STELLANTIS

1er année cycle ingénieur

Mémoire

2021-2022

Kilian **GOËTZ**

utbm
université de technologie Belfort-Montbéliard

Table des matières

Table des illustrations

Remerciements

Je tiens à remercier toutes les personnes qui ont contribué au succès de mon projet, de mon insertion professionnelle et qui m'ont aidé lors de la rédaction de ce mémoire.

Je voudrais dans un premier temps remercier, mes tuteurs entreprise M. CHANGEY et M. HOSATTE ainsi que ma tutrice académique Mme. CISTERNE, pour leur patience, leur disponibilité, leurs conseils judicieux ainsi que leur expertise.

Je remercie également toute l'équipe TIES pour son soutien permanent, ainsi que l'ensemble des enseignants-chercheurs de l'UTBM et de L'UIMM de m'avoir octroyé les compétences théoriques dans les différents domaines.

Je tiens à témoigner toute ma reconnaissance à mon entreprise Stellantis pour son accueil et son ouverture académique.

De surcroît, la réalisation de ce mémoire a été possible grâce au concours de plusieurs personnes à qui je voudrais témoigner toute ma gratitude :

Mes parents, Isabelle et David qui ont toujours été là pour moi. Mon petit frère Jimmy ainsi que mes amis Maxime, Axel et Vincent pour leurs encouragements.

Une page de remerciements est trop maigre pour citer l'ensemble des personnes m'ayant soutenu, c'est pourquoi je leur adresse mon plus profond respect.

1. Présentation et Analyse de l'entreprise

1.1. Stellantis une idée forte

Fondé le 16 janvier 2021, Stellantis est né de la récente fusion de deux anciens groupes automobiles PSA (Peugeot-Citroën) et FCA (Fiat Chrysler automobile). Le nouveau groupe s'inscrit comme le nouveau leader européen du marché de l'automobile en détrônant Volkswagen. Le nouveau groupe Stellantis se compose désormais d'un catalogue de plus de quatorze marques automobiles toutes autant mythiques les unes que les autres. À Hérimoncourt, Russelsheim, Turin, Londres, Bologne et Détroit les passions naissent. De la mythique Peugeot 205 en passant par l'Opel Manta, la Fiat 500, Lancia Stratos à la Jeep Willys, les quatorze marques ont chacune révolutionné le marché de l'automobile et sont devenues des symboles culturels, si bien que certaines automobiles sont connues de tous.

Stellantis hérite d'un grand passé et héritage culturel avec Peugeot fondé en 1896, Opel et Fiat fondé en 1899, Vauxhall fondé en 1903, Lancia fondé 1906, Alfa Roméo fondé en 1910, Maserati et Dodge fondé en 1914, Citroën fondé en 1919, Chrysler fondé en 1925, Jeep fondé en 1941, Abarth fondé en 1949, Leasys fondé en 2001, fiat Professional fondé en 2007, RAM fondé en 2009, DS Automobile fondé en 2014 et Free2Move fondé en 2016. Ensemble, ils représentent plus de 1000 ans d'héritage, sont présents sur plus de 130 marchés, une présence industrielle dans plus de 30 pays ainsi qu'une forte mainmise en Europe, Amérique latine et Amérique du Nord.

Les 300.000 hommes et femmes qui composent le groupe travaillent ensemble dans un esprit d'unité. Être agile et compétitif, ne pas chercher à être le plus gros, mais devenir meilleur, tel est leur philosophie : « Nous ne produisons pas seulement des voitures, nous créons des histoires. Des histoires qui font partie de nos vies, qui ont fait l'Histoire. Guidés par une vision claire : offrir la liberté de mouvement à toutes et tous. Aujourd'hui, la mobilité est en pleine transformation, cela touche chacun et rend nos rêves accessibles. Relier les villes, relier les personnes, connecter le présent et le futur. Il s'agit d'explorer de nouvelles opportunités, de répondre à de nouveaux besoins en harmonie avec le monde qui nous entoure. Dans cette nouvelle ère de la mobilité, il ne s'agit pas d'être le plus grand, il s'agit de chercher à devenir meilleur. C'est pourquoi, nous, un groupe de 300.000 hommes et femmes tous différents et talentueux travaillant dans 50 pays du monde et concevant des véhicules et des marques de mobilités emblématiques. Avec des millions d'idées, une créativité sans limite, une passion sans failles et l'esprit de compétition, nous sommes engagés à poursuivre de nouveaux horizons et à façonner un monde meilleur pour les générations à venir. Nous associons nos forces, nos talents et nos expériences, pour offrir à nos clients et à la société les solutions de mobilité les plus distinctives, les plus attractives, agiles et durables. C'est aujourd'hui que nous anticipons les besoins de nos clients et concevons les solutions de demain. Et nous savons qu'ensemble nous pouvons créer un avenir meilleur. Notre nouvelle histoire a commencé ». [1]

1.2. Organigramme et économie

Lors de la fusion de PSA et FCA, il a été décidé de délocaliser le siège social à Amsterdam. Stellantis a donc choisi comme statut juridique « Naamloze vennootschap » [2] abrégé NV est l'équivalent Français d'une société anonyme. Il a été décidé de subdiviser Stellantis en 9 comités : Conseil stratégique, business review, comité de programme global, comité industriel, comité allocations, comité régions, comité marques, revue de style et revue de marque.

1.2.1. Strategic and performance

Stellantis distingue 4 divisions chargées de la stratégie et des performances :

- Global corporate office
- Affiliates
- Performance
- Software

1.2.2. Regions chief and operating offices

Par sa dimension internationale, Stellantis a subdivisé sa direction par région dans le monde pour une meilleure opérabilité. Actuellement, 6 régions différentes composent ce comité :

- North America
- South America
- Enlarged Europe
- Middle east & Africa
- China
- India and Asia Pacific

1.2.3. Direction des marques

Stellantis se compose désormais d'un catalogue de plus de 14 marques, c'est d'ailleurs cette multitude de marques qui a inspiré le nom lors de la création de Stellantis. Le mot Stellantis vient du latin « stello » qui signifie « briller d'étoiles » ou « constellation d'étoiles » car chaque marque est une étoile qui forme toutes ensemble la constellation Stellantis. On peut pousser l'étude du terme Stellantis encore plus loin car « stello » a donné naissance au mot Italien « stela » qui signifie « étoiles », mais aussi au mot Français « stellaire », au mot Allemand « stern » et au mot Anglais « star ». De plus la constellation d'étoiles qui unit les marques est symbole d'unité comme le drapeau Américain qui unit les États d'Amérique. Ainsi à travers le nom Stellantis on retrouve tous les différents pays d'origine des différentes marques. Chaque marque du catalogue de Stellantis répond à un certain type de besoin. Plusieurs de ses marques collaborent dans leurs stratégies et sont donc regroupées en direction. [4].

Figure 1: Marques constellation Stellantis

1.2.3.1. Global SUV

Proposant une large gamme de SUV, Jeep a su se démarquer dans ce domaine proposant depuis ses débuts des SUV performants, fiables et indémodables. Personne n'oubliera les Jeep Willys devenues les valkyries libératrices de l'Europe et aujourd'hui adulées.

1.2.3.2. American Brands

Chrysler, Dodge et RAM furent fondées toutes trois aux États-Unis, aujourd'hui leur principal secteur de vente reste les États-Unis. Elles ont donc été regroupées en une entité « American Brands », toujours dans un esprit Américain avec des moteurs atmosphériques et de grosses cylindrées, laissant le choix entre des SUV, Pickup (Dodge RAM) et muscle car.

1.2.3.3. Core

Voici le « cœur » de Stellantis avec Citroën et Fiat qui offrent les plus bas prix pour des automobiles plus que correctes.

1.2.3.4. Upper Mainstream

Peugeot et Opel sont le « courant dominant » de chez Stellantis, ce sont eux qui génèrent le plus de ventes de tout le groupe. Ils proposent énormément de choix, un prix imbattable et un niveau d'expertise mondialement reconnu grâce à leur héritage. Peugeot 508, 3008, 208, Landtreck, Opel Corsa, Astra, etc. Il y en a pour tous les goûts à tout prix !

1.2.3.5. Premium

Offrant un plus haut niveau de qualité Alfa Roméo, DS et Lancia s'inscrivent comme les marques dites « Premium » de chez Stellantis. Marques prestigieuses qui ont même réussi à entrer dans les garages présidentiels, on n'oubliera jamais la DS19 de Charles de Gaulle, la DS 21 de Giscard d'Estaing, etc.

1.2.3.6. Luxury

Les voitures de course sont l'essence même de Maserati, plusieurs fois victorieuses aux évènements, mais aussi détentrices du record du monde de vitesse sur une décennie. Le constructeur italien est aussi à la ville le grand spécialiste de la GT (Gran Turismo) haut de gamme, une lignée de « travel » sportive, mais aussi de modèle exceptionnel voire hors norme. Elle est la gamme « Luxury » de chez Stellantis, un seul objectif pour la marque : la perfection. Elle propose une large gamme de véhicule pour tous les plaisirs, toujours dans un esprit GT. [3]

1.2.3.7. Mobility

Les services orbitant autour de l'automobile sont croissants, ils génèrent de plus en plus de bénéfices. On retrouve deux marques de service chez Stellantis : Leasys et Free2Move, ils proposent respectivement des services de leasing et de location de voitures. Ce sont également les marques les plus récentes, car le service automobile est un marché plutôt récent.

1.2.4. Global functions offices

Dernier comité principal de chez Stellantis, il est chargé de tout le fonctionnement interne de Stellantis avec plusieurs départements :

- Finance
- Human Resources & transformation
- General counsel
- Planning
- Purchasing & supply chain
- Manufacturing

- Design Chrysler/Dodge/Jeep/RAM/ Fiat Latin America
- Design Abarth/Alfa/Citroen/DS/Fiat Europe/Lancia/Opel/Peugeot/Vauxhall
- Engineering
- Sales & marketing
- Customer experience
- Communication & CSR
- Technology office

1.3. Systèmes d'informations :

Tous les secteurs ont été grandement impactés par le boum technologique qu'a importé les systèmes d'informations. Seules les entreprises qui ont entrepris une transition numérique ont pu subsister et profiter des innombrables avantages du numérique. Plus le temps passe et plus les besoins en informatique sont grands, il faut donc pour cela des experts dans tous les domaines. Stellantis a énormément investi dans le numérique et prévoit d'investir toujours plus. Aujourd'hui nous retrouvons deux principales entités qui s'occupent des systèmes d'informations et plus globalement de l'IOT (Internet of Things).

1.3.1. Software

L'entité software de chez Stellantis est la branche qui s'occupe de tous les services informatiques qui sont « monétisables » auprès du client. Ce sont tous les services qui orbitent autour de l'automobile. C'est une entité plutôt récente qui va grandement croître avec le temps, car les voitures sont de plus en plus connectées avec l'IoT et les pilotes automatiques. De surcroît, Stellantis va grandement investir dans cette entité, puisque récemment lors d'un communiqué, Stellantis a exprimé son desiderata d'investir 30 milliards d'euros à l'horizon 2024 pour le software, avec à la clef embaucher près de 4500 ingénieurs software supplémentaire en interne d'ici 2024. De plus, cet investissement promet une augmentation (selon les prévisions) du bénéfice de près de 20 milliards d'euros par année chez Stellantis. [5]

1.3.2. ICT (Information and Communicate Technologies)

ICT pour Information and Communication Technologies anciennement DDCE pour Digital Data & Connectivity Engineering, est l'entité chargée de développer les applications nécessaires au fonctionnement interne de Stellantis. Toutes les années, un budget est attribué au fonctionnement de chaque entité, lorsqu'une entité a besoin d'un service informatique, ils s'adressent à l'ICT qui facture l'entité bénéficiaire des services de l'ICT. L'ICT est le partenaire digital de nos entités business avec une intégration plus forte qu'une entité prestataire externe, car Stellantis utilise de nombreux outils qui lui sont propres ainsi qu'un mode de fonctionnement qui nécessite du temps d'adaptation. Avoir des équipes internes permet de gagner du temps et des coûts sur certains points, bien sûr lorsque les compétences ne se trouvent pas en interne ou que le coût est trop cher Stellantis s'adresse à des grands prestataires informatiques. Il faut toujours garder en mémoire que Stellantis n'est

pas une société de solution informatique mais un constructeur de voiture, il faut investir dans le numérique, mais sans pour autant y débourser de trop. L'ICT se décompose de la même manière que les branches de ventes présentées précédemment, c'est-à-dire par région. On retrouve donc 5 subdivisions dans l'ICT, trois de région et deux de support :

- EEID : Enlarged Europe ICT & Digital qui est la branche Européenne de l'ICT ;
- North America ICT & Digital qui est la branche Américaine de l'ICT ;
- Global Convergence ICT & Digital qui est une branche plus large au sens ou leur travail ne s'arrête pas à une région, ils jouent principalement le rôle de pont entre les deux régions informatiques ;
- Et les deux subdivisions de support qui sont HRT pour Human Resources & Tranformation et HURE pour Human Resources qui sont des subdivisions anecdotiques car composé de seulement quelques personnes chargées de la supervision de l'ICT.

1.3.2.1. Enlarged Europe ICT and Digital

La région Européenne de l'ICT est également découpée en plusieurs groupes spécialisés dans des domaines informatique précis :

- APDE : Application Delivery;
- CXCG : Customer Experience & CO2 Governance;
- ENGD : Engineering & Design;
- FSAP : Finance & SAP;
- HRCP : HR Corporate & Purchasing;
- MFG : Manufacturing;
- MKG : Marketing;
- OPDC : Operations & Data Centers;
- PS : Parts & Services;
- SC : Supply Chain;
- VST : Vehicles Sales & Trading;
- COM : Communication.

1.3.3. Affectation

Dans mon cas, j'ai été affecté à l'équipe TIES pour Technical Integration Expertise and Support faisant partie du groupe TECI pour Technical & Cloud Integration intégré à l'OPDC pour Operations & Datacenters.

1.3.3.1. Operations and Datacenters

Les missions principales de l'OPDC sont :

- Modifier les réglementations informatiques ;
- La gestion et l'orchestration des incidents ;
- La gestion et l'orchestration des problèmes ;

- Apporter un soutien technique local ;
- Gestion des infrastructures locales et outils pour l'exploiter ;
- Assurer l'intégration technique du support des applications ;
- Veiller à une surveillance permanente ;
- Gérer les datacenters et cloud provider ;
- Manager et apporter une expertise sur les servers, le stockage, produits, les middlewares et les bases données.

1.3.3.2. Technical & Cloud Integration

TIES pour Technical Integration Expertise and Support est l'équipe chargé de la standardisation et la mise en place des normes liés au continuous deployment chez Stellantis. Pour cela elle s'appuie sur un grand nombre de partenaire et possède des compétences variées dans de nombreux domaines. Elle a à charge l'innovation et l'évolution des méthodes des équipes. De ce fait, ses membres sont autonomes, curieux et font preuves d'initiative à des fins d'évolutivités.

Introduction

L'automobile connecte les Hommes depuis des décennies. Créée fin du 19ème siècle l'automobile a grandement évolué. La mobilité restera un besoin, car elle assoit le désir de liberté des Hommes. De la fin de l'ère industrielle à l'ère contemporaine, l'automobile comme tous les secteurs a dû s'adapter et cela passe par la numérisation et la connectivité. Dans notre monde ultra-connecté, la data et les services numériques augmentent drastiquement notre productivité. Les systèmes d'information ont connecté le monde, font tomber les barrières socioculturelles des sociétés et les uniformisent.

C'est dans cet esprit d'uniformité internationale que Stellantis s'est formé, impliquant un besoin informatique grandissant pour continuer à lier les personnes. Le nombre d'applications a augmenté de manière drastique, là où développer une application prenait du temps il y a quelques années, il est désormais plus simple de les développer que les déployer. De plus, maintenir les applications est compliqué et fastidieux, des évolutions ont été faites avec l'émergence depuis peu du DevOps qui consiste à ne plus dissocier le développement et l'opérationnel, néanmoins, il est vite compliqué lors d'un problème de reprendre une application faite il y a quelque temps.

Sentry traduction du Français « Sentinelle » est à l'image d'une sentinelle, un veilleur du moindre dysfonctionnement. Dédié à la veille applicative et utilisé à des fins de simplification de déploiement, Sentry répond aux problèmes actuels auxquels font face certaines grosses entreprises comme Stellantis. Cette nouvelle application web permet d'un simple coup d'œil de connaître l'état actuel des applications associées selon chaque rôle. Nom, description, Pilote fonctionnel d'application, développeur, Indus, etc.., tout est accessible en un clic.

Le développement de Sentry a pris du temps, à travers ce mémoire est relayé l'entièreté d'une année de travail chez Stellantis. Je vous souhaite une bonne lecture en espérant que mes travaux ne vous laisseront pas indifférents.

2. Travail réalisé

2.1. Problématique

Les outils d'exploitation, de livraison, de correction des anomalies chez Stellantis sont nombreux et sont basés sur des environnements différents. Il est complexe de connaitre précisément les caractéristiques d'un serveur et les besoins d'un développeur pour permettre une exploitation optimale. Ces critères sont basés sur des standards, normes (souvent internes) sur lesquels chacun s'appuie pour offrir les solutions adéquates aux utilisateurs.

À la suite de la récente création de Stellantis, issue de la fusion de PSA et de FCA le groupe est désormais présent tout autour du monde. Cette nouvelle dimension internationale a engendré un besoin informatique croissant et surtout une forte demande de résilience. La stratégie informatique du groupe s'est donc tournée vers les cloud publics, ceux-ci permettent un coût inférieur et une présence internationale comparé à nos datacenters. Il ne faut pas oublier que Stellantis est avant tout un constructeur automobile, le groupe ne peut pas se permette d'ouvrir des datacenters tout autour du monde. De surcroît il ne peut pas prétendre atteindre un meilleur niveau de sécurité et d'accessibilité que les providers cloud publics comme Microsoft ou Amazon.

Le problème qui sous-tend est que la plupart des normes et solutions évoqués précédemment ne sont plus valables pour les environnements de type cloud externe. En adéquation avec la stratégie informatique du groupe tournée vers le cloud first, la solution mise en application doit être cloud natifs tout en permettant une certaine promiscuité avec les outils internes.

2.2. Choix du cloud first

Aujourd'hui de nouveaux enjeux sont enclins à faire basculer Stellantis dans une solution de cloud first, c'est-à-dire migrer les applications cloud candidates depuis nos serveurs on-premise (on-premise signifiant interne/ dans nos murs) vers les cloud provider, il en est de même pour les nouvelles applications qui sont toutes destinées à être hébergé sur le cloud. Ce changement de stratégie informatique pour le cloud s'est fait ressentir comme une nécessité, depuis la fusion de PSA et FCA pour former Stellantis, le groupe a vu sa présence drastiquement élargie, de fait, le nombre de datacenter était insuffisant pour couvrir les secteurs de présences du nouveau groupe. De surcroît, le nouveau groupe avait besoin d'une solution informatique commune car les deux groupes possédaient des méthodes différentes. Ce changement de stratégie à un impact sur les équipes internes, TIES a donc vu ses effectifs formés sur les différents providers cloud public tel qu'AWS ou Azure (piste Google cloud non écarté). Une première migration de masse est en cours de prospection, les applications ont été jugés une à une pour vérifier leur potentielle conformité au cloud. Au vu du temps très court alloué et du nombre d'applications qui doivent être migrées, les différentes équipes prospectent la pléthore de méthode permettant de migrer rapidement. Stellantis ne peut pas faire une refonte complète de toutes ces applications pour l'instant permettant de les rendre

cloud natifs, la solution de migration qui sera choisie s'inscrit comme une migration de masse.

Figure 2 : Logo AWS, Azure et GCP

La question qui sous-tend est pourquoi ne pas avoir décidé de continuer sur du on-premise ou du cloud hybrid ? Comme évoqué plusieurs fois dans différentes parties, Stellantis est un constructeur automobile, ses bénéfices proviennent des voitures et non des solutions informatiques. La stratégie informatique est donc de favoriser les cloud provider public qui permettent avec un coût faible, une forte résilience, une scalabilité hors pair et un haut niveau de sécurité, d'accéder à des services SaaS, IaaS et PaaS (« Software as a Service », « Platform as a Service », « Infrastructure as a Service »). Ces différents services permettent chacun d'offrir des niveaux d'abstractions.

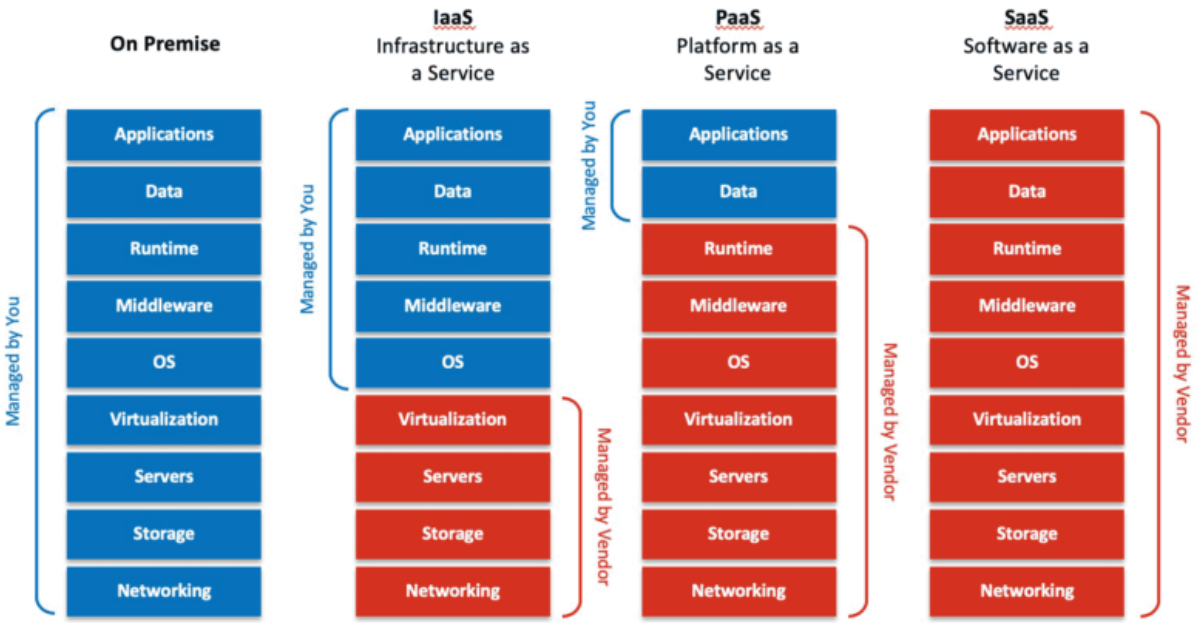

Figure 3: Différence On-premise, IaaS, PaaS et SaaS

Les cloud publics s'inscrivent comme la meilleure solution pour Stellantis. Nonobstant, il ne faut pas migrer l'entièreté sur un seul provider cloud pour garder la souveraineté de nos applications. Si du jour au lendemain les services proposés par un provider cloud ne nous convient plus on pourra facilement migrer nos applications d'un provider à un autre sans perdre la disponibilité et sans devoir se plier aux désirs des providers. De plus côtoyer une multitude de provider cloud permet de créer une ambiance concurrentielle qui octroie un plus important rapport de force pour les négociations contractuelles et ainsi économiser des coûts.

2.3. DevOps & CI/CD

Le DevOps est un mouvement qui concilie deux corps de métier : le développeur logiciel (dev) et l'administrateur de système et d'architecture (Ops). [6]

Le DevOps se divise en deux parties le Continuous Integration et le Continuous Deployment (ou Delivery), respectivement les parties gauche et droite du schéma ci-dessous.

Figure 4 : Schéma DevOps

2.3.1. Continuous Integration (CI)

L'intégration continue est une pratique qui consiste à créer automatiquement à partir du code source un livrable (artefact) prêt à être déployé. Ce dernier est ensuite stocké et versionné dans un référentiel de package. Ces Artefacts sont ensuite placés dans un dépôt tel que JFrog Artifactory.

Figure 5 : JFrog Artifactory logo

2.3.2. Continuous Deployment/Delivery (CD)

[7] Le Continuous Deployment ou Continuous Delivery pour « Livraison continue » est la possibilité d'obtenir des modifications de tous types, y compris de nouvelles fonctionnalités, des changements de configuration, des corrections de bogues et d'incidents et cela que ce soit en production ou entre les mains des utilisateurs en toute sécurité, rapidement et de manière durable.

L'objectif est de rendre les déploiements des affaires de routine effectuable sur demande et cela qu'il s'agisse d'un système distribué à grande échelle, d'un environnement de production complexe, d'un système embarqué ou d'une

application. Tout cela est réalisé en veillant à ce que le code soit toujours dans un état déployable.

Il est souvent admis que si l'on souhaite déployer des applications plus rapidement, nous devons accepter le fait d'avoir des niveaux de stabilité et de fiabilité inférieurs. Mais il s'avère que des recherches montrent que ce n'est pas le cas, les équipes hautement performantes fournissent systématiquement des services plus rapidement et de manière plus fiable que leurs concurrents. Et cela est vrai partout même dans des domaines hautement réglementés. Le Continuous Deployment est donc un avantage incroyable pour celles et ceux prêts à y investir des efforts.

Ainsi le Continuous Deployment permet :

- Un taux de risque plus faible : l'objectif du Continuous Deployment est de rendre les déploiements d'application indolore, par exemple le Blue/Green, qui permettent un déploiement sans temps d'arrêt et indétectable aux yeux des utilisateurs.
- Délai de mise sur le marché plus rapide : Les phases d'intégration, de test et de correction d'un Delivery traditionnel prend énormément de temps. Néanmoins, lorsque les équipes travaillent ensemble pour automatiser les processus de génération, de déploiement et d'approvisionnement, les développeurs peuvent réaliser des tests d'intégration dans leur travail quotidien et supprimer complètement ces phases.
- Meilleure qualité : Les outils automatisés permettent aux développeurs d'être plus libres et de concentrer leurs efforts sur les besoins des utilisateurs et les tests de plus haut niveau tels que les tests de performance ou d'expérience utilisateur.
- Coûts plus faibles : L'automatisation des déploiements permet de limiter les coûts « cachés » du développement, on va pouvoir limiter les pertes de temps des équipes et pouvoir économiser des coûts.
- Meilleur produit : Le Continuous Deployment rend plus économique le travail et moins long, ce qui permet aux équipes de pouvoir intégrer des demandes en plus en provenance des utilisateurs, de ce fait, le produit livré sera meilleur.
- Des équipes plus heureuses : aspect souvent négligé qui pourtant a un impact considérable et cela bien au-delà d'un projet, c'est le moral des équipes. La charge de travail liée au déploiement est une charge très pénible car c'est souvent de la résolution de bogue, or elle est très épuisante psychologiquement et souvent personne ne souhaite s'en occuper. Par conséquent, la livraison est stressante et crée une mauvaise ambiance de travail qui peut à l'avenir avoir des impacts bien plus importants comme le burnout. En réduisant cette charge de travail les équipes sont plus heureuses, donc plus efficaces et cela assure une pérennité pour l'entreprise.

2.3.3. Mise en place du Continuous Deployment/Delivery (CD)

Les mérites du « Continuous Delivery » sont innombrables, néanmoins il nécessite du travail et de l'investissement pour être mis en place. Lors d'un déploiement

traditionnel sans « Continuous Delivery » on suit une trame unidirectionnelle mais cela implique qu'au moindre problème lors de la trame on ne sait pas quelle étape en est la source. On se retrouve donc coincé, obligé de reprendre la trame depuis le début vérifier et modifier chaque étape, cela créer des délais supplémentaires et aigri les équipes d'industrialisateur.

Figure 6 : Delivery traditionnel

Pour pallier les problèmes de cette méthode surannée on met donc en place le « Continuous Delivery » dans lequel les étapes de production que sont le développement, le contrôle qualité et la livraison ne sont pas définitifs et se répètent automatiquement (ce que l'on appelle « production feedback ») au cours du processus de développement pour former une boucle, le tout forme une « pipeline » de Continuous Delivery. L'avantage de cette méthode est qu'elle permet d'effectuer régulièrement, graduellement et à intervalles réguliers des contrôles qualité aux produits logiciels. De surcroît, cette méthode permet d'avoir en continu des livrables alors que l'on travaille encore sur le produit, le client peut alors profiter rapidement du produit et n'a pas besoin d'attendre que la globalité du produit soit finie pour être déployé, l'avantage est donc double.

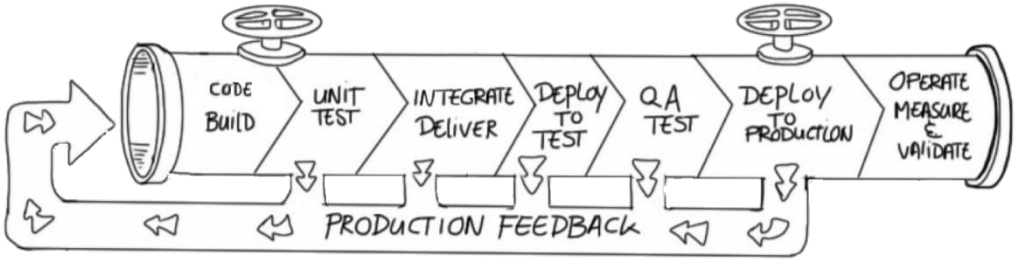

Figure 7 : Continuous Delivery Pipeline

Le Continuous Delivery est un réel avantage, néanmoins bien que la mise en œuvre théorique du pipeline semble simple il est vraiment compliqué de le mettre en place et cela est d'autant plus vrai pour des entreprises tel que Stellantis puis-ce que les applications sont nombreuses et chacune différentes, l'automatisation du pipeline en devient complexe. Cependant on peut simplifier les étapes et réduire les interventions humaines qui sont souvent la source principale des problèmes, comme disait Voltaire « Quel Homme est sans erreur ? Et quel roi sans faiblesse ? ». Ainsi Stellantis à construit avec le temps un pipeline pour assurer le Continuous Delivery souvent à partir d'outils externes à l'entreprise (PaaS voire partie 2.2). Il est à noter que ce schéma est dès lors complexe à comprendre dans sa globalité et que les différentes étapes seront développées a posteriori à travers le projet.

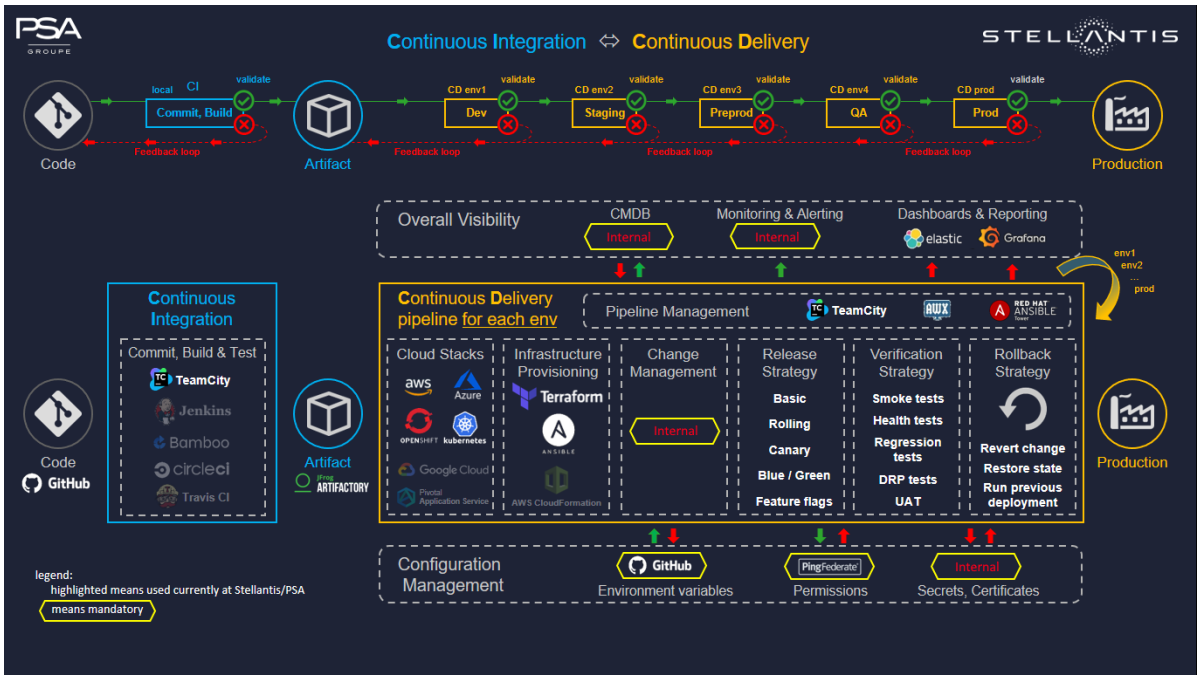

Figure 8 : Continuous delivery pipeline Stellantis

2.4. Introduction du projet

En adéquation avec les problèmes relatifs à la mise en place du Continuous Delivery, le projet confié est dans le continuum du consensus de démocratisation du DevOps et plus particulièrement du CI/CD chez Stellantis. Ce nouvel outil s'inscrit dans un esprit évolutif, il sera développé dans un premier temps en tant qu'outil de supervision apportant ainsi une solution aux problèmes de visibilité connue par l'entreprise. De facto le nombre important d'outils dans la facilitation du Continuous Delivery a complexifié la recherche d'informations, ce qui implique des désagréments lors des différentes étapes de pré-production, production et vie courante. Le projet sera donc dans un esprit de centraliser les données et jouera le rôle de superviseur. Dans un second temps l'application évoluera et permettra la visualisation de données jusqu'alors non répertoriées dans d'autre outils internes Stellantis, l'application étendra ses compétences aux référencements de données. Dans un avenir plus lointain l'application aura une grande montée en puissance et sera plus qu'un simple référentiel pour devenir acteur en automatisant le Continuous Delivery. L'application jouera le rôle d'intermédiaire entre les différents outils et cela grâce à son grand référentiel d'informations.

Le projet, d'une grande envergure, se déroulera sur l'ensemble de mon cursus d'apprentissage et perdura dans l'environnement Stellantis pour de nombreuses années. Un regard avisé sera donc nécessaire dans l'études préliminaire afin de faire les bons choix pour augmenter la longévité de l'application.

2.5. Nom de l'application

Le choix du nom de l'application est primordial, car un nom simple permet d'être retenu plus facilement. De plus un nom en rapport avec son utilité accroît la compréhension de l'utilité de l'outil. En concordance avec l'orientation du projet défini précédemment il a donc été décidé d'appeler l'application « Sentry » (prononciation : /ˈsentri/) est un mot anglais signifiant « sentinelle ». À l'image des sentinelles l'application joue le rôle principal d'informateur, gardant en permanence un état d'attention sur toutes les applications. Un petit mot, simple et efficace qui se retient facilement.

2.6. Logo de l'application

Le logo est la première image que verra l'utilisateur, il doit être unique pour être reconnaissable et il doit également évoquer les activités, les domaines, les produits et services de l'application. Le logo doit être le reflet parfait de la célèbre locution « une image vaut mille mots », c'est pourquoi sa conception a pris du temps, plusieurs essais et un sondage auprès des potentiels utilisateurs.

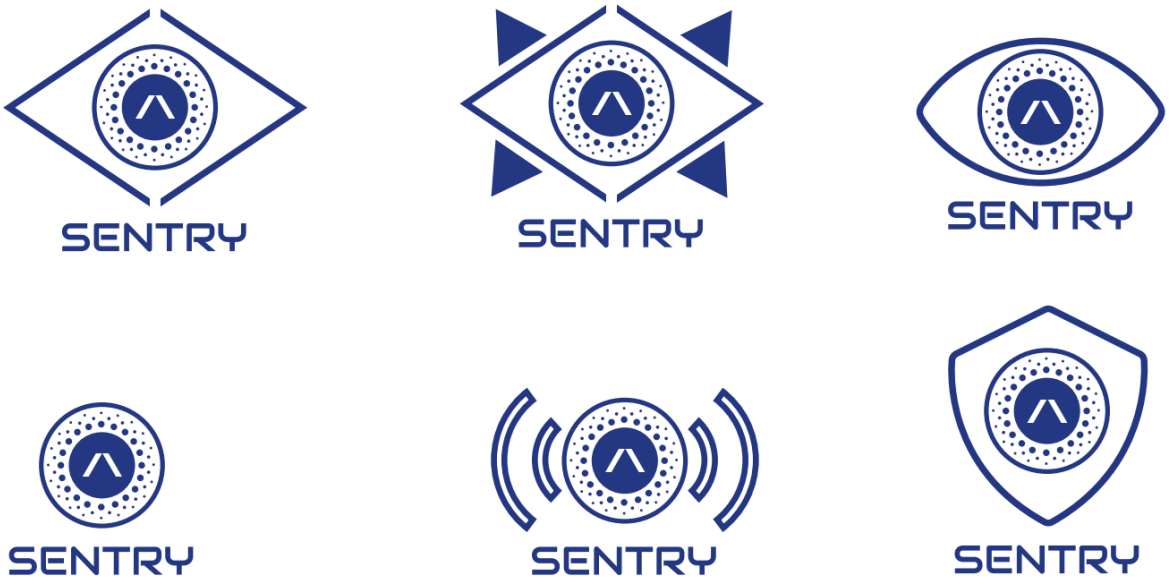

Figure 9 : Liste des logos pour Sentry

En tout 6 logos ont été réalisés, ils ont tous une signification particulière et rappellent tous, le rôle de Sentry et son secteur d'activité. On retrouve dans chacun des logos le A central de Stellantis avec les différents points qui constituent la constellation Stellantis (l'ensemble des 14 marques unis à travers une même constellation). De plus, la constellation forme l'iris de l'œil qui évoque le regard constant qu'apporte Sentry sur les applications. Les différents logos sont peu détaillés, c'est une règle universelle qui permet ainsi d'exporter le logo en format vectoriel (SVG) sans que cela ne soit trop lourd, le poids des différents logos varie donc entre seulement 5 à 10 ko. Le poids à un réel intérêt car cela impactera la vitesse

d'affichage lors du chargement du site (pour éviter d'alourdir les requêtes inutilement). Après concertation (sondage) le logo retenu pour Sentry est :

Figure 10 : Logo retenu pour Sentry

2.7. Gestion du projet

2.7.1. Cycle en V

Stellantis comme beaucoup de grosse entreprise fonctionne de deux manières différentes selon les projets, il y a la méthode de gestion dite cycle en V (qui est la plus ancienne et qui devient surannée) et la méthode dite agile (plus récente et palie certains problèmes rencontrés sur un cycle en V). Le choix de la gestion de projets est primordial, une mauvaise gestion de projet peut ralentir, désorganiser, voire suspendre un projet. Tandis qu'une bonne gestion de projets permet une meilleure efficience (efficacité et rapidité) donc par subséquent des coûts (car le temps c'est de l'argent).

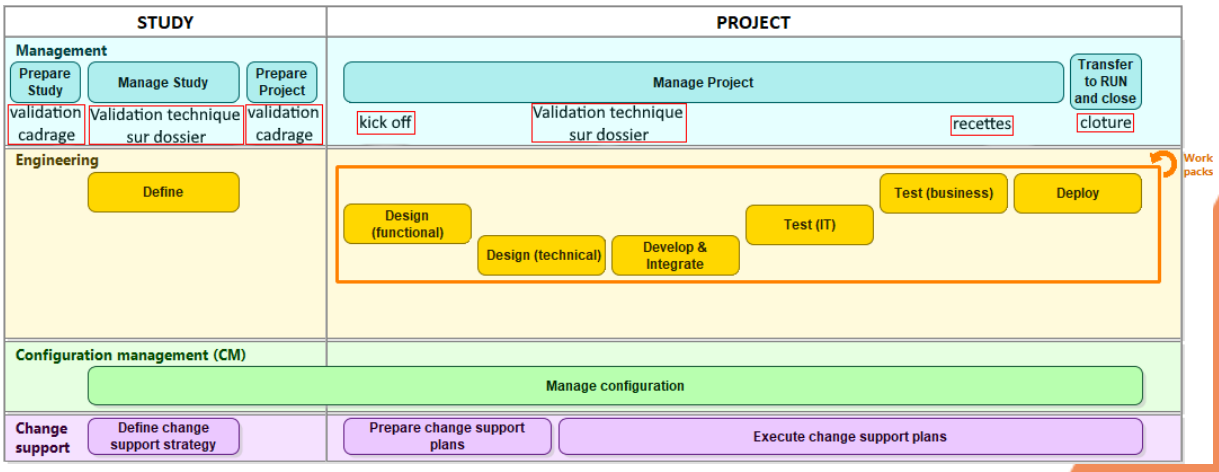

Figure 11 : Cycle en V

La *Figure 11* est le schéma représentant le cycle en V (waterfall ou monolithique) de chez Stellantis, cette méthode est risquée car elle fonctionne de manière linéaire et développée dans les années 1980 ce qui le rend dépassé par rapport aux

technologies actuelles . Cette méthode possède des avantages mais aussi de nombreux défauts. Elle a comme avantage de ne pas avoir besoin de revenir en arrière ce qui implique de faire une documentation précise et exhaustive lors de la phase de conception, c'est un processus simple à comprendre et facile à mettre en œuvre car il suffit de suivre des étapes. Cette méthode est pratique pour des entités séparées dans lesquelles les différents collaborateurs n'ont pas besoin de se côtoyer régulièrement (c'est le cas notamment chez Stellantis qui privilégiera un cycle en V quand c'est un projet fait par un prestataire, de cette manière Stellantis fourni la demande et le cahier des charges et le prestataire gère de la façon qu'il veut de son côté tant que le livrable est bien fourni à Stellantis dans le temps imparti). Cependant, cette méthode possède énormément d'inconvénients résumables en deux mots « l'effet tunnel », une fois le projet lancé plus de retour en arrière mais si le besoin du client change en cours de projet il ne pourra pas être pris en compte, de ce fait on peut se retrouver avec un projet répondant à aucun besoin du client. [8]

2.7.2. Méthode agile

La méthode agile est née aux États-Unis en 2001 par un groupe d'experts qui cherchait à mettre au point une nouvelle méthode permettant de diminuer les échecs engendrés par le cycle en V, la méthode agile fut publiée à travers le manifeste agile [9]. La méthode agile met en son cœur une plus grande implication du client et une meilleure réactivité des équipes définit en 4 valeurs fondamentales :

- Les individus et leurs interactions plus que les processus et les outils
- Des logiciels opérationnels plus qu'une documentation exhaustive
- La collaboration avec les clients plus que la négociation contractuelle
- L'adaptation au changement plus que le suivi d'un plan

Le groupe d'experts a également défini 12 principes sous-jacents aux valeurs précédentes :

- Le client est la priorité
- Accepter et prendre en compte les changements
- Effectuez des livrables à courte intervalle (quelques semaines à quelques mois maximum)
- Les utilisateurs et développeurs doivent être mis fréquemment en relation
- Le projet se réalise avec des personnes motivées
- Privilégié un dialogue direct
- L'avancement du projet se détermine à travers les évolutions fonctionnelles du projet
- Le rythme demandé à l'équipe projet doit être soutenable et réalisable à un rythme constant
- Avoir un regard continue sur l'excellence technique et la conception
- Rester simple
- L'équipe projet doit être responsabilisée

- Convenir à intervalles réguliers d'une réunion afin de définir clairement le comportement à adopter et les processus à réaliser

Les avantages de la méthode agile se résume en trois mots : flexibilité, collaboration et communication.

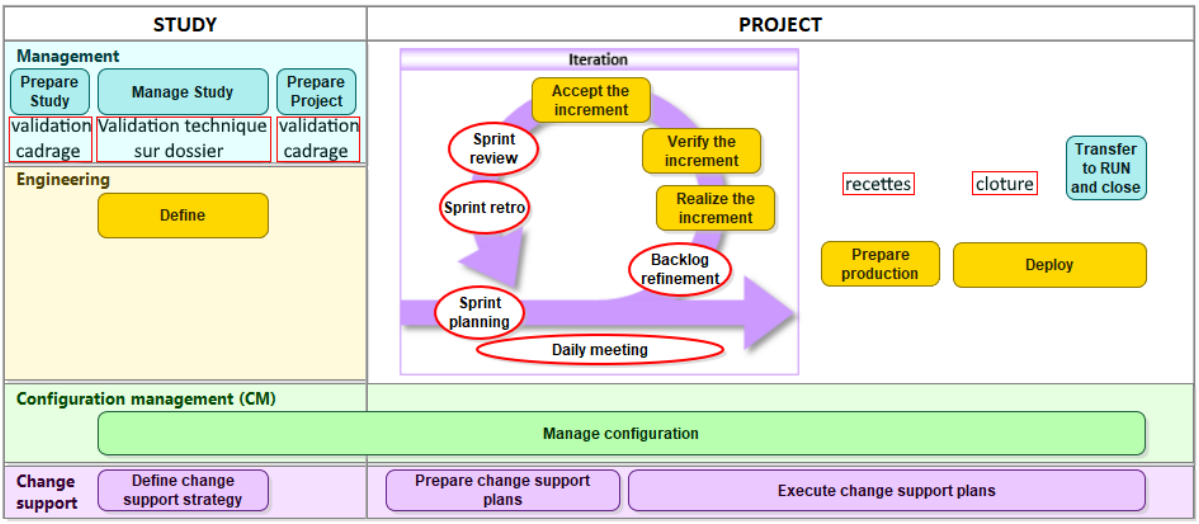

Figure 12 : Méthode agile

2.7.3. Prepare study

La *figure 12* du précédent chapitre représente la méthode agile au sein de Stellantis, de ce fait beaucoup de termes sont propres à Stellantis, c'est pourquoi cette partie et les suivantes sont là pour les expliciter. La première phase de n'importe quel projet chez Stellantis commence par la « prepare study » ou « pre-study », c'est une phase très importante durant laquelle deux réunions sont obligatoires représentant la validation de cadrage. L'objectif de cette première phase est de définir l'organisation de référencement des besoins métiers et étudier les solutions informatiques répondant à ces besoins. Pour cela plusieurs tâches sont demandées :

- Établir la portée des étapes
- Identifier les processus applicables à l'étude préliminaire et aux adaptations
- Faire l'organigramme des tâches du projet
- Faire les choix des sous-traitants/prestataires nécessaires
- Estimer les étapes de l'étude préliminaire
- Construire un planning global macroscopique de l'étude
- Initialiser les plans de mesure via des KPI
- Identifier et analyser les risques
- Identifier et analyser les problèmes
- Préparer le contrôle et le suivi du projet
- Préparer les canaux de diffusion d'informations (exemple Teams, sharepoint, etc.) et les implications des collaborateurs
- Élaborer les documents pour l'organisation du projet

- Obtenir l'engagement des collaborateurs.

Les réunions de validation de cadrages quant à eux sont les réunions permettant de faire valider le budget par les équipes budget. Une fois le budget validé une note de budget est distribué qui sera demandé à plusieurs moments lors du projet.

2.7.4. Manage study

Après avoir obtenu le financement lors des réunions de validation de cadrage durant la phase « Prepare study », la phase de « Manage study » peut commencer. Cette phase est très importante car elle contient les premières réunions de projet pour son lancement, mais cette fois-ci orientée technique. Les tâches de cette phase sont :

- Réaliser la réunion de lancement de projet
- Construire un planning détaillé
- Engager et coordonner les travaux
- Suivre l'avancement des activités (tâches et livrables)
- Surveiller l'adéquation des ressources
- Surveiller et contrôler les risques
- Gérer les actions et les problèmes
- Préparer et mener les revues de jalons
- Mettre à jour le plan de gestion de projet, les supports de la réunion de lancement et le planning
- Préparer et animer les réunions
- Estimer les différentes solutions
- Faire valider formellement les livrables

La réunion la plus importante est la réunion de lancement (ou « Plan management Project ») c'est celle-ci qui définit beaucoup de choses comme :

- Les objectifs, les défis, les hypothèses et les contraintes
- Les autres projets et applications impliqués
- Le coût initial
- Un macro-planning et ses dates buttoirs de validation
- Les membres de l'équipe projet et ses livrables
- L'endroit de stockage des documents
- La gestion des anomalies et des demandes de changement
- La gestion des risques
- Les stratégies de tests
- Les types de réunion à conduire durant le projet

Il y a également deux autres réunions d'une importance capitale durant la validation technique sur dossier.

La première a pour but de présenter :

- Le contexte : les défis, les contraintes, identifier les risques
- Les solutions possibles
- Les éléments d'architecture techniques et fonctionnels

- Les données personnelles
- Pour chaque solution : une description technique, un macro-planning, le coût et les investissements
- Et prendre en considération tous les éléments de la partie RUN (donc la mise en production)

Elle doit décider pour chaque solution technique si elle est retenue, refusé ou si elle a besoin d'une étude supplémentaire avant de prendre la décision. Elle doit également valider la solution retenue, son coût et son temps de développement.

La première traite beaucoup de l'aspect technique tandis que la seconde traite plus d'un aspect général du projet c'est-à-dire valider l'engagement du projet à la suite de la première réunion avec tous les collaborateurs.

Durant cette seconde réunion il doit être présenté :

- Le contexte : rappel des exigences, les risques, les indicateurs clés de performance attendus (KPIs).
- L'architecture fonctionnelle et technique
- Les données personnelles

Elle doit valider les résultats de l'étude préliminaire et la décision de la première réunion ainsi que les modalités d'implantation du projet (coût et investissement, planning, stratégie de test, etc.). Elle est en quelque sorte la réunion de validation complète qui met fin à la phase d'étude.

2.8. Sentry study

Le projet Sentry n'a pas dérogé à ces besoins, hormis pour la phase « prepare study » car le budget est prélevé depuis une note déjà existante dédié à la VCO (vie courante applicative) les réunions de validation de cadrage n'ont donc pas eux besoins d'être organisé. La réunion de lancement a été réalisé milieu janvier 2022 avec la participation de l'entièreté de l'équipe TIES afin qu'ils valident le projet. Cette réunion a nécessité plusieurs semaines de préparation avec plusieurs dossiers à complété :

- Le dossier de conception technique
- Le dossier technique de l'infrastructure
- Le dossier des spécifications fonctionnelles
- Le dossier de la stratégie des tests
- Le dossier de présentation de la réunion de lancement
- Ainsi que définir le but du MVP (« Most Viable Product ») et sa première date de mise en production

2.8.1. Dossier de conception technique

Le « Technical design file » est un des dossiers les plus importants lors du lancement du projet il regroupe l'entièreté des choix technologiques et leurs raisons.

C’est un dossier qui est demandé à plusieurs reprises notamment dans la réunion de lancement, lors du comité de contrôle technique et lors du comité d’orientation.

2.8.1.1. Choix du type de client

En informatique on distingue deux types de choix clients, les clients lourds et les clients légers. Néanmoins, avec le temps les clients lourds sont de plus en plus dépréciés au profit des clients légers. De nos jours, avec l’efficience des systèmes d’information et les connexions ultra-rapide, les applications web sont devenues une des meilleures solutions. Les applications web ont l’avantage de ne pas avoir besoin d’être installé, d’être ouvert partout et sur une bonne partie des systèmes d’informations actuels. Beaucoup d’applications vieilles en client lourd tendent à devenir des applications web et resteront des applications en client lourd que celles qui ne peuvent être supportées par un client web ou par besoin technique. Au vu de la pléthore d’avantages qu’offre le web Sentry a pris pour orientation de devenir elle-même une application web.

2.8.1.2. Choix de l’architecture

Une autre question s’offre désormais à nous, quelle architecture web choisir ? C’est un choix important car cela induit la manière dont les couches web vont interagir entre elle. Le choix de l’architecture est souvent lié à ce que l’on souhaite faire comme site internet. Est-ce que le site a besoin de stocker des données ? est-ce un simple site statique sans aucune interaction ? Dans notre cas Sentry va communiquer avec beaucoup d’applications qui sont des référentiels. De plus, Sentry devra également stocker certaines informations des applications et référencer des éléments qui ne sont pas encore référencés dans d’autre applications. Sentry aura donc besoin des trois couches web c’est-à-dire une partie Front-end (qui est en quelque sorte la couche d’interface homme-machine), la couche back end (le traitement des informations en arrière-fond et la récupération des informations auprès des applications) et la couche Database (Couche de la base de données). L’architecture qui correspond le mieux aux besoins de Sentry est une architecture dite MVC pour « Model View Controller », cela signifie que l’utilisateur va manipuler des « Controller » (front-end) qui vont mettre à jour les « Model » (back-end et database) qui eux-mêmes viendront remplir et mettre à jour le view (front-end). Le rendu sera hybride, une partie du rendu est fait en back-end pour les rendus volumineux afin d’éviter des temps de chargement trop longs et l’autre partie sera rendue depuis le front-end pour les rendus légers. Si on avait choisi un rendu qu’en back-end certaines requêtes auraient été trop volumineuses et si on avait choisi que le rendu en front-end, les actions auraient été limités. De surcroît, il ne faut pas oublier que le front-end est accessible par l’utilisateur dont le code source, donc choisir que du rendu en front-end est dangereux. Des actions critiques seront donc faites-en back-end plutôt qu’en front-end, certaines seront même doublées (exemple un champ utilisateur peut être vérifié en front-end et en back-end).

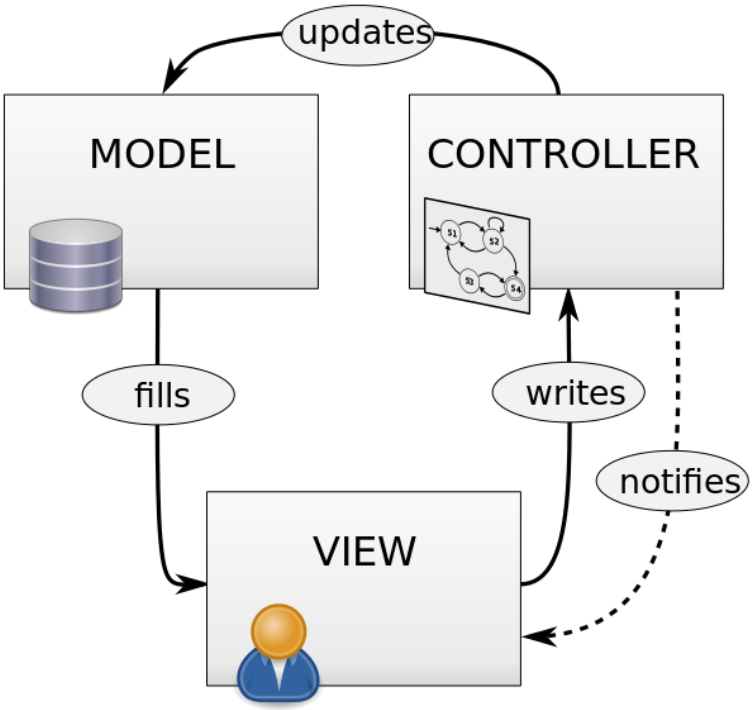

Figure 13 : Schéma d'une architecture MVC

L'application utilisera également la technologie Docker, permettant de créer une architecture dite micro-services. Docker est une plateforme logicielle open source qui permet de gérer, créer et déployer des conteneurs. Les conteneurs sont des environnements d'exécution léger qui offrent une alternative à la virtualisation classique. L'avantage des conteneurs c'est qu'ils permettent de créer des environnements décorrélés, chaque service est stocké dans un conteneur qui a son propre environnement d'exécution et bibliothèques. Cela permet de déstructurer les applications en plus petites d'où le nom d'architecture micro-services. C'est quelque chose d'assez nouveau dans le monde de l'informatique et très apprécié. Le fait que tout soit indépendant permet une meilleure résilience car chacun à son propre environnement, donc si l'environnement tombe toute l'application ne tombera pas.

L'utilisation de docker est vraiment simple, dans un premier temps il faut créer une image. Une image c'est l'OS ou l'environnement que l'on va faire tourner dans le container. Ensuite il suffit de créer le conteneur à partir de l'image puis de provisionner le code à exécuter dans le conteneur. Docker est une technologie très simple d'utilisation d'où sa grande popularité, néanmoins Docker se limite à l'orchestration de quelques conteneurs. Étant donné le nombre de conteneurs dont nous avons besoins, Docker est largement suffisant.

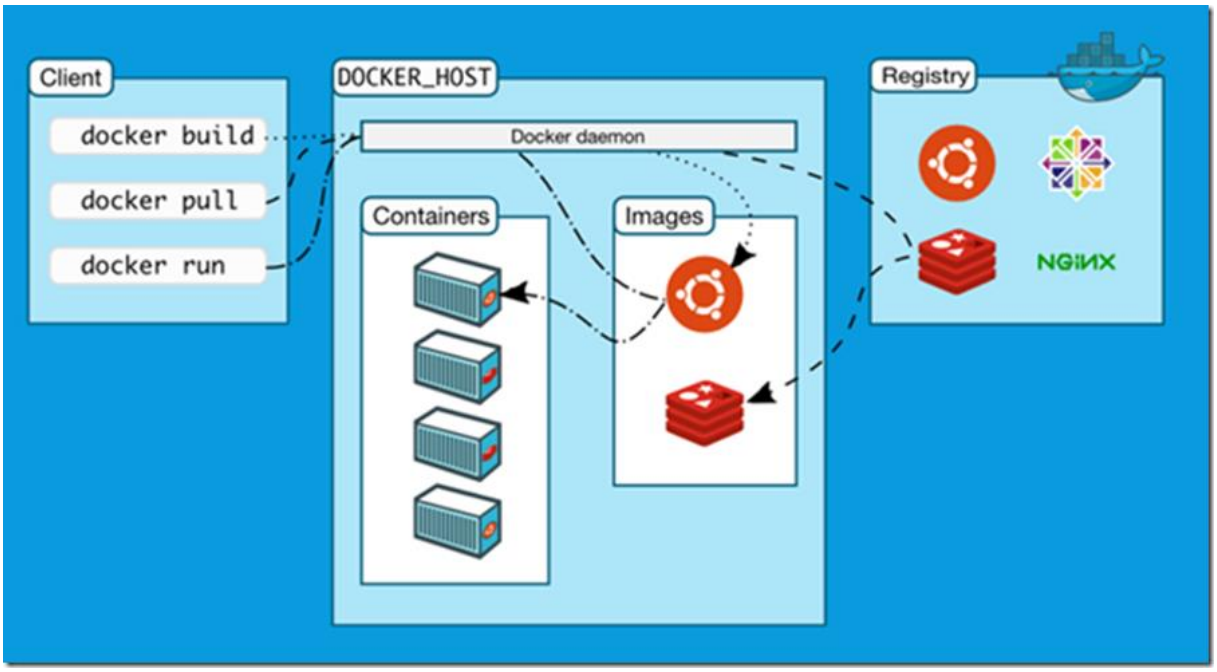

Figure 14 : Schéma fonctionnement Docker

2.8.1.3. Choix de l'hébergeur

En concordance avec la stratégie cloud first de Stellantis, Sentry sera hébergé sur Azure. Nous avons fait le choix d'Azure pour plusieurs raisons. La première raison est qu'au sein de l'équipe TIES on connaît déjà très bien le monde AWS, cependant on ne veut pas rester coincé chez un seul cloud provider. Sentry permet alors à l'équipe TIES de former des éléments internes chez Microsoft et de savoir comment se déroule une démarche projet complète chez Azure. C'est un choix gagnant pour tout le monde, pour TIES cela leur permet d'explorer des nouvelles technologies sans pour autant devoir provisionner une personne interne. Pour l'alternant, c'est très enrichissant il peut se former sur des technologies émergentes qui ne seront que bénéfique pour son futur et peut expérimenter une démarche projet complète. De surcroît, Sentry est un projet nécessitant peu de ressources (mais avec beaucoup d'ambitions), on a donc un projet « simple » à mettre en place chez un hébergeur inconnu et une application évolutive qui permet d'explorer plein de services. Enfin, la dernière raison est qu'Azure est géré par Microsoft et que Stellantis possède de nombreuses licences chez Microsoft (Office 365, Yammer, etc..) cela permet de lier plus facilement certains outils avec Sentry.

2.8.1.4. Choix du Framework web frontend

Le choix des technologies utilisés est crucial, il est important de choisir des technologies pérennes, commune et novatrice. Il est également important de choisir des technologies simples à maintenir, c'est-à-dire des technologies low-code ou alors automatisés les tâches qui nécessitent l'intervention d'un développeur. Chez Stellantis et globalement dans les entreprises, les équipes vie courante et

développeur ne sont souvent pas les mêmes personnes. Même si la tendance est celle du DevOps (*Cf partie* 2.3), il reste rare qu'un développeur s'occupe de la vie courante. Que quelques applications ont cette chance, dont Sentry. Mettre un développeur sur le support d'application signifie que l'on pourra moins exploiter le développeur pour ses compétences de développeur car il sera à mi-temps sur le support d'une application. Étant donné que mon alternance dure 3 ans et que Sentry a comme ambition de s'inscrire comme outil incontournable du CI/CD Stellantis. Il est important que les technologies soit simplement maintenables, d'où la nécessite du low-code. Pour la partie Front-end il a été décidé de choisir le Framework React JS avec la librairie graphique Elastic UI. React JS à une syntaxe très simple avec une notation en JSX [14] (qui ressemble à 99% à du HTML [15]). Il permet également de faire du web dynamique sans avoir une connaissance très poussée en Javascript [16] et surtout du fonctionnement du DOM [17] qui peut être très complexe pour des développeurs novices (Le DOM est le Document Object Model permettant de « charger » le rendu du navigateur et son contenu à partir du code provisionner). Autre avantage de React JS est qu'il offre un système de composant réutilisable et très simple à mettre en place. Une question légitime que l'on peut se poser est pourquoi avoir choisi React JS ? car il existe plusieurs Framework permettant d'effectuer un travail similaire.

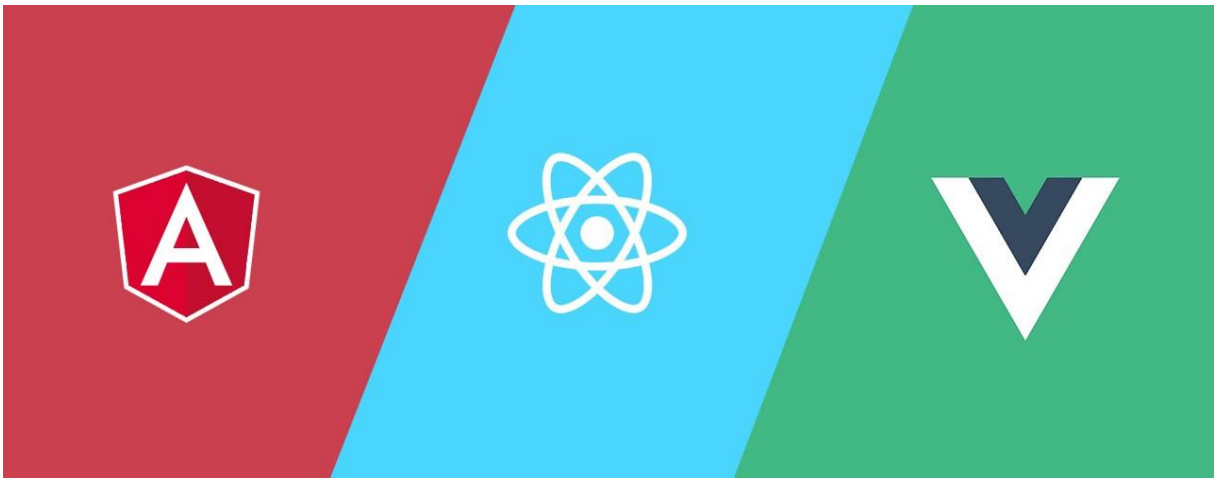

Figure 15 : Logo AngularJS, ReactJs et VueJs

Il existe trois principaux Framework pour le développement Web en Javascript : AngularJs [19], ReactJs [18] et VueJs [20].

- AngularJs est le plus vieux d'entre eux, il fut développé par Google en 2009.
- ReactJs est le deuxième plus vieux et développé par Facebook (Meta) en 2013.
- VueJs est le plus jeune et fût développé par un ancien employé de Google, il vient pallier plusieurs problèmes rencontrés sur AngularJs. Pour ces raisons il avait été renommé affectueusement le tueur d'AngularJs à sa sortie. Néanmoins, AngularJs a depuis grandement évolué, notamment avec une version 2.0 sortie en 2016 qui a nettement changé les choses. VueJs est donc plus une version simplifiée d'AngularJs, dans la version 3 de VueJs il tend à se rapprocher syntaxiquement et fonctionnellement à ReactJs.

Pour faire le choix du langage Front-end à utiliser il est important d'étudier pour chacun d'entre eux :

- Les tendances du langage
- Les parts de marché du langage, plus le langage possède des parts de marché plus il y a des chances que la future personne qui s'occupera de l'application connaîtra le langage.
- Les performances
- La courbe d'apprentissage et le temps pour maîtriser le langage

Les tendances des langages peuvent être étudiées via Google trends, un site permettant de voir le nombre de recherche sur un sujet, langage, etc. et de les comparer. Un grand nombre de recherche implique que les personnes sont intéressées par le langage. Un autre indicateur de la tendance est le nombre d'installations du Framework via un manager de package (npm ou yarn, ici on étudiera que npm car il est majoritaire dans l'utilisation des package manager.

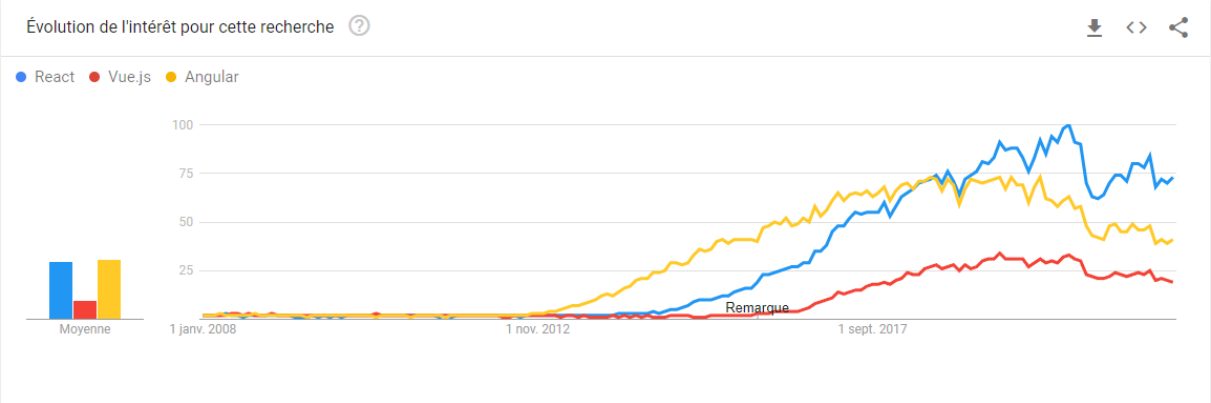

Figure 16 : Tendances ReactJs, VueJs, AngularJs

La *figure 16* représente les courbes extraites depuis Google trends du nombre de recherche concernant AngularJs, VueJs et ReactJs (depuis le 1er janvier 2008, google trends ne peut pas remonter plus loin et dans tous les cas, cela ne change rien puisque les langages sont sortis bien plus tard). On remarque directement que pas longtemps qu'après l'apparition de VueJs, les recherches liées à Angular JS ont décru (ce qui confirme que VueJs était bien un Angular Killer). De surcroît, on voit que ReactJS est le plus recherché sur google ce qui signifier qu'il représente un intérêt certain de la part des développeurs.

Néanmoins une forte recherche n'implique pas forcément que beaucoup de développeurs travaillent avec ReactJs. C'est pourquoi il faut regarder le nombre d'installations via les packages manager, ce qui donnera une idée du nombre de projet créer avec le langage.

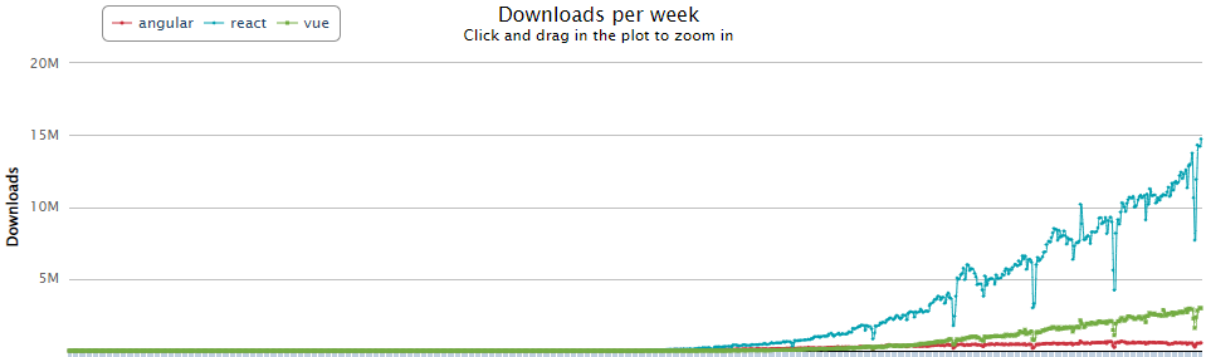

Figure 17 : Nombre de téléchargements par semaine de VueJs, ReactJs et AngularJs via npm

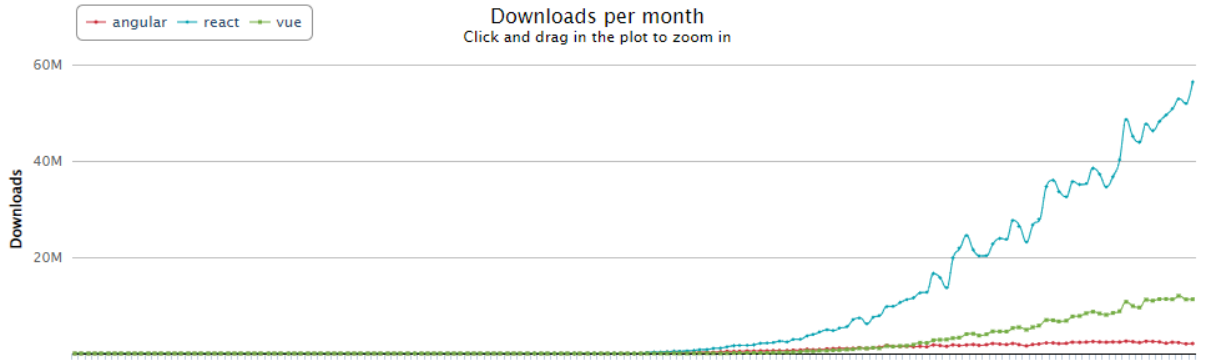

Figure 18 : Nombre de réléchargements par mois de VueJs, ReactJs et Angular JS via npm

Le nombre d'installations via npm nous montre que ReactJs est largement majoritaire dans l'utilisation des projets. On remarque également que le nombre d'installations de VueJs est bien supérieur à AngularJs alors que les tendances de recherches plaçaient AngularJs en tête par rapport à VueJs. Pour la partie des parts de marché ReactJs est largement en tête, suivi par Vue Js puis AngularJs.

Les tendances et parts de marché sont importantes, néanmoins cela ne suffit pas à faire le choix. Il faut étudier les performances des différentes technologies, une technologie performante permettra un affichage plus rapide et donc la possibilité de faire une application plus complexe sans qu'elle devienne lente. Les performances des Framework sont difficiles à calculer, car souvent il y a un écart entre les performances théoriques et pratiques. De surcroît, les performances dépendent énormément de la façon dont le développeur code. Un développeur qui maîtrise à la perfection le Framework produira un code très performant, tandis qu'un développeur novice aura tendance à créer du code peu performant. Je n'ai pas calculé les performances moi-même, elles sont extraites depuis un site internet [10] qui fournit les performances sur des cas de chargement basique ou lourd. Toutes les durées sont en millisecondes avec une intervalle de confiance de +/- 95%.

Name Duration for...	vue-v3.2.26	angular-v13.0.0	react-v17.0.2
Implementation notes			
create rows creating 1,000 rows	100.8 ± 2.5 (1.00)	113.4 ± 2.4 (1.13)	122.5 ± 3.5 (1.22)
replace all rows updating all 1,000 rows (5 warmup runs).	89.8 ± 0.6 (1.00)	101.9 ± 1.4 (1.13)	105.3 ± 1.0 (1.17)
partial update updating every 10th row for 1,000 rows (3 warmup runs). 16x CPU slowdown.	179.9 ± 2.6 (1.08)	165.9 ± 1.9 (1.00)	215.0 ± 4.4 (1.30)
select row highlighting a selected row. (no warmup runs). 16x CPU slowdown.	32.7 ± 0.9 (1.00)	55.7 ± 2.8 (1.70)	92.4 ± 2.0 (2.82)
swap rows swap 2 rows for table with 1,000 rows. (5 warmup runs). 4x CPU slowdown.	49.3 ± 0.7 (1.00)	324.5 ± 1.3 (6.58)	323.4 ± 1.1 (6.56)
remove row removing one row. (5 warmup runs).	21.5 ± 0.1 (1.07)	20.0 ± 0.3 (1.00)	22.3 ± 0.3 (1.12)
create many rows creating 10,000 rows	964.2 ± 2.6 (1.00)	1,088.0 ± 18.1 (1.13)	1,375.3 ± 23.4 (1.43)
append rows to table appending 1,000 to a table of 1,000 rows. 2x CPU slowdown.	197.1 ± 4.5 (1.00)	230.1 ± 1.5 (1.17)	242.0 ± 2.8 (1.23)
clear rows clearing a table with 1,000 rows. 8x CPU slowdown.	66.2 ± 1.3 (1.00)	139.7 ± 3.9 (2.11)	74.9 ± 0.8 (1.13)
geometric mean of all factors in the table	1.02	1.51	1.62
compare: Green means significantly faster, red significantly slower	compare	compare	compare

Figure 19 : Temps passé par les Framework pour une action précise

Name	vue-v3.2.26	angular-v13.0.0	react-v17.0.2
ready memory Memory usage after page load.	1.4 (1.00)	1.8 (1.29)	1.4 (1.05)
run memory Memory usage after adding 1000 rows.	3.2 (1.00)	4.0 (1.24)	4.2 (1.32)
update eatch 10th row for 1k rows (5 cycles) Memory usage after clicking update every 10th row 5 times	3.2 (1.00)	4.1 (1.27)	4.8 (1.51)
replace 1k rows (5 cycles) Memory usage after clicking create 1000 rows 5 times	3.2 (1.00)	4.4 (1.37)	4.8 (1.50)
creating/clearing 1k rows (5 cycles) Memory usage after creating and clearing 1000 rows 5 times	1.4 (1.00)	2.4 (1.66)	1.9 (1.32)
geometric mean of all factors in the table	1.00	1.36	1.33

Figure 20 : Mémoire alloué pour différente actions

Le site internet de comparaison met à disposition beaucoup de point de comparaison. Néanmoins, pour l'étude je n'ai pris en compte que le temps de traitement et l'allocation mémoire nécessaire. Je m'attarde que sur ces deux points car ce sont les plus importants pour avoir un ordre d'idées, le temps de traitement

influera directement sur le temps d'affichage et l'allocation mémoire influera sur le système du client. Quand on regarde la *figure 19* et la *figure 20*, on peut voir que VueJs offre les meilleures performances, suivi par ReactJs et AngularJs en dernier. VueJs est largement plus performant, tandis que ReactJs et AngularJs ont des performances assez similaires avec de légères meilleures performances pour ReactJs. Cependant il faut se rappeler que c'est une étude des performances théorique, il faut donc désormais les comparer en pratique. Pour cela je me suis basé sur un autre site internet [11] qui étudie les performances pratiques à partir d'un échantillon de 7 millions de sites internet.

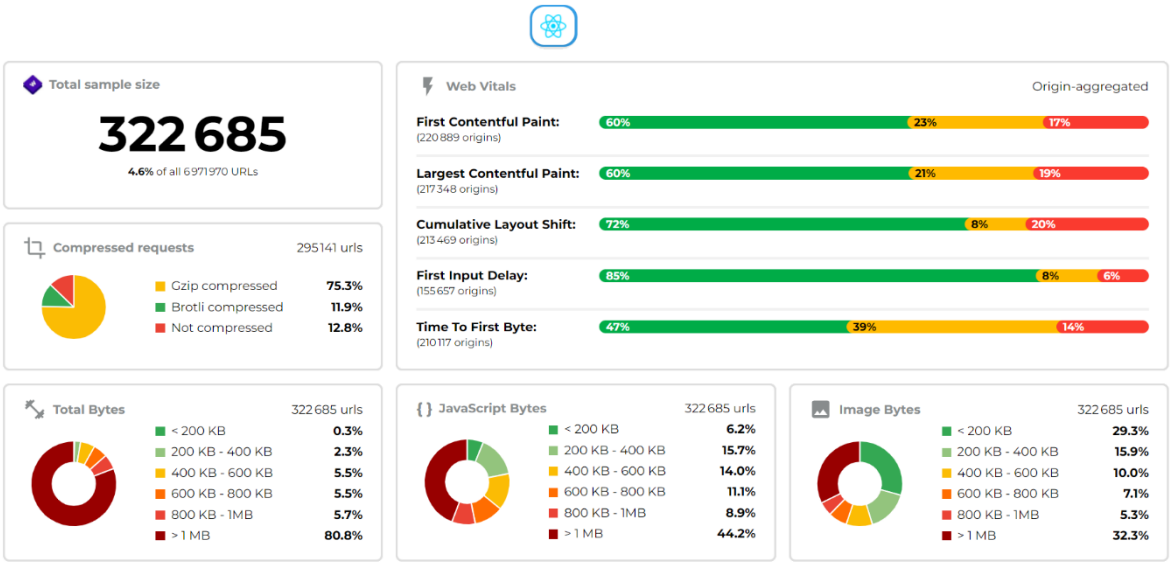

Figure 21 : Performances pratiques de ReactJs

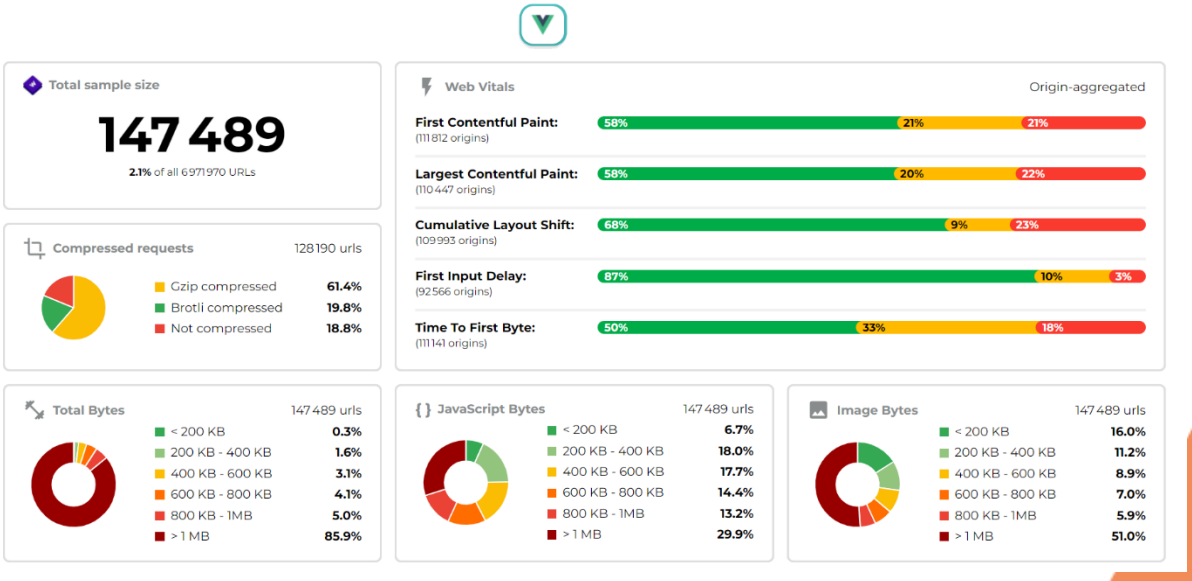

Figure 22 : Performances pratiques de VueJs

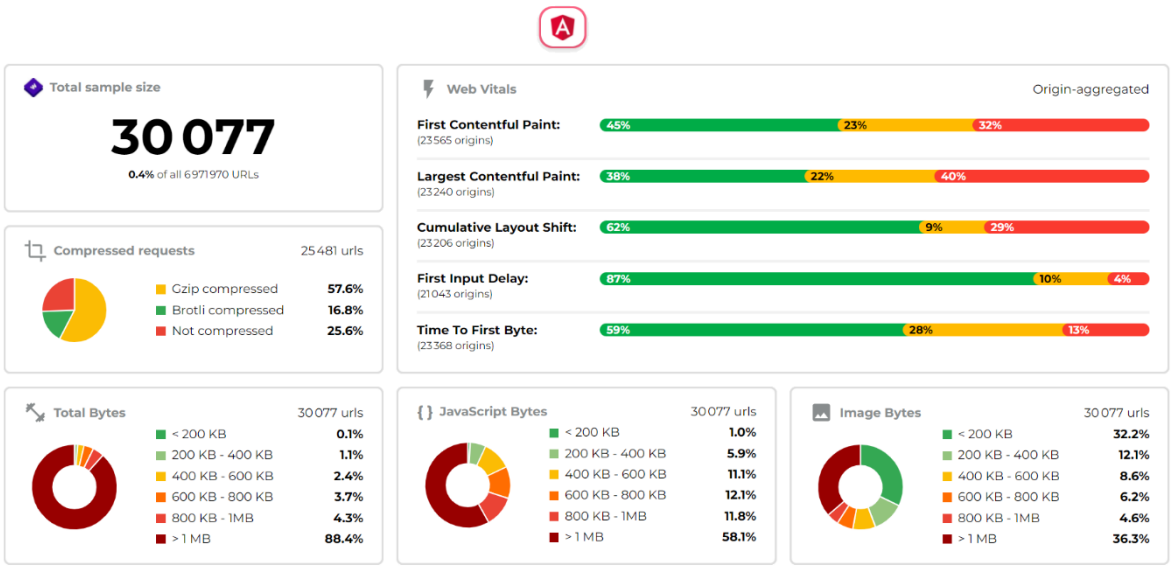

Figure 23 : Performances pratiques d'AngularJs

On remarque qu'en pratique ReactJs possède les meilleures performances, suivi par VueJs et AngularJs. Cependant, les échantillons restent petits, sur les 6971971 d'url étudier par le site internet, 322685 utilisent ReactJs, 147489 VueJs et 30077 AngularJs (ce qui reconfirme les tendances et parts de marché). Il reste difficile de départager les Frameworks, VueJs et ReactJs sont globalement au même niveau et AngularJs loin derrière.

Enfin le dernier point de comparaison est la courbe d'apprentissage et la difficulté à maitriser le langage. Selon plusieurs recherches menées j'ai pu construire le tableau récapitulatif suivant :

	AngularJs	ReactJs	VueJs
Performance	Moyenne	Haute	Haute
Scalabilité	Haute	Haute	Faible
Apprentissage	Difficile	Moyenne	Simple
Demande des développeurs	Haute	Haute	Faible
Communauté	Grosse	Très grosse	Petite
Acceptation et confiance	Haute	Très grande	Faible

ReactJS a beaucoup de points forts, une grande communauté, une grande demande de la part des développeurs, de grande performance et tout cela avec une simplicité d'apprentissage. Il s'inscrit donc comme le Framework avec le meilleur compromis.

Il reste un point que l'on n'a pas abordé dans l'étude, c'est la durabilité de la technologie. Il existe plusieurs manières d'obtenir un ordre d'idées de la durabilité d'une technologie. La plupart du temps dans d'autres domaines que l'informatique, la durabilité et l'espérance de vie d'un produit peut être estimées via l'effet Lindy. Néanmoins, l'effet Lindy s'applique très mal à des technologies informatiques car elles sont beaucoup plus volatiles. De fait, la meilleure façon de connaitre la pérennité d'une technologie est de savoir qui utilise cette technologie et en

particulier les entreprises. Voici donc un tableau non exhaustif des entreprises qui utilisent les différents Framework.

AngularJs	ReactJs	VueJs
Microsoft	Facebook	Google
Deutsche Bank	Yahoo !	Apple
Google	NewYork Times	Nintendo
Forbes	Netflix	Trivago
Paypal	Airbnb	GitHub
Samsung	DropBox	Trustpilot

Les entreprises sont un bon indicateur, car les grosses entreprises investissent beaucoup d'argent dans leur produit avec beaucoup d'équipes formées sur les technologies. Il est extrêmement coûteux pour une entreprise de migrer d'une technologie à une autre. C'est pourquoi ils ne changent de technologie qu'en cas de besoin majeur ce qui assure une pérennité de la technologie. Un exemple probant sont les systèmes bancaires qui sont programmés en COBOL un langage qui a quasiment disparu mais toujours utilisé pour des raisons historiques dans les banques.

Si on revient au tableau ci-dessus, on peut voir que Google utilise aussi bien AngularJs que VueJs, car VueJs trouve son intérêt dans de petits projets et par opposition AngularJs dans des projets volumineux. De plus, ils se ressemblent car VueJs avait été développé dans le même esprit qu'AngularJs. D'où le fait que Google utilise les deux.

Pour conclure le choix de la partie Front-end, nous avons de nos jours le choix entre trois principaux Framework web majeurs (en JavaScript), avec AngularJs qui tend doucement à disparaître. L'étude que j'ai menée montre que la première place est disputée entre ReactJs et VueJs, ReactJs est encore largement en tête mais peut être détrôné dans un avenir plus ou moins proche. Si on souhaite savoir avec plus de précision qui a le plus de chance d'être le numéro un dans le futur, il faut élargir les critères d'études. Une étude plus complète impliquera de prendre en compte le contexte géopolitique. Rappelons que qu'AngularJs fût développé par Google et que ReactJs fût développé par Meta (Facebook), des pays tels que la Chine adhère peu aux institutions Américaines. Cela est confirmé quand on regarde la tendance des recherches par pays, VueJs est en première position dans des pays comme la Chine. La Chine représentant 1.4 milliards d'habitants elle influe grandement ses pays voisins, de fait les pays situés en Asie ont plus une tendance à utiliser VueJs alors que le reste du monde utilise ReactJs et AngularJs. C'est un exemple parmi plein d'autres, on ne peut prendre en compte tous les paramètres. Une citation d'Edward Lorenz un météorologiste que j'apprécie énormément représente bien la volatilité des technologies numériques « Does the Flap of a Butterfly's Wings in Brazil Set off à Tornado in Texas ? » qui se traduit littéralement par « Les battements d'ailes d'un papillion situé au Brésil peuvent-ils déclencher une Tornade au Texas ? ».

2.8.1.5. Choix de la librairie des composants

Autre choix que j'ai dû expliciter dans ce dossier, fût le choix de la librairie de composant. Une librairie avec des composants simples et larges est nécessaire. Elle permet à une personne qui ne connaît pas les langages web basique (HTML, CSS, javascript) d'implémenter des composants graphiques poussés et responsifs. Pour Sentry, j'ai fait le choix d'utiliser Elastic UI. Ce n'est pas la librairie la plus connue, loin de là, elle cumule seulement 3.6 millions de téléchargements via npm sur la période 01/01/2008 au 30/01/2022, alors que react-bootstrap en cumule plus de 130 millions.

Néanmoins, Elastic UI est un choix très stratégique, car il tend à se démocratiser. Il possède des performances bien supérieures à celles de bootstrap qui est souvent décrié pour ses performances lentes. De plus Elastic UI est bien plus complet, il propose une pléthore de composant graphique avec beaucoup de documentation. Son site de documentation permet même de générer le JSX du composant avec les paramètres nécessaires en cochant les options que l'on souhaite. Globalement, le choix d'Elastic UI a été conditionné par sa simplicité de prise en main, de mise en place et par rapport à ce qu'il offre.

2.8.1.6. Choix du langage backend

Avec la démocratisation du web, beaucoup de langages offrent des Framework web backend :

- Javascript
- PHP
- Ruby
- Python
- Java
- Rust
- Solidity
- Go
- Kotlin
- Node JS
- Etc.

Pour les comparer j'ai besoin tout comme la partie *choix du Framework web front end* , de m'appuyer sur différentes études, étudier les tendances et de savoir lequel est le plus utilisé à travers le monde par les développeurs.

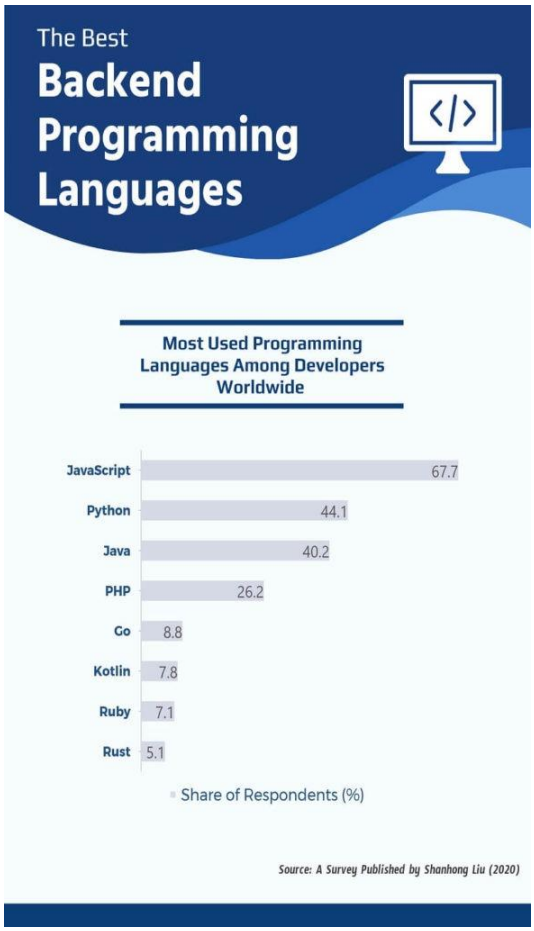

Figure 24 : The Best Backend Programming Languages

En me basant sur cette première étude, elle classe Javascript en première position, suivi par Python, Java et PHP. Ils représentent à eux 4 une proportion significative des développeurs. Bien sûr, je suis allé vérifier les résultats de cette étude par moi-même grâce à Google trends :

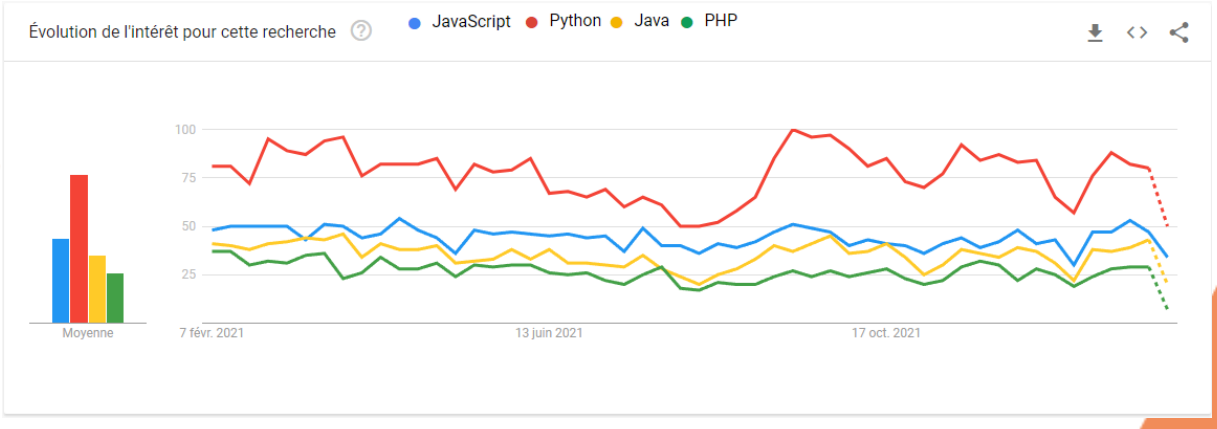

Figure 25 : Comparaisons de la part des recherches google entre Javascript, Python, Java et PHP

On remarque que les résultats sont légèrement différent comparés à l'étude, Python est en première position des recherches, Javascript en second, Java en troisième position et PHP clôture le classement. Cette différence est facilement explicable, elle vient du fait que google trends permet que de comparer des recherches basiques tels que des mots-clés. Donc ici il s'agit des langages dans leur globalité et non pas que

dans la partie web backend. Si on se concentre exclusivement sur la partie Web le classement de l'étude est plutôt bon.

À l'époque les architectures web étaient souvent les mêmes, avec un backend complètement décorrélé du frontend. Le backend générait les éléments et l'envoyait au frontend car il n'existait pas des langages frontend permettant de faire de l'interactif. Une structure dite basique était d'utiliser du HTML, CSS et javascript en frontend et du PHP ou Java en backend. PHP et Java offraient de nombreuses possibilités que d'autres langages ne pouvaient offrir. Mais avec le temps, les langages ont énormément évolué. C'est le cas notamment de Python et Javascript.

La communauté Javascript a grandi à la suite de l'émergence des Framework frontend permettant de faire du web interactif avec un système de composant simple. De là est né un Framework javascript backend qui a largement démocratisé JavaScript qui est NodeJs [12].

Javascript s'est vite retrouvé leader du marché en backend car couplé avec un Framework Javascript front-end, il permettait d'avoir un site internet gérer en un seul langage. C'est pourquoi j'ai décidé d'utiliser NodeJs pour la partie backend de Sentry et ainsi d'avoir mon application entièrement en JavaScript. Python aurait pu être aussi un choix intelligent, car Python est un langage avec une syntaxe très simple et très utilisée dans les domaines scientifiques et scolaire pour sa simplicité d'apprentissage. Néanmoins, il est moins bien optimisé pour du web que Javascript car il est prévu à la base pour des calculs conséquents ou de l'intelligence artificielle. D'autre part, chez Stellantis beaucoup d'applications étaient faites en Java ou PHP (nous avons nos propres dérivés de Framework en interne), l'alternant étant une source d'innovation il était donc normal de ne pas se tourner vers des langages déjà bien connus en interne. Le choix de la technologie backend pour Sentry s'est donc tourné vers NodeJs avec ExpressJs [13].

2.8.1.7. Choix du moteur de base de données

Pour choisir le moteur de base données j'ai dû au préalable estimer la taille des données qui seront stockées. Sentry n'a pas besoin de stocker énormément de données car la plupart d'entre elles sont récupéré depuis des applications référentielles en interne.

On estime stocker environ 100 informations par application à sauvegarder. Chaque information est stockée sur une moyenne de 30 caractères chacun encodé sur 2 octets. Selon le CMBD de Stellantis 17057 sont référencés on obtient donc une capacité de stockage utilisé de $17057 * 100 * 30 * 2 = 10234200\ octets = 10{,}2342\ Mo$. Étant donné le peu de stockage nécessaire, mon choix de moteur de base de données s'est tourné vers MySQL [21]. MySQL est le moteur de base de données relationnelles le plus répandu, il est très simple d'utilisation et aisément accessible à l'apprentissage. De plus, la mise en place d'un serveur MySQL est simple et peu chère.

2.8.1.8. Azure DevOps et CI/CD Stellantis

Dans un esprit d'innovation, j'ai choisi d'utiliser Azure DevOps [22] pour le CI/CD. Je ne me détache pas du CI/CD traditionnel de chez Stellantis (Github [24], Artifactory , TeamCity [25], JIRA [26], etc.), je le lie à Azure DevOps. De cette manière je conserve le CI/CD traditionnel tout en profitant d'un outil qui me permets de centraliser tous les autres outils. De surcroît, cela me permet de rendre compte auprès de TIES les avantages et inconvénients des outils DevOps Built-In des cloud provider.

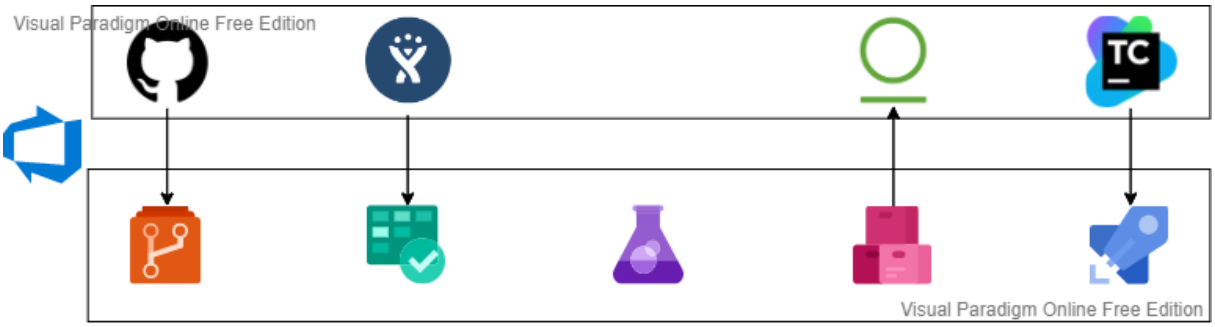

Figure 26 : CI/CD Azure DevOps et lien avec le CI/CD Stellantis

Il existe deux types d'Azure DevOps, Azure DevOps Services et Azure DevOps Server. Azure DevOps Server permet de bénéficier des services d'Azure DevOps depuis les datacenters. Alors qu'Azure DevOps Services est une plateforme en ligne offrant plusieurs services de CI/CD.

Parmi ces services on retrouve Azure Repositories, Azure Boards, Azure Test Plan, Azure Artifact et Azure Pipeline. Chacun d'entre eux a une utilité particulière. Azure repositories est la partie de gestion du code source, il permet notamment de lier les repositories GitHub au projet. Azure Boards permet de faire de la gestion de projets avec des tableaux de bord, de planifier les sprints, de gérer les back logs etc. Azure Test Plans propose de créer des pipelines de tests pour assurer des tests automatisés. Azure Artefact sert de stockage et de gestion des artéfacts. Et pour finir Azure Pipeline offre la possibilité de créer des pipelines de Build d'artéfact, de déploiement de code et d'architecture.

De surcroît, Azure DevOps propose de lier de nombreux outils extérieurs dont ceux utiliser dans le CI/CD traditionnel Stellantis. Si un outil ne peut pas être lié, il existe le Visual Studio Marketplace for Azure DevOps [23] permettant d'installer des extensions ajoutant de nouvelles fonctionnalités. C'est notamment le cas d'Artifactory, TeamCity et JIRA qui ne sont pas intégrés par défaut. Il faut néanmoins faire attention, certains outils nécessitent des extensions payantes. C'est le cas de JIRA que je n'ai donc pas pu lier, car il nécessite une extension payante dans la marketplace du côté d'Azure DevOps et une autre extension du côté d'Atlassian la société distributrice de JIRA.

Le prix d'Azure DevOps Services dépend du nombre d'utilisateurs de celui-ci, en dessous de 5 utilisateurs inclus, il est gratuit. Au-delà c'est un surcoût proportionnel au nombre d'utilisateurs et à la formule choisie (qui donne accès à certaines catégories énumérées au-dessus). Il en est de même pour Azure DevOps Server qui a

le même système de tarification à l'exception qu'il est gratuit dans le cas où la société possède une licence Visual Studio (car il est fourni avec).

2.8.2. Dossier technique de l'Infrastructure

Ce dossier regroupe l'infrastructure de Sentry ainsi que certains points de sécurité. Il est primordial et doit être validé lors de la première réunion de validation technique. L'infrastructure à beaucoup évoluer depuis le début du projet, je ne peux toutes les décrire pour la raison que cela rendrait trop lourd et indigeste cette partie. Je me contenterai d'expliquer la dernière infrastructure ainsi que l'authentification via les ID Stellantis.

2.8.2.1. Infrastructure Mono-région

Comme je l'ai dit précédemment, j'ai du designer plusieurs infrastructures et repasser plusieurs fois la première réunion de validation technique pour l'application car elle ne répondait pas aux standards de sécurité. Ici seront décrites essentiellement les dernières infrastructures.

Chez Stellantis la règle est que chaque projet sur Azure doit avoir sa propre souscription. Une souscription est un accord légal, un abonnement est une offre Azure. Ces offres présentent des plans tarifaires, avec des avantages et des conditions. C'est également un accord de paiement, chaque mois est débité les coûts engendrés par les ressources azure exploiter sur la souscription. La souscription permet de créer une limite d'échelle, c'est-à-dire le nombre de ressources que l'on peut créer sur cet abonnement. Et enfin c'est une limite administrative pour l'administration, la sécurité et la stratégie.

Chaque projet azure doit avoir sa propre ou ses propres souscriptions. Un projet conséquent aura plusieurs souscriptions pour chaque phase du projet, une pour le développement, une pour la préproduction et une pour la production. Pour des projets nécessitant peu de ressources comme Sentry, une seule souscription est suffisante. Les phases du projet sont délimitées via des groupes de ressources qui sont également des limites administratives. Il faut voir un groupe de ressource comme un dossier pour des services Azure, chaque service doit être associé à un groupe de ressource lors de sa création. On retrouve donc la souscription de Sentry avec 3 groupes de ressources, un pour le développement, un pour la préproduction et un pour la production.

Stellantis possède plusieurs cloud provider et est plus ou moins établi dans certain. Nous avons des projets chez AWS et chez Azure. Chez AWS un déploiement automatique de la landing zone a été mis en place. Une landing zone sont les ressources minimums qu'un projet doit configurer pour respecter les standards de sécurité. Cette landing zone fût automatisé chez AWS via un outil créer en interne permettant de générer un Template de l'infrastructure selon le besoin. Chez Azure c'est à la charge de l'équipe projet d'implémenter cette landing zone. Pour cela j'ai

dû m'appuyer sur les différents standards définis chez AWS et mettre en place un équivalent du côté Azure. À partir des critères de sécurité j'obtiens donc l'infrastructure présentée en *figure 27*.

Dans les groupes de ressources on retrouve un VNet pour Virtual Network qui est un réseau virtuel. Ce réseau virtuel permet de communiquer de manière sécurisée avec internet et d'autres réseau virtuel (au besoin).

Comme un réseau classique on peut créer des subnet ou sous-réseaux. Ces sous-réseaux peuvent être soit publics soit privé. Un subnet public permet la communication vers internet alors qu'un subnet privé ne peut communiquer qu'avec les autres subnet. Un subnet public suivi d'un subnet privé oblige un cheminement et ajoute ainsi de la sécurité. On retouve donc 3 subnet, deux public et un privé.

Le premier subnet public contient un Azure Firewall. Cet Azure Firewall agi sur les couches L3 à L7 (couches du modèle OSI). Il ajoute une inspection TLS (Transport Layer Security) qui est un chiffrement pour la confidentialité, l'intégrité et l'authenticité en utilisant des certificats. L'Azure Firewall ajoute également un système IDPS (Intrusion Detection and Prevention System), qui filtre les activités, détecte celles malveillantes, les bloques et les signales. L'Azure Firewall ajoute aussi la possibilité de faire du filtrage URL et de créer des catégories web pour refuser ou autoriser l'accès à certains site web. L'Azure Firewall est seul dans son subnet et est la première protection mise en place.

Il communique ensuite avec le second subnet public qui est composé d'un Application gateway ou passerelle d'application. L'application gateway est composé de deux services Azure, d'un Web Application Firewall ainsi qu'un L7 Load Balancer. Le Web Application Firewall permet de contrer les attaques web classiques tels que l'injection SQL et le Scripting inter-sites. Le L7 Load Balancer est un équilibreur de charge qui s'applique sur la plus haute couche réseaux, la couche application. L'équilibreur de charge permet d'agir comme un reverse proxy.

Enfin le dernier subnet est privé, celui-ci contient une machine virtuelle avec un serveur web, une seconde machine virtuelle avec un serveur de base de données et un azure containers instances pour la gestion des containers de l'architecture micro-services. Le contenu du subnet privé est la seule partie que l'on peut changer à notre guise, les deux subnet public sont les standards de sécurité et ne peuvent être modifiés.

Une autre architecture est possible représentée en *figure 28*. La différence se trouve dans le fait que j'utilise des services PaaS plutôt que IaaS. La seconde option permet de faire abstraction de l'OS en utilisant des sites managés Azure. J'utilise notamment un web app services avec un Plan app service. Le plan app service est une ressource de calcul, c'est le CPU pour la ou les applications liées. Une web application services ne coûte rien car les ressources de calculs viennent du plan app service qui est facturé. Le plan app service permet ainsi d'avoir une source de calcul que l'on peut lier à plusieurs applications, il peut ainsi jouer le rôle d'un balancer des ressources de calculs. De surcroît, le moteur de base de données est aussi un service managé qui permet également de faire abstraction de l'OS.

Les deux choix se valent, la première permet d'avoir une meilleure mainmise mais oblige plus de gestion. Néanmoins je suis partisan des services entièrement managés, car cela entre en adéquation avec la stratégie Stellantis qui consiste à privilégier dans l'ordre SaaS>PaaS>IaaS. Bien sûr dans la mesure ou les services managés ne coûtent pas beaucoup plus chers. C'est pourquoi j'ai dû estimer les coûts liés à l'infrastructure grâce à la calculatrice Azure [27] et estimer le SLA. Le SLA est le contrat de niveau de service (« Service-level agreements ») il permet de connaitre le temps de disponibilités de l'application. Par exemple un service peut avoir un SLA de 99.99%, ce qui signifie que sur une durée d'un an le système non exploitable :

$$365 * 24 * 60 * 60 - 365 * 24 * 60 * 60 * 0.9999 = 3153.6\ secondes$$
$$= 52\ minutes\ et\ 33\ secondes$$

Il faut voir le SLA comme un rendement du service. Plus il y a de service connecté ensemble plus le SLA sera faible. Microsoft fournit une documentation de calcul du SLA [28]. Azure garantie des SLA pour la majorité des services. Seuls quelques-uns n'ont pas de SLA, notamment les connexions avec des datacenters extérieurs car Microsoft n'a aucun pouvoir pour la connexion côté client il ne peut donc pas garantir un niveau de service.

2.8.2.2. Infrastructure Multi-région

Chez Azure comme chez tous les cloud provider, vous devez lors de la création d'une ressource, précisez la région sur laquelle elle sera stockée. Chaque région Azure est composée de plusieurs datacenters physiquement proche connecté avec une très faible latence. Chaque région Azure contient des zones de disponibilités. Ces zones de disponibilités sont des datacenters physiquement éloignés au sein d'une même région dont les données sont répliquées. Les zones de disponibilités ont une latence inférieure à 2ms ce qui permet une synchronisation quasiment instantanée des ressources. De ce fait, en cas de problème avec un datacenter, son homologue prendra le relai. Stellantis fonctionne un peu de la même manière sur ses applications on-premise avec le site de Bessoncourt répliquer à Poissy.

On arrive donc à un problème, Sentry est un projet nécessitant peu de ressources mais à énormément d'ambitions. Dans un premier temps les utilisateurs de Sentry seront majoritairement situés sur le site de Bessoncourt. Mais nous souhaitons que Sentry devient un référentiel du CD Stellantis. Ce qui impliquera que les utilisateurs seront du monde entier, une seule région ne sera plus suffisante car il y aura trop de latence pour les utilisateurs éloignés physiquement.

L'infrastructure de Sentry (représentée en *figure 29*) pour le multi-région diffère donc légèrement du mono-région. Les groupes de ressources pour la phase de développement et de préproduction ne sont pas représentés sur le schéma car il ne change pas.

Cette fois-ci on retrouve un nombre n de groupes de ressources chacun associé à une région. L'utilisateur est dirigé vers la région Azure la plus proche grâce à Azure Front Door. Les groupes de ressources sont structurés de la même manière que pour le mono-région à l'exception de la partie base de données.

Chaque région est un réplica d'infrastructure, on pourrait faire de même avec la base de données mais cela posera quelques problèmes. Imaginons une application X dont l'équipe en charge de cette application se trouve sur plusieurs régions différentes. Si pour l'infrastructure je me contente de faire un réplica de base de données, cela signifie que la partie de l'équipe dans la région A ne pourra pas voir les modifications faites par la partie de l'équipe sur l'application dans la région B. La solution sera d'interconnecter les bases de données. On pourrait le faire nous-mêmes ce qui est la bonne pratique quand on possède deux régions ou trois régions grand maximum. Au-delà Azure propose un service PaaS Azure Cosmo DB, qui permet d'avoir une ou plusieurs bases de données managées accessible depuis plusieurs régions. C'est un excellent service mais avec un coût élevé. C'est pourquoi, pour des applications nécessitant une petite base de données et peu de régions à interconnecter, il est préférable de faire cette interconnexion manuellement avec des private endpoint et de dupliquer la base de données par région.

Néanmoins, cette solution trouve sa limite quand on a beaucoup de régions avec une base de données très lourde (car l'interconnexion est souvent facturée aux données transmises). Bien sûr dans une optique FinOps, Sentry adoptera d'abord la solution de duplication des bases de données (non représenté ici) puis la solution cosmo DB quand celle-ci sera nécessaire. La solution cible est donc l'infrastructure ci-dessous. Il est à préciser que la première réunion de validation technique doit être réorganisé après chaque modification technique.

2.8.2.3. Authentification par ID Stellantis

L'authentification chez Stellantis utilise le principe d'authentification par token avec PKCE. Cette méthode permet d'assurer une très haute sécurité. Pour mettre en place ce système on utilise le protocole OAuth et la librairie Auth0. La librairie Auth0 est utilisée exclusivement pour les tests en développement car elle est gérée par une entité en interne qui à la suite d'une demande, une réunion et un dossier, ouvrira le droit d'authentification sur une ou plusieurs url provisionner lors de la demande.

Le fonctionnement est plutôt simple à comprendre. Tout d'abord le client clique sur login ce qui va appeler l'application. L'application génère deux codes, un code de vérification et un code de défi. L'application va ensuite envoyer le code de la requête ainsi que le code de défi à la librairie Auth0. La librairie va rediriger l'utilisateur sur la même url avec un /callback. Cette redirection permet à la librairie d'être sûr que c'est bien l'utilisateur qui veut se connecter et que le code n'a pas été intercepté. Ensuite l'utilisateur va recontacter la librairie mais cette fois-ci avec l'url /callback. La librairie va transmettre un code d'autorisation à l'application qu'elle lui retournera suivie du code de vérification. La librairie va ensuite valider les codes de vérification et de défi et renvoyer le token avec l'access token à l'application. Enfin, l'application peut contacter l'API avec le token qui lui donnera soit une réponse positive ou négative.

Le système peut paraître complexe et bizarre mais il est en soi logique. Il limite énormément la communication entre le client, l'application, la librairie et l'api ce qui complexifie grandement l'interception des codes. De plus la redirection en /callback

permet de vérifier que c'est bien une demande de l'utilisateur. Tout se passe en moins d'une seconde, donc pour un attaquant il est quasiment impossible d'intercepter, imiter les requêtes et rediriger son navigateur en une fraction de seconde.

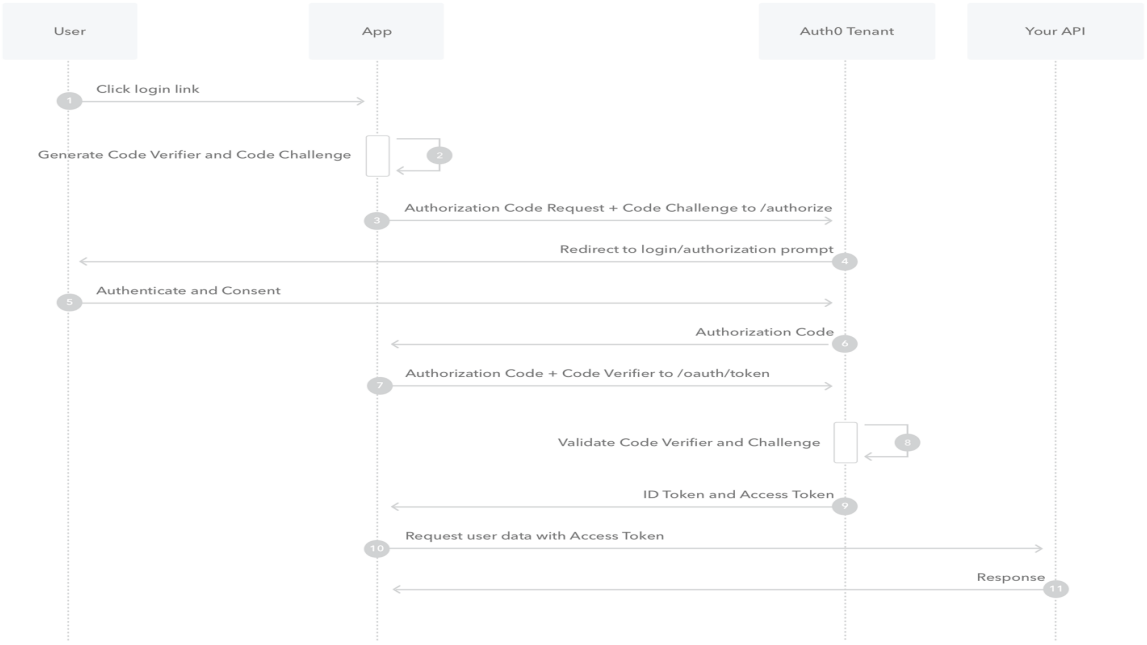

Figure 27 : Authentification via token PKCE

Conclusion

La récente création de Stellantis issue de la fusion PSA et FCA a grandement changé la structure interne de l'entreprise. L'entreprise a dû changer complètement sa stratégie informatique et s'adapter à sa nouvelle taille. Ce changement de stratégie à impliqué une mise en commun des standards informatiques et cela passe par le tout-cloud. Dans l'esprit d'uniformisation des standards et en adéquation avec le rôle de TIES, une nouvelle application référentielle en matière de continuous delivery est une nécessité.

De ce besoin est né Sentry, une application ayant pour but d'être dans un premier temps un superviseur pour ensuite élargir ses capacités à l'automatisation du continuous delivery. Dans l'esprit du tout cloud Stellantis Sentry se voit donc héberger sur Azure, permettant à l'équipe TIES d'avoir un retour d'expérience chez différents cloud provider. Le fait que Sentry soit une nouvelle application a induit le besoin de se plier aux nombreuses étapes projets de chez Stellantis. Réunions de validations technique, Prepare Study, dossier technique d'infrastructure, dossier de conception technique, Sentry a pris du temps à émerger. Le rapport n'est pas un retraçage exhaustif de l'entièreté des documents et des choix qui ont été pris, car ils ont été nombreux et quelquefois infructueuses.

Après 6 mois de travail, Sentry va enfin commencer à voir le jour. Les estimations, les dossiers, les réunions, les idées émergent et prennent vie. Cette année a aussi été rythmé par l'apprentissage du monde cloud. Un monde encore jeune avec de belles années devant lui. L'apprentissage est une part conséquente du travail, AZ-900, AZ-305, AZ-204, les certifications liées au monde azure étaient un défi. Sentry devra voir le jour au courant septembre 2022 avec un premier livrable. C'est une application qui a beaucoup d'avenir avec de grandes ambitions et c'est un plaisir de travailler dessus.

De surcroît, Stellantis et ses employés ont été d'un accueil incommensurable et d'une aide précieuse . Je les remercie encore une fois, car sans eux rien de tout cela n'aurait été possible.

Bibliographie

[1] À propos de nous - Stellantis [en ligne]. [Consulté le 20/12/2021] Disponible à l'adresse : https://www.stellantis.com/fr/groupe/a-propos-de-nous

[2] Pays-Bas : Les formes légales d'entreprises – BNP Paribas [en ligne]. [Consulté le 20/12/2021] Disponible à l'adresse : https://www.tradesolutions.bnpparibas.com/fr/implanter/pays-bas/creer-une-entreprise

[3] Maserati France – Maserati [en ligne]. [Consulté le 20/12/2021] Disponible à l'adresse : https://www.maserati.com/fr/fr

[4] L'organigramme de Stellantis organisé autour de neuf comités – Christophe CARIGNANO [en ligne]. [Consulté le 20/12/2021] Disponible à l'adresse : https://www.auto-infos.fr/article/l-organigramme-de-stellantis-organise-autour-de-neuf-comites.92304

[5] Software Day – Stellantis [en ligne]. [Consulté le 21/12/2021] Disponible à l'adresse : https://www.stellantis.com/fr/finance/evenements/sw-day-2021

[6] DevOps – Plusieurs auteurs [en ligne]. [Consulté le 12/01/2022] Disponible à l'adresse :

https://fr.wikipedia.org/wiki/Devops

[7] Continuous Delivery – Jez Humble [en ligne]. [Consulté le 12/01/2022] Disponible à l'adresse :

https://continuousdelivery.com/

[8] Cycle en V en gestion de projet : définition et méthode – Rémi Lardilleux [en ligne]. [Consulté le 05/05/2022] Disponible à l'adresse : https://www.manager-go.com/gestion-de-projet/cycle-en-v.htm

[9] Manifesto for Agile Software Developement – Kent Beck, James Grenning, Robet C.Martin, Mike Beedle, Jim Highsmith, Steve Mellor, Arie van Bennekum, Andrew Hunt, Ken Schwaber, Alistair Cockburn, Ron Jeffries, Jeff Sutherland, Ward Cunningham, Jon Kern, Dave Thomas, Martin Fowler et Brian Marick [en ligne]. [Consulté le 07/05/2022] Disponible à l'adresse : http://agilemanifesto.org/

[10] js-framework-benchmark – 198 collaborateurs GitHub [en ligne]. [Consulté le 17/05/2022] Disponible à l'adresse : https://krausest.github.io/js-framework-benchmark/current.html

[11] Perf Track – 7 contributeurs GitHub [en ligne]. [Consulté le 18/05/2022] Disponible à l'adresse : https://perf-track.web.app/

[12] NodeJs - NodeJs Foundation [en ligne]. [Consulté le 18/05/2022] Disponible à l'adresse : https://nodejs.org/en/

[13] ExpressJs – Fondation OpenJs [en ligne]. [Consulté le 19/05/2022] Disponible à l'adresse : https://expressjs.com/fr/

[14] Introduction à JSX - Facebook Open source [en ligne]. [Consulté le 19/05/2022] Disponible à l'adresse : https://fr.reactjs.org/docs/introducing-jsx.html

[15] HTML Living Standard – Tim Berners-Lee et de nombreux contributeurs cités dans la partie « Acknowledgmets » [en ligne]. [Consulté le 19/05/2022] Disponible à l'adresse : https://html.spec.whatwg.org/multipage/

[16] Javascript – Mozilla corporation [en ligne]. [Consulté le 20/05/2022] Disponible à l'adresse : https://developer.mozilla.org/fr/docs/Web/JavaScript

[17] Référence du DOM – Mozilla corporation [en ligne]. [Consulté le 20/05/2022] Disponible à l'adresse : https://developer.mozilla.org/fr/docs/Web/API/Document_Object_Model

[18] React Js – Facebook Open source [en ligne]. [Consulté le 22/05/2022] Disponible à l'adresse : https://fr.reactjs.org/

[19] Angular - 1570 contributeurs [en ligne]. [Consulté le 22/05/2022] Disponible à l'adresse : https://angular.io/

[20] Vue Js – 58 contributeurs [en ligne]. [Consulté le 22/05/2022] Disponible à l'adresse : https://vuejs.org/

[21] MySQL – Oracle [en ligne]. [Consulté le 23/05/2022] Disponible à l'adresse : https://www.mysql.com/fr/

[22] Azure DevOps – Microsoft [en ligne]. [Consulté le 23/05/2022] Disponible à l'adresse : https://azure.microsoft.com/fr-fr/services/devops/

[23] Extensions for Azure DevOps – Microsoft [en ligne]. [Consulté le 23/05/2022] Disponible à l'adresse : https://marketplace.visualstudio.com/azuredevops

[24] GitHub – Microsoft [en ligne]. [Consulté le 24/05/2022] Disponible à l'adresse : https://github.com/

[25] TeamCity – JetBrains [en ligne]. [Consulté le 24/05/2022] Disponible à l'adresse : https://www.jetbrains.com/fr-fr/teamcity/

[26] JIRA – Atlassian [en ligne]. [Consulté le 26/05/2022] Disponible à l'adresse : https://www.atlassian.com/fr/software/jira?bundle=jira-software&edition=free

[27] Calculatrice de prix – Microsoft [en ligne]. [Consulté le 26/05/2022] Disponible à l'adresse : https://azure.microsoft.com/fr-fr/pricing/calculator/?&ef_id=CjwKCAjw46CVBhB1EiwAgy6M4gcdkkthSi8s1XRIdkwUUrTz3Nw93eJP7R128gt5uPK9LX0jvdmmXBoCywIQAvD_BwE:G:s&OCID=AID2200187_SEM_CjwKCAjw46CVBhB1EiwAgy6M4gcdkkthSi8s1XRIdkwUUrTz3Nw93eJP7R128gt5uPK9LX0jvdmmXBoCywIQAvD_BwE:G:s&gclid=CjwKCAjw46CVBhB1EiwAgy6M4gcdkkthSi8s1XRIdkwUUrTz3Nw93eJP7R128gt5uPK9LX0jvdmmXBoCywIQAvD_BwE

[28] Utilisation des métriques métier pour concevoir des applications Azure résilientes – Microsoft [en ligne]. [Consulté le 30/05/2022] Disponible à l'adresse : https://docs.microsoft.com/fr-fr/azure/architecture/framework/resiliency/business-metrics

Annexes

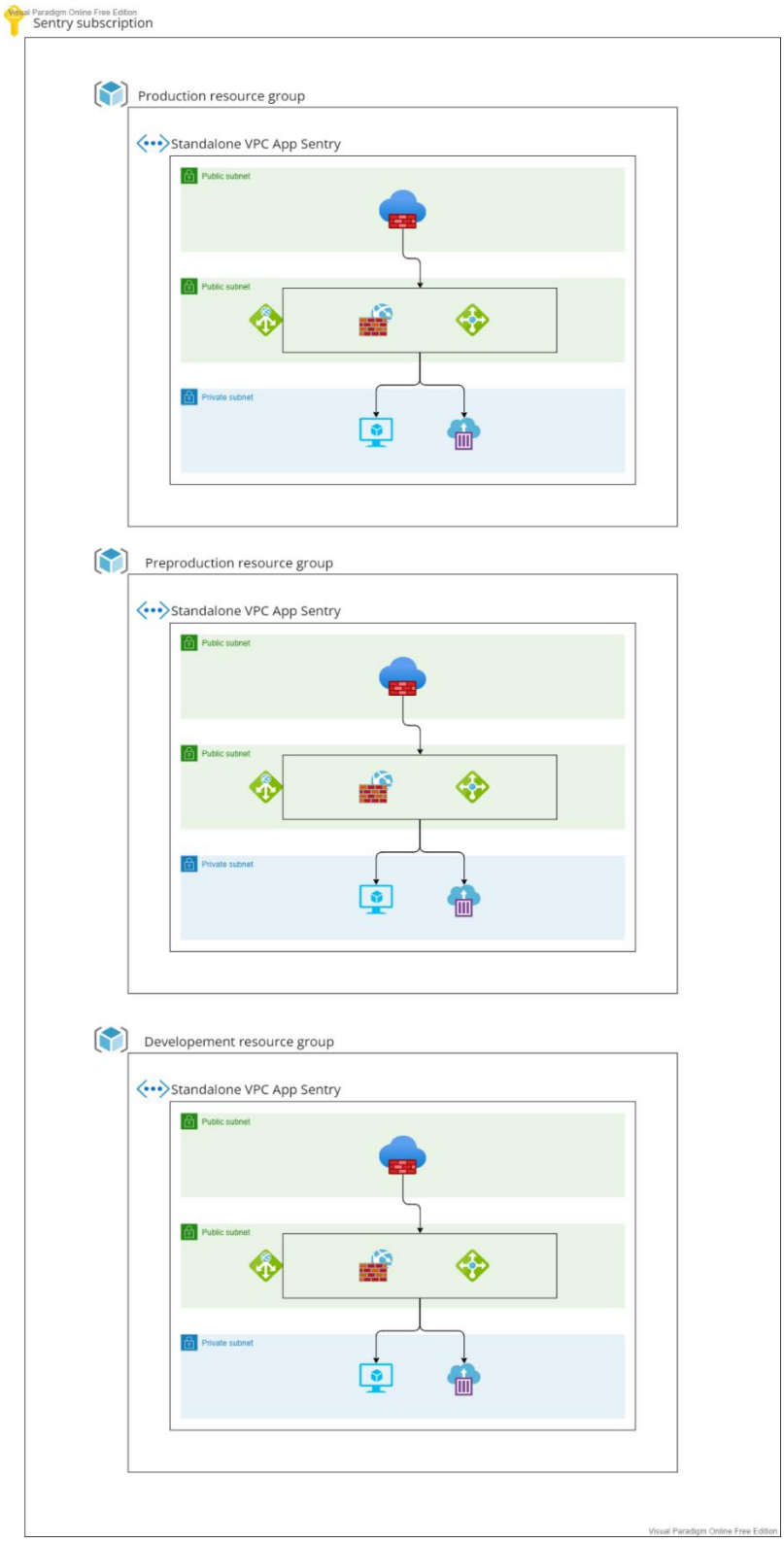

Figure 28 : Première option d'infrastructure

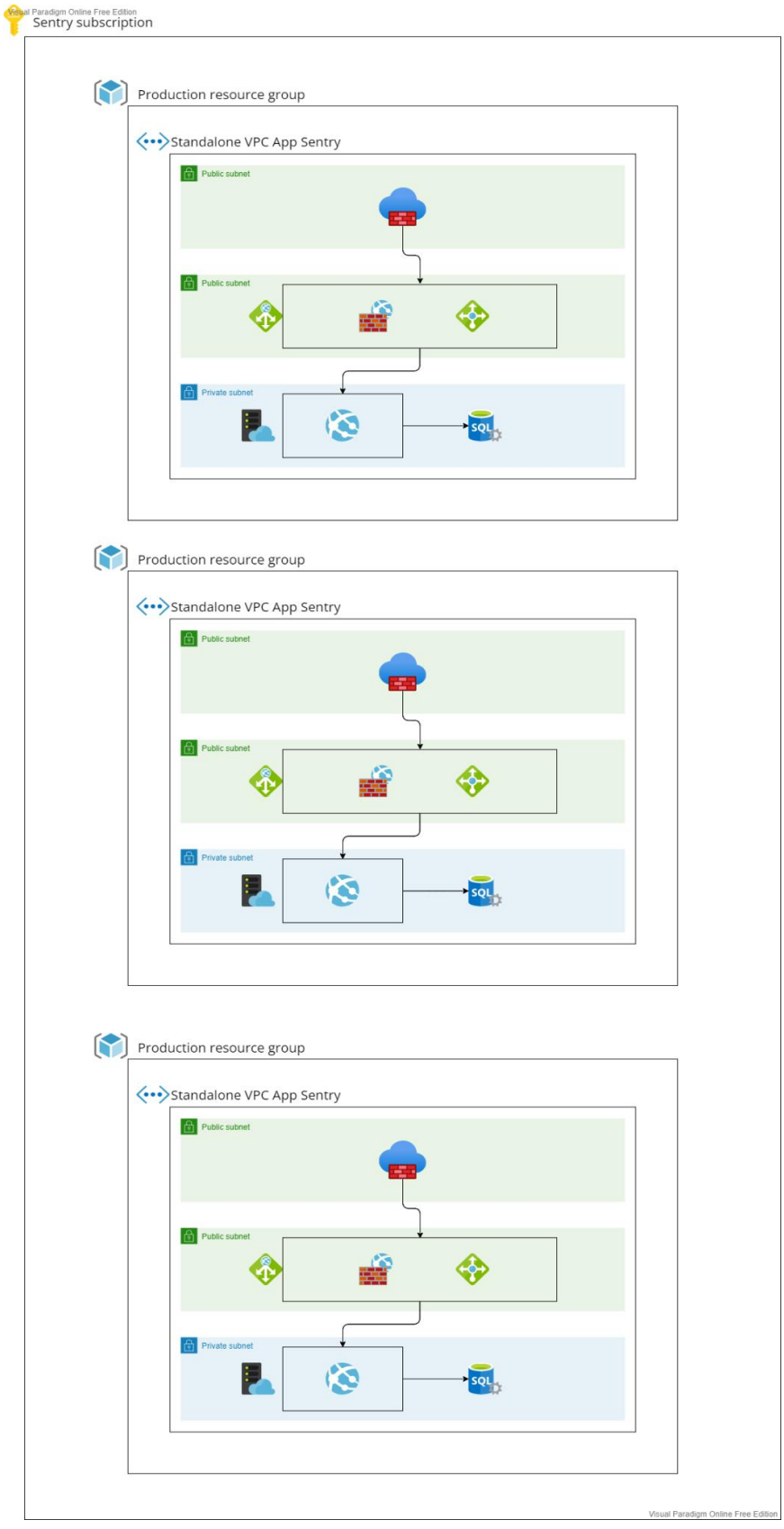

Figure 29 : Seconde option d'infrastructure

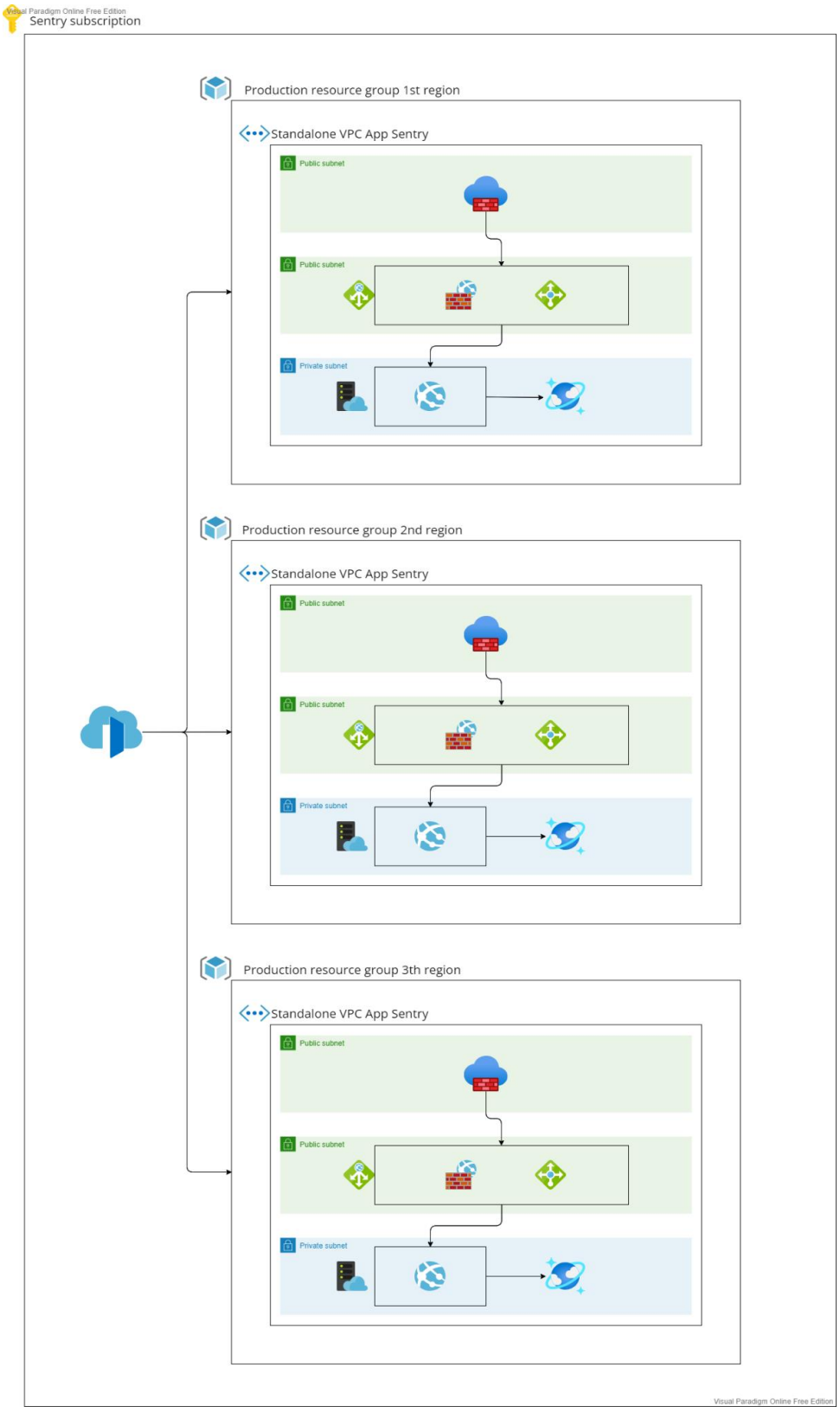

Figure 30 : Infrastructure multi-région

Mots clés

Azure – Stellantis – Web application

Résumé

Stellantis issue de la récente fusion de PSA et FCA à dû revoir son fonctionnement interne dont sa stratégie informatique. Cette nouvelle stratégie informatique du tout cloud et le besoin de standardisation pousse aujourd'hui Stellantis à revoir son CI/CD. L'équipe TIES en charge de la partie standardisation du delivery souhaite un nouvel outil permettant de répondre au besoin d'automatisation du continuous delivery. De là est né Sentry, une application web qui joue d'abord le rôle de superviseur et qui tend à devenir un futur outil du standard du continuous delivery. Son développement prend du temps, ce mémoire retrace ma première année de travail sur Sentry. Ce mémoire est orienté d'abord sur une partie générale de présentation d'entreprise puis sur la longue étude de lancement de projet qui a duré 6 mois en tout. J'espère que vous trouverez satisfaction à le lire comme j'ai trouvé satisfaction à le rédiger et à travailler sur Sentry.