De ce fait, il est possible d'avoir de multiples équipes localisées à plusieurs endroits, voir même en télétravail, qui sont responsables de leur application et de leur code. Ce qui aide à avoir des déploiements plus fréquents et simplifie la gestion de la responsabilité des composants et de leurs mises à jour.

3.2.8 Optimisé pour remplacer

D'après l'étude « *S'adapter ou disparaître : la nouvelle réalité d'un monde hyper numérisé* » [17], Les entreprises modifient leur fonctionnement tous les 15 mois environ. Or, il paraît impossible de changer en profondeur le mode de fonctionnement d'un logiciel monolithique tous les 15 mois. Ceux-ci ne sont pas adapté pour faire face à des changements fréquents. Cela prendrait extrêmement longtemps et serait tout aussi couteûx. Il est donc pertinent de préparer les outils du système d'information dans le but d'être remplaçable et modifiable facilement. Ce pourquoi les microservices sont particulièrement adapté.

De plus, les outils se doivent d'être remplaçable facilement. Le fait d'utiliser un outil sans pouvoir en changer induit une dépendance vis-à-vis de celui-ci. Que ce se passerait dans ce cas si des fonctionnalités venait à être supprimer ou si le mode tarification évolue de manière désavantageuse ? L'entreprise serait alors en état de dépendance partielle par rapport à une autre.

3.2.9 Composabilité et re-composabilité

Le monde du e-commerce n'est pas un monde fermé. Bien au contraire, il y a énormément de flux d'informations, de stock, d'argent… Comme vu précédemment, cela nécessite de bien cloisonner les éléments pour une bonne gestion. Mais cela permet aussi d'avoir de nouvelles opportunités pour être réutilisé. Par exemple, un micro-service de gestion des commandes peut être utilisé afin d'envoyer les commandes de certains produits au fournisseur de façon automatique, le tout via des appels HTTPs. Un revendeur pourrait utiliser le microservices des produits dans son site afin d'afficher les produits. Ce microservice deviendrait alors le référentiel produit et les modifications se répercutent de façon automatique et fluide. En bref, les microservices permettent de changer simplement et rapidement pour s'adapter aux besoins.

3.2.10 Meilleure distribution des données et des ressources

Avec les bases de données NoSQL et les microservices, il est donc possible d'avoir de multiples clusters d'instance partout dans le monde. On peut ainsi avoir un cluster à Tokyo, Paris et New York, chaque requête étant dirigé vers le cluster le plus proche et ayant le moins de charge serveur. Cela permet donc une réduction nette des temps de latence, car il y a moins de trajet à parcourir. De plus, en cas d'interruption des communications à cause de problèmes avec des câbles sous-marins, par exemple, on maintient une haute disponibilité, grâce aux multiples clusters.

3.3 Analyse

Les microservices et les bases de données noSQL permettent donc de nombreux avantages dans le cadre du e-commerce. Cependant, leur utilisation n'est pas une panacée qu'il suffit d'utiliser pour arriver à la perfection.

3.3.1 Avantages concurrentiels

Les avantages concurrentiels sont nombreux, le premier d'entre eux étant une meilleure gestion des ressources de l'infrastructure. En effet, il n'est plus nécessaire d'avoir un serveur web capable de tenir la charge des milliers de clients de la période de soldes tout au long de l'année, ce qui correspond à quelque pourcent de l'utilisation journalière. On peut donc ainsi faire des économies.

De plus Les instances des bases de données NoSQL et des microservices sont généralement dans le cloud, de ce fait, le prix est fonction de l'utilisation que l'on a de ses services cloud. Ce sont donc des charges variables pour l'entreprise, qui peut facilement les supprimés si un microservice se trouve être non rentable finalement.

De plus, cela permet une meilleure gestion des pics d'activité, qu'ils soient réguliers ou non. Le temps de latence en général est réduit, grâce à la proximité géographique et à la dénormalisation des données. Mais aussi à la mise en place de cache, tel que Redis. Quant aux risques d'indisponibilité en cas de défaillance du réseau ou de tentative de piratage, ils sont réduits grâce à une architecture redondante. Du fait de la séparation de façon fine les grands domaines, les microservices sont séparés, ce qui permet aussi de réduire les risques d'avoir de nouveaux éléments du SI de pirater à la suite d'un élément infecté.

De ce fait, l'expérience utilisateur en général est amélioré, car les utilisateurs on moins de ralentissement intempestif ou d'interruption de service. Ce qui peut vraiment faire perdre définitivement des clients et en faire des personnes qui risquent de dénigrer l'entreprise. Grâce à cette expérience utilisateur amélioré on peut donc avoir plus de clients promoteurs de l'entreprise, ce qui permet à terme d'obtenir plus de clients.

Ainsi, grâce au nouvelles technologies apportées par les bases de NoSQL et les microservices, de nouveaux services et fonctionnalités apparaissent, pour les clients finaux et le personnel de l'entreprise. Grâce a la communication évènementielle des microservices, un client peut par exemple être notifié en temps réel de la disponibilité d'un produit en magasin. Par l'utilisation des bases de données NoSQL au format graphique, il est plus simple d'analyser les comportements des groupes de clients.

En bref, cela apporte de nouvelles possibilités pour avoir de nouveaux services, que ce soit dans le but de satisfaire la clientèle ou comprendre ses envies du moment. Ces nouvelles possibilités peuvent être mise en place plus rapidement car les microservices sont plus petit et plus rapides à mettre en place. De plus, si le besoin change, cela est plus simple, plus rapide et plus sécurisé de faire le changement que sur un système monolithique. En effet, le déploiement et le développement sont simplifiés grâce aux outils d'automatisation et de la plus grande similarité entre les données réelles et celles en base de données.

Les nouvelles fonctionnalités permettent aussi d'améliorer l'automatisation des tâches, tel que la gestion des commandes impayées. Cette automatisation est applicable à l'ensemble des outils du SI grâce une interopérabilité accrue. Cette interopérabilité permet de mettre en place rapidement de nouveaux projets, mais aussi de faire communiquer les éléments du SI sans actions humaines. Cela signifie donc moins d'action humaine et moins d'oubli. Mais cela signifie aussi de nouvelles opportunités avec les sous-traitants et les fournisseurs. Il est par exemple possible d'effectuer un réapprovisionnement automatique des stocks de produits critique passé un certain niveau.

Le système de gestion des alertes est aussi plus pertinent et plus réactif. Chaque élément du SI peut être surveillé selon des paramètres précis. Cela peut être le nombre d'erreur journalière sur le site, ou la fréquence d'indisponibilité des produits les plus commandés. De ce fait, la sécurité du SI est améliorée et l'entreprise peut réagir plus rapidement face aux problèmes.

Le SI est maintenant plus complet, avec des outils spécialisés dans des domaines spécifiques, tels que les moteurs de recherches, afin de répondre au mieux aux besoins. Les outils obsolètes peuvent être retirés au profit de nouveaux outils plus profitables. Ses nouveaux outils sont aussi plus rapides à être mis en place et plus simple à maintenir, car il y a de petites équipes qui gère cela au lieu d'une seule grosse équipe. La maintenance des systèmes informatiques étant un point crucial pour la confiance des clients. En bref, améliorer la capacité de l'entreprise de mettre en place rapidement les changements nécessaires dans le cadre de son activité. Ce qui permet à terme de se démarquer de la concurrence et d'obtenir un avantage significatif dans le secteur changeant du e-commerce [17] [1].

3.3.2 Coûts liés à ses nouvelles technologies

Les microservices et les bases de données NoSQL apportent donc de nombreux avantages. A long ou moyen terme, en fonction de la taille des changements à effectuer, cela permet de gagner de l'argent grâce à la scalabilité et la rationalisation des dépenses du SI, entre autres.

Mais il faut cependant payer la mise en place des microservices et des bases de données NoSQL. Ce qui signifie donc de payer pour :

- la conception de la nouvelle architecture,
- le découpage des logiciels monolithiques si cela est pertinent,
- l'inévitable étape de développement pour adapter les différents logiciels,
- le développement des nouvelles fonctionnalités,
- le coût des licences, si nécessaire,
- si le développement est fait en interne, il faut aussi former les développeurs aux nouvelles technologies nécessaires au projets,
- le cout de gestion des équipes,
- le cout de maintien en état des logiciels et de l'infrastructure,
- l'ancienne infrastructure et ses composants le temps de la transition.

Il y a donc une période ou la mise en place de microservices et de bases de données NoSQL n'est pas rentable pour l'entreprise. Les nouveaux microservices ne sont pas encore forcément totalement utilisable.

Les instances des bases de données NoSQL et des microservices sont généralement dans le cloud, de ce fait, le prix est fonction de l'utilisation que l'on a de ses services cloud. Une entreprise avec une grande volumétrie de produit qui change fréquemment ses produits va donc payer plus qu'une autre qui change rarement son catalogue réduit.

De plus, le nombre d'instance étant beaucoup plus important, les moyens d'accéder a des informations sensibles ou de prendre le contrôle des logiciels sont plus importants. C'est pourquoi il est très important de chiffrer les communications entre les microservices et de les cloisonner. Par exemple, un microservice qui ne doit pas être appelé via une adresse publique sur internet ne doit pas avoir de moyen d'être appelé par ce biais. La mise en place des microservices et les bases de données NoSQL ne doit pas se faire au prix de la sécurité. [18] [19]

3.4 Aspects méthodologiques : points importants à considérer

3.4.1 Remise à plat au fil du temps du SI

Comme vu précédemment, on souhaite avoir un SI dont les éléments ont un couplage faible. Mais cependant, il est nécessaire que chaque élément ait une grande cohérence interne. Certains éléments du SI respectent déjà ses critères et il n'est pas toujours nécessaire de les modifier. Par exemple, un ERP qui gère les finances et la création des bilans financier est à sa place. Mais d'autre outils qui sorte de leur champ d'action normal peuvent être modifié, voire, remplacer.

Les raisons pour lesquelles on peut souhaiter de remplacer ou diviser une application monolithique peuvent être nombreuses. Cela peut être les facilités vis-à-vis des rythmes de changement de l'application, car une application plus petite est plus simple à tester et plus simple à modifier. Cela peut être aussi la facilité organisationnelle des équipes. Il n'est plus nécessaire d'avoir une grosse équipe à temps plein, mais quelques équipes qui peuvent être à plusieurs endroits. Ou encore la sécurité, car les vieux logiciels remplis de failles sont légions. De plus, les microservices peuvent être protégés par plusieurs couches de sécurité, mais il est aussi possible de séparer les risques liés aux piratages. Ainsi, les pirates informatiques qui ont réussi à pirater le système de commande n'auront pas forcément accès au système de paiement.

De ce fait, la modification de tout ou partie d'un SI n'est pas une tâche anodine, surtout avec des logiciels monolithiques, ou qui ne sont plus maintenus. En effet, il peut être compliqué de trouver des personnes compétentes pour cette tâche, qui du fait de la nature monolithique a tendance à créer des effets de bords lors des modifications.

Ainsi, la première étape consiste à trouver les éléments qui ont le plus de valeur à modifier. Ce serait par exemple un vieux site internet sur lequel de nouvelles fonctionnalités pourraient être rajoutées au passage, par exemple. Le tout de façon incrémentale afin de pouvoir s'assurer de ne pas causer de problèmes.

Une fois qu'un élément qui aurait de l'intérêt à être modifié ou séparé du monolithe est découvert, il est possible d'essayer de l'encapsuler. C'est-à-dire que le code va rester globalement le même hormis certaines adaptations est que le nouvel élément sera ensuite utilisé avec le reste. Ce qui peut nécessiter des modifications dans le monolithe. De ce fait, on cherchera avant tout les éléments ayant le moins de couplage possible afin de ne pas modifier des éléments du monolithe. En effet, il faudrait alors le refaire par la suite, ce qui serait une perte de temps. Il est possible que pour des raisons techniques toutes les fonctionnalités ne soit pas présentes dès la première itération, mais cela sera corrigé dès que possible.

Comme décris dans la figure suivante, il y a donc un site constitué de 4 grands domaines métiers. Ses domaines métiers sont ici simplifiés pour l'exemple, cependant le principe serait le même pour gérer une des dizaines de grands domaines métiers. Le premier domaine métier à être séparé sera le domaine client, car il a moins de couplage que le reste.

Mais avant de le séparer, il est nécessaire de créer un nouveau schéma dédié. Ainsi, tant que le microservice n'est pas finalisé, la structure de la base de données reste intacte. Il y a donc toujours la possibilité de revenir en arrière si nécessaire. Cela pourrait être l'occasion de changer de type de base de données si cela est pertinent. Le domaine client a été choisi entre autres car il y a peu de couplage avec le reste et le client souhaite améliorer la gestion des données clients au vu du nouveau RGPD (Règlement Général de Protection des Données de l'Union Européenne).

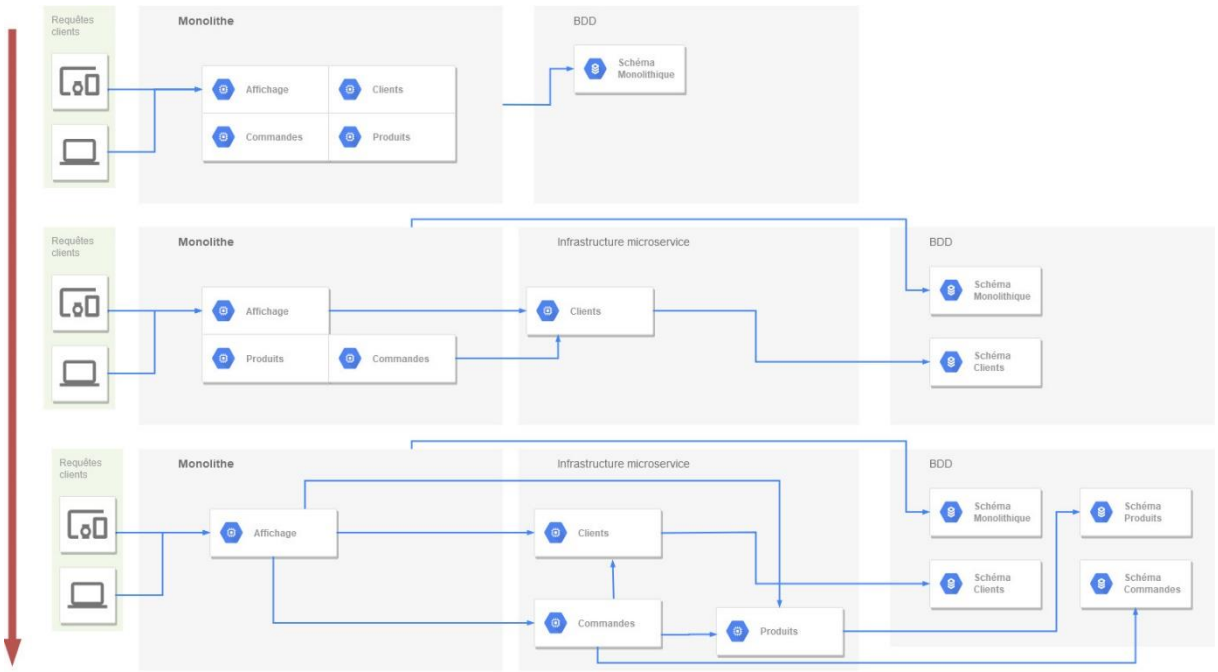

Figure 16, séparation d'un site monolithique de ventes en 3 étapes

Ensuite viennent les commandes et les produits. Pour les produits il serait pertinent de choisir une base de données ElasticSearch par exemple pour améliorer les temps de réponses des recherches des clients. L'affichage peut rester un élément de l'ex monolithe si cela est pertinent. En effet, les clients peuvent apprécier son design et il n'est pas forcément utile de modifier des éléments à ce niveau.

Ce processus pourrait continuer avec l'interconnexion des trois nouveaux microservices avec des prestataires ou encore d'autres éléments du SI comme un système d'analyse des ventes. Il est aussi envisageable de créer de nouveaux microservices, tels qu'un microservice de gestion des stocks, ou de suggestion ciblée.

En conclusion, les avantages des microservices combinés aux bases de données NoSQL sont nombreux et permettent de répondre à la majorité des problèmes décris précédemment à propos des systèmes monolithiques standard.

Ses systèmes permettent entre autres une tolérance aux pannes très importantes grâce à la redondance des instances. Les différentes structures de données proposées permettent des fonctionnalités adaptées en fonction du besoin, tel que le reporting de donnée ou la mise en cache de page internet.

Mais les microservices et les bases de données NoSQL reste des outils et ne sont pas la panacée qui va permettre de répondre à tous les problèmes. Il faut rester pertinent dans le choix et l'organisation des outils et méthodes afin de rester cohérent. Ceci permet d'établir des standards qui guideront le reste du travail.

3.4.2 Points de vigilance

Comme dit précédemment, l'utilisation de microservices et de bases de données NoSQL n'est pas une panacée qui suffirait à gérer tous les problèmes du SI de l'entreprise. Il reste toujours nécessaire de s'assurer que dans le contexte, les microservices sont pertinents vis-à-vis des besoins. Et évidemment, de déterminer la base de données la plus adaptée. Pour cela, il faut toujours se reporter au théorème CDP [10].

Le premier point de vigilance porte sur le NoSQL. Contrairement au SQL, il n'y a aucun standard pour le NoSQL. Ce qui signifie qu'en cas de changement entre deux bases de données NoSQL, il n'y a aucune assurance que les requêtes seront toujours fonctionnelles. Que ce soit à cause des fonctionnalités qui ne sont pas les mêmes ou purement le langage. De ce fait, il est nécessaire d'avoir toujours une couche d'abstraction des données. Ce peut être une librairie externe, des drivers… Il faudra bien évidemment s'assurer que celui-ci est mis-a-jour quand cela est nécessaire pour préserver la sécurité du microservice.

La sécurité quand a elle peut être améliorée grâce aux microservices… ou amoindrit. En effet, la présence de multiples microservices augmente la surface d'attaque pour les attaquants et en fait autant de portes d'entrée dans le SI. Evidemment, cela peut être plus long de prendre contrôle de l'ensemble du SI grâce à la défense en profondeur. Mais cela signifie qu'il faut être particulièrement pointilleux du point de vue de la sécurité [18] [19] [20].

De plus, comme dans tout logiciel, il convient de faire attention à la dette technique. Cette dernière, fruit des développements successifs peut être maitrisée grâce à l'architecture en microservice via la rationalisation. Mais seule l'architecture ne suffit pas. Il faut surveiller cette dette, par exemple avec des outils dédiés, tel que Sonnar.

On peut souvent trouver qu'il est difficile de trancher entre trop et trop peu de découpage dans les microservices. Cela peut rapidement devenir un piège. En effet, l'un et l'autre risquent de déclencher une perte des performances et une dégradation de la cohérence métier. Pour répondre à ce risque, il faut toujours penser d'un point de vue domaine métier.

Au-delà de la cohérence métier se pose aussi de la cohérence du SI. Si l'on utilise toujours des nouvelles technologies, langages et Framework pour chaque développement, le SI risque vite d'être très complexe à gérer. Il va falloir des personnes qualifiées dans toutes les bases de données et technologies choisies par le passé. Personnes qui en plus vont devoir gérer l'ensemble de l'architecture. Ce qui signifie être multi-compétent dans tous les domaines nécessaires.

De plus, les bases de données NoSQL sont bien souvent portées par une communauté et non pas une entreprise comme Oracle pour le produit du même nom. De ce fait, le support peut être assez variable, tout comme la fréquence des mises à jour. Cependant, certaines bases de données NoSQL, comme ElasticSearch ou BigTable, ont l'appui d'entreprise pour leur maintien.

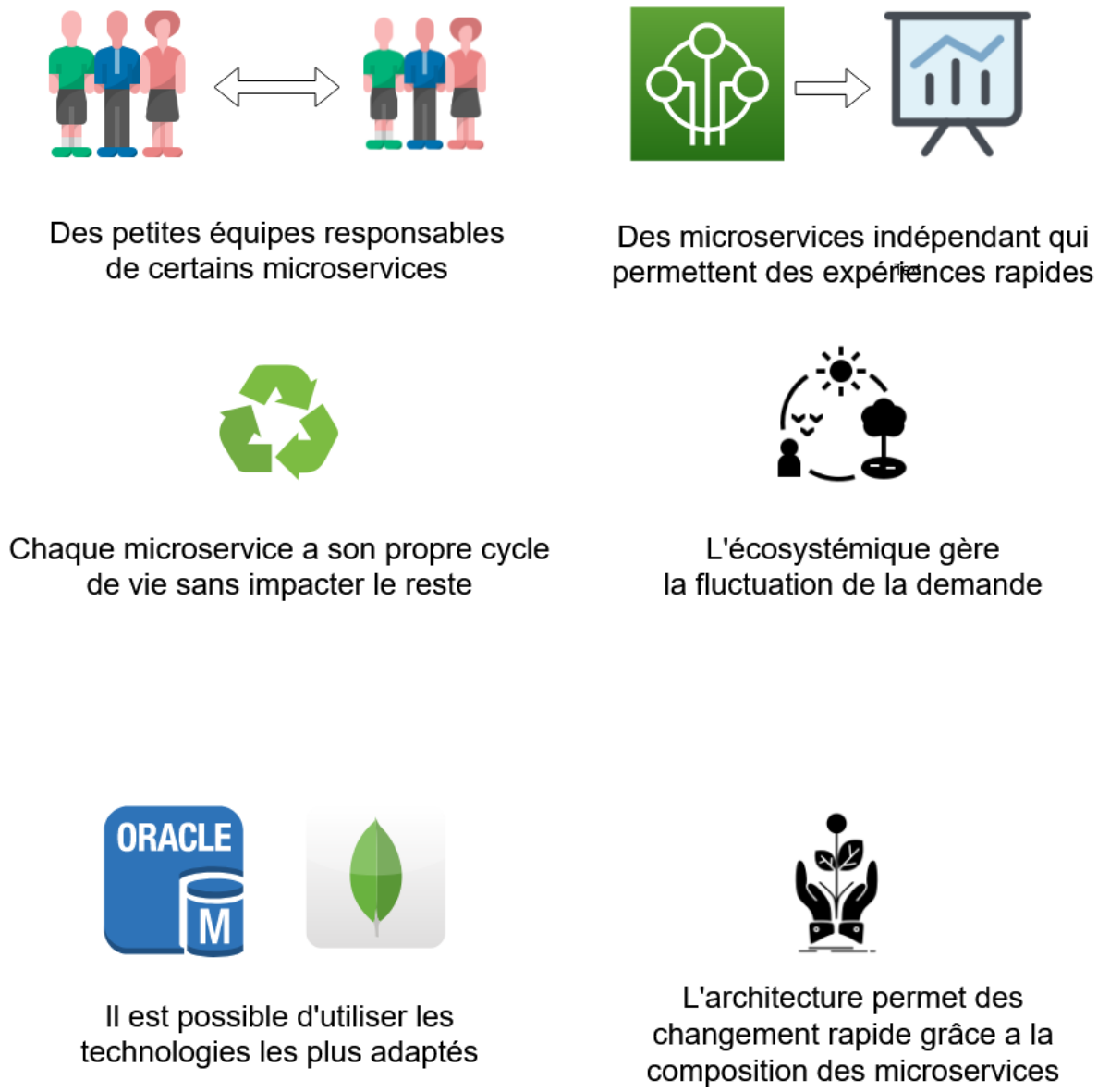

Figure 17, récapitulatif des avantages de l'architecture des microservices

En conclusion, La complexité d'un SI n'est pas dans le code de ses logiciels ou dans leur aspect métier, mais dans leur bonne organisation. Le point le plus important étant dans leur interactions. Des microservices mal coordonnés faisant des appels inutiles ne sont pas plus efficaces qu'un logiciel monolithique, bien au contraire.

4 Gouvernance de l'architecture

4.1 Vision sur le long terme

Un SI ne change et ne se créée pas du jour au lendemain. Cela demande de la préparation et une certaine planification. Cependant, il n'est pas possible de tout planifier à l'avance, car les besoins de l'entreprise sont changeants [17]. Surtout dans le secteur du e-commerce, où les entreprises peuvent subir des changements réglementaires ou des rachats.

La métaphore la plus pertinente vis-à-vis du SI est celle d'une ville. Quand un urbaniste construit une ville, il ne sait pas forcément les bâtiments ou le nombre précis d'habitants qu'il y aura dans quelques années. Cependant, il peut y prévoir une zone industrielle, une zone résidentielle…

Cela pousse à avoir une autre approche de la vision en cycle en V, qui pousse à tout prévoir à l'avance et de créer un cahier des charges descriptif pour chaque élément. Ce n'est pas réaliste si l'on souhaite pouvoir suivre les exigences croissantes du e-commerce. Pour ce faire, il faut une nouvelle vision afin de mettre en place l'architecture du SI du futur. La vision d'un architecte informatique évolutionnaire.

Pour expliquer cette vision évolutionnaire, revenons à la comparaison de la ville en tant que SI. Comme le disait Erik Doernenburg « *Le rôle d'un architecte dans l'industrie du logiciel devrait être vu plus comme un urbaniste. Si vous y réfléchissez, le rôle de l'urbaniste est de regarder la multitude de sources d'information et de tenter d'optimiser l'aménagement de la ville en tenant compte des besoins actuels et futurs des habitants.* ». [15] Et il est vrai qu'un architecte en informatique, doit être capable d'optimiser l'aménagement du SI pour les besoins actuels. Mais aussi et surtout des besoins futurs. Si l'on choisit une base de données qui n'est pas capable de gérer les besoins futurs de l'entreprise, cela peut la mettre en péril.

Voici la citation originale *« The role of an architect in the software industry should be seen more as that of a Town Planner. If you think about it, the role of the town planner is to look at the multitude of sources of information and attempt to optimize the layout of the town/ city keeping in mind the current and future needs of the people.* » [15]

De ce fait, un architecte informatique évolutionnaire devrait prévoir de façon générale comment le SI va évoluer d'ici quelques années. Tout en laissant quelques marges d'erreurs, car comme dit précédemment, les besoins sont fluctuants. Pour ce faire, on revient aux grands domaines métiers expliqués précédemment. Un SI typique aura autant de zone dédiée, ici, a la gestion des produits et la, l'envoi des commandes. L'architecte va donc se concentrer sur le gros plan général plutôt que les moyens d'implémentations. Cela est du domaine des développeurs, analystes et intégrateurs. A eux de déterminer par exemple si la mécanique du bouton d'ajout au panier doit être gérée avec tel ou tel librairie au vu des besoins et des bonnes pratiques.

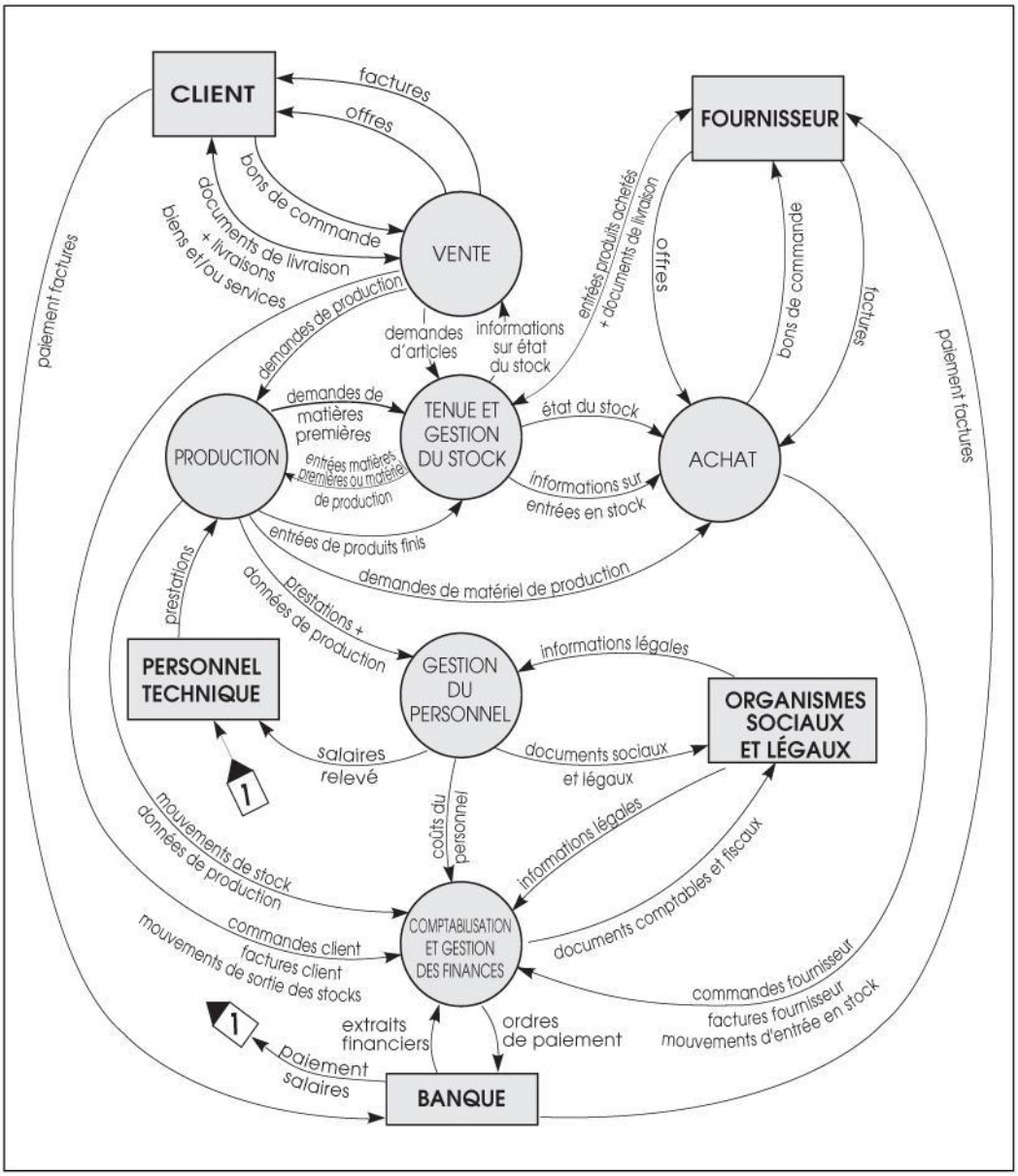

Figure 18, un système d'information vu sous l'angle des flux d'information, Cairn https://www.cairn.info/systeme-d-information-de-l-entreprise--9782804149727-page-11.htm#

Dans le schéma précédent, chaque bulle représente un grand domaine métier. Mais celui-ci ne contient pas forcément un seul microservice ou un seul logiciel. On peut avoir par exemple pour les achats un microservice pour les fournisseurs qui peuvent être interconnectés avec celui-ci et un autre logiciel pour les autres gérant les notifications du microservice de gestion des stock.

Au niveau des bonnes pratiques, le plus importants pour l'architecte sera la bonne gestion des flux d'informations. Tout comme les flux de personne font vivre une ville, les flux d'informations font fonctionner le SI. Et comme on peut voir dans la figure précédente, il y peut y avoir de très nombreux flux dans un SI. C'est ici que la vue globale de l'architecte est la plus utile afin d'optimiser le SI. Il est entre autres nécessaire que différents flux n'entrent pas en conflits et s'il y a besoin d'un ordonnancement, alors s'assurer que celui-ci est respecté.

Au-delà du respect de l'ordonnancement, il faut s'assurer que les interactions entre les microservices et donc les flux d'informations se font de manière correcte. Et cela passe par la mise en place et le respect des bonnes pratiques et des standards, eux-mêmes dérivés des principes architecturaux.

Ces derniers dépendent des objectifs stratégiques du SI de l'entreprise. Ceux-ci sont généralement déterminés par le CEO de l'entreprise. Par exemple, une entreprise pourrait avoir deux objectifs stratégiques pour son SI :

- soutenir l'entrée de l'entreprise sur de nouveaux marchés,
- supporter l'innovation dans les marchés existants.

Ces deux objectifs sont faisables avec des processus opérationnels flexibles et des nouveaux produits ou services. Pour les mettre en place, l'architecte va déterminer des principes architecturaux les plus pertinents au vu des objectifs. Par exemple :

- réduire l'inertie en faisant des choix qui favorisent des retours d'expérience et des changements rapides,
- réduire l'interdépendance entre les équipes en donnant des rôles bien définis à chacune d'entre elles.

Ainsi les objectifs ont maintenant des principes afin de pouvoir les remplir. Cependant, cela ne suffit pas. Il faut une manifestation de ses principes, et ce sont les bonnes pratiques et les standards. De ce fait, afin de mettre en application les principes, les bonnes pratiques seront :

- l'utilisation du standard REST/HTTP dans les microservices a venir,
- la création de microservices de faible taille et indépendant,
- la mise en place de l'intégration continue,
- le découpage des logiciels monolithiques via l'encapsulation de ses derniers.

Le tout pourrait être simplifié par la figure suivante :

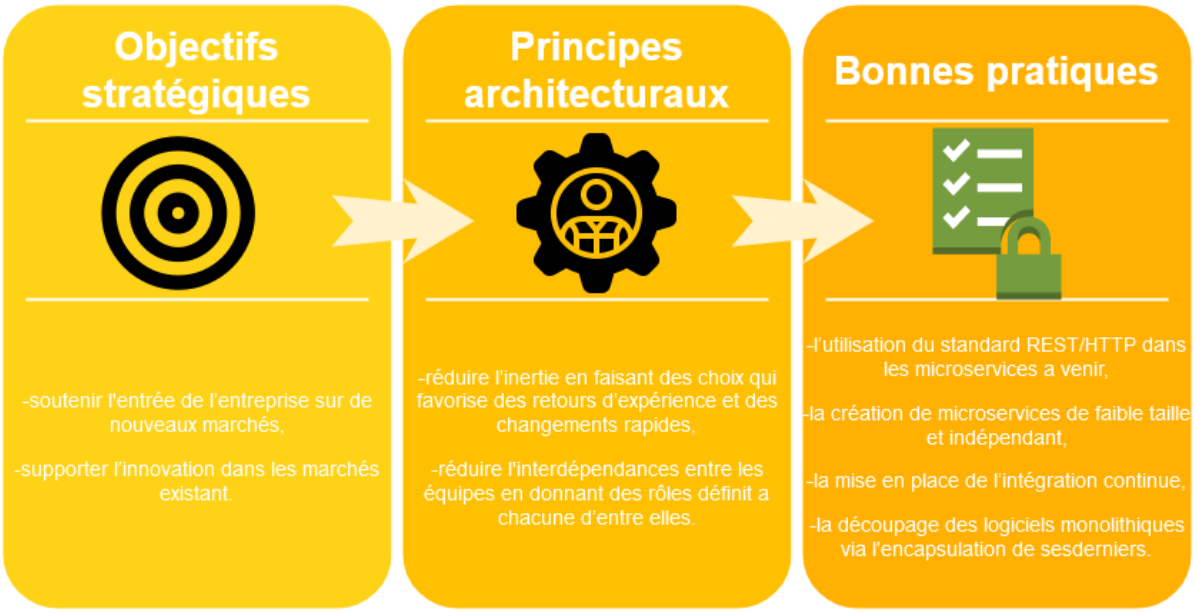

Figure 19, synthèse des étapes nécessaires pour la mise en place des bonnes pratiques

Mais finalement, le SI est aussi comparable à une créature vivante. En effet, il va changer au fil du temps et bien souvent de manière imprévue. De ce fait, il n'est pas possible d'avoir un contrôle total dessus et il est parfois nécessaire de faire des compromis. Parfois il sera nécessaire d'utiliser d'autres standards de communication que ce qui est prévu pour des raisons de performance. Evidemment, cela doit rester exceptionnel et le plus cohérent possible avec les objectifs, principes et bonnes pratiques mises en place.

4.2 La résistance au changement

Chez l'être humain, la résistance au changement est un phénomène qui se déclenche naturellement. Cette résistance apparait particulièrement dans les SI vétuste, car des connaissances peuvent être perdus au fil du temps. Afin d'en réduire l'impact, il est essentiel d'en déterminer rapidement les causes et les personnes impactées par ses changements. Pour ce faire, il est préférable de communiquer rapidement et clairement les motifs qui se profilent derrière les transformations. Le tout dans le but de faire comprendre aux personnes concernées qu'elles font et feront à l'avenir partie intégrante des projets d'amélioration.

Certaines personnes peuvent en effet avoir peur que les changements aillent leur prendre leur travail ou encore le modifier à leur détriment. En effet, avec les robots, internet… les habitudes de travail ont été particulièrement impactées au fil des ans par l'informatique.

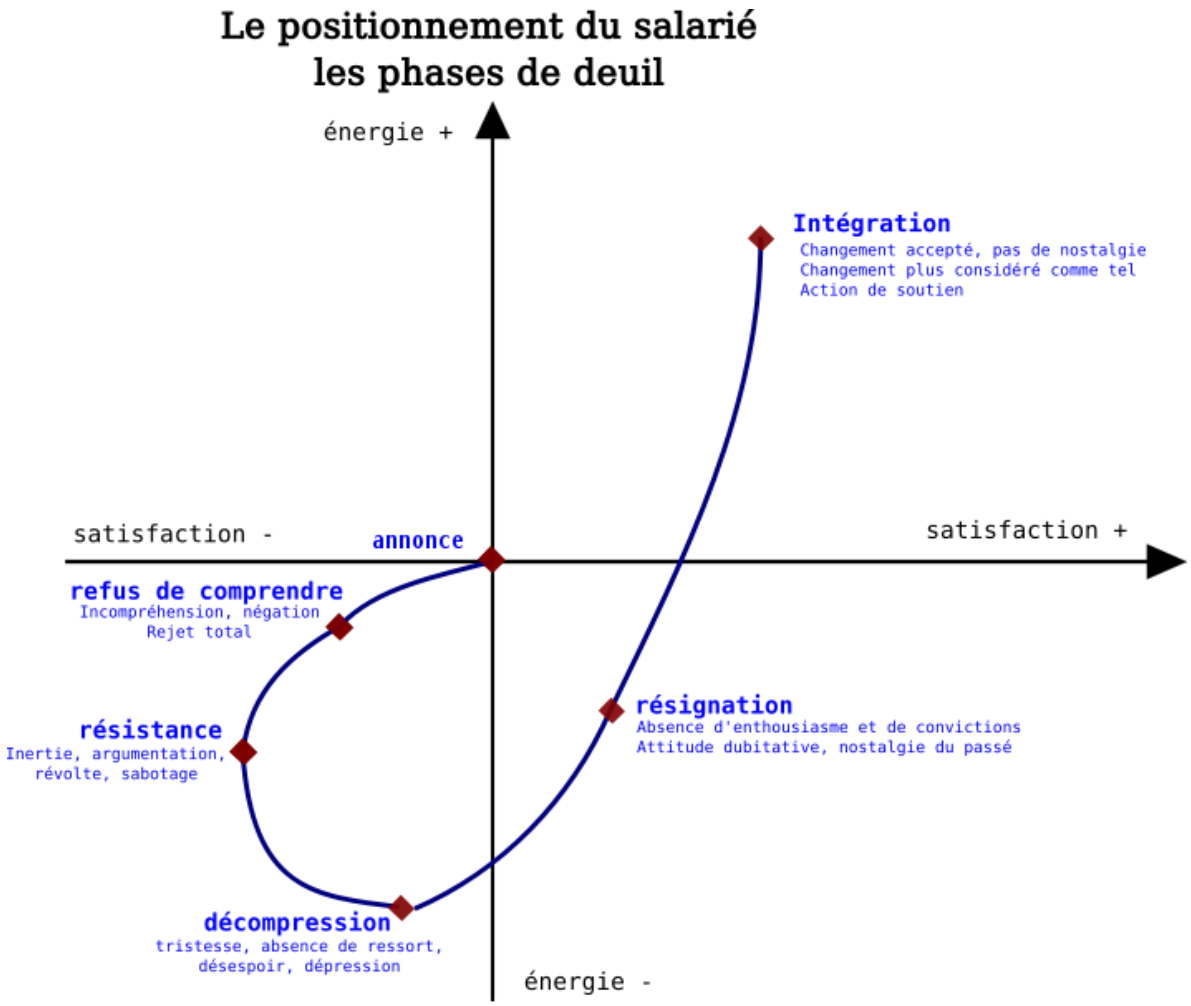

Figure 20, les positions des salariés face au changement, source : Eloge du changement : Guide pour un changement personnel et professionnel, Gérard-Dominique Carton

Dans le cadre du SI, les causes peuvent être multiple. Cela peut être pour faire des économies a court terme, par exemple. On ne souhaite pas injecter plus d'argent dans des changements dans le SI qui ne semblent pas nécessaires. Car les transformations inhérentes à tout changements coutent cher et ont des couts annexes, comme les formations du personnel. Cependant, comme le dit Francis Blanche, *« Face au monde qui change, il vaut mieux penser le changement que changer le pansement. »* [21] . Ainsi, il faut se préparer au changement. Car si on ne le prépare par, on va devoir en subir les conséquences, comme toute entreprise qui n'a pas réussi à voir suffisamment en avant.

De ce fait, afin de s'adapter le plus tôt aux changements qui vont arriver, avec ou sans notre bon vouloir, il est nécessaire d'avoir une politique d'entreprise vis-à-vis du changement. La direction du SI, notamment doit avoir une personne qui est en charge de la gestion des changements dans son domaine d'activité.

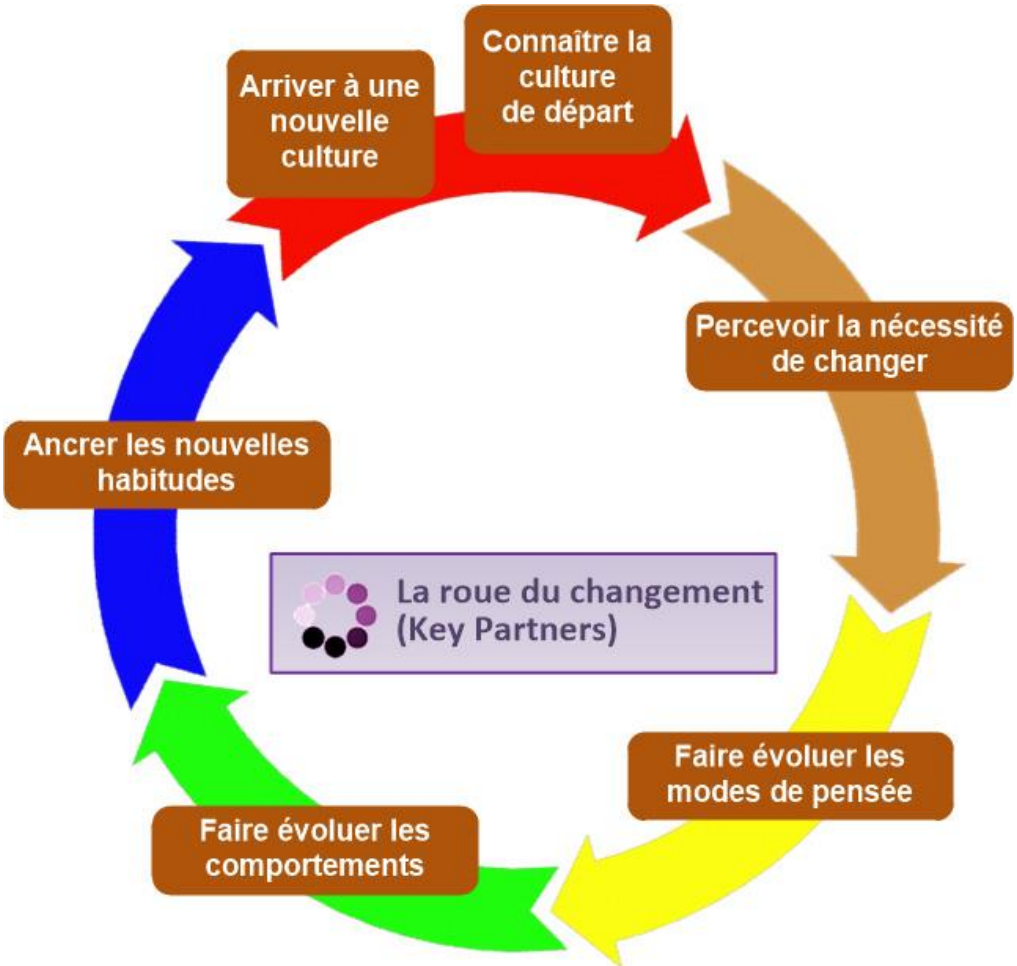

Figure 21, la roue du changement, https://coachingcommunication.fr/conduite-du-changement-informatique, Raphael Diaz,

5 Mise en pratique

Au sein de Pictime Retail, j'ai fait partie de l'équipe de développement de Bureau Vallée. C'est une enseigne de la grande distribution fondée en 1990 et spécialisée dans la papeterie, les fournitures de bureau, bureautique et consommables informatiques. En 2018, Bureau Vallée a effectué un chiffre d'affaires de 375 millions d'euros. Bureau Vallée compte actuellement 301 magasins dont plus de 250 sur le territoire français et de nombreuses ouvertures sont prévues dans le futur.

Sur le plan du catalogue, Bureau Vallée avait en 2018 environ 30 000 produits. Son catalogue est actuellement constitué de plus de 100 000 produits différents. L'objectif est d'arriver à 200 000 produits et à terme de concurrencer Amazon dans le domaine de la papeterie, les fournitures de bureau, bureautique et consommables informatiques.

Dans le cadre de sa transformation numérique, Bureau Vallée a souhaité changer de site internet afin de pouvoir se déployer à l'international et améliorer la satisfaction client. En effet, le site précédent avec Magento 1 a quelques limitations et ne permet pas la traduction, élément vital pour un déploiement à l'international. Ce site internet a été développé par Pictime et de nombreux membres de l'équipe originale de développement sont dans la nouvelle équipe de développement de Bureau Vallée. Cela permet entre autre de garder à l'esprit les objectifs d'origine du site internet et une meilleure compatibilité. Bureau Vallée a choisi des objectifs stratégiques similaires à ceux décrits dans le chapitre 4.1 et qui ont été appliqués ici.

Lors de mon arrivée, dans le cadre de cette mission, on m'a donné pour rôle de conceptualiser, pendant environ 1 mois, des éléments du nouveau site internet demandé par Bureau Vallée, développé avec Magento 2. J'ai ensuite put améliorer mes compétences sur l'autre facette du projet, ou j'ai développé et amélioré des microservices et système d'intégration qui servent de référentiel métier pour le site internet. Les dirigeants de Bureau Vallée savent que le nouveau site internet risque d'être obsolète et de ce fait, mise sur l'utilisation conjointe de microservices pour gérer le domaine métier nécessaire à Bureau Vallée.

Ses microservices gèrent notamment les stocks et les magasins de Bureau Vallée et sont interconnectés à de nombreux autres éléments, notamment le site internet et des prestataires.

L'ensemble de l'équipe était géré en mode agile SCRUM avec un Scrum Master et des instances régulières. Cela permettait, comme le voulait Bureau Vallée, d'accroître la fluidité des livraisons et la vélocité en réduisant les lourdeurs, afin de répondre à ses objectifs stratégiques.

La conception originale théorique de l'architecture à la fin du projet était la suivante :

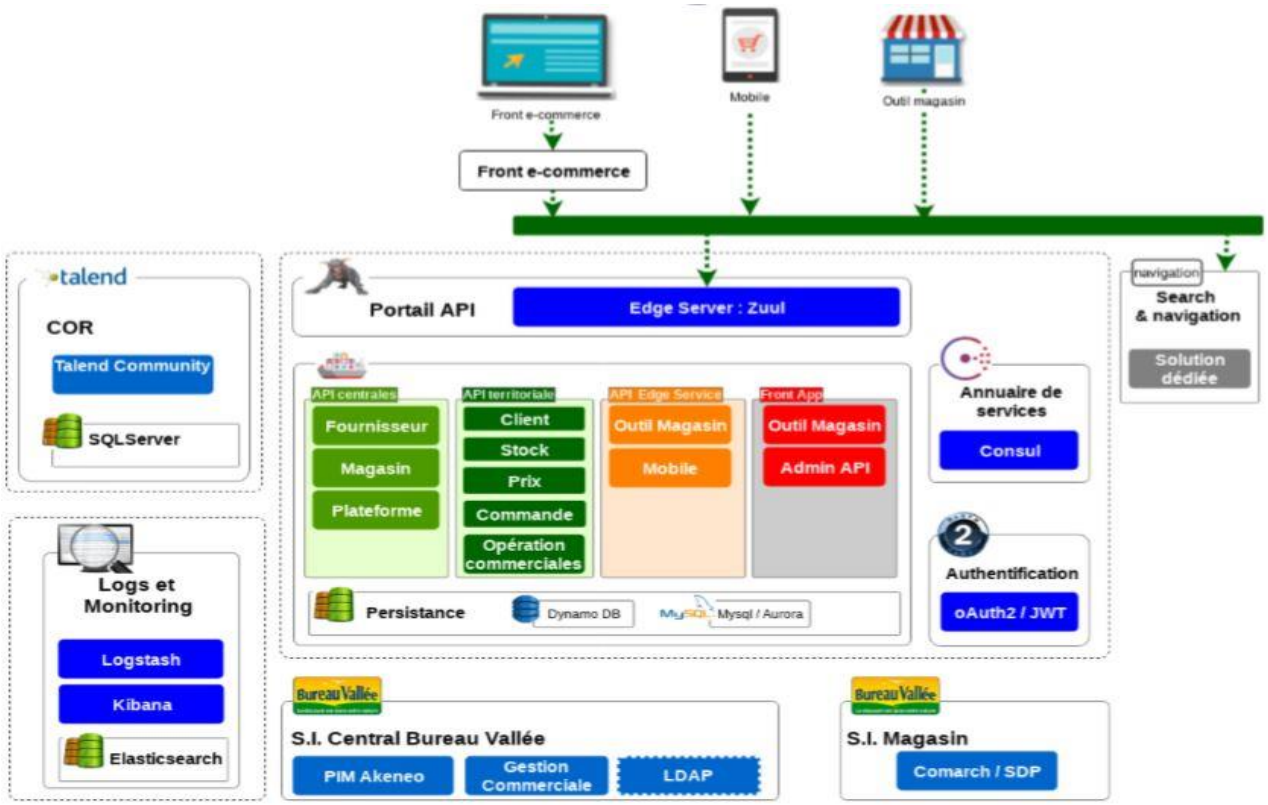

Figure 22, architecture originale théorique à la fin du projet, source Pictime Groupe

Cette figure ne reflète pas exactement la mise en place l'ensemble du projet, cependant cela montre bien l'étendue de la tâche, sachant que seul le SI central de Bureau Vallée était présent au commencement du projet.

Afin d'avoir un temps de réponse optimal et de ne pas utiliser inutilement des ressources, le site internet, appelé dans la figure précédente front e-commerce, utilise un système de mise en cache par blocs. Ces éléments mis en cache sont stockés dans Redis, qui est une base de données NoSQL de type clé-valeur. Cela permet de mettre en cache des éléments du site commun à tous les clients, comme par exemple la barre de navigation. Ces éléments en cache sont ensuite réutilisés dès que nécessaire.

Sachant que sans cache, Magento 2 met en moyenne environ trois secondes à générer une page et qu'avec l'utilisation du cache, ce temps est réduit à moins d'une seconde, cela triple ses performances !

Ce cache, constituée en blocs, est alors utilisé pour générer la page. Certains éléments de la page sont cependant totalement dynamiques et ne peuvent pas être mis en cache. Il s'agit des éléments avec un contour noir sur la figure suivante. Il y a par exemple le stock de produits disponible qui est géré par le microservice des stocks.

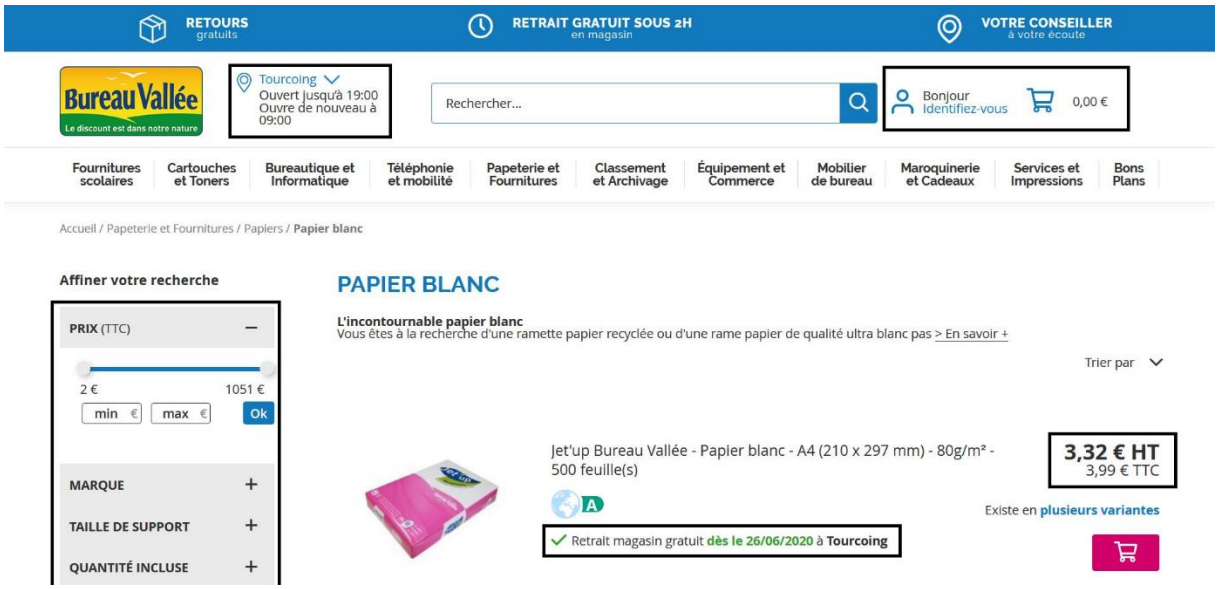

Figure 23, imprime-écran de la page des papiers blanc du site de Bureau Vallée, https://www.bureau-vallee.fr

Mais qui dit utilisation du cache signifie aussi gestion du cache. Car sa validité n'est pas éternelle. Pour s'assurer que les informations affichées sont pertinentes, Redis est encore une fois utilisé. Chaque élément en cache a un temps de vie, et quand son temps est dépassé, il est supprimé. Ce temps de vie est paramétrable. Plus une information change fréquemment, plus le temps de vie devrait être faible, ou alors il ne faudrait pas utiliser le cache.

La figure suivante montre l'architecture simplifiée nécessaire à l'affichage des pages de produits du site. Ce dernier appelé au frontweb, basé sur Magento 2, utilise divers modules de Magento et sert à faire les ventes et l'affichage des produits. Pour ce faire, il utilise Redis pour le cache, comme vu précédemment. Redis a été choisi car il peut stocker n'importe quel type de format. De ce fait, il est utile pour les microservice et le frontweb. ElasticSearch quant à lui est utilisé pour la recherche des produits car il y a un module dédié dans Magento 2 et qu'il est extrêmement efficace pour les moteurs de recherche. Enfin, il y a MySQL pour tout le reste ou les bases de données non-relationnelles ne sont pas pertinentes actuellement.

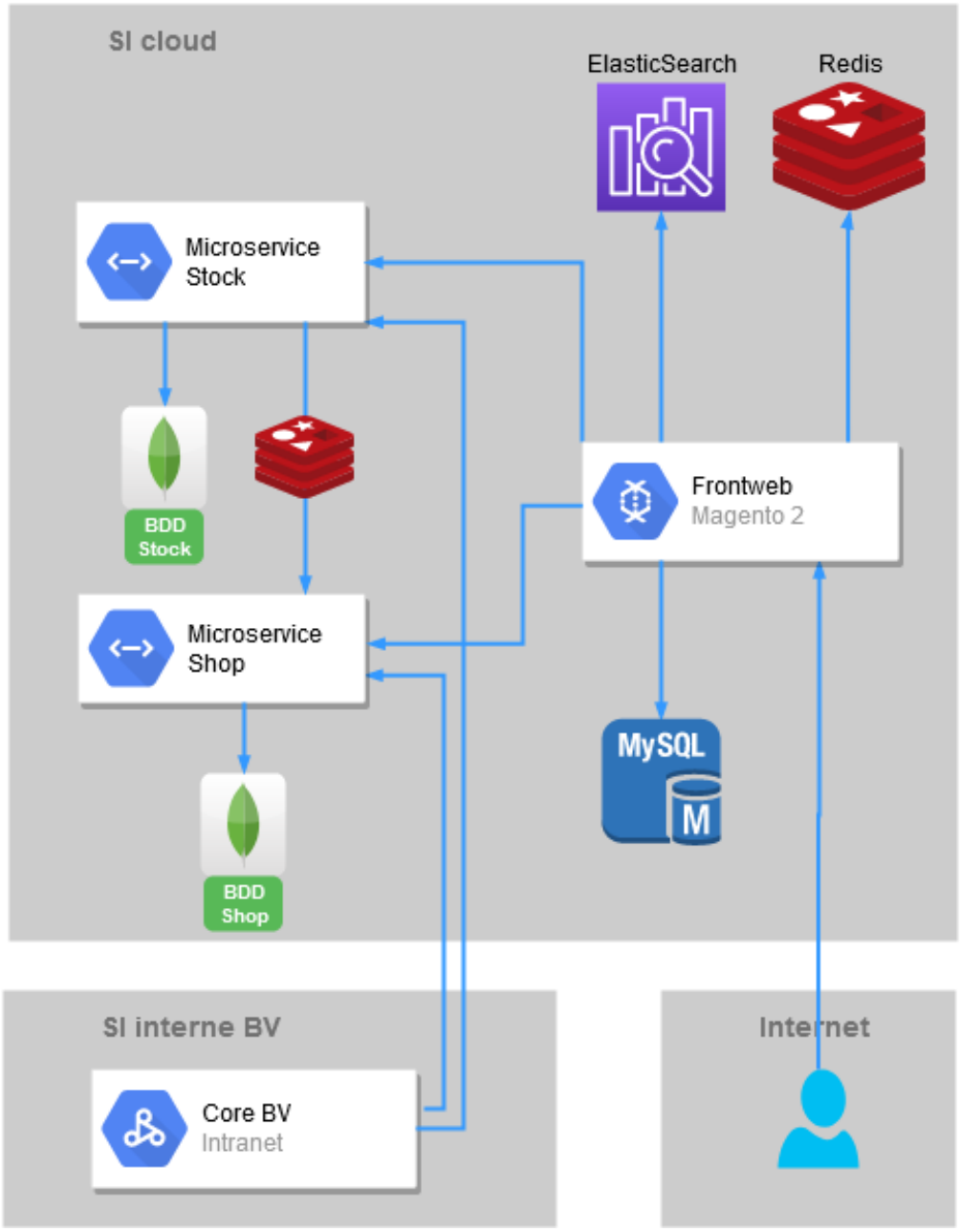

Figure 24, architecture simplifiée

Les deux autres microservices Shop et Stock sont ceux qui ont été créés en premier car ils ont la plus grande valeur métier. Ils permettent la gestion des stocks et des magasins en temps réel. Ils sont utilisés dans la figure 23 dans les éléments dynamiques de la page. Pour eux, la base de données MongoDb a été choisie, car elle est très rapide et a un format de donnée de type document. En effet, on sait que la structure de la donnée va changer au fil du temps, de ce fait, le type document est très pertinent.

Le SI est également divisé en deux grandes entités : le SI interne de BV, qui est basé sur des logiciels avec une architecture en couche ou monolithique. Celui-ci contient entre autre le Core BV. Il y a aussi le SI cloud qui contient tous les nouveaux logiciels et qui représente ce vers quoi les logiciels du SI interne de BV vont migrer au fur et à mesure, après parfois quelques modifications.

Le Core BV quant à lui, est un ERP fortement modifié ainsi que d'autres logiciels propres à Bureau Vallée. Il est probable que dans le futur, le prochain chantier soit sa division afin d'améliorer l'efficacité du SI.

On augmente ainsi l'interopérabilité du SI, ce qui permet de rapidement mettre en place des actions pour satisfaire les besoins et objectifs de l'entreprise. Il était par exemple nécessaire d'interconnecter Google LIA avec les microservices. Cela fut très simple grâce à l'utilisation d'auth0 et des librairies de Google.

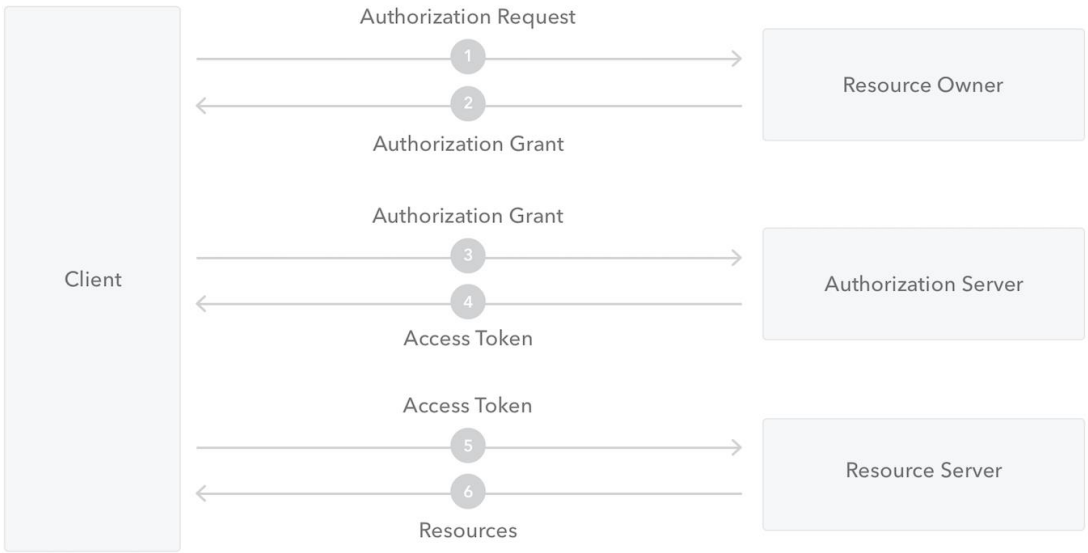

Figure 25 Schéma d'autorisation issu de la documentation technique du site internet auth0 © Google

Cependant, d'un point de vue sécurité, il faut noter que le nouveau site interne de Bureau Vallée a fait l'objet d'un piratage informatique entre le 26 mars et le 26 mai 2020 [22] [23]. Cette attaque a « *conduit à la mise en place frauduleuse d'un système de duplication des informations des cartes bancaires au moment d'un achat depuis notre site ».* Pour ce faire, « *le pirate reproduisait le formulaire de notre prestataire bancaire* ». La correction a été déployée dès que possible. De ce fait, le piratage est resté très localisé sur un élément du site internet, c'est-à-dire au niveau de la page de paiement [22]. Ainsi, le piratage n'a pas impacté l'activité de l'entreprise et son activité est resté normale sur la période du piratage. Ce piratage qui n'a pas eu de grand impact est peut-être dû à la séparation de l'architecture et également à l'utilisation de diverses technologies. En effet, la faille présente sur le site internet basé sur Magento 2 avec le langage PHP 7 n'était pas présent sur les microservices, qui eux sont basés sur le Framework Spring et le langage Java 9.

Cette nouvelle architecture a cependant permis à Bureau Vallée de répondre à ses objectifs stratégiques. Il y a maintenant un site dédié en Belgique et une autre à Nouméa, deux territoires ou Bureau Vallée souhaitait soit soutenir son entrée, soit supporter l'innovation, ce qui a été accompli. De plus, il est maintenant possible de gérer plus d'un million de produits et le stock associé, ce qui permet à Bureau Vallée d'avoir un catalogue plus étendu pour répondre aux demandes de sa clientèle et de ses associés. Cette extension du catalogue aurait été impossible précédemment sans l'utilisation de base de données NoSQL.

D'un point de vue critique, la gestion des grands domaines aurait pu être mieux gérée. Il y a eu de nombreux problèmes d'incohérence notamment au niveau d'un système d'intégration des données qui aurait pu être évité avec une meilleure gestion de l'architecture et une meilleure communication. En effet, des développements ont été effectués dans le SI cloud alors que sa place était en fait dans le SI interne de Bureau Vallée. Cela à causé des développements inutiles.

L'équilibrage des ressources aurait aussi pu être amélioré. Ainsi, de nombreuses autres choses auraient pu être mises en place, tel qu'un microservice dédié a la recherche des produits avec la base de données ElasticSearch, plutôt que d'implémenter directement le moteur de recherche avec un module Magento 2. Le site internet justement, n'étant qu'une solution temporaire qui était pensée à l'origine pour être remplacé rapidement. Cette solution temporaire risque fort de devenir permanente et d'être un nouveau logiciel assez monolithique. Cependant, les impératifs financiers de Bureau Vallée et la sortie du site internet, très liés au design et aux fonctionnalités ont prévalus.

Les développement effectués ne sont donc qu'une première étape dans la restructuration et la rationalisation du SI de Bureau Vallée dans le but de répondre à sa volonté de transformation numérique.

6 Conclusion

Les changements effectués sur le SI ont été bénéfiques pour Bureau Vallée. De nouvelles fonctionnalités très attendues ont été mises en place, comme le moteur de recherche optimisé avec auto-complétions. De plus, le site internet est en moyenne trois fois plus rapide, ce qui lui permet de rivaliser avec les autres entreprises du secteur.

Bureau Vallée a pu aussi agrandir significativement son catalogue de 30 000 produits à plus de 100 000. Ce changement de volume de catalogue aurait été infaisable avec l'ancien site internet au vu des besoins de l'enseigne. D'autre part, ses ventes e-commerce ont doublé depuis la mise en place du nouveau site internet. Pour l'enseigne, ceci est un gage de qualité de sa transformation numérique car Bureau Vallée tient à sa visibilité et à son image de marque.

A un niveau plus général, les microservices et les bases de données NoSQL sont de nouvelles technologies dont l'utilisation est pertinente afin de compléter les sites e-commerce actuels. Leur utilisation permet d'améliorer les performances et de s'adapter aux besoins des clients de façon rapide et permet d'améliorer la qualité de service.

Comme vue précédemment, cette adaptabilité est vitale dans le secteur changeant et compétitif du e-commerce. Grace à cette adaptabilité, une entreprise pourra se démarquer de la concurrence et s'imposer comme un leader dans son domaine.

Les microservices et les bases de données NoSQL ne permettent cependant pas de tout faire, car il faut toujours gérer les actions des utilisateurs. Cette gestion des actions utilisateurs restent au niveau du frontweb, à l'affichage. Cela permet donc une meilleure division des tâches qui va permettre à terme d'améliorer les performances.

Cependant, leur mise en place est plus pertinente pour une entreprise ayant un budget assez important et de grandes ambitions. A l'inverse, les PMEs ayant peu de moyens et devant commencer par vendre leurs produits afin de rester à flot, elles ont besoins d'un moyen de vente en ligne simple, ce qui correspond aux CMS. De ce fait, pour une PME, il est plus pertinent de débuter sur un site CMS grand public, puis, si besoin, commencer à restructurer le SI avec une vision à long terme des besoins.

Pour une grande enseigne qui cherche à améliorer ses services et répondre mieux aux attentes de ses clients, les microservices et les bases de données NoSQL sont donc tout à fait pertinents. En effet, leur utilisation va permettre de rationaliser l'ensemble du SI et mettre en place rapidement de nouveaux services pour se démarquer rapidement de la concurrence.

De ce fait, dans le cadre du e-commerce, l'utilisation des microservices et des bases de données NoSQL peut être un sérieux avantage pour une grande entreprise. Sa bonne utilisation lui permettra de devancer sa concurrence et de maintenir une satisfaction client optimale. Néanmoins leurs utilisations à mauvais escient et non coordonnés peut créer un système d'information disparate et moins performant.

Cette expérience professionnelle de deux ans m'aura permis d'acquérir et de développer de nouvelles compétences et d'autre part agrandir ma vision métier dans le domaine de l'architecture informatique.

Afin de terminer la transformation numérique de Bureau Vallée, je pense qu'il serait utile de poursuivre le projet en séparant les éléments monolithiques du SI interne de Bureau Vallée.

7 Table des matières

8 Bibliographie

[1] Fedav, «Chiffre clé é-commerce 2019,» [En ligne]. Available: https://www.fevad.com/wp-content/uploads/2019/06/Chiffres-Cles-2019_BasDef-1.pdf . [Accès le 10 03 2020].

[2] Capital, «Carrefour et Casino pourraient connaître un rattrapage,» 04 06 2020. [En ligne]. Available: https://www.capital.fr/entreprises-marches/carrefour-et-casino-pourraient-connaitre-un-rattrapage-le-conseil-bourse-du-jour-1371621. [Accès le 2020].

[3] Wikipédia, «Devops,» [En ligne]. Available: https://fr.wikipedia.org/wiki/Devops. [Accès le 12 03 2020].

[4] E-marketing.fr, «Résistance au changement,» [En ligne]. Available: https://www.e-marketing.fr/Definitions-Glossaire/Resistance-changement-242953.htm. [Accès le 10 06 2020].

[5] Wikimédia, «Propriétés ACID,» [En ligne]. Available: https://fr.wikipedia.org/wiki/Propri%C3%A9t%C3%A9s_ACID. [Accès le 08 05 2017].

[6] S. Crozat, «Dénormalisation,» [En ligne]. Available: https://stph.scenari-community.org/bdd/0/co/optUC004.html. [Accès le 12 03 2020].

[7] Wikimédia, «Théorème CAP,» [En ligne]. Available: https://fr.wikipedia.org/wiki/Th%C3%A9or%C3%A8me_CAP. [Accès le 24 03 2018].

[8] Wikimédia, «Modèle-vue-contrôleur,» [En ligne]. Available: https://fr.wikipedia.org/wiki/Mod%C3%A8le-vue-contr%C3%B4leur. [Accès le 03 01 2014].

[9] B. Hediard, «The evolution of software architecture,» 05 23 2015. [En ligne]. Available: https://medium.com/@benorama/the-evolution-of-software-architecture-bd6ea674c477. [Accès le 16 12 2019].

[10] N. T. Régis Behmo, «Maitrisez le théorème de CAP,» 28 11 2019. [En ligne]. Available: https://openclassrooms.com/fr/courses/4462426-maitrisez-les-bases-de-donnees-nosql/4462471-maitrisez-le-theoreme-de-cap. [Accès le 21 02 2020].

[11] P. m. E. P. David Hows, «MongoDB Basics,» [En ligne]. Available: http://file.allitebooks.com/20150523/MongoDB%20Basics.pdf. [Accès le 27 04 2020].

[12] C. BROUCHE, «MongoDB et le sharding,» 14 07 2017. [En ligne]. Available: https://www.supinfo.com/articles/single/4761-mongodb-sharding. [Accès le 19 10 2019].

[13] P. Neubauer, «Les bases de données orientées graphes, NoSQL et Noe4j,» 20 05 2010. [En ligne]. Available: https://www.infoq.com/fr/articles/graph-nosql-neo4j/. [Accès le 26 01 2020].

[14] O. G. Florent Biville, «Neo4J dans la mouvance NoSQL,» [En ligne]. Available: http://www.lateral-thoughts.com/DevInLove/prez/. [Accès le 22 01 2020].

[15] S. Newman, Building Microservices: Designing Fine-Grained Systems 1st Edition, O'Reilly Media, 2015.

[16] F. d. Peyrebrune, «Pourquoi la petite équipe est plus performante que la grande,» 19 03 2019. [En ligne]. Available: https://www.cadre-dirigeant-magazine.com/manager/pourquoi-la-petite-equipe-est-plus-performante-que-la-grande/. [Accès le 23 05 2020].

[17] A. Chandeze, «Les entreprises changent leurs processus en 15 mois,» 15 04 2020. [En ligne]. Available: https://www.lemondeinformatique.fr/actualites/lire-les-entreprises-changent-leurs-processus-en-15-mois-78761.html. [Accès le 20 05 2020].

[18] M. Kamaruzzaman, «Effective Microservices: 10 Best Practices,» 23 09 2019. [En ligne]. Available: https://towardsdatascience.com/effective-microservices-10-best-practices-c6e4ba0c6ee2. [Accès le 02 02 2020].

[19] d. space, «10 Best Practices for Building a Microservice Architecture,» [En ligne]. Available: https://www.devteam.space/blog/10-best-practices-for-building-a-microservice-architecture/. [Accès le 03 02 2020].

[20] G. Lea, «Microservices Security: All The Questions You Should Be Asking,» 29 06 2015. [En ligne]. Available: https://www.grahamlea.com/2015/07/microservices-security-questions/. [Accès le 06 05 2020].

[21] F. Blanche, «Citation Francis Blanche,» [En ligne]. Available: https://dicocitations.lemonde.fr/citations/citation-2456.php. [Accès le 01 06 2020].

[22] NextInpact, «Cyberattaque Bureau Vallée : « Le pirate reproduisait le formulaire de notre prestataire bancaire »,» 29 05 2020. [En ligne]. Available: https://www.nextinpact.com/brief/cyberattaque-bureau-vallee-----le-pirate-reproduisait-le-formulaire-de-notre-prestataire-bancaire---12551.htm. [Accès le 29 05 2020].

[23] Bureau Vallée, «Communiqué de Bureau Vallée sur son piratage,» 29 05 2020. [En ligne]. Available: https://bv-prd-fbi-fr-media.s3.amazonaws.com/pub/media/wysiwyg/CP-Bureau-Vallee-Intrusion-informatique-site-Bureau-vallee-fr.pdf. [Accès le 29 05 2020].

[24] R. Bruchez, Les bases de données NoSQL : Comprendre et mettre en œuvre, Eyrolles, 2013.

[25] M. F. Pramd J. Sadalage, NoSQL Distilled A brief guide to the ermerging world of polyglot persistence, Addison Wesley, 2013.

[26] CouchDB, «Cohérence finale,» [En ligne]. Available: https://guide.couchdb.org/editions/1/fr/consistency.html. [Accès le 26 05 2020].

9 Table des illustrations