Année 2023-2024

Dossier professionnel

Gestion de projet informatique

DELORME Luc-Louis
CA-GIP

Résumé

L'automatisation est devenue un enjeu majeur dans le monde actuel, touchant presque tous les secteurs de la société. En effet, l'automatisation des tâches et des processus à de très nombreux avantages à faire valoir. L'un d'entre eux est l'augmentation significative de la productivité. En automatisant des tâches répétitives et chronophages, les entreprises d'aujourd'hui peuvent accomplir le même travail beaucoup plus rapidement, avec moins d'erreurs. Cela libère du temps pour les employés, leur permettant de se concentrer sur des activités à plus fortes valeurs ajoutées. Les processus automatisés permettent également d'optimiser l'utilisation des ressources, augmenter le temps d'opération (travail soutenu des machines), d'améliorer la conformité, la précision, les coûts, la rentabilité et la compétitivité.

C'est dans le cadre de cette tendance que le projet du référentiel des connectivités a vu le jour. Celui-ci est centré sur des solutions visant à répondre aux besoins d'automatisation de l'équipe OER. La transformation numérique engendre le besoin de tenir à jour un référentiel dynamique de connectivités WAN. L'objectif du projet est donc de proposer une nouvelle solution plus performante, automatisée et évolutive pour venir remplacer des procédés laborieux exécutés manuellement. Employant une méthodologie rigoureuse, le projet est basé sur l'utilisation de technologies de pointe comme Kubernetes, Redis, Django Rest Framework et des APIs.

Les activités présentées dans ce document en lien avec la gestion d'un projet informatique n'ont pas pu être toutes pratiquées concrètement dans le cadre de mon alternance. Cependant, mon dossier reflète la façon dont j'aurai procédé devant une telle tâche en ma qualité de consultant *DevOps*.

Mots-clés : Réseaux étendus, Connectivités WAN, Automatisation, Transformation numérique, Intégration et déploiement continu, Méthodologie Agile

Abstract

Automation is becoming a main issue in the modern world, affecting almost every sector of society. Indeed, automation offers many advantages that are well worth the detour. One of them is the significant increase of the productivity. By automating repetitive and time-consuming tasks, modern companies are available to do the same work most efficiently, with less human mistakes. Employees have more time to concentrate on higher-value activities. Automated processes optimize resources use, increase uptime, improve compliance, profitability, competitiveness and reduce costs.

It is in this context that the referential connectivity was designed. It focuses on solutions to meet the OER team's automation needs. The digital transformation brings need for a dynamic WAN connectivity referential. The main goal of this project is to propose a new high-performance, automated, and adaptable solution to replace a laborious manual process. By adhering to a rigorous methodology, this project is based on advanced technologies like Kubernetes, Redis, Django Rest Framework and APIs.

Not all the activities presented in this document were carried out during my work-study placement. However, it reflects how I would handle such this task as a *DevOps* consultant.

Keywords: Wide Area Networks, WAN connectivity, Automation, Digital transformation, Continuous Integration/Continuous Delivery, Agile method

Table des matières

1 – Introduction

1.1 - Présentation de l'entreprise

L'essor des technologies numériques a considérablement bouleversé le secteur bancaire. De De nouveaux défis et enjeux apparaissent, tels que la numérisation des transactions, l'amélioration de l'expérience client, la concurrence accrue ou la sécurité des données.
Fondé le 1er janvier 2019, CA-GIP (Crédit Agricole Group Infrastructure Platform), qui est une filiale du Crédit Agricole, regroupe plus de 80% de la production informatique, des infrastructures et des plateformes technologiques de la banque. L'objectif du groupe est de répondre aux enjeux de la transformation digitale.
Pour cela, il s'agit de développer de nouvelles plateformes adaptées aux nouvelles pratiques du digital, tout en garantissant un haut niveau de sécurité et de confidentialité pour les Entités du Groupe. La création de CA-GIP permet de disposer de nouveaux moyens pour faciliter la construction de solutions convergentes et innovantes au niveau du Groupe tout en gagnant en agilité et en efficacité.
CA-GIP est une force de travail de plus de 4000 personnes dont 1900 collaborateurs répartis sur 16 sites en France. Plus 7 millions de visiteurs sont reçus chaque jour sur les applications hébergées. Le schéma ci-dessous illustre l'emplacement des différents sites sur le territoire métropolitain.

Figure 1 : Sites CA-GIP sur le territoire métropolitain

1.5 - Projet Référentiel des connectivités

C'est dans ce contexte que j'ai pu travailler sur la mise en place d'une application dans le cadre du projet Référentiel des connectivités.
Le projet est centré sur des solutions visant à répondre aux besoins d'automatisation de l'équipe OER. La transformation numérique engendre le besoin pour les équipes de tenir à jour un référentiel de connectivités WAN. Celui-ci était à l'origine basé sur une gestion manuelle d'un fichier Excel. L'objectif du projet est de proposer une nouvelle solution plus performante, automatisée et évolutive. Employant une méthodologie rigoureuse, le projet implique l'utilisation de plusieurs technologies de pointe visant à aborder et à résoudre des problématiques d'importance stratégique, tout en assurant une surveillance et un suivi de la performance au fil le temps.

Le cœur de l'application réside dans la gestion automatisée de trois types de connectivité WAN essentielles : Les liaisons spécialisées (LS), les tunnels VPN/IP sec et les mobilités. Chacune de ces trois catégories répond à des besoins spécifiques de communication et de sécurité au sein de l'entreprise. Les liaisons spécialisées fournissent une connectivité dédiée et hautement performante, les tunnels VPN/IP sec garantissent la confidentialité des données lors de la transmission sur des réseaux publics et les mobilités offrent une utilisation flexible aux utilisateurs, leur permettant d'accéder en toute sécurité aux ressources internes du Crédit Agricole.

1.5.1 - Besoins d'automatisation croissants

La complexité croissante des infrastructures réseaux exige des solutions innovantes pour leur gestion efficace. Mon projet se concentre sur la création d'une solution d'automatisation pour améliorer la gestion du référentiel des connectivités WAN.
La gestion manuelle d'un parc aussi diversifié de connectivités montre ses limites, générant des erreurs, des problèmes dans les mises à jour ou des problèmes d'intégrité des données. L'utilisation d'un fichier Excel pour gérer ce référentiel a prouvé son inefficacité face aux exigences croissantes de l'entreprise. Ma solution d'automatisation vise à challenger ces limites en rationalisant les processus de gestion, en éliminant les erreurs humaines et en fournissant une vue en temps réel sur les états et les performances des connectivités.

1.5.2 - Suivre l'état des connectivités

L'illustration ci-dessous permet de visualiser les différents changements d'états et de sous-états auxquels sont confrontés les connectivités durant leur mise en service.

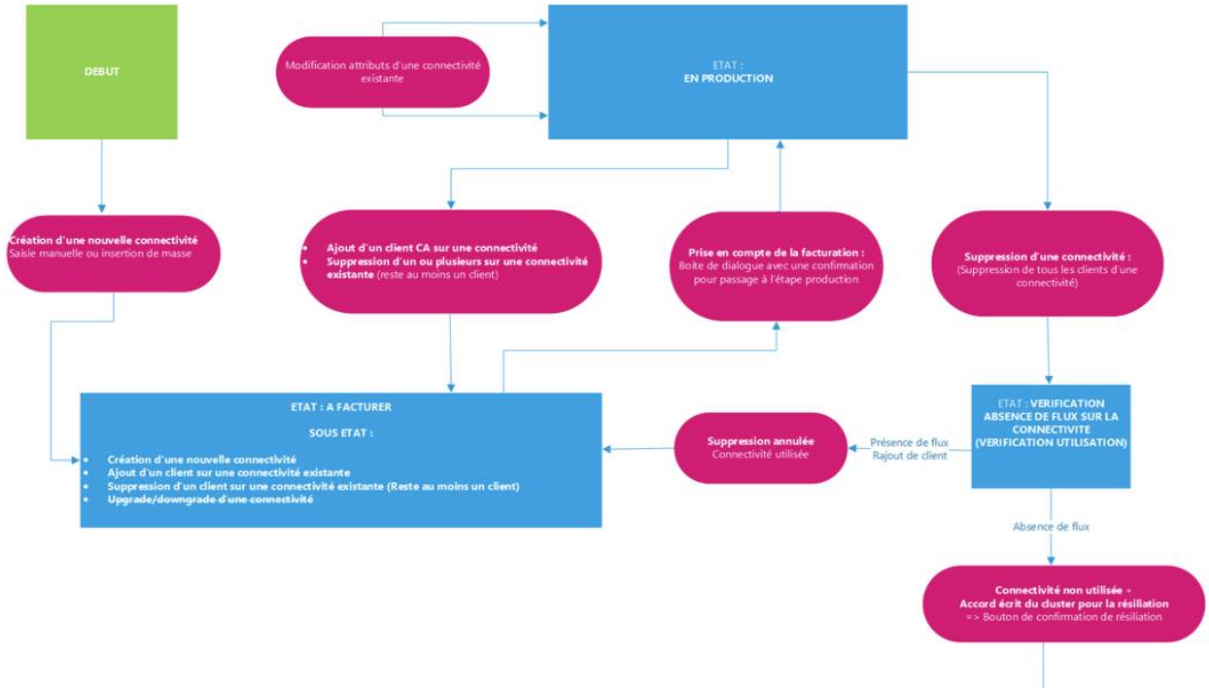

Figure 4 : Changements d'état des connectivités

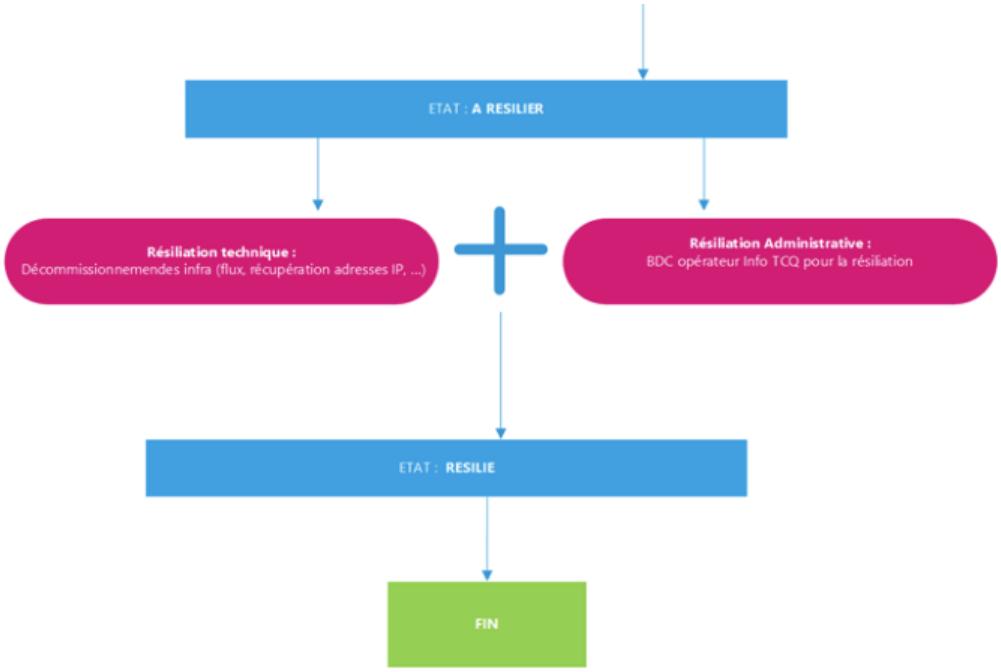

Figure 5 : Résiliation en cas de non-utilisation

En effet, la première nécessité pour l'équipe OER est de pouvoir de suivre avec plus de faciliter et d'exactitude la situation et l'état de chaque connectivité enregistrée. Ces connectivités sont associées à des contacts (les représentants légaux de la connectivités) et des clients (les utilisateurs facturés). L'état de ces connectivités peut varier dans le temps en fonction de certains évènements déclencheurs, comme des ajouts ou des suppressions de clients, générant des facturations supplémentaires. Il faut également pouvoir suivre l'état en cas de demande de résiliations. C'est à dire :

- Observer l'utilisation de la connectivité. Si elle n'est plus utilisée,
- Déclencher la résiliation technique et administrative.
- Attester que la connectivité est bien clôturée.

Dans la partie suivante, nous allons examiner le cahier des charges afin de d'examiner de façon plus procédurale les objectifs, les contraintes et les attentes du projet.

2 – Cahier des charges

2.1 - Introduction

2.1.1 - Projet

Le projet vise à mettre en place une application web basée sur des requêtes API depuis une interface web, permettant d'interagir avec une base de données en toutes simplicité pour les utilisateurs. Les tâches de traitement automatisées (inventaires de connectivités VPN, inventaires de routeurs, gestion de clients et de contacts) améliorent la vitesse d'exécution et la fiabilité des enregistrements. Pour cela, il est nécessaire de mettre en place l'infrastructure informatique appropriée ainsi que la chaîne de déploiement continu.

2.1.2 - Client

Les clients de ce projet sont les collaborateurs internes de CAGIP c'est à dire d'autres équipes du Socle, mais également des équipes d'autres socles et clusters ayant des besoins d'automatisation validés.

2.1.3 - Dates

Le temps de mon alternance. C'est à dire 16 moins, de septembre 2024 à janvier 2025.

2.2 - Objectifs

2.2.1 - Contexte

L'équipe OER offre et gère pour le compte de ses clients plusieurs type de connectivités VPN (lignes spécialisées, tunnels, mobilités). Ces connectivités permettent aux partenaires d'accéder à diverses ressources informatiques du Crédit Agricole. Actuellement, les liaisons spécialisées, les tunnels VPN/IP sec et les Mobilités constituent les principaux types de connectivités utilisées dans l'infrastructure. Mais la gestion de ces connexions repose sur un fichier Excel, ce qui engendre des inefficacités et des erreurs potentielles au niveau des facturations. La nécessité d'une solution automatisée se fait ressentir pour optimiser cette gestion, éliminer les erreurs humaines et libérer du temps pour d'autres tâches importantes.

2.2.2 - But

Le but de ce projet est de fournir à la fois une application web basée sur des appel d'API et l'infrastructure informatique requise pour la faire fonctionner. Cette infrastructure doit être résiliente, sécurisée et conçue selon l'approche *Devops* :

- L'application est une application web interne à l'organisation. Elle repose sur un backend et un frontend.
- L'application doit pouvoir être mise continuellement à jour selon l'approche *Devops*. Elle doit également permettre aux collaborateurs de l'équipe OER d'automatiser la gestion du référentiel de leurs connectivités VPN.
- Une fois fonctionnelle, cette application sera en mesure d'acquérir d'autres fonctionnalités et répondre à des besoins d'automatisation d'autres acteurs internes.

2.3 - Périmètre

2.3.1 - Fonctionnalités

Certaines fonctionnalités répondant à la demande du client doivent être présentes. Celles-ci sont les suivantes :

- L'application doit permettre d'afficher les connectivités existantes, avec la possibilité de filtrer la recherche.
- Le système doit permettre l'ajout ou la suppression des connectivités, mais également la visualisation ou la modification de ces connectivités.
- Le système doit permettre de suivre l'état de chacune des connectivités pour le suivi de leur facturation.
- L'application doit être un référentiel de toutes les connectivités WAN existantes en temps réel.

2.3.2 - Contraintes techniques et technologiques

Les contraintes technologiques sont détaillées dans la section 2.4 : "Contraintes techniques et technologiques" du document ci-présent.

- Les communications entre les différents VLAN et les sous-réseaux hébergeant des services clés doivent faire l'objet d'une demande d'ouverture de flux auprès des administrateurs systèmes et réseaux en lien avec la SSI.
- Les postes de travail pour les développeurs sont fournis par le Département Informatique de l'entreprise.
- Les ressources informatiques virtualisées, comme des ressources ou accès Kubernetes, ainsi que d'autres services virtualisés, sont fournis par le Socle Native DataCloud (NDC) de l'entreprise.

2.3.3 - Fréquences des livraisons

Les livraisons sont partielles et continues, conformément à la méthodologie agile Scrumban. Plus précisément, elles auront lieux toutes les 2 semaines, temps nécessaire et conventionnel pour une itération classique.

2.3.4 - Contraintes Opérationnelles

Voici la liste des contraintes opérationnelles dont l'application doit être pourvue :

- Le temps de fonctionnement doit correspondre aux heures ouvrables,
- En cas d'incident, l'équipe IAC doit faire preuve d'initiative et investiguer,
- Mise en place d'un système de protection des données,
- Accès des utilisateurs par authentification,
- Optimiser la maintenabilité,
- Tolérance aux pannes,
- Assurer un support technique aux utilisateurs requis,
- S'intègre dans l'écosystème de l'infrastructure informatique existante.

2.4 - Contraintes technologiques

2.4.1 - Backend

Voici la liste des technologies principales qui ont été retenues pour le backend dans le cadre de ce projet

- Django est un framework web basé sur Python choisit dans le cadre de ce projet. Il met à disposition aux développeurs un ensemble de bibliothèques et de composants facilitant le développement de sites web. Son architecture MVC (Modèle-Vue-Contrôleur) simplifie la gestion des données. De plus, sa fonctionnalité ORM (*Object-*

Relational-Mapping) facilite grandement la gestion des bases de données, indépendamment du fournisseur.

- Django Rest Framework est une bibliothèque complète pour créer des API RESTful. Elle s'appuie sur Django pour fournir de nouvelles fonctionnalités comme les opérations CRUD (*Create, Read, Update, Delete*), les sérialisations et désérialisations, les authentifications et permissions d'accès aux API.
- MariaDB est un système de gestion de bases de données relationnelles (SGBDR) open-source. Il s'agit d'un fork de MySQL par ses développeurs originels à la suite du rachat de la technologie par Oracle Corporation. Compatible avec MySQL, ce système offre des fonctionnalités qui s'intègrent pleinement dans le cadre de projet.
- Redis est un système de gestion de base de données en mémoire open source hautement disponible. Son appellation signifie "*REmote DIctionary Server*". Il est souvent utilisé comme cache mémoire pour améliorer les performances des applications en stockant des données fréquemment appelées.
- L'utilisation de l'API FortiManager permet d'interagir avec les équipements existants Fortinet. Cela permet d'intégrer la surveillance et la gestion des tunnels en production directement depuis le portail de l'application.
- InfluxDB a été choisi pour le stockage de métriques SNMP, fournissant une base de données de séries temporelles optimisée pour le stockage et la requête de données de performance en temps réel (débit, taux d'utilisation).

2.4.2 - Frontend

Voici la liste des technologies principales qui ont été retenues pour le backend dans le cadre de ce projet. Angular s'est avéré être le choix optimal pour le développement du Frontend en raison de sa structure modulaire, de sa gestion efficace des composants et de sa facilité, d'intégration avec les API.

2.4.3 - Architecture logicielle

L'architecture hexagonale est un modèle de conception logicielle visant à créer des applications modulaires, testables et indépendantes des technologies externes appelées pour récupérer des données. Cette flexibilité, organisée autour d'un emboîtement de Port et d'Adapteur, permet de gérer plus facilement les versions d'une technologie sans avoir à réécrire l'ensemble du code. Cela améliore la maintenabilité de l'application à long terme.
(Voir la partie 7.3 - Mise en place de l'architecture hexagonale pour avoir une représentation visuelle dans la façon dont le code de l'application est organisé)

2.4.4 - Conteneurisation

Kubernetes est une plateforme open source de gestion de conteneurs. Il est conçu pour automatiser le déploiement, la mise à l'échelle et la gestion des applications conteneurisées. Dans le cadre de ce projet, il permettra de déployer continuellement de nouveaux conteneurs dans une infrastructure résiliente, virtualisée, indépendante et hautement disponible.

2.4.5 - Surveillance et supervision

- Surveillance du temps de réponse, du temps de chargement, de l'utilisation des ressources, des activités suspectes, des vulnérabilités, du nombre et type de requêtes, des erreurs HTTP et des exceptions levées dans le code.
- Utilisation d'outils comme Elasticsearch, Logstash et Kibana (ELK) pour surveiller les journaux de l'application.
- Utilisation d'outils comme Prometheus et Grafana pour collecter et visualiser des métriques.

2.5 - Exigences fonctionnelles

2.5.1 - Exigences fonctionnelles prioritaires

Voici la liste des fonctionnalités qui doivent intégrer prioritairement l'application, et sur lesquelles axer les efforts de développement.

- SF01 : Le système doit permettre de consulter les connectivités existantes, avec la possibilité de filtrage et de recherche.
- SF02 : La solution doit permettre de modifier les attributs des connectivités existantes, voire d'ajouter ou de supprimer des clients ou des contacts associées à ces connectivités.
- SF03 : Le système doit pouvoir prendre en charge des méthodes flexibles pour l'ajout d'une nouvelle connectivité avec la saisie des informations pertinentes.
- SF04 : La solution d'automatisation doit concrétiser le diagramme de changement d'état afin de gérer les différentes phases de connectivité.
- SF06 : Le système de facturation doit prendre en charge les connectivités à facturer, en les basculant vers l'état « en production ».
- SF07 : Le système doit assurer la gestion des différentes étapes de vérification de l'utilisation des connectivités avec des confirmations, voire des résiliations administratives et techniques afin d'assurer une résiliation complète.
- SF08 : Le système de contrôle doit interagir avec l'API du FortiManager pour vérifier la présence des tunnels en production et afin d'identifier les tunnels absents de la production mais présents dans notre référentiel.

2.5.2 - Exigences fonctionnelles secondaires

Voici la liste des fonctionnalités secondaires qui doivent intégrer l'application après l'implémentation des fonctionnalités principales.

- SF09 : Le système de gestion doit offrir une insertion de masse via un fichier CSV pour une mise à jour efficace du référentiel.
- SF10 La solution doit solliciter l'API InfluxDB pour extraire l'état de la supervision des connectivités, en se basant sur les métriques SNMP.

2.6 - Exigences non fonctionnelles

Voici une liste qui décrit les critères de performance, de qualité ou de contrainte attendue concernant l'application.

- SF01 : Le Backend du système doit être développé en utilisant Django, un framework Python robuste et éprouvé.
- SF02 : Le Frontend doit être construit à partir du langage Angular, et offrir une interface utilisateur confortable, dynamique et réactive.
- SF03 : Le système doit pouvoir utiliser le concept d'API pour interagir avec les outils externes tels que FortiManager, InfluxDB et ICINGA.
- SF04 : La solution d'automatisation doit utiliser une base de données MySQL pour stocker les informations relatives aux connectivités.
- SF05 : Les données du référentiel doivent être solidement modélisées, avec des tables pour les connectivités, les liaisons spécialisées, les tunnels, la mobilité, les clients et les contacts (MCD).
- SF06 : Mise en place de workflows automatisés pour le processus de facturation et de résiliation
- SF07 : Envoi de courriels de confirmation aux équipes concernées en cas de résiliation ou de prise en charge.
- SF08 : Interaction avec l'API du FortiManager pour vérifier l'état des tunnels en production.
- SF09 : Sollicitation de l'API InfluxDB pour obtenir les métriques SNMP de surveillance.
- SF10 : Intégration avec ICINGA pour alimenter la base de données InfluxDB avec des données d'inventaire.

2.7 - Chaine de déploiement continu

2.7.1 - Environnement

GitLab est utilisé pour la gestion de version et le contrôle de code source à travers une chaine de déploiement. Il permet une collaboration efficace au sein de l'équipe en facilitant la fusion de code, le suivi des modifications et la gestion des incidents. Le serveur GitLab peut être connecté à des éditeurs de code de haute qualité comme VS code.

2.7.2 - Suivi du déploiement

- L'application est testée dans plusieurs environnements différents avant d'être envoyée dans l'environnement de Production
- Un environnement de développement est utilisé par les développeurs pour effectuer des tests fonctionnels et non fonctionnels avant le déploiement en production.

2.7.3 - Rollback

Un retour à des versions antérieures de l'application reste possible en cas de dysfonctionnement du code source. Les différentes versions du code sont stockées sur une instance GitLab sécurisée. Les différentes images docker sont stockées dans le registre interne. L'objectif principal en cas de dysfonctionnement et/ou de régression est de pouvoir minimiser l'impact sur les utilisateurs et leurs données en générant des conteneurs à partir de code source fonctionnel.

2.8 - Mesures de sécurité

- L'utilisation d'Active Directory pour l'authentification sécurisée. Des groupes AD sont configurés pour contrôler l'accès aux différentes parties de l'application.
- Les droits d'accès aux points de terminaison du serveur Django sont associés à des groupes Active Directory existants auxquels l'utilisateur appartient.
- Des logs d'accès et d'erreurs sont générés à chaque fois qu'un point de terminaison est sollicité.
- Des audits de sécurités sont régulièrement effectué pour confirmer l'intégrité et la sécurité de l'application dans le temps.

2.9 - Livrables

Voici la liste des éléments attendus lors de la livraison finale et complète de l'application.

- Une base de données sécurisée dans chaque environnement de déploiement,
- Des micro-services fonctionnels orchestrés avec Kubernetes,
- Une chaine de déploiement continu dans un environnement de développement et dans un environnement de production,
- Une application backend et frontend adaptable hautement disponible répondant aux besoins de l'équipe OER,
- Une documentation complète et à jour visant à décrire les différentes démarches, proposer des tutoriels d'*on-boarding*, ou d'établir la liste inventaire des machines et des services et des différents composants de l'application.

3 – Etude de faisabilité

3.1 - Résumé du cahier des charges

L'objectif du projet est de mettre en place une application web conteneurisée à disposition des équipes internes de l'entreprise afin de pouvoir répondre aux besoins d'automatisation de tâches. Dans ce cas présent, les fonctionnalités à implémenter doivent permettre aux collaborateurs de l'équipe OER de gérer un référentiel de connectivités VPN de façon plus efficace et productive. L'application est évolutive et peut répondre à d'autres besoins au fils du temps. Basée sur des API, elle est constituée d'une partie Web et d'un backend. Elle doit pouvoir accéder à différents outils externes. Les ressources informatiques nécessaires pour la réalisation de ce projet doivent être commandées auprès du Socle NDC. Le matériel de travail et les licences Office 365 pour les collaborateurs de l'équipe du projet sont fournis par le support.

3.2 - Analyse technique

Les technologies choisies dans le cadre de ce projet sont robustes, open-sources, documentées, largement utilisées par une large communauté d'utilisateurs particuliers ou professionnels à travers le monde. Ce qui est le cas pour des technologies comme Ansible, Python, Angular, Redis, MariaDB, GitLab.
Le Framework Django, codé en Python pour la partie backend, est adapté pour la conception de points de terminaisons sécurisés prenant en charge les opérations CRUD grâce à son

architecture MVC. De plus, il assure une communication sécurisée et interchangeable avec la base de données grâce à sa fonctionnalité intégrée ORM.

Kubernetes est une plateforme open-source de gestion de conteneurs qui automatise le déploiement, la mise à l'échelle et la gestion des applications conteneurisées. Développé initialement par Google et maintenant maintenu par la *Cloud Native Computing Foundation*, Kubernetes permet de gérer efficacement des clusters de conteneurs sur plusieurs hôtes, offrant une haute disponibilité et une résilience accrues. Il simplifie la gestion des micro-services, facilite les déploiements continus et les mises à jour sans interruption de service, et optimise l'utilisation des ressources grâce à son orchestration intelligente.

3.3 - Analyse économique

L'application est conçue pour permettre d'améliorer considérablement la productivité de l'équipe OER grâce à la gestion automatisée de certaines tâches manuelles. Le temps d'exécution des tâches et les erreurs humaines peuvent être considérablement réduits. Les éléments déclencheurs de facturations peuvent être consignés de façon automatisée, avec un meilleur suivi, ce qui a pour effet d'améliorer les bénéfices, de réduire les incertitudes et les litiges.

De plus, l'application peut accueillir d'autres projets d'automatisation de masse, ce qui peut potentiellement renforcer la force de travail et les bénéfices d'autres secteurs.

3.4 - Analyse opérationnelle

Cependant, l'architecture de l'application est complexe et celle-ci doit s'adapter à des standards de sécurité élevés avant d'être opérationnelle. Un, ou plusieurs experts en architectures systèmes, réseaux et sécurité informatique doivent être recrutés pour construire l'infrastructure sur laquelle repose l'application, grâce aux technologies citées précédemment. D'autres professionnels, comme le Responsable produit et le *Scrum Master*, doivent être également recrutés. Enfin, l'équipe de développement doit avoir un certain nombre d'années d'expérience dans le domaine, mais aussi être à l'aise dans les procédures à suivre qu'implique l'adoption de la méthodologie Agile.

3.5 - Matrice SWOT

La matrice SWOT, également connue sous le nom d'analyse SWOT, est un outil stratégique puissant utilisé par les professionnels pour évaluer les forces, les faiblesses, les opportunités et les menaces liées à une entreprise, un projet ou une initiative. L'acronyme SWOT provient des termes anglais *Strengths* (forces), *Weaknesses* (faiblesses), *Opportunities* (opportunités) et *Threats* (menaces). Cette méthode offre une vision générale et large, qui permet aux entreprises de faire les bons choix, mais également de tirer profit de leurs principaux avantages, tout en minimisant les risques.

Facteurs internes	Forces	Faiblesses
	Forte demande d'automatisation Service cloud à la demande Sécurité déléguées à des équipes spécialisées Utilisation de techologies web éprouvées Réseau robuste et sécurisé Langages de programmation populaires Recrutement possible de développeurs qualifiés Méthodologie de gestion de projet adaptative Possibilités de développement illimitées	Politique de gestion de flux réseau rigoureuse (zéro trust) Infrastructure virtualisée complexe Besoin de passer par un support en cas de problème technique Dépendances aux décisions des équipes Cloud, support, SSI Dépendances aux décisions des équipes Réseaux Possibles problèmes de communication inter-équipes
Facteurs externes	**Opportunités**	**Menaces**
	Améliorer la productivité de l'ensemble du Socle Améliorer la réputation du Socle Diminuer les erreurs humaines Améliorer la fiabilité des données Possibilités de répondre à tout type de besoin Diversification des technologies Formation de personnel ou d'étudiants Faciliter le cadre de travail des collaborateurs	Cyber attaques Maintenances prévues ou imprévues Migration techniques (de VM, de sous-réseaux, de cluster) Changements techniques liés au besoin du cloud Nouvelles versions (outils, librairies Python) Coupure de service Migration ou changement forcée des outils d'API externes

Figure 6 : Matrice S.W.O.T.

3.6 - conclusion

Bien que la mise en place d'une telle application soit complexe et demande de réelles compétences pour être fonctionnelle, tant dans la complexité des technologies utilisée que par les hauts standards de qualités et de sécurités demandés par l'entreprise, son déploiement est tout à fait réalisable et peut apporter de nombreux bénéfices tant économiques que techniques à l'ensemble du Socle.

4 – Management

4.1 - Établir le constat des compétences recherchées

La matrice RACI est un outil qui va me permettre de définir les responsabilités entre les acteurs du projet. Elle permet également de clarifier et de mettre en exergue les rôles, les compétences et les tâches. Chaque tâche sera associée à :

- Un responsable (*Responsible*) chargé d'effectuer le travail pour la réaliser,
- Un responsable ultime (*Accountable*) de tâche est chargé de rendre les comptes à la hiérarchie,
- Des personnes consultées (*Consulted*) dont l'avis est soit important ou soit indispensable pour réaliser la tâche,
- Des personnes informées (*Informed*) qui doivent recevoir une communication lorsque la tâche est réalisée.

	RH	Equipe IaC					Equipe Cloud	RSSI
Tâches	Recruteurs	Chef de projet	Scrum Master	PO	NetDevops	Développeurs	AR*	CC*
PLANIFICATION								
Organiser les réunions		A	R	C	I	I		
Définir le cahier des charges fonctionnel		A	I	R	C	C	C	C
Analyser la faisabilité d'un besoin		A	I	R	C	C	C	C
Rédiger les user stories		A	I	R	I	I		
Planifier la feuille de route		A	I	I	I	I		
Evaluer les risques		A	I	C	R	C	C	C
Gérer le budget	C	A	I	R	I	I		
Recrutement du PO, Scrum master, Netdevops	R	A	I	I	C	I	I	
Assurer la formation technique de l'équipe		A	R	I	R	R		
Rédiger les backlogs produits		A	I	R	I	I		
Rédiger les backlogs de sprint		A	I	R	I	I		
Demande d'ouverture des flux réseaux		A	I	I	R	I	C	C
Segmentation du réseau		A	I	I	R	I	C	C
Demande d'accès au noeud Kubernetes		A	I	I	R	I	C	
Gestion des licences logicielles		A	R	I	I	I	C	
Founir la base de données		A	I	I	R	I	C	I
Assurer le support utilisateur		A	I	I	C	R		

	RH	Equipe IaC					Equipe Cloud	RSSI
Tâches	Recruteurs	Chef de projet	Scrum Master	PO	NetDevops	Développeurs	AR*	CC*
DEPLOIEMENT DE L'INFRASTRUCTURE								
Configurer les ressources Kubernetes		A	I	I	R	I		
Estimer les tâches d'infra		A	I	R	C	C		
Configurer l'accès aux ressources collaboratives		A	I	I	R	I	C	
Configurer l'accès aux postes de travail		A	I	I	R	I	C	
Mettre en place les outils CI/CD		A	I	I	R	I		
Superviser le registre d'images Docker		A	I	I	R	I		
Superviser le serveur Gitlab		A	I	I	R	I		
Configurer l'environnement de développement		A	I	I	R	I		
Configurer l'environnement de production		A	I	I	R	I		
Concevoir les Dockerfiles		A	I	I	R	I		
Recruter les développeurs	R	A	I	I	C	I		
Instructions pour configurer les postes de tavail		A	I	I	R	R		
Ecrire la documentation générale		A	I	I	R	R		

Figure 7 : Matrice RACI partie 1

	RH	Equipe IaC					Equipe Cloud	RSSI
Tâches	Recruteurs	Chef de projet	Scrum Master	PO	NetDevops	Développeurs	AR*	CC*
DEPLOIEMENT DE L'APPLICATION								
Déploiement continue de l'application		A	I	I	I	R		
Organiser des revues avec le client		A	I	R	I	I		
Automatisation des tests de qualités		I	I	I	A	R		
Surveiller les performances techniques		A	I	I	R	C		
Mettre à jour les librairies de code		A	I	I	R	I		
Intégrer et mettre à jour les tests de la pipeline		A	I	I	R	I		
Estimer les tâches de développement		A	I	R	C	C		
Gérer les incidents techniques		A	I	I	R	C	I	
Mise en intégration		A	I	C	R	R		
Mise en pré-production		A	I	C	R	R		
Mise en production		A	I	C	R	R		

	RH	Equipe IaC					Equipe Cloud	RSSI
Tâches	Recruteurs	Chef de projet	Scrum Master	PO	NetDevops	Développeurs	AR*	CC*
SECURITE								
Effectuer des tests de sécurités du code		I	I	I	A	R	C	C
Sauvegarde et chiffrement de la base de données		A	I	I	R	I	C	I
Effectuer des tests de performances		I	I	A	R	C	C	I
Veille technologique des outils		I	I	I	R	R	A	I
Veille technologique cybersécurité		I	I	I	I	R	R	A
Monitoring de l'infrastructure		I	I	A	R	I		
Monitoring de l'application		I	I	A	R	R		
Monitoring du réseau		I	I	A	R	I		
Réalisations d'audits de sécurité		A	I	C	C	I	I	R

	RH	Equipe IaC					Equipe Cloud	RSSI
Tâches	Recruteurs	Chef de projet	Scrum Master	PO	NetDevops	Développeurs	AR*	CC*
MANAGEMENT								
Résoudre les conflits		A	R	I	I	I		
Animer les rétrospectives		A	R	I	I	I		
Animer le daily		A	R	I	I	I		
Rédiger les compte-rendus de réunion		A	R	I	I	I		
Animer les séances de planning poker		A	I	R	I	I		
Animer les revues de sprint		A	I	R	I	I		
Surveillance des performances humaines	I	A	R	C	C	C		
Rédiger les tickets d'on-boarding	I	A	R	I	I	I		

Figure 8 : Matrice RACI partie 2

4.2 - Définition des rôles et des responsabilités

Voici la liste des cinq rôles qui ont été déterminés au sein de l'équipe dont le travail collaboratif va permettre de réaliser avec succès les objectifs du projet. Cette équipe est nommée l'équipe IAC (*Infra As Code*) :

- Le chef de projet,
- Le *Scrum Master*,
- Le Responsable du projet (*Product Owner*)
- L'équipe des développeurs, dont les ingénieurs *DevOps*.

L’équipe IAC est en relation constante avec des équipes d'autres socles : l'équipe RH, les équipes du Socle NDC et l'équipe SSI (sécurité des réseau). Les architectes réseau (AR) et les consultants en cybersécurité (CC) jouent des rôles complémentaires mais distincts dans la conception, la mise en œuvre et la sécurisation du projet. En travaillant en collaborant étroite, l’application peut profiter d’une infrastructure optimisée pour la performance, la sécurité et la résilience. Elle répond ainsi aux besoins de l'organisation tout en optimisant sa protection contre les menaces potentielles.

En collaboration avec l'équipe RH, cette matrice permet de dégager la liste des compétences recherchées. Celle-ci permet d'élaborer les fiches de postes. Les candidats rencontrent à la fois les recruteurs de l'équipe RH et les collaborateurs de l'équipe IAC, afin de permettre la bonne analyse des performances techniques et relationnelles.

4.3 - Recrutement et accompagnement des collaborateurs

4.3.1 - Déroulement des entretiens

Le recrutement des collaborateurs s'effectue en coopération entre l'équipe IAC et l'équipe des ressources humaines. Deux entretiens ont lieu pour les candidats potentiels. Un entretien général avec l'équipe RH, et un entretien technique avec un technicien de l'équipe IAC.
Une première étape de recrutement se focalise sur le recrutement des leaders opérationnels comme le Responsable Projet et le *Scrum master*.
Une seconde étape se focalise sur le recrutement des leaders techniques : les *DevOps*.
Enfin, une troisième étape est consacrée au recrutement des développeurs lorsque l’infrastructure virtuelle est déployée.

L'équipes RH se chargent de sélectionner les candidats disponibles sur le marché du travail en fonction des compétences recherchées. Ils peuvent pour cela s'appuyer sur les candidatures spontanées, les candidatures internes de l'entreprise ou des offres de prestataires comme les ESN.

Lorsque le premier entretien est concluant avec le personnel de l’équipe RH est concluant, le candidat peut être présenté pour le second entretien individuel.
La première phase de recrutement est dédiée au Responsable produit et au *Scrum master*.
La deuxième phase de recrutement est dédiée aux ingénieurs *DevOps*. Pour cela, je peux être accompagné d’un leader technique d'une autre équipe. Cela me permet d'avoir une meilleure approche et de réduire ainsi les risques d'erreurs dans l'analyse en confrontant mon opinion avec des points de vue des collaborateurs plus expérimentés.
Enfin, dans la troisième phase de recrutement, les *DevOps*, le *Project Owner* et le *Scrum master* peuvent m'assister pour analyser plus précisément les compétences des candidats

4.3.2 - Test des aptitudes

Le premier entretien du candidat avec le service RH dure une trentaine de minutes. Il s'agit d'un échange global avec le candidat pour évaluer son parcours, ses réalisations, ses compétences, ses motivations et sa communication (verbale et non-verbale).
Si le premier entretien est concluant, un deuxième entretien de 40 minutes se déroule à quelques jours d'intervalle. Il s'agit d'évaluer son expertise technique sur les technologies utilisées dans le cadre du projet, ainsi que ses connaissances théoriques sur le développement d'application, et la sécurité des réseaux.

Un questionnaire est remis au candidat dans les dernières minutes de l'entretien afin de tester ses connaissances pratiques dans une mise en situation.
Enfin, si les entretiens sont concluants, le candidat est invité à rejoindre l'équipe pour son *on-boarding* dans l'équipe.

4.3.3 - On-boarding

Une période de formation est nécessaire pour permettre nouveau collaborateur de s'imprégner de l'ambiance de travail et des procédés inhérents à l'entreprise.
Le responsable d'*on-boarding* effectue la création des tickets à destination des services spécialisés. Cela permet au collaborateur de recevoir dès son arrivée le matériel informatique, ses licences Office 365 et ses accès aux diverses ressources partagées.

Ensuite, il est invité aux différentes cérémonies agiles. Il dispose d'une période d'essai pour s'adapter à l'organisation existante, échanger avec les autres collaborateurs. Des solutions de *pair-programming* peuvent être proposées pour faciliter l'intégration humaine et technique.

De plus, un wiki interne est progressivement construit afin de fournir plus d'explications informelles concernant à la fois des procédures à suivre ou des indications concernant l'infrastructure. Cette solution offre plus d'autonomie et de flexibilité aux collaborateurs. Parmi ces procédures, il est possible d'y détailler comment effectuer les paramétrages du poste de travail de développement, ou les moyens d'accéder aux ressources collaboratives.

4.3.4 - Recrutement progressif

Les estimations et la réalisation des tâches sont effectuées par les experts techniques de l'équipe. Il s'agit des ingénieurs *DevOps* et des développeurs. Cependant, il convient de noter que lorsque le projet démarre, l'équipe technique n'est pas complète. Il n'est d'ailleurs pas nécessaire qu'elle le soit, car le déploiement de l'infrastructure s'effectue avant le déploiement de l'application.

Dans un premier temps, les recrutements se focalisent donc sur le *Project Owner*, le *Scrum master* et les ingénieurs *DevOps*. Le *Project Owner* a pour mission de rédiger les tâches techniques dans le *backlog* permettant d'élaborer, de déployer et de rendre fonctionnel l'environnement de production. Il estime, en collaboration avec les ingénieurs *DevOps*, et sous la supervision du *Scrum Master*, la complexité des tâches associées au déploiement de l'infrastructure avec des story points. Cet aspect de la méthodologie *Scrum* en agilité est détaillée dans la section 5.3.5 dédiée au *Planning poker*.

Dans un second temps, lorsque l'infrastructure est déployée, les recrutements se focalisent ensuite sur les développeurs back, front et full-stack. Ils sont chargés, en collaboration avec le PO et les ingénieurs *DevOps*, d'estimer la complexité des tâches de développement de l'application, comme des ajouts de fonctionnalité, des mises à jour de bibliothèques de code, des intégrations de nouveaux tests, des corrections de bugs.

4.4 - Gestion opérationnelle de l'équipe

4.4.1 - Méthode agile Scrumban

Dans le cadre du projet, c'est la méthodologie agile *Scrumban* qui a été retenue par sa nature flexible, adaptative et visuelle. En tant que méthode de gestion de projet Agile, elle s'intègre très bien une équipe de travail collaboratif. En effet, elle décrit des rôles, des réunions et des outils à mettre en place pour aider à structurer un travail dans le cadre d'un projet. L'intérêt de la méthode *Scrumban* est l'association de la pile des tâches de la méthode Scrum avec un tableau visuel Kanban. L'organisation des tâches dans différentes colonnes colorées rend la gestion et le suivi quotidien plus facile, agréable et confortable pour l'équipe.
Cette méthodologie est très efficace dans le cadre de mon projet de développement logiciel, car elle a été pensée pour fournir des livraisons de fonctionnalités de manière cyclique, récurrentes et quotidiennes, tout en favorisant l'amélioration continue et la communication régulière entre les parties prenantes.

4.4.2 - Les artefacts

Les artefacts sont les éléments auxquels mon équipe doit réfléchir pour permettre d'optimiser l'avancée du projet : les *backlog*s et l'incrément. Le *backlog* produit est une liste dynamique de toutes les tâches recensées (*User stories*, Epic ou correctifs) pour la réalisation du projet. Cette liste dynamique contient l'ensemble des différentes demandes du client traduites sous la forme de tâches techniques appelées *User Stories*. De nouvelles tâches peuvent y être ajoutées en cours de développement si nécessaire.
Le *backlog* de sprint est une liste de ces tâches qui ont été sélectionnées depuis le *backlog* produit par l'équipe de développement. Ces tâches doivent être réalisées durant la durée du sprint. En début de sprint, une réunion est programmée pour déterminer quels éléments du *backlog* produit prioriser sur le prochain cycle de travail.

Enfin, les incréments sont les fonctionnalités ou les corrections qui ont été livrées à la fin du sprint. Celle-ci sont exploitables par le client.

4.4.3 - Les rôles

Les membres de l'équipe que je souhaite mettre en place sont le Responsable produit, le *Scrum master*, deux ingénieurs *DevOps* et trois développeurs. Le responsable produit, ou *Project Owner* gère le *backlog* produit. Directement à l'écoute du client, il fournit à l'équipe les directives claires et priorisées sur les fonctionnalités à livrer. Il est le représentant du client au sein de l'équipe, et doit communiquer avec succès sa vision du produit aux développeurs. Le quotidien de l'équipe est animé par une suite d'évènements et de réunions. Ces réunions jouent un rôle fédérateur et apportent de la cohérence en permettant de synchroniser les membres de l'équipe avec la vision du projet.

Le *Scrum master* joue un rôle de médiateur. Il prévoit les ressources nécessaires pour organiser, coordonner et animer les différentes réunions et équipes. Il peut également intervenir pour gérer les conflits ou répondre à des problématiques humaines et techniques.

Les ingénieurs *DevOps* sont les leaders techniques de l'équipe. Grâce à leurs expériences et compétences transverses dans la sécurité, la gestion des systèmes et des réseaux et le développement logiciel, leur rôle consiste à gérer l'infrastructure virtuelle, tout en veillant aux tâches liées au déploiement continu et à l'optimisation des performances de l'application.

Les développeurs produisent les fonctionnalités de l'application une fois l'infrastructure déployée. En plus de participer à l'enrichissement continu de l'application, ils élaborent également les tests d'intégrité backend et frontend venant attester de la qualité du code à long terme, tout en étant proactifs et forces de proposition.

4.4.4 - Les événements agiles

La méthodologie Agile prévoit la mise en place de plusieurs réunions hebdomadaires qui visent à apporter aux membres de l'équipe les moyens de planifier, de suivre et d'analyser l'avancée des objectifs du projet. Ces réunions sont organisées sur toute la durée d'un sprint, c'est à dire sur deux semaines.

- Le planning poker (ou réunions d'estimation de tâches) est un évènement ponctuel qui peut être programmé durant la semaine de sprint. Le Responsable projet faire évaluer les nouvelles tâches de son *backlog*. Il peut également demander à réévaluer des tâches qui existent déjà. Il est possible que cette activité ait lieu durant la

planification du sprint, si les tâches ne sont pas nombreuses, afin d'éviter de planifier davantage de réunions. La durée de ce type d'échange ne doit pas excéder une heure. Elle est programmée manuellement via la messagerie Outlook.

- La planification de sprint est une réunion programmée à chaque début de sprint (toutes les deux semaines). Les membres de l'équipe actuelle sont invités à décider, de manière collaborative, et sous le contrôle du *Scrum master* et du Responsable projet, quelles tâches inclure dans le *backlog* de sprint. Toutes ces tâches doivent être réalisées dans le temps imparti du sprint. La durée de cette réunion doit être située entre 45 minutes et 1h30. Elle est programmée automatiquement via la messagerie Outlook.

- Le Daily est la première réunion de la journée. Elle se passe le matin lorsque tous les membres de l'équipe sont présents. Ceux-ci vont discuter de l'état et de l'avancement de chaque tâche qui leur est assignée. Les membres de l'équipe peuvent communiquer sur les difficultés ou les points de blocages rencontrés. Enfin, la progression des tâches est suivie plus facilement grâce aux colonnes du tableau Kanban. Ces tâches doivent être progressivement placées dans la colonne "terminées" à la fin des 2 semaines. La durée de cette réunion ne doit pas dépasser 15 minutes.

- La revue de sprint est une réunion qui se tient dans les derniers jours du sprint. Elle est supervisée par le Responsable produit. Celui-ci présente aux clients les fonctionnalités et les éléments du *backlog* produit qui ont été complétés par l'équipe de développement pendant le sprint. Ce temps d'échange permet aux parties prenantes d'apprécier les progrès réalisés. Les clients peuvent alors valider que les fonctionnalités répondent à leurs attentes et fournir des retours immédiats. En fonction de l'échange, le Responsable projet peut ajuster son *backlog* produit pour refléter les nouvelles priorités et les nouvelles fonctionnalités à développer. La durée de cette réunion doit être située en 45 minutes et 1h30. Elle sera programmée manuellement dans les derniers jours du sprint, à la place du Daily journalier.

- La rétrospective est une réunion qui permet à l'équipe de se rassembler afin d'analyser ce qui a fonctionné ou non dans le sprint. L'idée est de se concentrer sur les éléments à améliorer, et non sur les échecs. Plusieurs outils méthodologiques peuvent être utilisés pour répondre au besoin d'amélioration des rapports humains techniques. C'est ce que nous allons dans le prochain chapitre.

4.5 - Amélioration continue

Les rétrospectives sont des temps d'échange important pour les membres de l'équipe. Plusieurs méthodes peuvent être utilisées dans le cadre de son animation. Celle-ci se déroule sous la supervision du *Scrum master*.

4.5.1 - Gestion des conflits

De profonds désaccords peuvent survenir entre les collaborateurs. Cela peut entrainer de graves dysfonctionnements au sein de l'équipe, impactant sa cohésion ainsi que sa productivité. Il est donc important de savoir détecter les points de conflit pour les régler rapidement. La rétrospective est une réunion qui peut aider à gérer ce type de situation. Par exemple, la méthode *DESC* est une technique de communication simple et facile à mettre en place. Celle-ci permet de rationnaliser la communication entre les différentes parties en conflit. Dans mon équipe, c'est le *Scrum master* qui intervient en tant que médiateur.

DESC est un acronyme et désigne les termes suivants en français : Décrire, Exprimer, Spécifier et Conclure. Les acteurs concernés vont pouvoir échanger et se mettre d'accord sur une solution en organisant le temps d'échange autour de quatre étapes.
La première étape "Décrire" permet d'exprimer de manière objective et factuelle la situation ou le comportement qui pose problème.
La deuxième étape "Exprimer" permet au collaborateur d'exprimer ses émotions et son ressenti face à la situation problématique de façon claire et encadrée par un médiateur.
La troisième étape "Spécifier" offre une proposition de changement que la partie impliquée souhaite voir pour que le conflit soit résolu.
Enfin, la quatrième étape "Conclure" évoque les aspects positifs que ce changement apporte pour toutes les parties concernées.

4.5.2 - Rétrospectives en 4 "L"

Les 4 "L" de la rétrospective sont une méthode utilisée pour encourager des réflexions structurées, constructives et collaboratives sur les expériences passées. L'objectif est de pouvoir améliorer continuellement les processus de travail.
Ces points d'échanges s'articulent autour de quatre idées motrices : « *Liked* », « *Learned* », « *Lacked* », « *Longed for* ». Les points soulevés peuvent être regroupés dans un tableau de quatre colonnes. :

- La colonne « *Liked* » met l'accent sur les aspects positifs du sprint, qui ont été appréciés par les membres de l'équipe. Le nom de cette première colonne pourrait être traduit en français par « Ce que j'ai aimé ».

- La colonne « *Learned* » évoque les connaissances et les compétences qui ont été acquises durant le sprint. Cela permet de mettre en avant le savoir-faire de l'équipe. Le nom de la deuxième colonne pourrait être traduit par « Ce que j'ai appris ».
- La colonne « *Lacked* » met en évidences les ressources (informatiques ou soutiens) qui ont manqué aux collaborateurs pour qu'ils puissent pleinement réaliser leurs tâches. Cela aide à cerner les domaines qui demandent un point de vigilance pour les prochains sprints. Le nom de cette troisième colonne pourrait-être « Ce qu'il m'a manqué pour réussir ».
- Enfin, la colonne « *Longed for* » permet aux membres de l'équipe d'exprimer leurs désirs ou leurs aspirations afin de mettre en avant des points d'améliorations spécifiques qu'il faut intégrer lors des prochaines itérations. Le nom de la quatrième colonne pourrait-être « Ce que je souhaiterai pour la prochaine fois ».

4.5.3 - Les chapeaux de Bono

Les six chapeaux de Bono sont une méthode de résolution de problème et de créativité. Elle permet aux participants d'analyser un problème en fonction de six rôles, représenté par des chapeaux colorés sur un support visuel. Ces rôles sont attribués aux collaborateurs de façon aléatoire et interchangeable. Ainsi, chacun s'exprime selon des angles d'approche et des points de vue différents. Le but est de laisser émerger des idées créatives ou ingénieuses qui vont apporter des solutions ou dégager des axes d'amélioration.

- Le chapeau blanc : le collaborateur énonce des faits purement et simplement. Il donne au groupe des informations objectives. Il représente la neutralité et répond à la question suivante : Quels sont les faits ?
- Le chapeau rouge : le collaborateur apporte des informations teintées d'émotions, de sentiments, d'intuitions ou de pressentiments. Il n'a pas à se justifier auprès des autres chapeaux. Il représente la passion et répond à la question suivante : Qu'est-ce que je ressens ?
- Le chapeau noir : le penseur adopte une approche pessimiste de la situation. Il insiste sur les dangers, les risques, et la nécessité de prudence. Sa réflexion, toujours logique, aide à repérer les éventuels freins et obstacles. Il répond à la question suivante : Pour chacune de ces solutions, quels sont les risques, les avantages, les inconvénients ?
- Le chapeau jaune : le penseur adopte une posture optimiste. A l'instar du chapeau noir, ses commentaires sont constructifs, positifs et tentent de rendre les idées des autres membres concrètes. Il répond à la question suivante : Pour chacune de ces solutions, comment les mettre en œuvre ?
- Le chapeau vert : le penseur est à la recherche de solutions créatives, en dehors des sentiers battus, qui peuvent répondre aux critiques du chapeau noir. Il représente la créativité sans limite et la fertilité des idées. Il répond à la question suivante : Quelles sont les solutions possibles, y compris les plus farfelues ?

- Le chapeau bleu : le penseur est l'animateur de la réunion, le meneur de jeu. Il représente l'organisation et la canalisation des idées. Il est très important car il maintient la discipline et veille à ce que les participants utilisent bien leurs chapeaux. Il répond aux questions suivantes : quelles sont les solutions à retenir ? Comment la mettre en application ?

4.5.4 - Le tableau des humeurs

Le tableau des humeurs est un outil visuel utilisé dans les méthodologies Agile pour suivre l'humeur et le bien-être des membres de l'équipe quotidiennement. En favorisant une communication honnête, sincère et transparente, il est possible de détecter rapidement les problèmes humains, techniques ou les sources de stress. Si plusieurs membres de l'équipe signalent une humeur négative, cela peut indiquer un problème sous-jacent qui nécessite une attention particulière.
Le *Scrum master* peut mettre à disposition de l'équipe différentes images contenant des formes visuelles spécifiques. Ces formes peuvent être utilisée par les membres de l'équipe pour traduire leur sentiment ou leurs émotions. La description des humeurs s'effectue ensuite à tour de rôle dans une optique d'échange et d'apport de solutions.

5 - Planification

5.1 - Étapes du projet

5.1.1 - Recrutement des collaborateurs

La livraison du projet s'effectue en cinq étapes. Celles-ci sont détaillées dans les sections suivantes.

La première étape pose les fondations sur lesquelles le projet va se construire. Elle prévoit le recrutement des premiers collaborateurs de l'équipe, la rédaction des tâches du *backlog* et la mise en place de l'environnement de travail. Les premières personnes qui doivent être recrutées sont le Responsable projet (PO), le *Scrum Master* et les *DevOps* (experts informatiques avec compétences transverses).
Pour cela, des tickets à destination des équipes supports sont rédigés pour l'obtention des postes de travail, la création des comptes dans l'Active Directory et l'obtention des d'accès aux services collaboratifs. Une commande est passée auprès du Socle NDC pour recevoir une livraison de ressources IT comme des accès des machines virtuelles, et des accès à des clusters Kubernetes.

5.1.2 - Déploiement de l'infrastructure

La deuxième étape consiste à mettre en place et à configurer les environnements de développement et de production c'est à dire les ressources Kubernetes, les bases de données de production, le registre Artifactory (stockage privé des images Docker), l'instance GitLab (stockage privé des versions du code source), les environnements de travail pour les développeurs, le HashiCorp Vault (stockage des mots de passe de centralisé) et le wiki interne pour centraliser la documentation technique et les services de surveillance (ELK, Prometheus, Grafana).

Des tickets de demandes d'ouvertures de flux sont effectués auprès des équipes réseaux pour s'assurer que les différentes instances, situées dans des sous-réseaux différents communiquent bien ensemble. Un fois l'infrastructure fonctionnelle, il est donc possible de passer à l'étape suivante.

5.1.3 - Mise en place d'une première version de l'application

La troisième étape se concentre sur la livraison d'une première version fonctionnelle de l'application. Celle-ci pourra être testée par le client, qui pourra alors partager son ressenti durant la première revue de sprint consacrée à la livraison des premières fonctionnalités de l'application. Parallèlement, l'équipe des développeurs est recrutée progressivement, sous la supervision des leaders techniques et des recruteurs des Ressources Humaines.

5.1.4 - Mise en place d'une chaine de déploiement continu

La quatrième étape consiste à mettre en place la chaine de déploiement continu sur l'instance GitLab. Cette chaine de validation logicielle doit effectuer plusieurs tâches, comme la construction d'une image Docker dans un environnement d'exécution isolé, tester le code source, générer des images Dockers pour chaque type d'environnements.
Une fois cette opération terminée, il est possible de se concentrer sur l'amélioration continue de l'application, c'est à dire livrer continuellement des correctifs de sécurité, des optimisations ou de nouvelles fonctionnalités conforment aux besoins du client. En même temps, des métriques sont collectées grâce aux *logs* et aux instances chargées de surveiller les performances de l'application. Des tests de performances, d'intégrité et de sécurité sont effectués dans le but d'optimiser le comportement et la disponibilité l'application ainsi que la qualité de son code. De nouvelles tâches sont ajoutées au *backlog* produit, puis diffusées progressivement dans la liste des tâches à réaliser en sprint.

5.1.5 - Livraison continue de l’application

La cinquième étape concerne la clôture du projet dans son aspect fonctionnelle. Celle-ci permet d'attester que le logiciel a été livré, que la chaîne de déploiement continu est fonctionnelle et que le client est satisfait du produit livré. La supervision des améliorations continuellement apportées à l’application est prise en charge par le responsable projet et par l’équipe de développement. Ceux-ci sont chargés d’adapter continuellement le projet aux nouvelles exigences du ou des nouveaux clients potentiels.

5.2 - Suivi des étapes de déploiement

Lors de son cycle de déploiement, le code source de l'application évolue à travers plusieurs environnements avant d'intégrer la production. Les deux premiers environnements sont situés sur un cluster Kubernetes dédiés au développement et aux tests d'intégration. Les deux autres environnements sont situés sur un autre cluster dédié à la production. Ces quatre environnements sont intitulés « Developement », « Integration », « Preproduction » et « Production ».
Ces étapes de déploiement sont réparties sur quatre semaines (deux sprints). Elles sont nécessaires pour que l'équipe puissent prendre le temps d’observer le comportement de l'application et de détecter les éventuels dysfonctionnements. Ces environnements sont représentés par quatre branches principales sur GitLab : les branches intitulées « develop », « integration », « preprod » et « production ». La branche « production » est la branche principale, également appelée par défaut « master ».

Les images générées à partir du code source des branches « develop » et « integration » sont utilisées dans le l’espace de nom Kubernetes dédié au développement. De même, les images générées à partir des branches « preproduction » et « production » sont utilisées dans le l’espace de nom Kubernetes dédié à la production.

Durant la semaine de sprint, le code source de l'environnement « integration » est d'abord fusionné dans la branche principale de l'environnent « preproduction ».
Ce processus est nommé la mise en préproduction (MEPP). La mise en production s’effectue manuellement en changeant le tag de l’image dans la configuration de Kubernetes. Il s’agit de la mise en production (MEP) et intervient lorsqu’aucun dysfonctionnement n’est signalé. Ensuite, dans un second temps, la branche de l’environnement de développement « develop » est fusionnée dans la branche de l'environnement « integration ».
Ce processus est nommé la mise en intégration (MEI). Ce transfert progressif de code source dans les différents environnements est nécessaire pour effectuer la surveillance de l’application. Il s’agit aussi de vérifier qu’elle s’intègre bien avec les outils et les technologies existantes (base de données, services de conteneurs, API externes...)

5.3 - Outils de gestion des tâches

5.3.1 - Interface collaborative

L'outils de gestion de travail utilisé dans le cadre de ce projet est *Jira Software*. Il s'agit d'un logiciel spécialisé dans le suivi des tickets (incidents, bugs, *user stories*). Il est adapté pour des projets agiles. Populaire auprès des équipes de développement logiciel, il peut être utilisé et adapté à tout type de projet dans tout type de domaine. Il offre de nombreuses fonctionnalités permettant de structurer le travail d'équipe agile, comme un *backlog*, un kanban personnalisable, des rapports de performances, des notifications, des commentaires pour les tâches, des priorisations, une vaste gamme de *plugins* ou des contrôles d'accès.

Grâce à cette application, j'ai la possibilité de créer différents types de tickets (des *Epics*, *User stories*, bugs) et d'y associer toutes sortes d'informations qui viennent faciliter le suivi : Un parent (si le ticket est un *Epic*), une description, des *Story Points*, un responsable, une priorité. Je peux également créer des sprints facilement et y associer ces tickets. Le système de couleur du kanban selon les types de stories ou la criticité permet d'en faciliter la compréhension.

5.3.2 - Gestion du Backlog

La première étape consiste à rédiger l'ensemble des tâches à réaliser dans le *backlog* produit de manière la plus exhaustive possible. Cela permet de mettre en évidence les exigences, les besoins des parties prenantes, afin de favoriser une vision du produit qui soit la plus complète possible pour tout le monde. Le Responsable Produit est la personne en charge pour organiser le *backlog* produit du projet. A l'écoute du client, il sait comprendre et analyser ses besoins pour le retranscrire en des termes plus techniques l'information aux développeurs en charge de créer les fonctionnalités. La méthode S.M.A.R.T peut être utilisée pour créer des tâches spécifiques, mesurables et réalisables.

5.3.3 - Approche Kanban

L'approche Kanban, dans la gestion de projet agile, met à disposition un tableau pour aider à visualiser un travail en organisant les tâches dans un espace divisé en plusieurs colonnes. Chaque colonne renseigne l'état de la tâche : « A faire », « en cours », « validation techniques », « validation fonctionnelle », « terminée ». Au fils du sprint, les tâches doivent être déplacées dans la colonne « terminée ».

5.3.4 - Méthode MoSCoW

La méthode MoSCoW est un outil permettant la classification des exigences d'un projet en fonction d'un degré de criticité. Utilisé dans les équipes agiles, cette méthode permet de prioriser les tâches en fonction de plusieurs termes, ce qui facilite la prise de décision. Voici chacun des termes exprimés dans cet acronyme :

- « *Must have this* » pour la lettre M. Il s'agit des tâches critiques qui doivent être traités en priorité. Leur report peut compromettre la réalisation du projet. Leur réalisation n'est pas négociable.
- « *Should have this if at all possible* » pour la lettre S. Ces tâches permettent d'ajouter une vraie valeur ajoutée, contribuant à l'atteinte des objectifs. A la différentes des « Must Have this », le traitement de ces tâches peut être différées dans le temps si besoin.
- « *Could have this if it does not affect anything else* » pour la lettre C. Il s'agit des tâches qui peuvent être traitées en l'absence d'autres tâches prioritaires. Ce sont généralement des tâches de confort qui contribuent à la satisfaction du client, mais qui ne sont pas indispensable pour la réalisation du projet.
- « *Won't have this time but would like in the future* » pour la lettre W. Ces tâches sont exclues du projet, mais elles pourront être réalisées plus tard selon les besoins futurs du projet.

Les lettres O ont été ajoutée afin de former un mot cohérent, ce qui facilite la mémorisation du terme.

5.3.5 - Le planning poker

Lorsque les tâches du *backlog* produit sont réalisées, l'équipe technique, le Responsable Projet et le *Scrum master* se réunissent pour évaluer la charge de travail d'un ticket ou d'une tâche à réaliser. Les points de vue sont confrontés pour permettre d'attribuer une note de complexité à une tâche de la façon la plus précise possible. Si une tâche est jugée trop complexe, elle est redécoupée en *user stories* plus petites.

L'estimation d'une tâche s'effectue de façon collaborative à l'aide de cartes, virtuelles ou physiques. Ces cartes représentent une note de complexité que chaque collaborateur souhaite attribuer. Elles ont généralement une valeur comprise entre les premiers nombre de la suite de Fibonacci (0, 1, 2, 3, 5, 8, 13, 20). La note de complexité, appelée *Story point*, correspond à la moyenne des votes de toute l'équipe. Voici la description classique associée à chaque note.

- 0 : Une tâche ne demande aucun effort, ou elle est déjà terminée,
- 1 : Une tâche ne demande quasiment aucun effort,
- 2 : L'effort à fournir est très faible,
- 3 : L'effort à fournir est moyen,
- 5 : L'effort à fournir est significatif,
- 8 : L'effort à fournir est important,
- 13 : L'effort à fournir est très important,
- Joker (café) : le collaborateur ne souhaite pas estimer la tâche.

Le but de cette réunion est de noter la difficulté des tâches pour répartir de manière équilibrée la charge de travail entre les collaborateurs, tout en considérant la vélocité de l'équipe. Ces réunions peuvent être programmées une fois par semaine, en physique ou en distanciel. Elles sont habituellement programmées avant le démarrage d'un nouveau sprint, lorsque le Responsable Projet ajoute de nouvelles tâches dans le *backlog* produit. Les tâches estimées peuvent ensuite être embarquées dans le sprint suivant.

5.4 - Outils de travail collaboratifs

Les principaux outils de collaboration et de communication utilisés sont les logiciels de la suite Microsoft office 365. Il s'agit de Word, Excel, Powerpoint pour l'élaboration de documents, Teams pour la messagerie instantanée et les groupes de travail. Outlook est utilisé pour la gestion des mails, du calendrier et la planification des évènements quotidiens, comme les réunions.

Dans le cadre de mon projet mené dans une banque de grande envergure, les administrateurs peuvent mettre en place des politiques de sécurité strictes afin de protéger les données échangées au sein du groupe. Avec l'application Teams, plusieurs canaux peuvent être créés selon le besoin, ce qui permet de faciliter l'organisation du travail.

5.5 - Suivi de l'évolution des performances

Grâce à Jira, il est possible de collecter des statistiques qui viennent mesurer objectivement les performances de l'équipe dans l'avancée du projet, comme sa productivité ou sa cohésion. Il s'agit des KPI « *Key Performance Indicators* ».

Les KPI sont des mesures quantifiables utilisées pour évaluer la performance d'un projet ou d'un individu par rapport à des objectifs opérationnels. Ces mesures permettent de suivre les progrès et de déterminer quel(s) domaine(s) ont besoin de support ou d'amélioration. Pour moi, en tant que chef de projet, ils favorisent la prise de décision.

5.5.1 - Les KPI techniques

En méthodologie agile, plusieurs indicateurs clés peuvent être utilisés pour suivre l'évolution d'un projet. La vélocité, par exemple, est une mesure de la quantité de travail que l'équipe peut accomplir pendant un sprint. Elle est généralement exprimée en *story points* ou en heures de travail. Elle est calculée à la fin de chaque sprint en additionnant les points de story des tâches complétées. Cet indicateur est utile pour planifier les prochaines livraisons.

Le graphique de *Burnup* montre la quantité de travail complété et la quantité totale de travail à faire au fil du temps. Il permet de visualiser l'avancement global du projet et de voir si l'équipe est sur la bonne voie pour atteindre ses objectifs.

Le Diagramme de flux cumulatif montre le nombre de tâches dans chaque état (à faire, en cours, terminé) au fil du temps. Il aide à identifier les goulots d'étranglement et les blocages dans le processus de développement.

Le *Lead Time* mesure le temps total entre la demande d'une tâche et sa livraison. Il inclut le temps d'attente et le temps de traitement. Cet indicateur est utile pour évaluer l'efficacité du processus de développement et à identifier les opportunités d'amélioration.

Le taux de réalisation des objectifs de sprint mesure le pourcentage d'objectifs de sprint atteints. Il aide à évaluer l'efficacité de la planification et de l'exécution des sprints.

La couverture de code mesure le pourcentage de code testé par les tests automatisés. Cet indicateur est utile pour évaluer la robustesse des tests et la qualité du code. Il permet de s'assurer que le code est bien testé afin de minimiser les risques de bugs ou d’incidents.

La dette technique permet d'identifier et de gérer les risques associés à un code de mauvaise qualité introduit dans l'application. Pour cela, elle permet à l'équipe de prioriser et de planifier des efforts de refactorisation qui seront pris en compte dans le cadre de l'amélioration continue du développement logiciel. Ces efforts à fournir seront matérialisés par des tâches à compléter lors des sprints.

5.5.2 - Les KPI humains

Les KPI humaines, ou indicateurs clés de performance humaine, sont des métriques utilisées pour évaluer la performance, la satisfaction et le bien-être des membres d'une équipe. Dans un contexte agile, ces KPI favorisent l'identification des facteurs qui affectent le bien-être général des membres de l'équipe, et permet de prendre des mesures pour améliorer leur qualité de vie au travail.

La satisfaction des parties prenantes est également un KPI d'importance. Elle permet notamment de mesurer le niveau de satisfaction des clients, des utilisateurs, ou des managers par rapport au travail de l'équipe, et donc d'identifier les points qui doivent être améliorés. Les méthodes de collecte d'informations traditionnelles sont les enquêtes de satisfaction ou les feedbacks directs.

La satisfaction de l'équipe mesure le niveau de satisfaction des membres de l'équipe par rapport à leur travail et à leur environnement. Cela permet d'identifier les domaines où l'équipe est heureuse et ceux où des améliorations peuvent être apportées. Les méthodes de collecte d'informations comprennent des enquêtes de satisfaction ou les rétrospectives de sprint.

Le taux de turnover mesure le pourcentage des membres de l'équipe qui quittent l'équipe sur une période donnée. Il permet de mettre en évidence les raisons pour lesquelles les collaborateurs quittent l'équipe et d'identifier les domaines nécessitant des améliorations.

L'analyse de la qualité de la communication et de la collaboration permet d'évaluer la qualité de la communication et de la collaboration au sein de l'équipe, de comprendre comment les membres de l'équipe interagissent entre eux et d'identifier les obstacles d'une communication efficace et transparente. Pour cela, des observations participantes, des enquêtes ou les rétrospectives peuvent être utilisées.

La formation continue permet d'évaluer les opportunités de développement des compétences et de formation au sein de l'équipe afin de planifier des montées en compétences. Cette montée en compétence peut être faite en associant temporairement plusieurs personnes dans le cadre d'un travail, ou via des sessions de formation en ligne ou à distance.

5.6 - Mise en place d'un système de veille technologique

Avec la vitesse d'évolution des technologies aujourd'hui, il devient primordial de suivre les innovations techniques et technologiques qui peuvent avoir un impact significatif sur la performance et la sécurité d'un système informatique, afin de rester compétitif. C'est ce qu'on appelle la veille technologique. Celle-ci peut être passive c'est à dire être une accumulation de connaissances, ou bien active, c'est à dire favoriser l'adoption des changements et des nouvelles méthodes de manière soutenue au sein de l'entreprise. La veille technologique se déroule habituellement en trois étapes : l'acquisition d'informations, la transmission ou le stockage de ces informations, et la synthèse des informations collectées.

5.6.1 - Intérêts

Le but principal de la veille technologique est de renforcer la productivité. En effet, elle permet de récolter les informations pertinentes le plus tôt possible, mais aussi de donner une vision d'ensemble sur les évolutions d'un marché, ses acteurs et ses technologies, afin d'éclairer les prises de décision. Son rôle est donc stratégique et indispensable pour la survie de l'entreprise à long terme. La veille technologique permet de :

- Déterminer quelles sont les autres entreprises et ce qu'elles font,
- Détecter les tendances et anticiper les prochaines avancées.
- Accumuler des informations stratégiques fiables et mises à jour en permanence
- Se démarquer des concurrents

5.6.2 - Outils de veille technologique

Une veille technologique peut s'appuyer sur de nombreux outils, que ce soit par des applications installées sur l'ordinateurs ou des services web dédiés. Voici une liste des principaux canaux et outils qui m'ont aidé à récupérés les informations les plus actuelles sur en rapport avec l'approche *DevOps* :

- Des configurations de recherche avancées sur les moteurs de recherche, pour affiner ses recherches selon différents paramètres (format de document, dates, mots-clés...),
- Des systèmes d'alerte dans Google pour suivre des mots-clés précis, afin d'être informé de la parution de nouveaux contenus en rapport avec ces requêtes.
- Des souscriptions à des newsletters pour obtenir un condensé d'informations essentielles sur les sites et blogs,
- Une gestion de flux RSS via des lecteurs pour centraliser la récolte de nouveaux articles publiés à partir de nombreuses sources différentes, comme *Feedly*.
- Des abonnements sur les réseaux sociaux, ce qui permet d'être notifié de l'activité des personnalités et comptes suivis.
- Des participations à des évènements publics comme des séminaires ou des conférences.

5.6.3 - Les revues de presse technico-fonctionnelles

Les revues de presse technico-fonctionnelle sont des réunions qui peuvent être effectuées une fois par trimestres avec l'ensemble de l'équipe. L'objectif principale est de pouvoir proposer une façon collaborative une transmission de nouvelles connaissances entre les membres de l'équipe. En effet, chaque collaborateur peut décider de présenter, sur la base de ses recherches personnelles ou de ses centres d'intérêts, de nouvelles méthodes, innovations ou tendances qu'il a repérées lors d'une veille technologique personnelle.

Le Socle NDC gère la mise en place et la maintenance de tous les services : Vault, Gitlab et Artifactory, les bases de données, les ressources Kubernetes. Ils sont situés dans des sous-réseaux dédiés. Les accès aux clusters Kubernetes sont fournis grâce à des fichiers de configuration déployables sur la machine du client. Ils contiennent les contextes, des composants permettant de spécifier à quel(s) cluster(s) et à quel(s) espace(s) de nom un utilisateur peut accéder. Une allocation de ressources (CPU, mémoires, nombre de Pods, services, configMap ou volumes persistants maximums) y est également assignée.
Un cluster Kubernetes de développement héberge les environnements « Developement » et « Integration ». Le deuxième cluster Kubernetes héberge les environnements « Preproduction » et « Production ». Ces quatre espaces sont isolés les uns des autres, mais peuvent accéder au serveur Gitlab et à l'instance Artifactory, situés dans les autres parties du réseau. Les paramètres de configuration de chaque environnement sont stockés dans les *overlays* du manifest Kubernetes (voir la partie *6.7.2 : Structure du dossier de projet*)

La partie Frontend de l'application est déployée dans un Pod dédié. Celui-ci est le premier service auquel les utilisateurs de l'application accèdent. L'interface web, développée en Angular, offre une interface ergonomique, simple et efficace pour interagir avec la base de données via le backend. Il communique avec le conteneur uWSGI situé dans le Pod Dango.
Le *swagger* est également un conteneur de service web mettant à disposition aux utilisateurs techniquement plus avancés d'autres équipes, une documentation visant à communiquer les points de terminaisons disponibles. Il est ainsi possible de connecter directement sur le conteneur Django des tâches et des scripts d'automatisation externes.

La partie backend de l'application contient trois Pods. L'application Django et le service uWSGI associé, le Pod Redis et le Pod Celery.

Le Pod Django est composé d'un conteneur mettant à disposition le serveur d'APIs Django. Il communique avec le conteneur uWSGI, le Pod Redis, le Pod Celery et la base de données. Le conteneur uWSGI (*Unbit Web Server Gateway Interface*) est un serveur d'application offrant de la rapidité, de l'efficace et de la haute disponibilité dans le traitement de requêtes. Il peut gérer un grand nombre de requêtes simultanées avec une faible latence, ce qui en fait un excellent choix pour les applications web à fort trafic. Il reçoit les requêtes HTTP du serveur web, les transmets au conteneur Django, puis retourne les réponses au format JSON au serveur web afin qu'elles soient affichées au client grâce au frontend.
En combinaison avec Prometheus, il est possible d'extraire de nombreuses métriques comme le temps de réponse des requêtes, l'utilisation du CPU, l'utilisation de la mémoire, le taux d'erreur, le nombre de requêtes, le nombre de connexions à la base de données.

Le Pod Redis est une solution déployée en amont pour répondre au besoin futur des utilisateurs de la haute disponibilité des données. En effet, avec le temps, celle-ci va acquérir un volume d'enregistrement conséquent. Redis mets à disposition un cache de données en mémoire vive qui réduit le temps d'accès en lecture pour les utilisateurs de l'application.

Cependant, son état asynchrone oblige un rafraichissement constant des données par des tâches automatisées de Celery, qui lit les données dans la base, puis les injecte dans le cache.

GitLab est une forge logicielle mettant à disposition de nombreux outils aux développeurs afin d'en faciliter la gestion de leurs codes. Elle est basée sur Git et propose de nombreuses fonctionnalités comme la gestion des versions de code source, un système de branchement, un wiki, des outils de développement continue (GitLab CI/CD, Gitlab Runners, GitLab Pipeline). La plateforme se présente comme une solution open source favorisant la collaboration entre développeurs sur divers projets de logiciels. Elle permet, en effet, d'assurer l'hébergement de tous les projets et favorise l'approche *DevOps*.
Le serveur GitLab stock les différentes versions du code source de l'application, via les quatre branches que nous avons vu précédemment. Chaque espace de nom déploie quatre versions de l'application simultanément sous forme d'applications distinctes et indépendantes.
Les espaces de nom « Developement » et « Integration » déploient les dernières images générées avec le code source des branches « develop » et « integration ». Le tag de ces images est « staging » signifiant que l'application est en cours d'intégration progressive et que celle-ci est suivie grâce à des observations technico-fonctionnelles régulières.
Les espaces de nom « Preproduction » et « Production » déploient les dernières images générées avec le code source des branches « preproduction » et « production ». Le déploiement de la dernière image avec le tag « stable » en production s'exécute manuellement. Les secrets (comme les *tokens*, les identifiants, les mots de passe, les IP ou nom de domaines des services externes) sont récupérés depuis le Vault, puis injectés par Kubernetes dans les conteneurs au moment de la construction des Pods grâce aux variables d'environnements.

Artifactory est un gestionnaire de dépôts d'artefacts open-source développé par JFrog. Il permet de stocker, de gérer et de distribuer des artefacts et des binaires. Les artefacts peuvent inclure des bibliothèques, des paquets, des images de conteneurs, des fichiers de configuration, et d'autres types de fichiers nécessaires pour le développement et le déploiement de logiciels. Artifactory prend en charge de nombreuses technologies de gestion de paquets comme Maven, Gradle, npm, NuGet ou Docker. Doté d'une API REST complète, il s'intègre avec des outils comme Gitlab pour créer et stocker des images de conteneur en lien avec un environnement dédié.

HashiCorp Vault est un outil de gestion des secrets qui permet de stocker, de gérer et de contrôler l'accès aux informations sensibles telles que les mots de passe, les clés API, les certificats, ou d'autres données confidentielles. Vault est conçu pour sécuriser, stocker et contrôler l'accès à ces secrets de manière centralisée et sécurisée. Vault s'intègre facilement avec diverses infrastructures et outils, y compris les systèmes de gestion de configuration, les orchestrateurs de conteneurs, et les plateformes cloud.

Les bases de données relationnelles MySQL s'exécutent sur deux instances Linux distinctes. Elles sont isolées sur des sous-réseaux dédiés, en dehors des clusters.
Communiquant avec le Pod Django et le Pod Celery, elles ne sont pas accessibles directement aux utilisateurs. Une base de données est dédiée pour les environnements de test, une autre pour les environnements de production.

6.3 - Environnements de travail pour les développeurs

Les développeurs de l'application bénéficient d'un environnement virtualisé fonctionnant avec le dernier système d'exploitation LTS d'Ubuntu. Ils se connectent depuis leur poste Windows à un portail web propulsé par un partenaire grâce à leurs identifiants Windows. Ils configurent ensuite leur environnement de travail, en effectuant les mises à jour du système, installent leur éditeur de code préféré et établissent une connexion avec l'instance GitLab. Les privilèges d'administration de la machine leur sont accordés. Ces machines virtuelles Linux sont situées dans des sous-réseaux spécifiques, et disposent des autorisations requises pour se connecter aux différentes serveurs et outils. Les requêtes vers internet sont cependant filtrées par un proxy privé, avec une liste des sites autorisées.

6.4 - Outils de surveillances

6.4.1 - Configuration d'ELK

ELK est un acronyme qui représente une suite de logiciels open-source utilisée pour la collecte, l'analyse et la visualisation des données de logs. ELK est composé de trois principaux composants : Elasticsearch, Logstash, et Kibana. Ensembles, ces outils permettent de gérer efficacement les *logs* et de fournir des *insights* précieux sur les opérations et les performances des systèmes.

Logstash est un pipeline de traitement des données qui collecte, transforme et envoie les logs et les données vers Elasticsearch. Elasticsearch est un moteur de recherche et d'analyse distribué qui stocke et indexe les logs et les données pour améliorer le temps de recherche. Enfin, Kibana est une plateforme de visualisation qui permet de créer des tableaux de bord interactifs pour visualiser et analyser des journaux et des données stockées dans Elasticsearch.

6.4.2 - Configuration de Prometheus et Grafana

Prometheus est un système de surveillance et d'alerte open source conçu pour collecter et stocker des métriques de performance en temps réel. Il est particulièrement bien adapté pour les environnements dynamiques et à grande échelle, comme ceux utilisés dans les architectures de micro-services et les infrastructures *cloud*.

Grafana est une plateforme de visualisation open source qui permet de créer des tableaux interactifs et personnalisés pour la visualisation des métriques et des *logs* provenant de

diverses sources. Grafana intègre bien de nombreuses sources de données, comme Prometheus, Elasticsearch ou InfluxDB.

6.5 - Documentation technique

Dans le cadre de ce projet, un wiki interne partagé est utilisé pour documenter l'infrastructure. C'est à dire lister des informations concernant les serveurs, comme les noms de domaines, les ports disponibles, les adresse IPv4, les systèmes d'exploitation ou les moyens d'accès aux services. Mais aussi, établir des tutoriels pour les nouveaux membres, des procédés techniques, ou décrire de manière informelle l'état des différents services ainsi que leur utilité dans l'architecture du projet. Cet outil présente plusieurs avantages, notamment la facilité d'accès, la collaboration, et la mise à jour en temps réel.

6.6 - Flux des données

6.6.1 - Fonctionnement général

Dans cette partie, je vais expliquer comment les données transmises par les utilisateurs circulent dans l'infrastructure et dans l'application. Une première partie permettra d'en avoir une vision globale. Dans un second temps, nous verrons de façon plus détaillée comment Django opère. Voici ci-dessous une illustration du cheminement des requêtes HTTP dans un contexte global, c'est à dire des utilisateurs authentifiés jusqu'à la base de données.

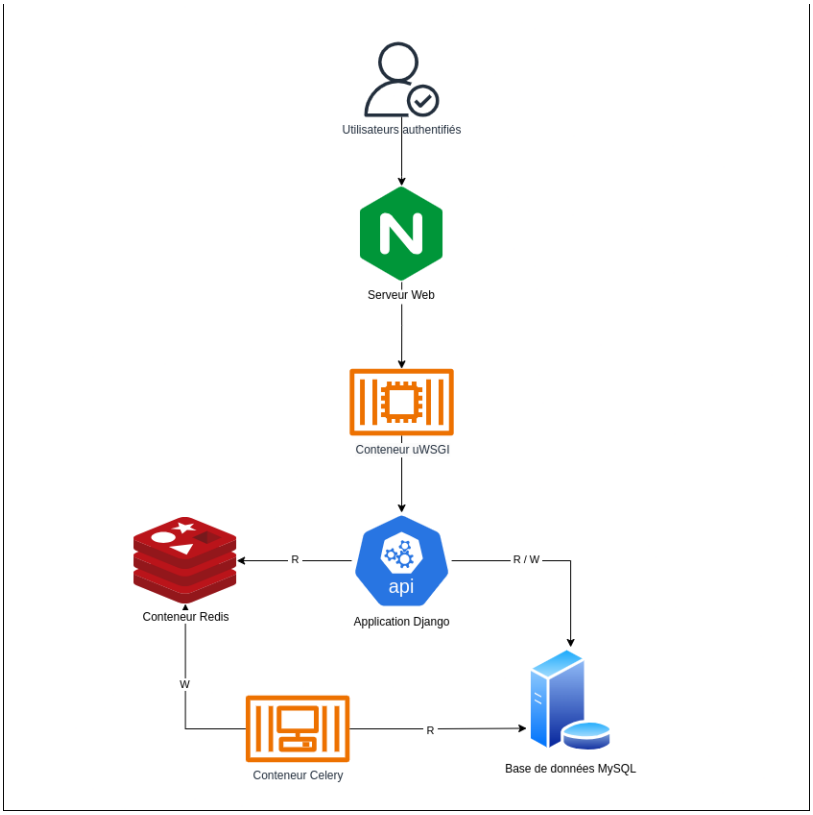

Figure 10 : flux des données

Les utilisateurs authentifiés peuvent ouvrir une session sur le serveur web. Ils reçoivent également un *token* temporaire dans le cache de leur navigateur. Selon les paramètres sélectionnés par les utilisateurs, le serveur web construit et envoie les requêtes HTTP au Pod uWSGI. Celui-ci redirige les requêtes http sur le Pod Django. Ces requêtes contiennent de nombreuses informations, comme les méthodes HTTP utilisées, les en-têtes, le corps du message et le *token*. Celui-ci contient les éléments permettant à Django de déterminer la validité d'une requête. Si c'est le cas, Django lit ou enregistre de nouvelles informations dans la base de données.

Le Pod Celery, non accessible depuis l'extérieur du réseau, récupère les données enregistrées dans la base de données puis les injecte dans le cache du conteneur Redis. Le pod Redis est disponible en lecture seule par Django. Il lui permet d'accélérer la vitesse de récupération des listes d'objets.

6.6.2 - Approche spécifique à Django

Voici une illustration du cheminement des requêtes HTTP dans un contexte spécifique à Django, c'est à dire leur routage, leur contrôle et leur transformation.

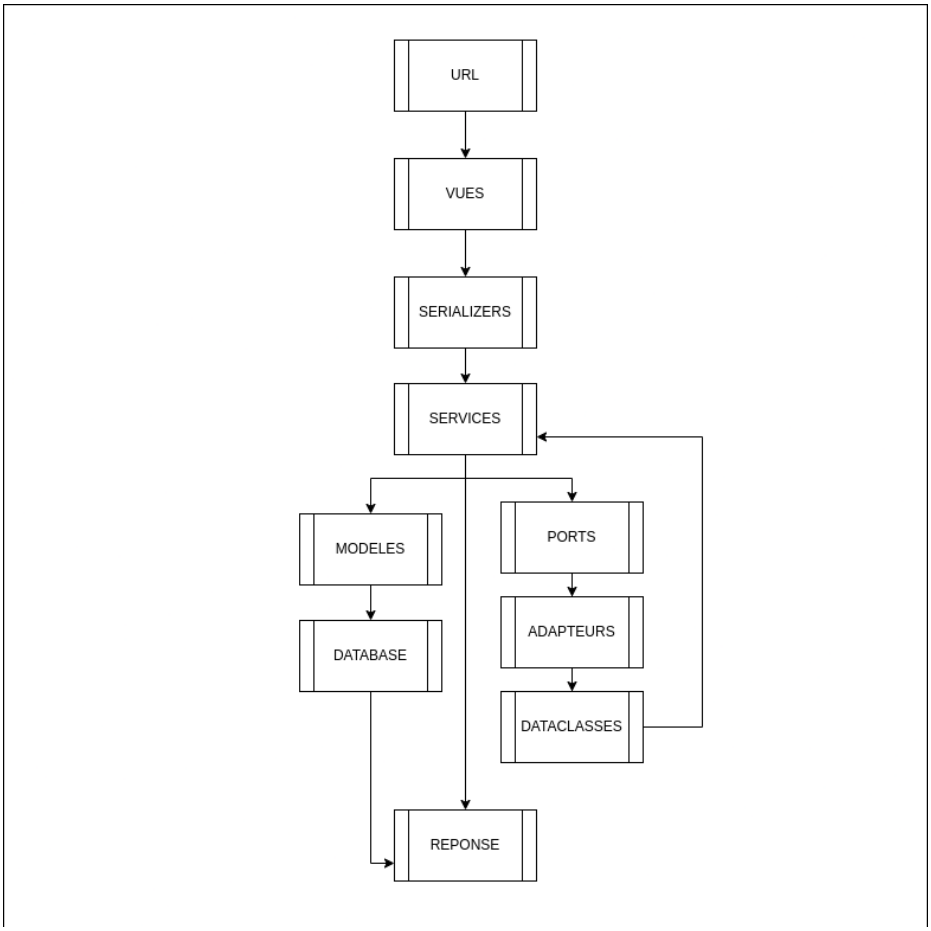

Figure 11 : flux des données dans Django

Les requêtes HTTP sont composées par divers éléments : des en-têtes (un ensemble de données organisées en paire de clé-valeur), une URL cible et une méthode (GET, POST, PATCH et DELETE). Celles-ci peuvent contenir un corps de requête, c'est-à-dire un ensemble de clé-valeur au format JSON utilisé lors de la création ou de la modification d'objets.

Dans un premier temps, la requête atteint l'url du serveur Django. Celle-ci est ensuite routées vers la vue correspondante. C'est à ce niveau que le *token*, qui contient les permissions d'un utilisateur, est déchiffré et analysé par Django.

Dans un second temps, un sérialiseur d'entrée ou de sortie est appliqué aux données selon la méthode utilisée. Le premier formate des objets python en dictionnaire JSON. Le second contrôle les données fournies dans le corps de la requête. Les services appliques des modifications aux données avant de les enregistrer.

6.6.3 - Traçabilité

Lorsqu'une requête aboutie ou lorsqu'elle génère une erreur, Django retourne automatiquement une réponse adaptée, contenant le message et le code HTTP à l'utilisateur.

Un *logger* est placé dans toutes les vues Django. Celui-ci enregistre tous les événements qui se produisent pendant l'exécution de l'application, comme des appels d'API. Les informations rapportées peuvent inclure des messages de débogage, des avertissements, des erreurs, et des informations critiques, des codes HTTP. L'utilisation de *loggers* permet le débogage, la gestion des erreurs, la surveillance, et la maintenance de l'application, mais peut également aider à établir des statistiques d'utilisation.

6.7 - Gestion des manifests Kubernetes

Un manifest Kubernetes est un fichier YAML ou JSON qui décrit comment des ressources Kubernetes doivent être créées ou modifiées. Les manifests sont utilisés pour définir des objets Kubernetes tels que des pods, des déploiements, des services, des ConfigMaps, des secrets... Combiné avec Kustomize, l'infrastructure virtuelle devient très flexible et adaptable selon les besoins rencontrés durant la chaîne de développement.

6.7.1 - Kustomize

Kustomize est un outil de gestion de configuration open-source pour Kubernetes. Il permet de de gérer et de personnaliser des manifests Kubernetes de manière déclarative. Il est conçu pour simplifier la gestion des différentes configurations Kubernetes en permettant aux développeurs de superposer des modifications spécifiques d'un environnement sur une base de configuration commune. Les valeurs définies dans les overlays complètent et/ou

remplacent celles définies dans la base pour les champs spécifiés (secrets, url de l'image docker, nombre de réplicas). Il s'intègre également avec kubectl qui est l'outils en ligne de commande de Kubernetes et offre de nombreux avantages.

Kustomize permet de superposer des configurations spécifiques à l'environnement (comme le développement et la production) sur une base commune de manifests. Cela permet de maintenir une configuration de base et d'appliquer des modifications spécifiques en fonction de l'environnement.

Kustomize peut gérer de nombreux types de ressources Kubernetes, comme les déploiements, les services, les ConfigMaps, les Secrets, les Ingress, les Pods, les Jobs, les PersistentVolumeClaims ou les HorizontalPodAutoscalers.

De plus, il permet d'appliquer des patches (modifications) aux manifests Kubernetes. Cela peut inclure des modifications de valeurs spécifiques, l'ajout de nouvelles sections, ou la suppression de certaines parties des manifests.

Kustomize peut également générer des manifests Kubernetes complets en combinant la base commune avec les superpositions spécifiques à l'environnement. Ces manifests peuvent ensuite être appliqués à un cluster Kubernetes.

Enfin, il s'intègre directement dans kubectl, ce qui permet d'utiliser les fonctionnalités de Kustomize sans avoir à installer d’outils supplémentaires. Des lignes de commandes spécifiques peuvent dont être utilisées pour déployer les ressources à la demande.

6.7.3 - Configurations

Les fichiers de configuration dans le dossier "base" contiennent les informations globales pour tous les environnements. C'est à dire principalement les ressources Kubernetes à déployer comme les Services, les Deployments (Les pods Redis, Celery, le Swagger, le serveur uWSGI et le serveur Django) ainsi que les Ingress, les ResourceQuota et le ServiceMonitor.

Les services Redis et Celery sont déployés sur dans des Pods indépendants. Les services Django et uWSGI sont déployés dans le même Pod. Le type ServiceMonitor est une ressource Kubernetes personnalisée (Custom Resource Definition, CRD) utilisée par Prometheus Operator pour surveiller les services Kubernetes. Prometheus Operator est un outil qui facilite la gestion et la configuration de Prometheus dans un cluster Kubernetes.

Les fichiers de configuration dans les dossiers *overlays* contiennent des paramétrages plus spécifiques aux différents environnements, commes les URLs, les versions des images à déployer, le nombre de réplicas, les metadata et les secrets. Ceux-ci permettant d'accéder aux différents services (Redis, Celery, API de FortiManager et Icinga)

Lorsque les quatre environnements sont déployés et qu'une première version de l'application est disponible, alors il est possible de commencer le développement des premières fonctionnalités en créant de nouvelles branches locales à partir de la branche de développement « develop ».

7 - Développement continu

7.1 - Gestion du backlog

7.1.1 - Création d'une Epic Story

J'ai créé sur Jira une *Epic story* représentant l'ensemble des fonctionnalités de l'application à coder dans un projet dans Django. En effet, les *Epic Stor*ies sont utilisées en méthodologie agile pour désigner une grande initiative ou un objectif à long terme qui ne peut pas être réalisé en un seul sprint. J'ai ensuite créé plusieurs *User Stories*.
Il est ensuite possible d'associer l'*Epic Stor*y aux différentes *User Stories* afin d'en faciliter la visualisation. La description de l'*Epic Story* doit être concise et contextualisée, afin que les autres développeurs puissent comprendre rapidement les intérêts et les enjeux des travaux. Voici par exemple ci-dessous description :

Afin de rationaliser et d'optimiser la gestion des connectivités WAN de l'équipe OER, un référentiel des connectivités doit être mis en place sur une application web pour permettre l'automatisation de tâches d'importances stratégiques, c'est à dire le suivi des états des

connectivités existantes et le suivi des éléments déclencheurs de facturation. Voici les différentes étapes planifiées pour cette *Epic Story* :

1) Améliorer le suivi des états des connectivités VPN
   - Ajuster la facturation au plus proche des besoins du client,
   - Limiter les erreurs,
   - Standardiser la création des connectivités,
   - Obtenir des métriques de performances.

2) Gérer une liste de clients et de contacts associés à ces connectivités
   - Enregistrer des données privées comme des numéros de téléphones,
   - Associer des clients et des contacts aux connectivités,
   - Standardiser la création des différents objets en base de données.

3) Mettre en place des mécanismes de réduction d'erreur
   - Analyse en temps réel des connectivités VPN grâce à FortiManager.
   - Synchroniser le référentiel sur les données de FortiManager.
   - Import et export de masse à partir d'un modèle de CSV.

4) Proposer une interface web simple d'utilisation et intuitive
   - Gérer des objets dans la base de données avec facilités.

7.1.2 - Création des User Stories

L'application est un projet est complexe c'est pourquoi il est nécessaire de diviser les différentes fonctionnalités en plusieurs tâches courtes et réalisables. Ces tâches sont ensuite insérées petit à petit dans le *backlog* de sprint au cours des différentes semaines de sprint. L'objectif, c'est qu'elles soient toutes terminée à la fin de chaque itération de deux semaines. Voici la liste des différentes *User Stories* pour backend, ainsi que leur DoD respectif *(Definition Of Done*) qui délimitant l'ensemble des éléments à implémenter. Le DoD est un ensemble de critères ou de conditions que l'équipe de développement doit satisfaire avant de considérer une *User Story* comme terminée.

US 1 : Démarrage du projet.

DoD :

- Création d'un nouveau projet dans l'application Django,
- Configuration des paramètres de base de l'application,
- Mise en place de l'architecture hexagonale (fichiers et dossiers).

US 2 : Gestion des clients.

DoD :

- Ajout d'un nouveau modèle « Client »,
- Création des points de terminaison CRUD pour gérer les clients,
- Ajouter des permissions d'accès aux vues,
- Réalisation des tests unitaires.

US 3 : Gestion des contacts.

DoD :

- Ajout d'un nouveau modèle « Contact »,
- Création des points de terminaison CRUD pour la gestion des contacts,
- Ajout des permissions d'accès aux vues,
- Réalisation des tests unitaires.

US 4 : Gestion des lignes spécialisées.

DoD :

- Création d'un nouveau modèle « SpecializedLine »,
- Création des points de terminaison CRUD pour la gestion des lignes spécialisées,
- Ajout des permissions d'accès aux vues,
- Réalisation des tests unitaires.

US 5 : Gestion des tunnels VPN.

DoD :

- Création d'un nouveau modèle « VPNTunnel »,
- Création des points de terminaison CRUD pour la gestion des tunnels VPN,
- Ajout des permissions d'accès aux vues,
- Réalisation des tests unitaires.

US 6 : Gestion des mobilités.

DoD :

- Création d'un nouveau modèle « Mobility »,
- Création des points de terminaison CRUD pour la gestion des mobilités,
- Ajout des permissions d'accès aux vues,
- Réalisation des tests unitaires.

US 7 : Création du service.

DoD :

- Ajouter le service permettant d'effectuer la gestion automatique des états des connectivités,
- Effectuer les tests unitaires.

US 8 : Création du connecteur FortiManager.

DoD :

- Création d'un connecteur FortiManager retournant la liste des Tunnels VPN existants,
- Mettre en place des *dataclasses* pour traiter les données récupérées,
- Ajouter le service synchronisant les tunnels entre la base de Django et la base FortiManager,
- Effectuer les tests unitaires.

US 9 : Création du connecteur InfluxDB.

DoD :

- Création d'un connecteur pour retourner les métriques SNMP des différents équipements utilisés par les connectivités,
- Mettre en place des dataclasses pour traiter les données récupérées,
- Effectuer les tests unitaires.

Voici quelques exemples d'*User Stories* pour le frontend :

US 1 : Démarrage du projet.

DoD :

- Création d'un nouveau projet Angular à partir d'un modèle.

US 2 : Gestion des clients.

DoD :

- Création d'un menu dédié à la gestion des clients,
- Création du service (requête HTTP) et du composant (affichage),
- Réalisation des tests unitaires.

US 3 : Gestion des contacts

DoD :

- Création d'un menu dédié à la gestion des contacts
- Création du service (requête HTTP) et du composant (affichage)
- Réalisation des tests unitaires

7.2 - Démarrage d'un nouveau projet

Une fois que les branches des différents environnements sont prêtes, et que les tâches sont définies, il est alors possible de commencer à élaborer les différentes fonctionnalités du projet.
Pour cela, après avoir effectué la mise à jour du dépôt distant sur mon poste de travail local, je commence le développement de l'application « *Connectivity Referential* ». Pour cela, je créé une nouvelle branche sur mon poste local en partant de la branche de développement « develop ». A ce niveau précoce de développement, il s'agit simplement d'une copie de la branche « production » :

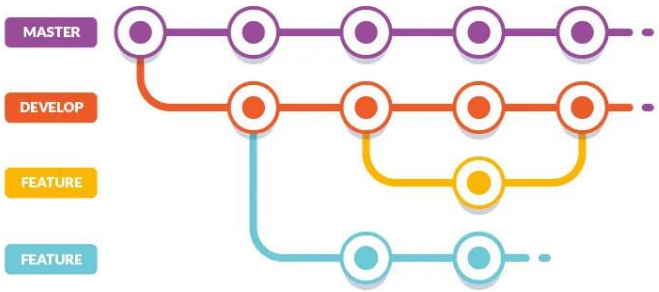

Figure 13 : Workflow git

L'image ci-dessus est une représentation d'un workflow git. Les nouvelles branches de travail sont créées à partir de la branche « develop ». Les conventions de nommage du nom des branches et de l'intitulé des *commits* ont été décidées de façon collaborative, afin d'en favoriser une compréhension rapide. Les branches doivent incorporer le type de fonctionnalité à implémentation, le nom du projet, le numéro de l'US, et l'intitulé de la fonctionnalité ajoutée.
L'ensemble est séparé par le caractère slash "/" ce qui permet d'améliorer la visibilité des différents blocs. Voici un exemple ci-dessous pour la création de la branche de développement. Par exemple :

```
feat/CR-6657/connectivity_referential
```

```
$ cd connectivity_referential
$ git pull
$ git checkout -b feat/CR-6657/connectivity_referential
$ python manage.py startapp connectivity_referential
```

En utilisant ces commandes, mon dépôt local se configure sur une nouvelle branche à partir de la branche de développement. Je vais ensuite pouvoir apporter des modifications, en créant un nouveau projet d'application en utilisant la commande Django *startapp* dans le dossier racine du projet. Différents fichiers de configuration sont automatiquement créés :

- views.py est relatif aux vues
- urls.py est relatif aux URLs de l'application
- models.py est relatif aux objets dans la base de données, appelés modèles,
- app.py permet de définir des paramètres spécifiques de l'application,
- admin.py permet de définir la mise en page des modèles dans l'interface, d'administration web de Django,
- tests.py contient les tests unitaires,
- Le dossier migrations contient un historique en Python de toutes les modifications apportées aux objets dans les modèles.

Je peux indexer les premières modifications apportées par ma branche, générer un premier *commit*, et la publier sur le dépôt Gitlab de l'équipe. Pour cela, dans le dossier du projet CR, je peux exécuter les commandes suivantes :

```
$ git add .
$ git commit -m "feat: initialization of the connectivity referential application "
$ git push --set-upstream origin feat/CR-6657/connectivity_referential
```

7.3 - Mise en place de l'architecture hexagonale

Une fois la branche déployée et l'application initialisée, je vais structurer les dossiers selon un modèle d'architecture hexagonale définit de façon collaborative. Voici un exemple ci-dessous permettant de représenter l'organisation des dossiers et les fichiers de code, un rectangle étant la représentation schématique d'un dossier.

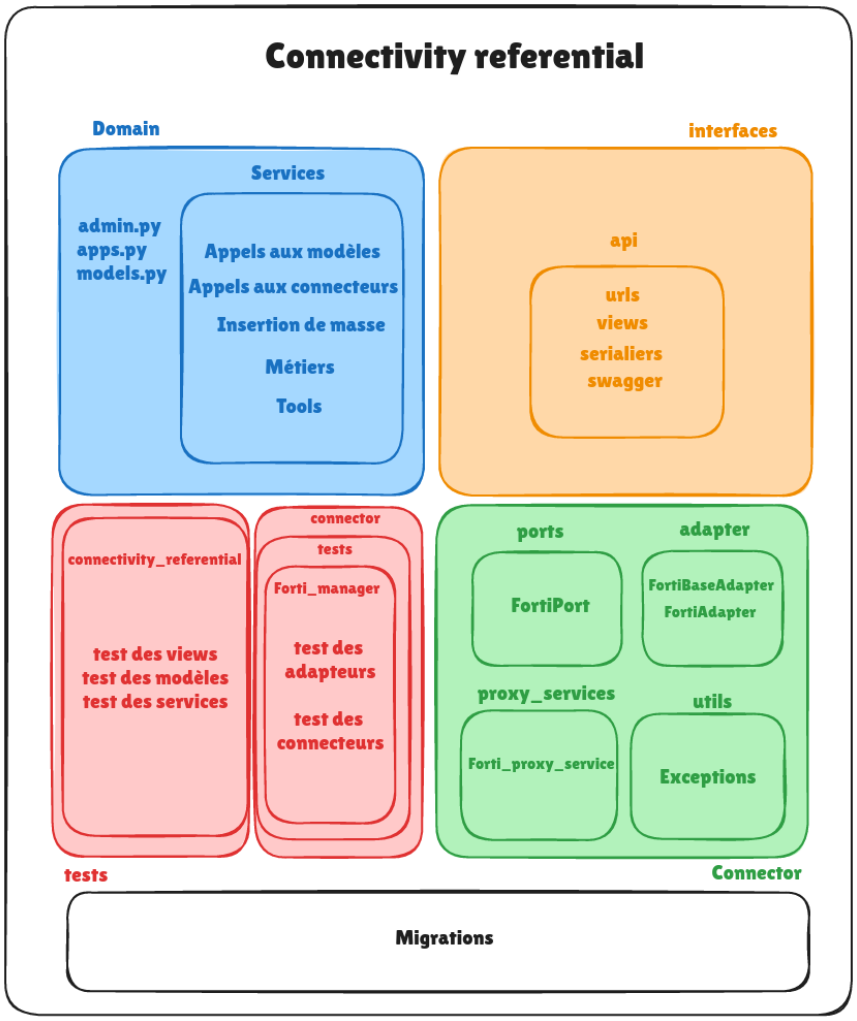

Figure 14 : Architecture du code

Le code de l'application est structuré autour de cinq parties, représentées par des dossiers. Cette disposition permet de favoriser la maintenabilité du code ainsi que sa clarté en l'organisant selon une logique de modules indépendants les uns des autres : le domaine, les interfaces, les connecteurs, les tests unitaires et les migrations.

Le bloc *Domaine* regroupe les fichiers de configuration, les modèles et les services. Les services sont des éléments très modulables (des classes ou des fonctions) qui répondent aux besoins du client. Ils traitent la logique métier. Ils contactent les modèles, et/ou aux connecteurs.

Le bloc *Interfaces* regroupe les modules exposés aux utilisateurs externes de l'application. Il assure le traitement et le routage sécurisé des données contenues dans le corps des requêtes HTTP des utilisateurs vers la base de données. Le code stocké est associé aux vues, aux URLs, aux sérialiseurs et au *swagger*.

Le bloc *Connector* regroupe les parties du code chargées d'établir des connexions aux outils externes. Structurés en port et adapteurs pour une meilleure adaptabilité dans le temps. Appelés par les services, ils leur retournent les données brutes reçues. Un contrôle sur la structure des données reçues est effectué grâce aux *dataclasses* de Python.

Le bloc Tests regroupe l'ensemble des tests unitaires permettant d'attester l'intégrité, la qualité et le bon fonctionnement des différents modules du code de l'application dans le temps. En effet, ils font offices de constat en cas de litige. Chaque module est testé séparément dans de nombreuses situations fonctionnelles, et doit avoir le comportement attendu en toutes circonstance. Pour une meilleure vision et une plus grande sécurité, la couverture doit être la plus large possible.

Enfin, le bloc *Migrations* regroupe l'ensemble des fichiers spécifiques que Django utilise pour configurer la base de données. Il s'agit d'un historique contenant l'ensemble de toutes les modifications qui ont été apportés aux modèles et donc, à la structure de la base de données.

7.4 - Développement des nouvelles fonctionnalités

7.4.1 - La méthode TDD

Le *Test-Driven Development* (TDD), ou développement piloté par les tests, est une méthodologie de développement logiciel qui repose sur l'écriture de tests avant l'écriture du code fonctionnel. Il s'agit d'une approche modulaire visant à améliorer la qualité du code dans son ensemble, le code étant finalement conçu de manière progressive et ordonnée. Dans le développement du test, plusieurs questions doivent-être posées :
Quel est l'objectif de ce test ? De quoi mon test a-t-il besoin pour pouvoir fonctionner ?

En effet, avec cette méthode, l'écriture du code est orientée par les besoins fonctionnels du test. Par exemple, les échecs répétitifs du test me permettent de cibler les éléments manquants, comme une URL, une vue ou un modèle. Analyser les retours permet ensuite d'effectuer les modifications dans le code de manière progressive.

Si je souhaite mettre en place un nouveau point de terminaison, les échecs successifs du test vont me guider jusqu'à ce que tous les modules soient présents et opérationnels (c'est à dire les URLs, les vues, les sérialiseurs, les services métiers et le retour attendu).

7.4.2 - Les modèles

Les modèles sont des objets Python essentiels définis dans les fichiers « models.py ». Ils définissent les tables de la base de données et les relations entre elles ainsi que leurs attributs. De plus l'ORM de Django (Object-Relational Mapping) ajoute une couche d'abstraction qui permet aux développeurs d'interagir plus facilement et de façon plus sécurisée avec la base de données, en utilisant le langage Python natif plutôt que des requêtes SQL parfois longues et complexes.
Voici quelques rôles que les modèles gèrent généralement :

- Ils structurent les tables de données de l'application,
- Ils interagissent directement avec la base de données,
- Ils sécurisent l'envoie et la validation des données,
- Ils automatisent la production de formulaire.

7.4.3 - Les URLs

Les URLs de Django sont définies dans les fichiers de configuration appelés « urls.py ». Hautement personnalisables, elles prennent en charge le routage des requêtes HTTP vers les vues appropriées auxquelles elles sont associées.

7.4.4 - Les vues

Les vues jouent un rôle central dans le modèle MVC (*Model-View-Controller*) de Django. Définies dans les fichiers « views.py », elles sont chargées de recevoir, contrôler, traiter le flux des données qu'elles reçoivent et de renvoyer la réponse appropriée au format JSON. A la fois très flexibles et adaptables, elles peuvent être basées sur des fonctions ou des classes. Django fournit des vues génériques pour simplifier la gestion des tâches courantes des développeurs, comme l'obtention d'une liste d'objets, la suppression ou l'envoie de formulaires.
Les vues peuvent traiter différents types de requêtes HTTP (GET, POST, PUT, PATCH, DELETE). Elles peuvent également prendre en charge des permissions d'accès ou être surchargée avec des fonctionnalités supplémentaires, comme un *logger*, un sérialiseur, ou des fonctionnalités ORM (gestion d'objets).

7.4.5 - Les sérialiseurs

Dans Django, les sérialiseurs sont des composants utilisés pour convertir des objets Python, comme des instances de modèles, en formats de données plus simples, comme JSON ou XML, et inversement. Il est également possible de définir plusieurs types de sérialiseur : des sérialiseurs de sortie, permettant de mettre en forme les données renvoyées en réponse d'une requête. Mais également des sérialiseurs d'entrée qui permettent de contrôler la conformité des données entrantes, avant de les traiter.
Définis dans les fichiers « serializers.py », ils sont particulièrement utiles dans le contexte des API RESTful, où les données doivent être échangées entre le serveur et le client sous forme de représentations sérialisées.

Le sérialiseur de sortie transforme un objet ou une liste d'objets python en données structurées au format JSON, puis les retournent en réponse à l'utilisateur.
Ce sérialiseur-ci s'applique lorsqu'une requête GET est effectuée sur cet objet spécifique.
Le sérialiseur d'entrée détermine quels sont les champs obligatoires à remplir dans le cadre d'une création ou d'une modification d'objet avec la méthode POST et PATCH.

Un service est appliqué sur les données pour modifier certains champs automatiquement avant leur enregistrement en base. Selon la méthode utilisée par la requête http (POST ou PATCH), la méthode « *create* » ou « *update* » du sérialiseur sera surchargée avec un service. Les paramètres passés à la méthode et au service sont les données d'entrée validées par le sérialiseur (ainsi que l'objet ciblé par la modification dans le cas d'un PATCH).

7.4.6 - Les services

Les services sont un ensemble de classes et de méthodes très spécifiques qui viennent apporter des modifications sur les données avant de les enregistrer en base. Très modulaires, ils répondent directement à un besoin fonctionnel spécifique du client en effectuant des modifications sur les données à enregistrer.

Dans le cas de mon projet, par exemple, le service « *update_connectivity_state* » est une méthode statique de la classe « *ConnectivityServices* ». Il permet à l'application d'automatiser la gestion du statut d'une connectivité VPN selon des paramètres passés dans le corps de la requête HTTP. Ce système améliore donc le suivi de chaque connectivité VPN, en repérant d'une part les évènements qui déclenchent une facturation, d'autre part en mettant à jour les attributs de l'objet avant de l'enregistrer dans la base de données.

7.4.7 - Les connecteurs

Les connecteurs sont les parties du code chargées de se connecter aux différents outils externes disponible sur les réseaux internes. Appelés par les services, ils leur retournent les données extraites dans un dictionnaires brute.

Conçu selon une architecture en ports et adapteurs, ils apportent plus de flexibilité dans le temps, favorisant la maintenance à long terme. Si un connecteur change, le service instancie simplement le bon adapteur, avec des méthodes universelles définies dans le Port. Ainsi, un changement au niveau des connecteurs, comme une montée de version, nécessite simplement l'implémentation du bon adapteur. Les adapteurs peuvent donc continuer à fonctionner ensemble.

Le Port est une classe abstraite qui définit quelles méthodes doivent être implémentées dans les Adapteurs. De cette façon, chaque adapteur dispose d'une interface commune qu'il est possible de surcharger spécifiquement.

Ainsi, si une mise à jour change le comportement d'un outil externe, il suffit de mettre à jour ou de créer un nouvel adapteur. Le problème s'est localisé au niveau du connecteur.

7.4.8 - Les dataclasses

Dans le contexte de ce projet Django, les *dataclasses* sont utilisées pour définir des modèles de données attendues lorsque celles-ci sont retournées par les connecteurs.
En effet, les connecteurs effectuent des requêtes spécifiques sur des points de terminaisons d'outils externes. Ils retournent ensuite ces données brutes (sans transformations) au service émetteur formatées en JSON.
Les dataclasses, intermédiaires situées entre les services et les connecteurs, jouent le rôle de sérialiseurs d'entrée. C'est à dire qu'elles viennent dans un premier temps, transformer ces données en dictionnaire, puis dans un second temps, attester (ou non) que ces dictionnaires possèdent bien les clés et les types de valeurs attendus.
Lorsque les données ne correspondent pas au modèle attendu, Django lève automatiquement une erreur, ce qui permet de faciliter la détection de l'emplacement de l'erreur si elle est liée aux connecteurs.

7.4.9 - Les permissions

Les permissions déterminent quel(s) groupe(s) d'utilisateur peuvent utiliser les points de terminaison des API. Lorsqu'un utilisateur n'a pas d'autorisation d'accès, le serveur Django peut lui retourner plusieurs code HTTP d'erreur selon les situations rencontrées :

L'erreur 401 indique qu'un utilisateur souhaite effectuer une requête HTTP sur un point de terminaison de l'application sans présenter le *token* temporaire, normalement attribué après avoir effectué avec succès une demande d'authentification. Ce *token* est enregistré dans le cache du navigateur web, et confirme que l'utilisateur existe bien dans *l'Active Directory* de l'entreprise.

L'erreur 403 indique qu'un utilisateur authentifié souhaite effectuer une requête HTTP sur point de terminaison spécifique, mais sans pour autant avoir les droits de lectures et/ou d'écriture sur l'objet spécifique. Cela arrive s'il n'est pas enregistré dans au moins un groupe Django ayant ces attributions.

Par défaut, les permissions d'accès sont automatiquement créées par Django pour chaque nouveau modèle. Il faut ensuite les ajouter manuellement aux groupes et aux utilisateurs depuis l'interface d'administration.
Il est également possible de les personnaliser pour faciliter leur gestion à long terme, surtout lorsque les demandes d'autorisation à gérer quotidiennement sont nombreuses. Ces permissions spécifiques sont déclarées dans la sous-classe Meta du modèle Django concerné.

Attribuer des autorisations à des groupes calqués sur ceux de l'Active Directory, plutôt que sur des utilisateurs standards, permet de faciliter et d'optimiser cette gestion à long terme.

7.4.10 - L'interface d'administration

Le fichier « admin.py » dans le projet Django est utilisé pour personnaliser certains éléments de l'interface d'administration de Django, comme la gestion de l'affichage des modèles et de leurs propriétés, la gestion des permissions, les utilisateurs ou les groupes. En effet, par défaut, Django met à disposition une interface web facile d'utilisation et conviviale pour gérer tous les objets de la base de données de façon manuelle.

7.4.11 - Les tests unitaires

Les tests unitaires permettent d'une part de vérifier que chaque module du code de l'application (une vue, une méthode, une fonction, un service) fonctionne de manière attendue dans un cas spécifique, mais également d'anticiper d'éventuels dysfonctionnements ou de failles de sécurités potentiels pour mieux les corriger.

Plus les différentes parties de l'application sont couverts par les tests, plus l'intégrité et la fiabilité de son code augmente. C'est pourquoi chaque URLs, modèles, vues et services sont testés sous différents cas d'utilisation pour attester que l'application répond bien de la façon attendue. Seuls les appels aux outils externes, via les connecteurs, sont ignorés (ou *mockés)* pour éviter une éventuelle surcharge des machines. Une réponse type est alors automatiquement retournée à l'adapteur en fonction des besoins fonctionnels du tests.

Les tests basés sur les vues suivent le même modèle : des requêtes HTTP sont construites puis rejouées dans les différents points de terminaison disponibles, en faisant varier certaines données dans le corps de la requête, les méthodes, les en-têtes. Les réponses et les codes HTTP retournés par l'API sont récupérés, puis comparés. Le test est réussi lorsque ces données correspondent à chaque fois à celles attendues.
Les tests basés sur les modèles se concentrent sur la création d'objet, et attestent que les contraintes de création ou de modification du modèle ont bien été respectées. C'est à dire le type de variable attendu pour chaque attribut, les critères d'unicités de certains champs, la relation avec des clés étrangères.
Les tests basés sur les services attestent que les données sont bien transformées avant leur enregistrement en base si les conditions sont remplies.
Enfin, les tests sur les connecteurs valident que les données récupérées des outils externes sont bien routées vers les services.

7.5 - Validation d'une User Story

7.5.1 - Chaîne de déploiement CI/CD

La pipeline CI/CD est configurée sur GitLab. Elle automatise les processus de développement, de test, et de déploiement. Elle permet aux développeurs d'effectuer sur leurs modifications des tests automatisés de sécurité et de qualité. Ces tests sont effectués dans des environnements isolés, à partir d'une image de test générée avec le nouveau code. Si toutes les étapes sont validées, le code est fusionné dans le dépôt central, ce qui génère une nouvelle image déployée automatiquement par Kubernetes.

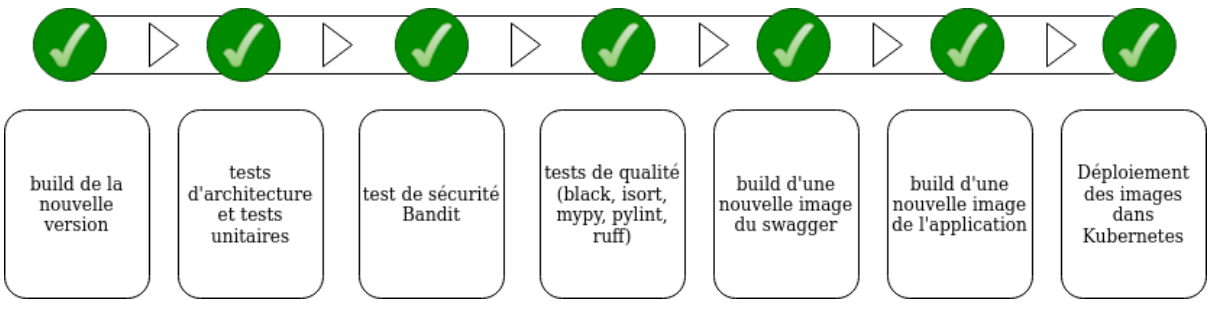

Figure 15 : Chaine de déploiement

7.5.2 - Revue de code

La revue de code est essentielle dans le cadre d'une amélioration continue. Elle consiste à faire examiner le code par d'autres développeurs avant de le fusionner (*merge*) dans la branche principale du dépôt de code. Ceux-ci peuvent ouvrir des points de discussion et proposer des éléments d'amélioration qu'il est nécessaire de prendre en compte. Cette pratique est bénéfique et offre de nombreux avantage :

- Amélioration de la qualité
- Partage de connaissance
- Réduction des risques
- Responsabilisation collective.

Lorsque tous les points bloquants ont été résolus (*threads)*, que les tests de la chaîne de déploiement sont validés, et que le code est approuvé par au moins deux développeurs, la branche de développement peut être fusionnée avec une plus grande assurance.

8 - Sécurité

8.1 - Infrastructure

8.1.1 - Réseau

Sécuriser un réseau étendu en utilisant des VLAN (*Virtual Local Area Networks*) et des pares-feux est une stratégie efficace pour protéger les données et les communications. Les VLANs permettent de segmenter le réseau en plusieurs sous-réseaux virtuels, isolant ainsi différents services ou types de trafic. Cela permet de limiter la propagation des menaces et d'améliorer la gestion du réseau.

Les pares-feux, quant à eux, agissent comme des barrières de sécurité en filtrant le trafic entrant et sortant selon des règles prédéfinies. Ils peuvent bloquer les accès non autorisés, détecter et prévenir les intrusions, et surveiller les activités suspectes.

Les sondes IDS (*Intrusion Detection System*) et IPS (*Intrusion Prevention System*) se placent à des points d'entrée ou de sorties critiques du réseau (comme les passerelles, les pares-feux, sur les serveurs, ou les points d'accès WIFI) et permettent d'analyser le trafic pour détecter et prévenir les activités malveillantes. Les sondes IDS fonctionnent principalement en mode passif, surveillant le trafic réseau pour identifier des comportements suspects ou des signatures d'attaques connues, et alertant les administrateurs en cas de détection alors que les sondes IPS opèrent en mode actif. Elles détectent non seulement les menaces, mais prennent aussi des mesures immédiates pour les bloquer, comme le blocage du trafic malveillant ou la réinitialisation des connexions suspectes.

En combinant ces trois technologies, on peut créer un environnement réseau plus sécurisé, où les données sensibles sont protégées et les risques de cyberattaques considérablement réduits.

8.1.2 - Accès

La sécurisation des accès aux ressources internes, comme des services ou des applications destinées aux employés, est importante pour protéger les données sensibles et prévenir les accès non autorisés. L'utilisation de moyens d'authentification forts, comme l'authentification multifactorielle (MFA), ajoute une couche supplémentaire de sécurité en exigeant plusieurs formes de vérification d'identité, telles qu'un mot de passe et un code envoyé sur un appareil mobile.

Une autre approche efficace mise en place dans la structure est l'implémentation d'un système de Single Sign-On (SSO) basé sur les utilisateurs de l'*Active Directory*. Le SSO permet aux utilisateurs de s'authentifier une seule fois pour accéder à plusieurs applications et services, réduisant le nombre de mots de passe à gérer, minimisant ainsi les risques de compromission. En intégrant le SSO avec l'AD, il est possible de centraliser la gestion des identités et des accès, appliquer des politiques de sécurité globale, et surveiller les activités de connexion pour détecter et répondre rapidement aux comportements suspects.

En cas d'un besoin d'accès aux ressources de l'entreprise à distance, comme en télétravail, l'utilisation d'une connexion VPN (*Virtual Private Network*) est un moyen toujours très efficace. En effet, le VPN chiffre toutes les données échangées entre l'appareil de l'utilisateur et le réseau de l'entreprise, protégeant ainsi les informations sensibles contre les tentatives d'interception. De plus, il permet aux travailleurs à distance d'accéder aux ressources internes de l'entreprise, telles que les applications, les fichiers et les services, comme s'ils étaient physiquement présents au bureau. Cette technologie est particulièrement utile lorsque les employés se connectent depuis des réseaux Wi-Fi publics ou non sécurisés.

8.1.3 - Serveurs

La sécurisation des serveurs liés au bon fonctionnement de l'application est essentielle pour protéger les données sensibles et assurer la continuité des services. Plusieurs moyens peuvent être mis en œuvre pour renforcer leur sécurité :

- Maintenir le système d'exploitation et les applications à jour avec les derniers correctifs de sécurité.
- Utiliser des pares-feux pour filtrer le trafic entrant et sortant en bloquant le trafic non autorisé.
- Centraliser des logs en collectant et en analysant les journaux de tous les serveurs dans un système centralisé, comme un SIEM (Security Information and Event Management),

les administrateurs peuvent détecter rapidement les activités suspectes et répondre aux incidents de sécurité de manière proactive.

- Implémentation de contrôles d'accès stricts, l'utilisation de l'authentification multifactorielle (MFA) sur Windows, ou l'utilisation d'une clé SSH avec passe phrase générée avec des protocoles de chiffrement robuste et performant comme l'ed25519.

8.1.4 - Contrôleur de domaine

La configuration de certains rôles du contrôleur de domaine principal est nécessaire pour assurer la sécurité du système informatique de l'entreprise. Un aspect essentiel de cette configuration est la création et la gestion des groupes selon le modèle AGDLP (*Accounts*, *Global groups*, *Domain Local groups, Permissions*). Conseillé par Microsoft, ce modèle permet de structurer les permissions de manière organisé et sécurisé. Voici une liste qui permet de comprendre son intérêt :

- Intégrer des comptes utilisateurs dans des groupes globaux,
- Ajouter ces groupes globaux à des groupes locaux de domaine (DL),
- Assigner des permissions spécifiques à ces groupes locaux. Ainsi, les administrateurs peuvent gérer les accès de manière granulaire et cohérente. Cette approche facilite la gestion des permissions, réduit les risques d'erreurs de configuration, et améliore la traçabilité des accès.

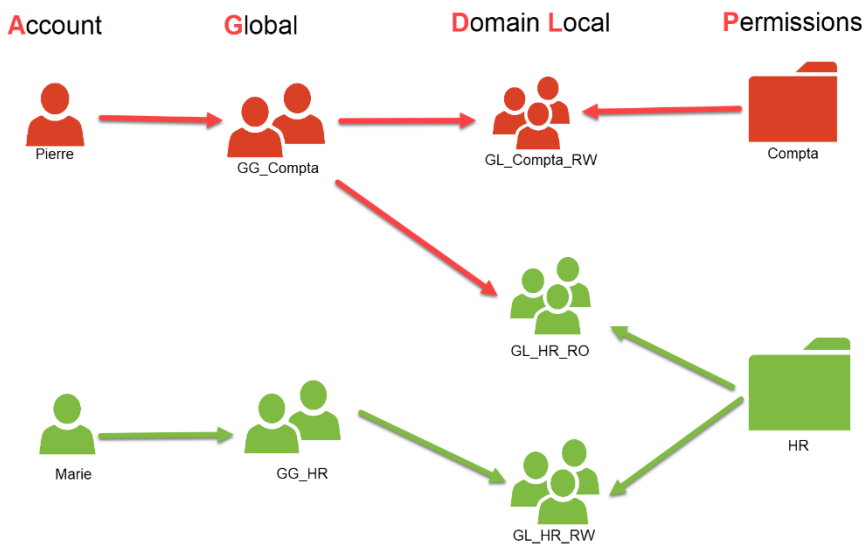

Figure 16 : Représentation visuelle de la méthode AGDLP

8.1.5 - Utilisation d'un proxy

L'utilisation d'un proxy offre une combinaison de sécurité, de performance et de flexibilité qui en fait un outil précieux pour la gestion du trafic Internet. En effet, il peut filtrer le contenu et bloquer les sites web malveillants ou inappropriés, contribuant améliorer la sûreté de la navigation. En termes de performance, il peut également mettre en cache les pages web fréquemment consultées, réduisant le temps de chargement et économisant ainsi la bande passante. Pour les entreprises, un proxy permet de centraliser et de surveiller le trafic Internet, facilitant l'application des politiques de sécurité et la conformité aux réglementations.

8.2 - Application

8.2.1 - Normes de codage

La sécurisation du code Python permet de prévenir les vulnérabilités et assurer la robustesse de l'applications. Plusieurs outils et pratiques peuvent être mis en œuvre pour atteindre cet objectif. Les normes de codage de Python PEP8 permettent d'avoir avec soi une liste des bonnes pratiques. De plus, l'utilisation de mypy permet de vérifier les types des variables, aidant à détecter les erreurs de typage et à améliorer la qualité du code. Pylint est un autre outil d'analyse de code qui permet d'identifier les erreurs de programmation, les mauvaises pratiques et les potentielles vulnérabilités de sécurité. SonarQube peut être utilisé pour effectuer des audits de code complets, détectant les failles de sécurité et les problèmes de qualité. Les tests unitaires jouent également un rôle important en validant le comportement du code et en assurant que les modifications n'introduisent pas de nouvelles vulnérabilités. Enfin, il est impératif de ne jamais inclure de mots de passe ou d'informations sensibles en dur dans le code. L'utilisation de variables d'environnement ou de gestionnaires de secrets comme le Vault d'HashiCorp permet de protéger ces informations sensibles.

8.2.2 - Sécurisation des postes de travail

Différentes méthodes existent pour sécuriser un poste de travail. Cela permet de protéger les données sensibles et prévenir les accès non autorisés. Par exemple, le chiffrement du disque dur, avec BitLocker sur Windows ou FileVault sur macOS, assure que les données stockées sont illisibles sans la clé de déchiffrement appropriée, même en cas de vol ou de perte de l'appareil. Interdire l'utilisation des clés USB non autorisées via des politiques de sécurité empêche les transferts de données non contrôlés et réduit les risques d'introduction de logiciels malveillants. Une politique stricte de gestion des mots de passe, incluant des exigences de complexité, des changements réguliers et l'utilisation de l'authentification multifactorielle (MFA), renforce la sécurité des accès.

8.2.3 - Gestion des bases de données

La mise en place d'un système de sauvegarde permet de prévenir la perte de données en cas de défaillance matérielle, de corruption de données ou d'attaques malveillantes. MySQL offre plusieurs méthodes pour réaliser des sauvegardes, telles que l'utilisation de l'outil mysqldump pour créer des fichiers de sauvegarde SQL, ou l'utilisation de mysqlhotcopy pour des sauvegardes rapides de tables MyISAM. Pour les bases de données volumineuses, des outils comme Percona XtraBackup ou MariaDB Backup permettent des sauvegardes incrémentielles et différentielles pour les moteurs de stockage InnoDB et XtraDB.

La restauration, quant à elle, peut être effectuée en important les fichiers de sauvegarde SQL via l'utilitaire mysql ou par le biais d'autres outils spécifiques pour les sauvegardes incrémentielles. Tester régulièrement les procédures de restauration permet de s'assurer que les sauvegardes sont complètes et que les données peuvent être récupérées en cas de besoin.

Une sauvegarde complète de chaque base de données est effectuée de façon automatique via un service Ansible une fois par semaine, le dimanche. L'emplacement des sauvegardes respecte la « règle des 3-2-1 » (C'est à dire trois copies de données, effectuées sur au moins deux types de supports différents dont un en hors-ligne). Les données sont transférées soit sur des disques dur mécanique soit sur des mémoires SSD depuis une instance placée dans un sous-réseau logique isolé. La base de données de production est sauvegardée de manière complète tous les soirs à partir de 10 heures (de façon transactionnelle)

Enfin, des exercices de restauration de base de données sont mis en place afin de vérifier si les sauvegardes stockées sont complètes, cohérentes et récupérables une fois par semaine. Ils aident à identifier et corriger les éventuels problèmes dans les procédures de sauvegarde et de restauration, assurant que le processus est fiable et efficace. Ces exercices permettent également de former le personnel aux procédures de récupération, réduisant ainsi le temps de réponse en cas de panne réelle.

8.2.4 - Mise en place d'un support utilisateur

Un support utilisateur efficace permet de résoudre rapidement les problèmes techniques, de répondre aux questions des utilisateurs et de fournir des conseils sur l'utilisation de l'application. Cela contribue à réduire les temps d'arrêt et à minimiser les interruptions, assurant ainsi une productivité continue. De plus, un support utilisateur réactif et compétent renforce la confiance des utilisateurs dans l'application. Il est également possible de recueillir des feedbacks utiles dans le cadre de l'amélioration continue.

8.2.5 - Assurances cyber

Les cyberattaques peuvent entraîner des pertes de données sensibles, des interruptions de service, des violations de la vie privée des clients et des coûts élevés de récupération. Une assurance cyber offre une couverture financière pour les coûts liés à la réponse à l'incident, la restauration des systèmes, la notification des parties prenantes et les éventuelles poursuites judiciaires. Elle permet de se concentrer sur la reprise rapide des opérations sans être paralysées par les coûts imprévus. En intégrant une assurance cyber dans la stratégie de gestion des risques, il est possible de renforcer la résilience face aux menaces cybernétiques et protéger leur réputation et leur viabilité à long terme.

8.2.6 - KPI de sécurité

KPI Toutes les mesures citées plus haut visent à améliorer significativement les métriques de l'application liées à la sécurité. En effet, ils permettent de mesurer et d'évaluer l'efficacité des mesures de sécurité en place, d'identifier les domaines nécessitant des améliorations et de garantir une protection continue contre les menaces de sécurité. En surveillant régulièrement ces indicateurs, je peux renforcer la posture de sécurité de mon application et réduire les risques associés aux cyberattaques et aux incidents de sécurité. Parmi ces KPI, je peux citer en citer quelques-unes particulièrement importantes, comme le nombre d'Incidents de sécurité détectés, le temps moyen de réponse (MTTR), le nombre de vulnérabilités découvertes et le taux de correction des vulnérabilités.

8.3 - Plan de réponse aux incidents

Le plan de réponse à une cyberattaque est un document stratégique dans la sécurité informatique de l'entreprise. Il détaille des rôles à tenir ainsi que des procédures à suivre en cas d'incident de cybersécurité (intrusion, fuite de données ou une attaque par ransomware). Ce plan permet de minimiser les dommages en fournissant des instructions claires et immédiates pour identifier, contenir, éradiquer et récupérer les services après une attaque.

8.3.1 - Préparation

Les ingénieurs *DevOps* en collaboration avec le Socle NDC et la SSI sont chargés de mettre au point une équipe de réponse. L'équipe de réponse obtient régulièrement des formations sur les procédures et les bonnes pratiques à adopter. Elle maintient à jour et assure la disponibilité des outils de détection, d'analyse et de récupération. De plus, elle est chargée d'effectuer des tests réguliers et des simulations d'incidents pour évaluer et si besoin, améliorer le plan.

8.3.2 - Détection et Analyse

L'équipe de réponse doit mettre en place des systèmes de surveillance pour détecter les activités suspectes. Des alertes automatiques peuvent être configurées pour cibler des

comportements anormaux. Les incidents détectés sont ainsi analysés et leur impact est déterminé.

8.3.3 - Contenir la menace

En cas de compromission, la première bonne pratique est d'isoler les systèmes affectés pour empêcher la propagation de l'attaque. Des contrôles temporaires sont mis en place pour limiter les dégâts (désactivation des comptes compromis, isolement d'un sous-réseau, débranchement d'une machine). Toutes les décisions prises ainsi que toutes les actions entreprises sont documentées.

8.3.4 - Éradication de la menace

La nature de l'incident (attaque ou menace) est identifiée et éliminée (suppression du code, du logiciel ou de la bibliothèque malveillante). Le système impacté est nettoyé. Les points d'entrée et les angles d'attaques sont listés de façon exhaustive. L'ensemble des mots de passe ou des secrets associés à des comptes ou des services initialisés depuis la machine compromise sont changés.

8.3.5 - Récupération

L'ensemble des paramètres ont été restaurés. La configuration est restaurée à partir d'une source de données fiable et non impactée. Les systèmes sont testés pour s'assurer qu'ils fonctionnent correctement. Enfin, les opérations reviennent à la normale de manière progressive et contrôlée.

8.3.6 - Post-Incident

L'équipe de réponse rédige un rapport détaillé sur l'incident, incluant les causes, les actions prises et les leçons apprises. Elle propose des améliorations basées sur cet enseignement afin de renforcer la sécurité. Les parties prenantes, internes et externes, sont informées des mesures prises et des résultats observés.

8.3.7 - Révisions

Le plan de réponse est mis à jour régulièrement selon l'évolution des technologies et des menaces afin que celui-ci soit toujours adaptés au mieux. L'intelligence collective est exploitée en collectant les feedbacks de tous les membres de l'équipe sont recueillis afin d'élargir la vision de l'ensemble, et la liste des possibilités à prévoir.

9 - Conclusion

La satisfaction du client est un élément essentiel pour déterminer si les solutions apportées par le projet sont convaincantes ou non. Cette satisfaction n'englobe pas uniquement la livraison du produit, mais également l'expérience globale du client durant toute la conception de l'application. En comprenant et en répondant aux besoins et aux attentes du client, l'équipe de développement obtient généralement des recommandations positives, une meilleure réputation, plus de crédibilité et donc la possibilité de travailler sur des projets futurs plus ambitieux et challengeant.

L'architecture globale de la solution a permis d'optimiser la gestion, le suivi et la facturation des connectivités WAN en offrant un contrôle optimal sur les données. En effet, chaque menu sur l'interface web offre une gamme complète de fonctionnalités, comme l'ajout, la modification ou la suppression d'objet en quelques clics.

Le projet a également mis en évidence des possibilités d'améliorations à long terme, comme la mise en place d'un menu « *Capacity Planning* » permettant d'inspecter avec précision le débit moyen d'émissions et de réception des connectivités, ou l'extraction automatique des Deltas, c'est-à-dire la synchronisation automatique entre le référentiel des connectivités de FortiManager et celui de la base de données utilisée par l'application.

La satisfaction client est donc un indicateur clé de performance qui permet de s'assurer que les méthodes adoptées par l'équipe sont efficaces, que les besoins ont bien été compris et que les solutions apportées renforcent les perspectives de succès du projet à long terme.

10 – Sources

[1] CodeHeroes [En ligne]. *Comment nommer ses branches et ses commits*. URL: https://www.codeheroes.fr/2020/06/29/git-comment-nommer-ses-branches-et-ses-commits. Consulté le 15 mai 2024

[2] *Django Project* [En ligne]. URL : https://www.djangoproject.com. Consulté le 4 décembre 2023.

[3] *Django Rest Framework* [En ligne]. URL : https://www.django-rest-framework.org. Consulté le 11 décembre 2023.

[4] Atlassian [En ligne]. *Qu'est-ce que la méthodologie Agile*. URL: https://www.atlassian.com/fr/agile. Consulté le 8 février 2024.

[5] Scrum Guides. [En ligne]. *The 2020 Scrum Guide*. URL: https://scrumguides.org/scrum-guide.html. Consulté le 15 février 2024.

[6] Crédit Agricole [En ligne]. *Notre positionnement stratégique*. URL : https://www.credit-agricole.com/notre-groupe/notre-projet-de-groupe/notre-positionnement-strategique. Consulté le 21 juin 2024.

[7] Crédit Agricole [En ligne]. *Notre stratégie RSE : être acteur d'une relation durable*. URL : https://www.credit-agricole.com/responsable-et-engage/notre-strategie-rse-etre-acteur-d-une-societe-durable. Consulté le 13 juillet 2024.

[8] Crédit Agricole [En ligne]. *Crédit Agricole Group Infrastructure Platform, acteur de la transformation digitale du groupe Crédit Agricole*. URL : https://www.credit-agricole.com/marques-et-metiers/toutes-nos-marques/credit-agricole-group-infrastructure-platform. Consulté le 15 juillet 2024.

[9] Using Git. URL : https://docs.gitlab.com/ee/topics/git/. Consulté le 3 octobre 2023.

[10] Python [En ligne]. *Style Guide for Python Code*. URL: https://peps.python.org/pep-0008. Consulté le 17 octobre 2023.