



REPUBLIQUE TUNISIENNE

MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE

Direction Générale des Études Technologiques

Institut Supérieur des Etudes Technologiques de Jendouba

Département des Technologies de l'Informatique

PROJET DE FIN D'ÉTUDES

Présenté en vue de l'obtention du

Diplôme de Licence en Technologies de l'Informatique

SUJET

**Développement d'une application desktop Myroad avec l'architecture
DevOp**

Spécialité : Développement des Systèmes d'Information

Réalisé par :

Mohamed Aziz BJAoui

Wassim Askri

Sous la direction de:

Mr. Riadh BOUSLIMI

Au sein de (Organisme d'accueil) :



Année universitaire : 2021/2022

Dédicace

Nous offrons ce modeste travail :

À nos chers parents,

Aucune dédicace ne pourra faire témoin de notre profond amour, notre immense gratitude et plus grand respect à votre égard. On n'oubliera jamais la tendresse et l'amour dont vous nous avez entourées depuis notre enfance et l'encouragement dans les moments les plus difficiles de notre scolarité.

À tous nos amis, et à tous ceux qu'on aime et à toutes les personnes qui nous ont encouragée et se sont données la peine de nous soutenir durant cette formation.

À nos chers enseignants sans exception.

À ceux qui nous sont chers.

Wassim & Aziz

Remerciements

Nous tenions tout d'abord à remercier DIEU de nous avoir donné le courage, la force et la volonté pourachever ce travail.

Nous adressons nous vifs remerciements à toute personne contribuant de près ou de loin à la réalisation de ce travail.

Nous tenions à remercier vivement l'institut supérieur des études technologiques de Jendouba

Nos remerciements à tous nos professeurs, Ensuite, Nous remercions les membres de jury pour avoir accepté d'évaluer le résultat auquel nous sommes arrivés.

De même, nous adressons nos remerciements à la société ACCENT pour nous avoir permis d'effectuer notre stage au sein de ses services, nous remercions plus particulièrement Mme· Chaima GHAZOUANI notre encadrant professionnel.

Enfin, nos remerciements le plus sincères à Mr· Riadh BOUSLIMI notre encadrant académique à l'institut supérieur des études technologiques de Jendouba, pour son encadrement, ses remarques constructives tout au long de notre travail.

Wassim ASKRI & Aziz BJAOUI

Sommaire

Introduction générale.....	1
Chapitre1 : Cadre général du Projet.....	3
Introduction	3
1.1 Présentation de l'organisme d'accueil	3
1.1.1 Fiche d'identité.....	3
1.1.2 Services	3
1.1.3 Activités	4
1.2 Analyse de l'existant	4
1.2.1 Description des applications similaires.....	4
1.2.2 Comparaison et critique	8
1.3 Solution Proposée	8
1.4 Environnement matériels et logiciels	9
1.4.1 Environnement matériels	9
1.4.2 Environnement logiciels	10
Conclusion.....	13
Chapitre 2 : Etat de l'art.....	14
Introduction	14
2.1 DevOps	14
2.1.1 Définition.....	14
2.1.2 Fonctionnement de DevOps.....	14
2.1.3 Les avantages	15
2.1.4 Intégration Contenu (CI/CD)	16
2.2 La notion de versionning	17
2.2.1 Définition.....	17
2.2.2 Les versions.....	17
2.2.2 Les branches.....	17
2.3 La gestion des infrastructures	18
2.3.1 Notion d'orchestration	18
2.3.2 Automatisation	19

2.4 Virtualisation	19
2.5 Conteneurisation	20
2.5.1 Notion de conteneurisation	20
2.5.2 Orchestration des conteneurs	20
Conclusion.....	21
Chapitre 3 : Etude préliminaire.....	22
Introduction	22
3.1 Les acteurs intervenants	22
3.2 Description sommaire des besoins	23
3.2.1 Besoins fonctionnels.....	23
3.2.3 Besoins non fonctionnels	24
3.3Modélisation des besoins.....	24
3.4Méthode de conception.....	25
3.4.1 Les méthodes agiles.....	25
3.4.2 Framework Scrum	27
3.5Planning de travail.....	32
3.5.1 Backlog de produit	32
3.6.2 Planification des sprints	35
3.6 Conclusion	36
Chapitre 4 : Mise en œuvre de Release 1.....	37
Introduction	37
4.1 Sprint 1	37
4.1.1 Backlog du Sprint.....	37
4.1.2 Exécution de sprint	38
4.1.3 Déroulement de sprint.....	39
4.1.3.1 Sprint Review	39
4.2 Sprint 2	41
4.2.1 Backlog du sprint.....	41
4.2.2 Analyse	42
4.2.3 Conception	45
4.2.4 Réalisation.....	49
4.2.5 Déroulement de sprint.....	52
Conclusion.....	54
Chapitre 5 : Mise en œuvre de Release 2	55

Introduction	55
5.1 Sprint 1	55
5.1.1 Backlog du Sprint.....	55
5.1.2 Architecture de l'environnement.....	55
5.1.3 Réalisation.....	59
5.1.3 Déroulement de sprint	66
5.2 Sprint 2	66
5.2.1 Backlog du sprint.....	66
5.2.2 Architecture de l'environnement.....	67
5.2.2.2 Architecture d'Ansible	68
5.2.3 Réalisation.....	71
5.2.4 Déroulement de sprint.....	73
Conclusion.....	74
Conclusion générale et perspectives	75

Liste des figures

Figure 1: Logo de la société "Accent"	3
Figure 2: Captures d'application "WeenCar".....	6
Figure 3: Captures d'application "MycarTracks "	7
Figure 4: Logo de notre application.....	9
Figure 5: Fonctionnement de DevOps	15
Figure 6: L'approche CI/CD	17
Figure 7: Notion des branches	18
Figure 8: Notion de virtualisation	20
Figure 9: L'acteur de l'application (Administrateur).....	22
Figure 10: Diagramme de cas d'utilisation globale	25
Figure 11: Processus SCRUM.....	29
Figure 12: Team Membres	30
Figure 13: Planification des sprints.....	35
Figure 14: Diagramme de Gantt	36
Figure 15: Le huit de DevOps	39
Figure 16: Sprint Brun-dow-charte de sprint 1 (Release 1).....	40
Figure 17: Diagramme de cas d'utilisations de sprint 2 (Release 1)	42
Figure 18: Diagramme de classe de sprint 2 (Release 1)	46
Figure 19: Diagramme de séquence s'authentifier"	47
Figure 20: Diagramme de séquence "ajouter une personne".....	48
Figure 21: Diagramme de séquence de "supprimer une personne"	49
Figure 22: Interface d'inscription.....	50
Figure 23: Interface d'authentification	50
Figure 24: Interface liste des personnes	51
Figure 25: Interface de suppression d'une personne	51
Figure 26: Interface d'ajout d'une localisation.....	52
Figure 27: Interface d'historique des trajets (filtre par date).....	52
Figure 28: Sprint Brun-down-charte de sprint 2 (Release 1).....	53
Figure 29: Architecture docker	56
Figure 30: Architecture conteneurisé	57
Figure 31: Architecture Jenkins	59
Figure 32: Notion de clustering	60
Figure 33: git push code master	61
Figure 34: Install docker-Jenkins container with data log volume	61
Figure 35: unlock Jenkins cat	62
Figure 36: setup Jenkins server	62
Figure 37: create admin user	63
Figure 38: getting start Jenkins	63
Figure 39: install plugin manager docker pipeline	64
Figure 40: Dashboard Jenkins	64
Figure 41: select new pipeline to application « myroad ».....	65

Figure 42: build console run output CI	65
Figure 43: Sprint Brun-dow-charte de sprint 1 (Release 2).....	66
Figure 44: Architecture optimisée	68
Figure 45: Architecture Ansible	69
Figure 46: Architecture générale	70
Figure 47: Sprint Brun-dow-charte de sprint 2 (Release 2).....	74

Liste des tableaux

Tableau 1: Analyse ergonomique de "WeenCar"	5
Tableau 2: Analyse ergonomique "MycarTracks".....	7
Tableau 3: Comparaison entre les applications similaires	8
Tableau 4: Environnement matériels	10
Tableau 5: Environnement logiciels	10
Tableau 6: Les méthodes agiles	26
Tableau 7: Backlog de produit.....	33
Tableau 8: Backlog de sprint 1(Release1).....	37
Tableau 9: Tableau de validation de sprint 1 (Release 1)	40
Tableau 10: Backlog de produit de sprint 2 (Release 1)	41
Tableau 11: Description de cas "s'authentifier"	42
Tableau 12:Description de cas « Ajouter véhicule »	43
Tableau 13: Description de cas d'utilisation "ajouter une localisation"	44
Tableau 14: Description de cas d'utilisation "supprimer véhicule"	44
Tableau 15: Tableau de validation de sprint 2 (Release 1)	53
Tableau 16: Tbaleau de validation de sprint 1 (Release 2)	66
Tableau 17: Tableau de validation de sprint 2 (Release 2)	74

Introduction générale

LE CLIENT EST ROI... Traitez donc vos clients avec tous les égards qui leur sont dus, car sans clients votre entreprise n'a aucune raison d'être. Ce slogan a obligé les grandes entreprises d'admettre des profonds changements dans leur stratégie, dans leurs modèles d'affaires et dans leur modèle de gouvernance pour satisfaire « CE CLIENT ROI ».

Ceci a engendré une évolution technologique dans la gouvernance de l'entreprise qui ne se limite plus à la gestion des ressources matérielles et humaines, mais aussi à la gestion des cycles de vie de ces projets.

Auparavant la gestion de projet se base sur un cycle basique qui consiste à recueillir les besoins, définir le produit, le développer et le tester avant de le livrer. Dans ce cas, les risques sont détectés tardivement et le client n'est plus satisfait.

L'ouverture des frontières et la concurrence constatée suite à la mondialisation a poussé les entreprises à suivre une course folle à l'innovation pour se positionner davantage sur le marché.

Réagir vite et mieux, adapter son organisation et anticiper ses évolutions sont devenus les slogans de la bonne gouvernance. Les défis à relever sont devenus importants et les grandes questions tels que comment livrer un logiciel plus rapidement répondant véritablement aux besoins des utilisateurs et comment mettre à jour une application avec le moindre risque.

À cet égard, Les entreprises auront recours à l'approche DevOps. Le DevOps permet d'en finir avec les projets à rallonge dont les deadlines n'étaient que rarement tenues. Il touche tant la gestion de projets, les outils utilisés et les modèles organisationnels favorisant la rapidité et l'efficacité d'un cycle de production. Les conteneurs sont par exemple des moyens techniques pour architecturer des solutions répondant non seulement aux besoins variables, mais encore qui s'adaptent rapidement aux évolutions requises par les environnements actifs afin de se positionner davantage sur le marché.

Introduction générale

C'est dans ce cadre que s'inscrit notre Projet de Fin d'Études qui a pour objectif à la mise en place de l'intégration et déploiement continue d'une application web. Notre stage est effectué au sein de l'entreprise ACCENT en Tunisie.

Notre rapport sera structuré en cinq chapitres qui décrivent les différentes étapes suivies pour la réalisation de notre projet. Nous commençons par un premier chapitre dans lequel nous présenterons l'organisme d'accueil Accent et son domaine d'activité ainsi que le contexte et la problématique du projet afin de proposer des solutions. Nous y présentons aussi la méthodologie de travail adopté "Scrum".

Nous passerons par la suite au deuxième chapitre qui présentera les concepts théoriques relatifs à notre projet savoir les notions de DevOps, versionning, conteneurisation, etc.

En ce qui concerne le troisième chapitre, nous identifierons les besoins fonctionnels et non fonctionnels de notre projet ainsi que ses acteurs suivi de la spécification de l'environnement matériel et logiciel.

Ensuite, nous enchaînerons avec la partie pilotage, qui nous mènera vers une planification pour la mise en place de notre projet.

Le quatrième chapitre, Sprint 1 : « Formation d'initiation à la culture DevOps » et Sprint 2 « Développement de l'application Desktop avec Flutter ».

Pour le cinquième chapitre, Sprint 3 : Mise en place d'u environnement Docker et automatisation» et Sprint 4 : « Optimisation de l'image et mise en plage de configurations des serveurs »

Le présent rapport sera clôturé par une conclusion générale qui résumera le travail réalisé, évoquera les problèmes rencontrés et dégagera quelques perspectives concernant ce stage.

Chapitre1 : Cadre général du Projet

Introduction

Ce chapitre se compose de quatre parties. La première pour but de présenter la société dans laquelle nous avons effectué notre stage de fin d'études, la deuxième de présenter le cadre général, la troisième de présenter l'étude de l'existant et la dernière partie d'expliquer le choix méthodologique et le formalisme adopté.

1.1 Présentation de l'organisme d'accueil

1.1.1 Fiche d'identité

ACCENT est une entreprise industrielle spécialisée dans l'industrie électronique faisant intervenir les technologies de communication M2M/IoT et le développement et recherche afférents aux systèmes d'électroniques embarqués. Cette entreprise est fondée en 2018 en Tunisie dans le technopôle de Manouba. La figure ci-dessous présente le logo de cette entreprise.



Figure 1: Logo de la société "Accent"

1.1.2 Services

La société ACCENT est spécialisée dans plusieurs domaines décrits par :

- **Plateforme IoT Accentify** : cette une plateforme IoT prête à l'emploi et qui sert à connecter les entreprises et les surveiller ;
- **Conception matérielle** : concevoir des systèmes connectés en fonction des exigences et des besoins du client ;
- **Intégration de logiciels** : adaptation de la connectivité aux entreprises et extraire de plus en plus d'informations des données disponible ;
- **Accélération IoT** : pour démarrer une activité IoT, l'expertise de cette société peut aider à avoir le délai de mise sur le marché le plus rapide ;
- **Intégration IoT** : les appareils connectés peuvent être transférés sur une plateforme dédiée pour avoir plus de données et plus de flexibilité pour les surveiller ;
- **Prototypage** : chaque prototype est produit avec des partenaires PME hautement qualifiés afin d'optimiser le coût et le temps de fabrication.

1.1.3 Activités

ACCENT propose plusieurs solutions dans le domaine de la communication M2M et de l'IoT tels que:

- **Transport intelligent** : réclamation BUS CAN et intégration BLE Network pour connecter des capteurs sans fil ;
- **Surveillance de l'énergie** : collecte de données multi-dimensions et projection métier client et traitement à la volée pour générer des notifications et des alertes ;
- **Éclairage des rues** : solution de maillage basée sur BLE 5.2, déploiement à faible coût et intégration de capteurs.
- **Industrie 4.0** : connecter la machine et affichez la preuve de travail et contrôle à distance et collecte de données.

1.2 Analyse de l'existant

1.2.1 Description des applications similaires

Avant de commencer à concevoir notre application, nous présentons une analyse sur quelques applications similaires existante dans la littérature. Nous citons principalement deux applications à savoir :

1.2.1.1 Analyse d'application « «WeenCar» »

✚ Description de site

C'est une application web et mobile tunisienne de suivis des véhicules en temps réel.

✚ Analyse fonctionnelle

Cette application permet aux utilisateurs de :

- Commander ses appareils WeenCar OBD-ligne.
- Personnaliser les informations de véhicule.
- Suivre ses véhicules en temps réel.

✚ Analyse ergonomique

Tableau 1: Analyse ergonomique de "WeenCar"

Dénotations	Description
Logo	Il est coloré de 2 couleurs (blanc et bleu), sa position à gauche.
La page d'accueil est disposée verticalement	Cette disposition donne un sens de lecture qui rend la page plus simple.
Liste de menu se trouve horizontalement.	Menu organisé et facilite aux utilisateurs la recherche
La gamme de couleurs utilisés est blanche et bleu.	Le Blanc est pour l'utilisation dans l'arrière-plan et dans le logo d'application. Le Bleu est utilisé à l'entête. L'assemblage de ces couleurs donne une cohérence aux différentes interfaces d'application

✚ Quelques captures de l'application

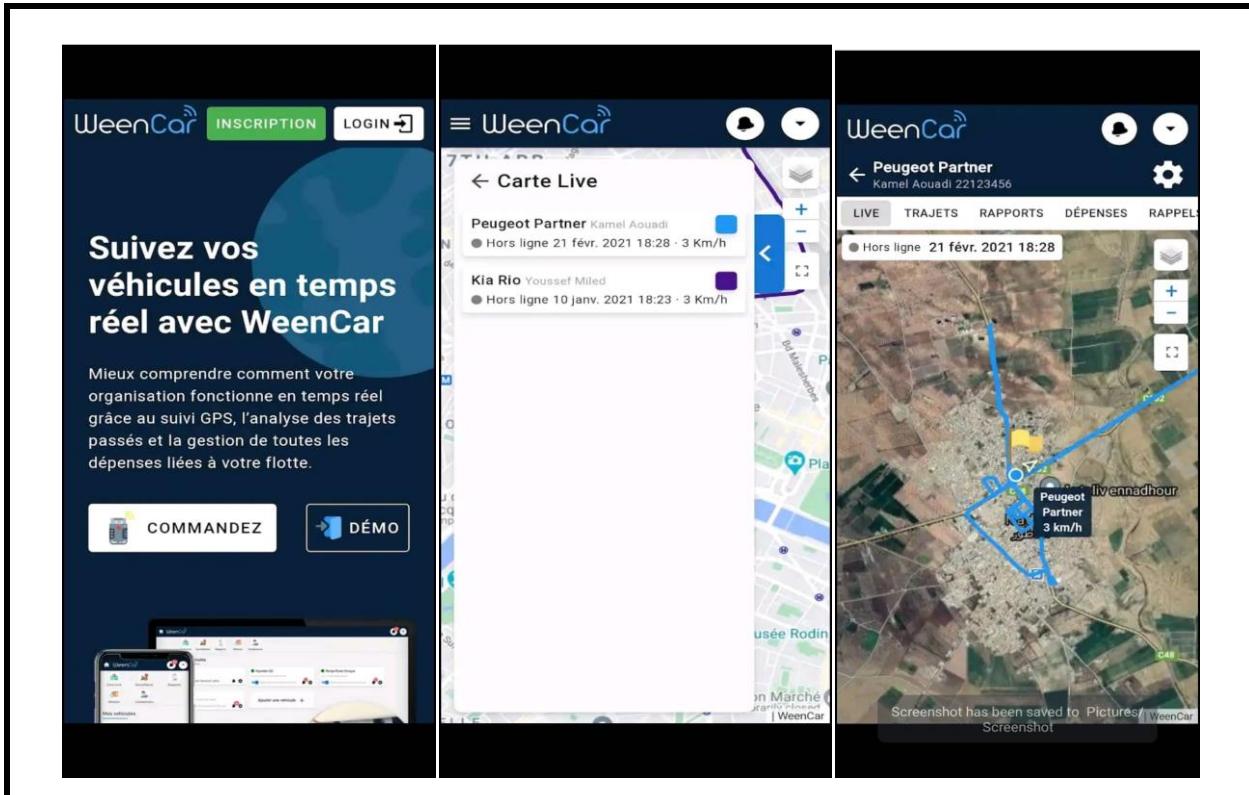


Figure 2: Captures d'application "WeenCar"[1]

1.2.1.2 Analyse de l'application «MycarTracks»

✚ Description de l'application

C'est une application mobile de gestion et de suivi des véhicules.

✚ Analyse fonctionnelle

Cette application permet aux Utilisateurs de :

- Gérer et suivre ses véhicules.
- Partager ses localisations.
- Consulter un rapport qui contient des statistiques.

✚ Analyse ergonomique

Tableau 2 : Analyse ergonomique « MycarTracks »

Dénotations	Description
Logo	Il est coloré de 4 couleurs (bleu, vert, jaune et blanc)
Les pages sont disposées verticalement	Cette disposition donne un sens de lecture qui rend la page plus simple
Liste de menu se trouve verticalement dans la page d'accueil.	Menu organisé et facilite aux patients la recherche.
La gamme de couleurs utilisés est blanche et bleu, vert, rouge et gris.	Le Blanc est pour l'utilisation dans l'arrière-plan. Le Gris est utilisé dans la liste de menu et le design. Le bleu à l'ente et le NavBar. Le vert et le rouge pour les boutons.

✚ Quelques captures de l'application

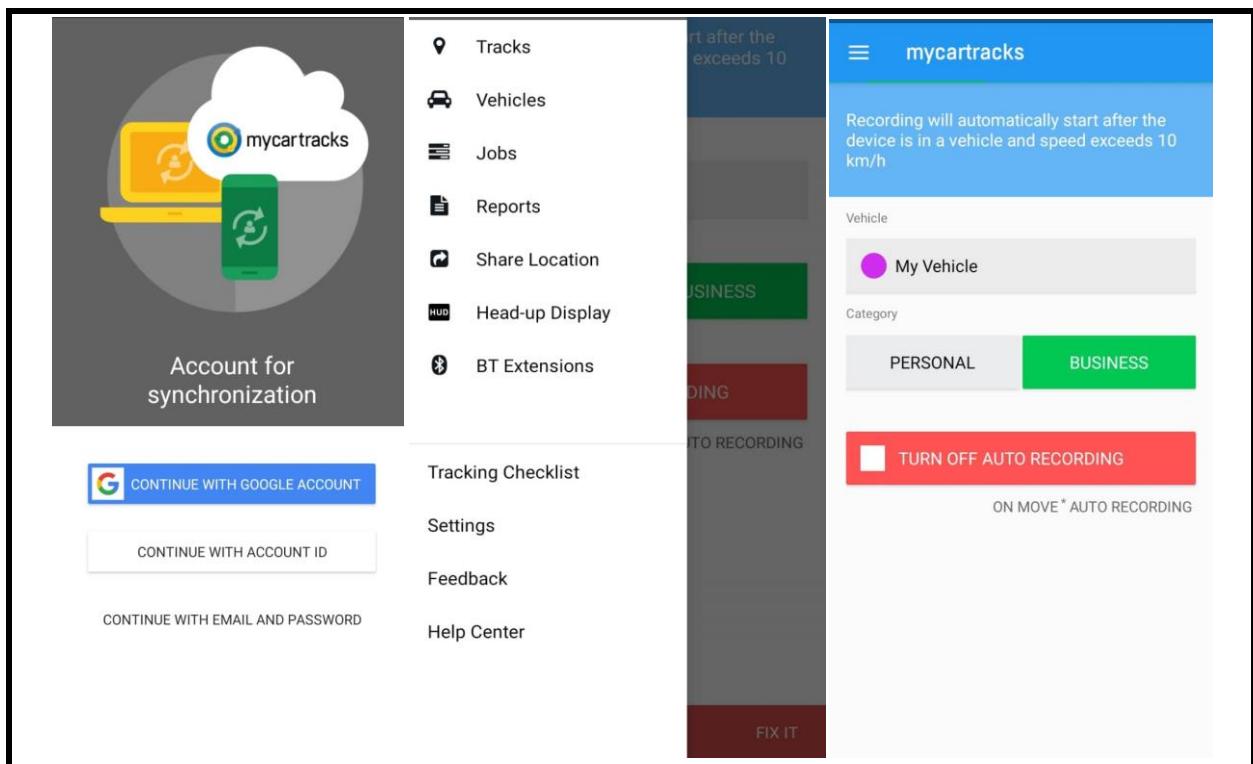


Figure 3 : Captures d'application « MycarTracks » [2]

1.2.2 Comparaison et critique

On compare les applications précédentes suivant des précises et on résume leurs comparaisons dans le tableau suivant :

Notes : 1 (très limité), 2 (limité), 3 (moyen), 4 (bon), 5 (excellent)

Tableau 3: Comparaison entre les applications similaires

Nom de l'application	WeenCar	MycarTracks
Performance	3	2
Fonctionnalités	3	4
Facilité d'utilisation	2	3
Documentation	2	2
Sécurité	3	4

Bien que les applications susmentionnées répondent aux quelques besoins et comme toute autre application, elles présentent également des inconvénients. Citons parmi eux:

- WeenCar :
 - Trop difficile et pas pratique à utiliser.
 - Sécurisation faible
- MyCarTracks :
 - Pas pratique à utiliser.

1.3 Solution Proposée

- En se concentrant sur ce thème et pour remédier aux problèmes précédemment cités, nous a cloné la mission de concevoir et de réaliser une application Desktop que nous avons appelé « **My ROAD** » permettant principalement de :
 - ✓ Gérer les véhicules et les personnes.

- ✓ Accéder au Maps et gérer localisation.
- ✓ Consulter historique des trajets.
- Pour accélérer les tâches requises à la mise en place de notre application, nous avons décidé de réaliser un système de **déploiement** automatique basé sur la méthode **DevOps**. En effet, il s'agit d'une technologie efficace, flexible et rapide qui permet de résoudre les diverses situations autour des utilisateurs et d'aider l'équipe Accent à automatiser l'acceptation et le déploiement contenu de notre solution dans l'environnement de recette et de production.
- Conception graphique de logo de notre application



Figure 4: Logo de notre application

Signification de notre logo :

- Nous avons choisi les couleurs, blanc, gris et rouge
 - ✓ Le rouge est associé à la passion et pousse les personnes à passer à l'action.
 - ✓ Le gris est l'une des teintes les plus neutres qui existent, les marques le choisissent souvent pour son caractère pratique et impartial.
 - ✓ Le blanc est une couleur réfléchissante qui représente la pureté et l'efficacité.

1.4 Environnement matériels et logiciels

1.4.1 Environnement matériels

Pour la réalisation de notre projet nous avons disposé de deux ordinateurs caractérisé par :

Tableau 4: Environnement matériels

	Asus m570d	Dell 3500
Processeur	Ryzan 5	I5
Mémoire	12 Go ram 500Go	8 Go ram 500 Go
Système d'exploitation	Windows 11	Ubuntu 20.04
Type de système	64 bits, processeur*64	64 bits, processeur*64

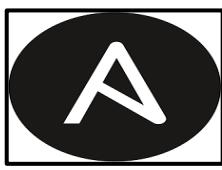
1.4.2 Environnement logiciels

Tableau 5: Environnement logiciels

Thème	Logo	Description
Gestion de projet	 GantProject	GantProject est un logiciel libre de gestion de projet écrit en Java, ce qui permet de l'utiliser sur divers systèmes d'exploitation (Windows, Linux, MacOs). Il permet d'édition un diagramme de Gantt qui organise le projet selon des dates fixées. [3]
	 Anydesk	AnyDesk est un logiciel permettant de prendre votre PC en main à distance depuis un autre appareil. C'est un programme portable que vous pourrez emmener partout avec vous sur un support USB et qui peut être installé sur plusieurs appareils. Il est également possible de contrôler votre bureau à distance depuis un appareil mobile grâce aux applications AnyDesk pour Ios et Android. [4]

Conception	 StarUml	<p>StarUml est un logiciel de modélisation UML (Unified Modeling Language) open source qui peuvent remplacer dans bien des situations des logiciels commerciaux et coûteux comme Rational Rose. Étant simple d'utilisation, nécessitant peu de ressources système, supportant UML 2, ce logiciel constitue une excellente option pour une familiarisation à la modélisation. Cependant, seule une version Windows est disponible. [5]</p>
	 Visual Studio code	<p>Vs Code se présente comme un éditeur de code optimisé pour Windows, OS X et Linux. Il est décrit comme « un nouveau type d'outil qui combine la simplicité d'un éditeur de code avec tout ce dont les développeurs ont besoin pour leur cycle basique modifier-construire-déboguer ». En plus d'offrir une édition complète, Visual Studio Code est venu avec le support d'IntelliSense (un système de complétion de code intelligent), le débogage et avec l'intégration de GIT. [6]</p>
	 Flutter	<p>Flutter est un framework développé par Google, le plus récent de tous. ce framework est utilisé pour tout ce qui est interface utilisateur. Mais aujourd'hui Flutter se fait surtout connaître pour sa capacité à concevoir des applications natives multiplateforme pour Android et Ios (Windows/Mac/Linux sont également supportés). [7]</p>
	 Dart	<p>Dart, le nouveau langage orienté Web de Google, a été dévoilé au monde en octobre 2011. Il se veut un langage structuré, non pas révolutionnaire, mais facile d'apprentissage pour tout développeur, quel que soit son background (C#, Java ou JavaScript) puisque Dart est un agrégat de ces trois langages avec d'autres, tel que Small talk. [8]</p>

Développement	 Aqueduct	<p>Aqueduct est conçu pour le développement piloté par les tests - la meilleure façon d'écrire une application est d'écrire des tests à l'aide d'un harnais de test et d'exécuter ces tests après avoir implémenté un point de terminaison. Vous pouvez également exécuter la commande Aqueduct document client dans le répertoire de votre projet pour générer un client Web pour votre application. [9]</p>
	 Supabase	<p>Supabase est construit sur un logiciel open source. Si les outils et les communautés sont disponibles sous une licence MIT, Apache 2 ou compatible, ils les utiliseront et en feront la promotion. Ils construiront et publieront le programme s'il n'existe pas déjà. Supabase n'est pas une cartographie individuelle de Firebase en termes de fonctionnalités. Leur objectif est de donner aux développeurs une expérience de développement similaire en utilisant des technologies open source. [10]</p>
	 PostgreSQL	<p>PostgreSQL (prononcer « post-gress-Q-L ») est un système de gestion de base de données relationnelle (SGBDR) open source développé par une équipe internationale constituée de bénévoles. PostgreSQL n'est détenu par aucune entreprise ni autre entité privée, et son code source est accessible librement et gratuitement. [11]</p>
Déploiement	 Docker	<p>Docker est un outil qui peut empaqueter une application et ses dépendances dans un conteneur isolé, qui pourra être exécuté sur n'importe quel serveur. [12]</p>

	 Jenkins	<p>Jenkins est un outil d'intégration continue open source développée en Java, conçue pour orchestrer des pipelines de déploiement. Équipé d'une API, il propose plus que 1500 plugins. A chaque modification de code d'une application dans le gestionnaire de configuration, Jenkins se charge automatiquement de la re-compiler, la déployer et de la tester. [13]</p>
	 GitLab	<p>GitLab est un logiciel libre de forge basé sur git proposant les fonctionnalités de wiki, un système de suivi des bugs, l'intégration continue et la livraison continue. [14]</p>
	 Ansible	<p>Ansible est une open source DevOps créé en 2012 par Michael DeHaan. Il est aujourd'hui géré par l'entreprise Red Hat (groupe IBM). Cet outil écrit en Python permet via le protocole SSH de gérer et configurer un ensemble de machines. [15]</p>

Conclusion

Tout au long de ce chapitre, nous avons présenté l'organisme de d'accueil et ses principales activités. Par ailleurs, nous avons pu dégager le cadre général du projet, et de présenter l'étude de l'existant et citer l'environnement logiciel et matériel qu'on va utiliser. Le chapitre suivant sera consacré à l'étude préliminaire.

Chapitre 2 : Etat de l'art

Introduction

Dans ce chapitre, nous commençons par la présentation de l'approche DevOps notre domaine d'étude. Ensuite nous allons présenter les notions de versionning et la gestion des infrastructures. Enfin, nous présentons l'approche de la virtualisation et la conteneurisation.

2.1 DevOps

2.1.1 Définition

DevOps est un ensemble de pratiques qui met l'accent sur la collaboration et la communication entre les développeurs de logiciels et les professionnels des opérations informatique, en automatisant le processus de livraison de logiciels et les changements d'infrastructure.

Le DevOps permet de faire évoluer et optimiser les produits plus rapidement par opposition à l'utilisation des processus traditionnels de développement de logiciels et de gestion de l'infrastructure par les entreprises. Cette vitesse permet à ces dernières de mieux servir les clients et de gagner en compétitivité. [16]

Au cours des dernières années, les équipes de développement et d'exploitation ont amélioré leur façon de travailler. Aujourd'hui, le besoin de réaligner ces deux équipes se renforce.

DevOps transforme totalement la façon dont les informaticiens perçoivent la stabilité et le fonctionnement du système qu'ils gèrent, ainsi que leur propre rôle dans le flux de valeur ajouté du début à la fin.

2.1.2 Fonctionnement de DevOps

Dans un modèle DevOps, les équipes de développement et d'exploitations ne sont pas isolées. Il arrive qu'elles soient fusionnées en une seule et même équipe. Les ingénieurs qui la

composent travaillent alors sur tout le cycle de vie d'une application, de la création d'exploitation, en passant par les tests et le déploiement, ils doivent ainsi développer toute une gamme de compétences liées à ces différentes fonctions.

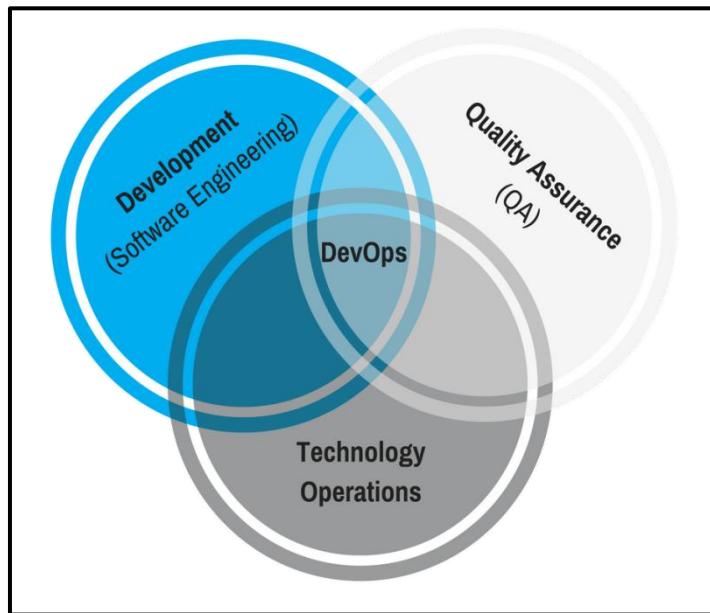


Figure 5: Fonctionnement de DevOps

Dans certains modèles DevOps, les équipes d'assurance qualité et de sécurité peuvent également s'intégrer étroitement au développement et aux opérations, comme la montre le figure 5.

Ces équipes utilisent des pratiques pour automatiser des processus qui étaient autrefois manuels et lents. Elles exploitent une pile technologique et des outils qui les aident à faire fonctionner et à faire évoluer les applications de façon autonome des tâches qui nécessiteraient l'idée d'autres équipes, ce qui augmente encore d'avantage la productivité de l'équipe d'ingénieurs. [17]

2.1.3 Les avantages

En plus d'améliorer et optimiser le travail en équipe, l'approche DevOps permet :

- **d'améliorer l'expérience client** : une entreprise doit pouvoir obtenir et répondre en continu au retour des clients, mais également des utilisateurs, partenaires, fournisseurs... et cela implique donc la mise en place de processus impliquant toutes les parties prenantes.

- **d'innover et optimiser l'utilisation du temps** : en réduisant les coûts de mise en œuvre et en automatisant les déploiements, cela laisse l'opportunité aux équipes de tester différentes versions ou fonctionnalités et de passer moins de temps à corriger les erreurs ou tout recommencer.
- **d'accélérer le retour sur investissement** : en déployant les logiciels plus rapidement et en misant sur des processus fiables et automatisés.
- **d'avoir un meilleur rendement informatique** : moins d'erreurs et d'échecs dans les déploiements avec des délais de production plus courts.
- **d'intégrer la sécurité en continu** : les objectifs de sécurité sont connus à l'avance et sont intégrés en continu dans le travail au quotidien des équipes. [18]

2.1.4 Intégration Contenu (CI/CD)

L'approche intégration contenu(CI)/CD se base sur un processus simple appelé « pipeline ». Il est constitué d'une série d'étapes à réaliser pour créer et fournir une nouvelle version d'un logiciel.

Dans un premier temps, les développeurs(**Devs**) écrivent le code. Dès qu'ils le jugent de qualité et pertinent pour faire évoluer le logiciel, ils l'intègrent. Il sera alors soumis automatiquement à des tests pour s'assurer qu'il fonctionne et n'entre pas en conflit avec l'existant sur l'application. S'il fonctionne, cette partie de code partira en production.

C'est ce que l'on appelle l'intégration continue – ou continuons intégration : les développeurs ajoutent progressivement du code, des fonctionnalités qui vont venir se fusionner à l'application existante.

Puis, les équipes opérationnelles (**Ops**) prennent le relais et de multiples séries de tests sont effectuées automatiquement dans différents environnements. Cela permet de s'assurer encore une fois qu'il n'y aura aucun problème. Si tel est le cas, ces fonctionnalités vont être automatiquement soumises à approbation. C'est ce que l'on appelle la distribution continue – continuous delivery.

D'autre part, elles peuvent aussi être automatiquement déployées sans approbation préalable si elles ont passé les tests avec succès. C'est ce que l'on appelle le déploiement continu – continuous deployment. [19]

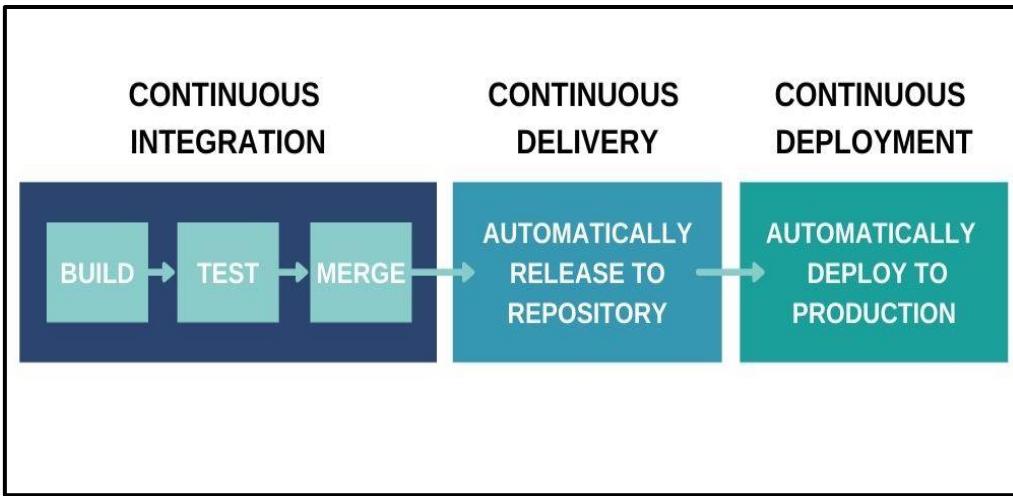


Figure 6: L'approche CI/CD

2.2 La notion de versionning

2.2.1 Définition

La versionning est une activité qui consiste à maintenir toutes les versions de logiciels. Essentiellement utilisé dans le domaine de la création des logiciels, impliquant du code source mais peut être utilisé pour tout type de document informatique. [20]

2.2.2 Les versions

Les différentes versions sont liées à travers des modifications.

Une modification peut correspondre à des ajouts, modifications, suppressions ou une combinaison de trois sur une version donnée. On passera de la version N à la version N+1 en appliquant une modification M.

Un logiciel de gestion des versions nous aidera alors à soustraire la modification M à la version N+1 pour retrouver la version N. [21]

2.2.2 Les branches

La branche version correspond à l'axe d'évolution de la version. Il est attaché à la branche source et peut provenir de plusieurs sous-branches. La gestion de toutes les branches et versions du produit constitue le contrôle de versions et fait l'objet de la gestion de la configuration.

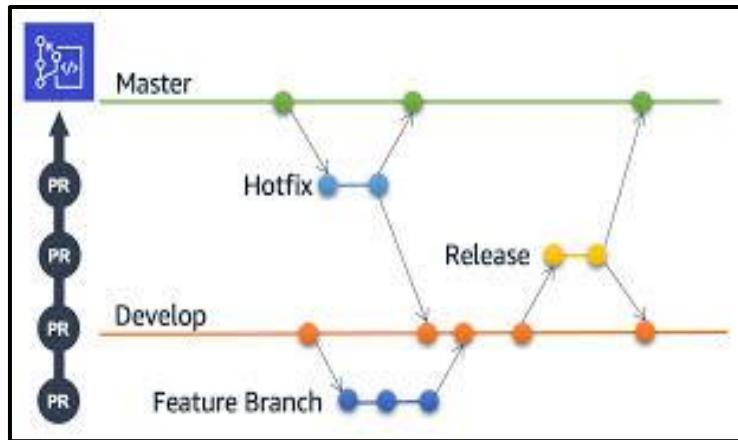


Figure 7: Notion des branches

Par défaut, l'évolution du produit à travers l'historique de ses versions est un phénomène linéaire : le développeur modifie le code du produit, un autre modifie le même code, et enfin le tout sera fusionné en un sur l'emplacement. C'est ce qu'on appelle la branche principale ou branche « Master » comme le montre la figure 7.

Cependant, dans le processus de développement logiciel, la ligne principale peut avoir à suivre son processus en même temps, comme le montre la figure 6 illustrée par les points en vert. Nous devons créer une autre branche qui évolue parallèlement à la branche principale. L'origine de ces contraintes peut être externe : un des clients veut une modification qui lui est spécifique ou interne, nous avons besoin de pouvoir continuer à développer la branche principale tout en permettant de corriger de défauts sur une branche en cours de tests. [22]

2.3 La gestion des infrastructures

2.3.1 Notion d'orchestration

L'orchestration est la configuration, la gestion automatiques des systèmes informatiques, des applications et des services.

L'orchestration permet aux informaticiens de gérer plus facilement des tâches et des flux de travail complexes : l'équipe du service informatique est responsable de nombreux serveurs et applications et sa gestion manuelle limite les possibilités de développement. Plus l'environnement informatique est complexe, plus la gestion de tous ses pièces mobile est complexe. [23]

2.3.2 Automatisation

L'automatisation optimise l'efficacité en limitant les interactions humaines ou en les remplaçant par des systèmes informatiques qui utilisent des logiciels pour effectuer les tâches, avec l'objectif de réduire les coûts, la complexité et le risque d'erreur.

En général, l'automatisation concerne une tâche unique. C'est ce qui la différencie de l'orchestration, qui décrit comment automatiser un processus ou un workflow constitué de nombreuses étapes réalisées sur plusieurs systèmes disparates.

Lorsque nous commençons l'automatisation des processus, nous pouvons les orchestrer afin qu'ils s'exécutent automatiquement. L'orchestration informatique peut également aider à simplifier et à optimiser les processus et workflow couramment utilisés.

Compatible avec l'approche DevOps, elle permet à l'équipe de déployer des applications plus rapidement. On peut utiliser l'orchestration pour automatiser les processus informatiques. [24]

2.4 Virtualisation

La virtualisation est la création d'une version virtuelle – plutôt que réelle – de quelque chose, comme un système d'exploitation (OS), un serveur, un périphérique de stockage ou des ressources réseau.

La virtualisation utilise un logiciel qui simule les fonctionnalités du matériel afin de créer un système virtuel. Cette pratique permet aux organisations informatiques d'exploiter plusieurs systèmes d'exploitation, plus d'un système virtuel et diverses applications sur un seul serveur. Les avantages de la virtualisation comprennent une plus grande efficacité et des économies d'échelle.

La virtualisation du système d'exploitation est l'utilisation d'un logiciel permettant à une pièce de matériel d'exécuter plusieurs images de système d'exploitation en même temps. Cette technologie a fait ses débuts sur les ordinateurs centraux il y a plusieurs décennies, permettant aux administrateurs d'éviter de gaspiller une puissance de traitement coûteuse. [25]

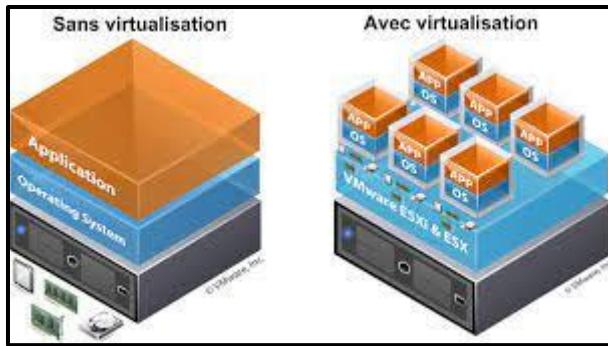


Figure 8: Notion de virtualisation

2.5 Conteneurisation

2.5.1 Notion de conteneurisation

La conteneurisation est un type de virtualisation au niveau de l'application, qui permet de créer plusieurs instances d'espace utilisateur isolées sur un même noyau. Ces instances sont appelées conteneurs.

Un conteneur est un système qui permet de stocker et d'isoler des objets avant d'être transportés ou déployés dans un environnement d'exploitation étendu applications, logiciels, librairie, etc. Il permet également au code applicatif d'être transporté de l'environnement de développement vers celui de production de manière aisée et sûre.

La conteneurisation est de plus en plus populaire, car les conteneurs sont :

- **Flexible** : même les applications les plus complexes peuvent être conteneurisées.
- **Léger** : les conteneurs exploitent et partagent le noyau hôte.
- **Portable** : Nous pouvons créer localement, déployer sur le cloud et exécuter n'importe où notre application.
- **Évolutif** : Nous pouvons augmenter et distribuer automatiquement les répliques (les clones) de conteneur.
- **Empilable** : Nous pouvons empiler des services verticalement et à la volée. [26]

2.5.2 Orchestration des conteneurs

L'orchestration des conteneurs permet d'automatiser le déploiement, la gestion, la mise à l'échelle et la mise en réseau des conteneurs. Cette technologie a fait ses épreuves avec les entreprises qui ont besoin de déployer et de gérer des centaines ou des milliers de conteneurs Linux et d'hôtes. [27]

Conclusion

Nous avons exposé dans ce chapitre les notions et les définitions indispensables pour la compréhension du contexte de notre projet. En effet, nous avons présenté l'approche DevOps, le CI/CD, la notion de versionning, la gestion des infrastructures et finalement la virtualisation et la conteneurisation. Dans le chapitre suivant, nous allons entamer une étude préalable, dégager les besoins et passer à la planification.

Chapitre 3 : Etude préliminaire

Introduction

Ce chapitre présente la planification et l'architecture de projet qui consiste une phase d'analyse du projet.

Nous allons montrer tout d'abord, l'identification des acteurs ainsi que la description des besoins.

Puis, nous allons présenter le diagramme de cas d'utilisation et de classe générales.

Ensuite, nous allons présenter les prototypes de quelques interfaces ainsi que l'architecture matérielle et logicielle

Enfin, nous allons clôturer le chapitre par la mise en œuvre qui consiste à présenter le Backlog office et la planification des sprints.

3.1 Les acteurs intervenants

Un acteur est une personne ou un autre système informatique qui attend un ou plusieurs services offerts par l'application. Par conséquent, nous identifions un seul acteur dans notre application. [28]



Figure 9: L'acteur de l'application (Administrateur)

- **Administrateur :** C'est un super utilisateur qui joue un rôle important dans l'assurance du bon fonctionnement de l'application ainsi que la gestion des différents modules.

3.2 Description sommaire des besoins

L'identification des besoins consiste à traduire les objectifs du projet en un ensemble de fonctionnalités ciblées par l'outil à réaliser. Ceci procurera une compréhension plus approfondie des tâches à mettre en œuvre.

3.2.1 Besoins fonctionnels

Nous présentons dans ce qui suit les différentes fonctionnalités du système. Ces fonctionnalités doivent répondre aux attentes du Product Owner. Les besoins spécifiés doivent être persistants, spécifiques et réalisable. Les besoins fonctionnels, répondre aux questions suivantes :

- ✓ A quoi sert le système ?
- ✓ Quelles sont les fonctionnalités du système ?

Suite à plusieurs réunions avec le Product Owner représenté par Mme. Chaima GHAZOUANI, nous avons cernées besoins fonctionnels d'application.

- Inscription
- Authentification
- Gestion des véhicules
- Gestion des personnes
- Gestion des localisations
- Consulter historique des trajets

Pour le déploiement nous avons cernées les besoins suivants :

- Accélérer les processus de développement.
- Assurer la création des environnements préconfigurés.
- Automatiser le déploiement et la mise à jour du code.
- Assurer l'orchestration des différentes machines.
- Automatiser le test sur les environnements de recettes et de productions.
- Assurer la réutilisabilité du code.

3.2.3 Besoins non fonctionnels

Les besoins non fonctionnels décrivent les objectifs liés aux performances du système et aux contraintes de son environnement. Ses exigences techniques sont souvent exprimées sous forme d'objectifs spécifiques que doit atteindre le système. En plus des besoins fondamentaux, notre système doit répondre aux exigences suivantes :

- **La robustesse** : une utilisation incorrecte n'entraîne pas de dysfonctionnement.
- **Fiabilité** : c'est-à-dire la capacité d'un logiciel de rendre des résultats corrects quelles que soient les conditions d'exploitation. En font partie la tolérance aux pannes (logicielle ou matérielle).
- **Flexibilité** : c'est-à-dire facilité avec laquelle des fonctions peuvent être ajoutées, modifiées ou supprimées dans un programme.
- **L'extensibilité** : facilité du logiciel à s'adapter aux changements de spécification.
- **L'efficacité** : est le degré de réalisation des objectifs. On considère qu'une activité est efficace si les résultats obtenus sont identiques aux objectifs définis.
- **Sécurité**: Faculté d'un logiciel à être protégé contre des altérations.
- **Maintenabilité** : caractère d'un logiciel qui définit la facilité avec laquelle un défaut peut être localisé, identifié et corrigé.

3.3 Modélisation des besoins

Nous avons donc identifié nos acteurs, maintenant il s'agit de définir plus en détails le besoin de chaque acteur en répondant à la question : QUI devra faire QUOI ?

Le diagramme de cas d'utilisation représente les fonctionnalités du système c'est-à-dire l'ensemble des actions que devront réaliser nos acteurs.

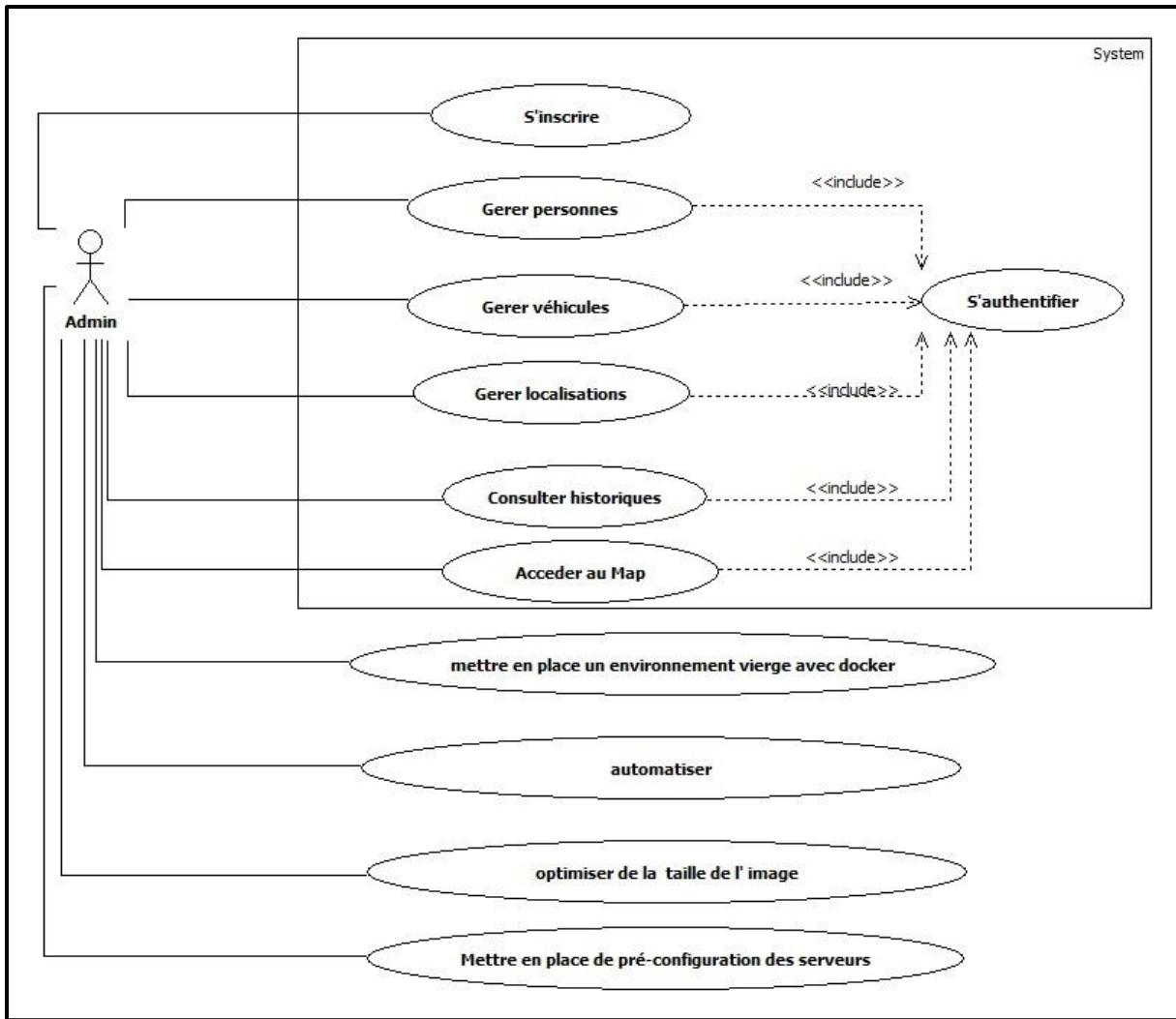


Figure 10: Diagramme de cas d'utilisation globale

3.4 Méthode de conception

La gestion du projet nécessite une méthodologie de travail claire. Cette méthodologie sert à spécifier comment nous pouvons développer un logiciel informatique. C'est-à-dire, les étapes que nous devons suivre pendant la période d'un projet.

3.4.1 Les méthodes agiles

« Les méthodes agiles sont des méthodologies essentiellement dédiées à la gestion de projets informatiques. Elles reposent sur des cycles de développement itératifs et adaptatifs en fonction des besoins évolutifs du produit.

Elles permettent notamment d'impliquer l'ensemble des collaborateurs ainsi que l'utilisateur dans le développement du projet. Ces méthodes permettent généralement de mieux répondre aux attentes du produit en un temps limité (en partie grâce à l'implication de

celui –ci) tout en faisant monter les collaborateurs en compétences. Ces méthodes constituent donc un gain en productivité ainsi qu'un avantage compétitif tant du côté utilisateur que du côté du fournisseur. » (Presse 2015).

Les valeurs fondamentales des méthodes agiles sont :

- Personnes et interactions plutôt que processus et outil : la communication est une notion fondamentale ;
- Logiciel fonctionnel plutôt que documentation complète ;
- Collaboration avec l'utilisateur plutôt que négociation de contrat ;
- Réagir au changement plutôt que suivre un plan : la planification initiale et la structure du logiciel doivent être flexibles afin de permettre l'évolution de la demande du produit tout au long du projet ;

Les méthodes agiles les plus connues sont :

- Extrême Programming (XP)
- Two Tracks Unified Process (2 TUP)
- Scrum [29]

Tableau 6: Les méthodes agiles

	Description	Points forts	Points faibles
Two Tracks Unified Process (2 TUP)	<p>S'inscrit autour de l'architecture.</p> <p>-Proposer un cycle de développement en Y.</p> <p>-Cible de projet de toutes tailles.</p>	<p>-Définit les profils des intervenants : les prototypes, les livrables, les plannings.</p> <p>-Reserve une large place pour la technologie et la gestion des risques.</p> <p>-Itératif.</p>	<p>-Superficiel sur les phases situées en amont et en aval du développement.</p> <p>-Ne propose pas des documents type.</p>
Extrême Programming (XP)	<p>Processus de développement complet.</p> <p>-Premier bouts de code</p>	<p>-Refactoring : Devoir faire la conception du système en continue pour améliorer sa</p>	<p>« -Focalisation sur l'aspect individuel du développement.</p> <p>-La documentation</p>

	<p>délivrés deux à trois semaines du démarrage.</p> <ul style="list-style-type: none"> -L'utilisateur fait partie de l'équipe. -Développement itératif au maximum. 	<p>performance et sa capacité de réponse aux changements.</p>	<p>ayant moins d'importance. »</p>
Scrum	<ul style="list-style-type: none"> -Sert à développer des offices, généralement en quelques mois. -Une vision est produite par une série d'itération d'un pois appelés des sprints. 	<p>Livrer fréquemment une application qui fonctionne.</p> <ul style="list-style-type: none"> -Besoins évolutifs -Satisfaire l'utilisateur en livrant tôt et régulièrement des logiciels utiles, qui offrent une véritable valeur ajoutée. 	<p>-Faible documentation et donc face à détourner.</p> <ul style="list-style-type: none"> -L'évolution des besoins amène les utilisateurs à exiger de plus en plus de fonctionnalités. -Les propriétaires du projet perdant le contrôle sur le sujet. <p>[30]</p>

3.4.2 Framework Scrum

a. Présentation de Scrum

C'est une méthodologie agile itérative basée sur des itérations de courtes durées appelées Sprints. Lorsqu'on dit Scrum, il faut comprendre les mots clés suivants :

- Responsable Produit : le représentant des clients et des utilisateurs. Il détermine ce qui doit être réalisé.
- Scrum Master : le responsable du déroulement du processus. Il garantit la motivation de l'équipe et l'efficacité de la collaboration entre les membres.

- Équipe projet : Les développeurs chargés de la construction du logiciel et d'en faire une démonstration.
- Backlog du produit : tout le travail est encadré par le Backlog. En effet, tout le projet est découpé en un ensemble de "User Stories" classés par priorité et listés dans le Backlog.
- Backlog du sprint : une sélection de tâches retenues du "Backlog du produit" pour construire l'objectif du sprint.
- Daily Meeting : c'est un point quotidien qui permet de mettre le point sur ce qui a été réalisé, les problèmes rencontrés et les objectifs de la journée.
- Démonstration du sprint : on la nomme aussi "Sprint Review". C'est une réunion programmée à la fin de chaque sprint durant laquelle l'équipe projet peut présenter son travail. Sur la base de cette démonstration, le responsable produit valide ce qui a été réalisé et détermine le nouvel objectif en se basant sur le Backlog du produit et si jamais il y a un ajout ou bien une modification dans ce dernier.

La figure 11 illustre à la fois les différents rôles de Scrum que nous venons d'exhiber et le déroulement du processus :

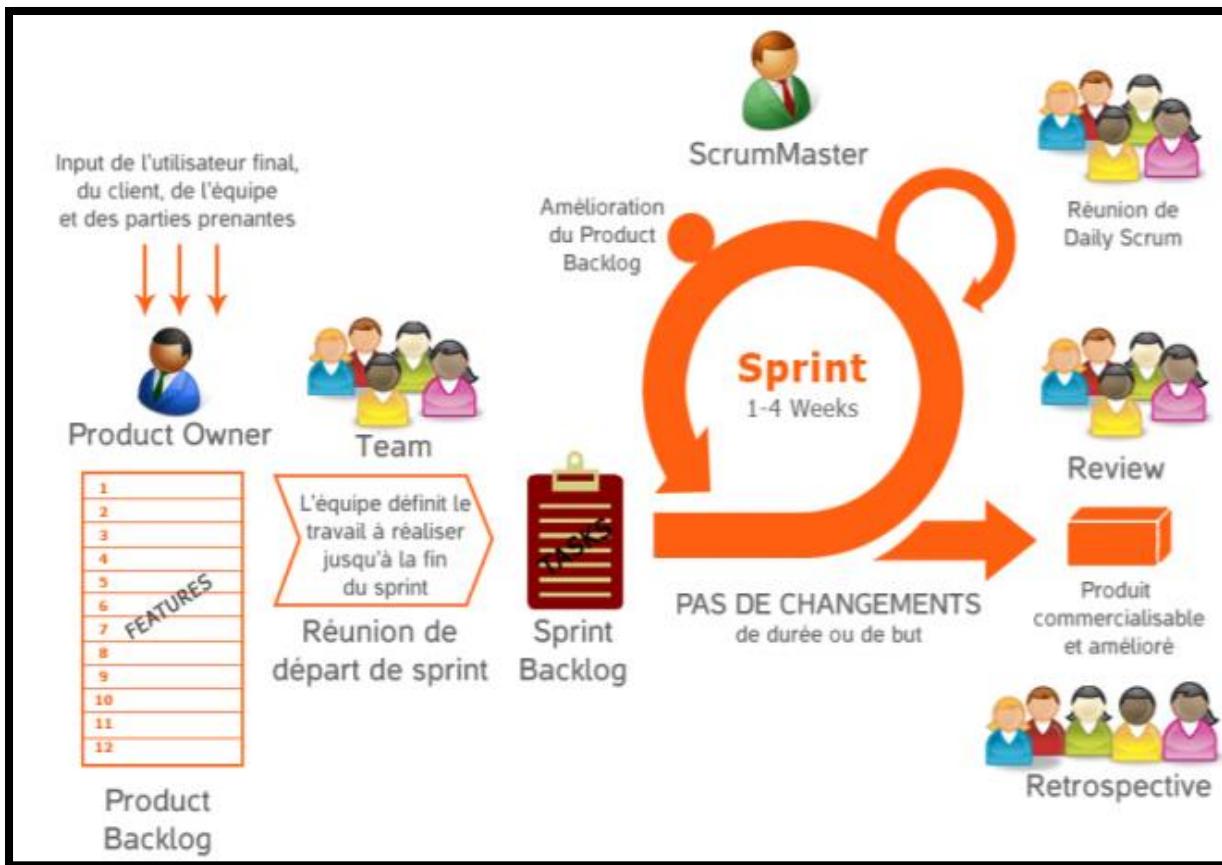


Figure 11: Processus SCRUM

b. Description des éléments de base de Scrum

✚ Principe de SCRUM :

Il existe 3 grands principes de Scrum qui sont :

- **L'équipe :**
 - Composée des développeurs (dont un Scrum master) et du propriétaire (Product Owner) ;
 - Elle est responsable du développement.
- **Des itérations :**
 - L'équipe livre périodiquement les versions du logiciel ;
 - La progression du projet est calculée sur les itérations.
- **Basée sur le temps :**
 - Les itérations sont bornées dans le temps ;
 - Toutes les réunions sont bornées dans le temps ;

- Le cout est borné dans le temps.

➤ **Présentation des rôles de Scrum**

Scrum est considéré comme un cadre ou « Framework » de gestion de projet. Ce cadre est constitué d'une définition des rôles, il s'articule autour des trois rôles qui sont principalement les suivants

➤ **Product Owner :**

Il représente à la fois le client et les utilisateurs. C'est donc lui qui définit les attentes et les besoins du projet. Ainsi, il définit les tâches permettant de répondre à ces besoins et il mettra en place leur priorisation.

➤ **Scrum Master :**

Le Scrum Master assure globalement le bon déroulement des programmes et protège l'équipe de tout problème extérieur. Il assure également l'organisation des réunions et la bonne application de la méthode agile.

➤ **Equipe ou Team Membres :**

Ce sont les personnes chargées de la réalisation du Sprint. Elle est composée des professionnels et caractérisée par une forte coopération et une haute communication entre les différents membres. Dans notre cas, les rôles sont répartis comme suit :



Figure 12: Team Membres

➤ **Présentation des évènements de Scrum**

❖ **Sprint :**

Le cœur de Scrum est le Sprint, qui a une boîte de temps (time-box), une durée, d'un mois ou moins au cours de laquelle un Incrément Produit « Fini » fonctionnel et potentiellement publiable est créé. Les sprints ont une durée cohérente durant la phase de développement. Un nouveau Sprint commence immédiatement après la conclusion du Sprint précédent.

❖ **Sprint Planning :**

Le travail à effectuer durant un Sprint est défini durant la réunion de Planification du Sprint (Sprint Planning). Ce plan est créé de manière collaborative par tous les membres de l'équipe Scrum.

❖ **Daily Meeting :**

La mêlée quotidienne (Daily Scrum) est un événement de 15 minutes (time-boxé) destiné à l'équipe de développement. La mêlée quotidienne est tenue tous les jours du Sprint. L'équipe de développement planifie le travail pour les prochaines 24 heures.

❖ **Sprint Review :**

Une revue de Sprint (Sprint Review) est tenue à la fin du Sprint pour inspecter l'incrément réalisé et adapter le Backlog Produit si nécessaire. Pendant la revue de Sprint, l'équipe Scrum et les parties prenantes échangent sur ce qui a été fait durant le Sprint. La Revue de sprint comprend les éléments suivants :

- Les participants incluent l'équipe Scrum et les principales parties prenantes invitées par le Product Owner ;
- Le Product Owner indique quels éléments du Backlog Produit ont été « Finis » et ceux qui n'ont pas été « Finis » ;
- L'équipe de développement discute de ce qui s'est bien passé pendant le Sprint, quels problèmes ont été rencontrés, et comment ces problèmes ont été résolus ;
- L'équipe de développement démontre le travail « Fini » et répond aux questions sur l'incrément ;

- Le Product Owner discute de l'état actuel du Backlog Produit tel qu'il est. Il ou elle projette les dates prévisionnelles et celles de livraison en fonction des progressions réalisées à ce jour (si nécessaire).

❖ **Sprint Rétrospective :**

La rétrospective de Sprint (Sprint Rétrospective) est une opportunité pour l'équipe Scrum de s'auto-inspecter et de créer un plan d'améliorations à adopter au cours du prochain Sprint.

- **Présentation des artefacts de Scrum**
- **Backlog du produit :**
- Tout le travail est encadré par le Backlog. En effet, tout le projet est découpé en un ensemble de "User Stories" classés par priorité et listés dans le Backlog.
- **Backlog du sprint :**
 - Une sélection de tâches retenues du "Backlog du produit" pour construire l'objectif du sprint. **Incrément :**
 - L'incrément est constitué des éléments du Backlog produit « Finis » pendant le sprint ainsi que de la valeur cumulative des incréments livrés dans les sprints précédents. [31]

3.5 Planning de travail

3.5.1 Backlog de produit

Le Backlog de produit correspond à une liste priorisée des besoins et des exigences du produit. Les éléments du Backlog de produit, appelé aussi les histoires utilisateurs, sont formulés en une ou deux phrases décrivant de manière claire et précise la fonctionnalité désirée par l'utilisateur, généralement, écrit sous la forme suivante : en tant que X, je veux Y, afin de Z.

Dans le tableau 5, nous présentons la liste des histoires utilisateurs ainsi que leurs estimations. Nous avons listé les besoins et les exigences du produit selon l'ordre croissant de priorité, ce tableau énumère les champs suivants :

- ✓ **Identifiant** : c'est un nombre unique et auto-incrémenté pour chaque User Story.
- ✓ **Cas** : la tâche à réaliser.
- ✓ **User Story** : c'est une description courte de la tâche à réaliser.

- ✓ **Priorité** : c'est l'importance attribuée par le Product Owner à cette tâche (remarque : pour que le tableau soit lisible et clair, nous allons nommer la " très haute priorité " par l'abréviation " TH ", la " Haute priorité " par l'abréviation " H ", la " moyenne priorité " par l'abréviation " M ", la " basse priorité " par l'abréviation "B ").
- ✓ **Story Point** : la durée nécessaire pour réaliser une tache.

Tableau 7: Backlog de produit

ID	Cas	USER STORY	Priorité	STORY POINT (j)
1	Rechercher et développer sur la culture, l'historique et les outils DevOps	En tant qu'administrateur je veux rechercher et développer sur la culture, l'historique et les outils DevOps.	TH	5
2	Préparer une présentation PowerPoint	En tant qu'administrateur je veux préparer une présentation PowerPoint.	TH	4
3	S'authentifier	En tant qu'administrateur je veux s'authentifier pour sécuriser l'application.	TH	3
4	S'inscrire	En tant qu'administrateur je veux m'inscrire pour bénéficier des services offerts par l'application.	B	4
5	Gérer les véhicules	En tant qu'administrateur je veux gérer les véhicules.	H	5

6	Gérer les personnes	En tant qu'administrateur je veux gérer les personnes.	H	4
7	Gérer les localisations	En tant qu'administrateur je veux accéder au Maps et gérer localisation.	H	8
8	Consulter historique des trajets	En tant qu'administrateur je veux consulter l'historique des trajets	M	6
9	Accéder au Maps	En tant qu'administrateur je veux accéder au Maps	B	3
10	Mise en place d'un environnement conteneurisé avec Docker	En tant qu'Admin, je veux préparer un environnement conteneurisé et construire une image Docker de la solution existante	H	10
11	Automatiser	En tant qu'Admin je veux mettre en place un pipeline CI/CD (en utilisant Jenkins).	H	8
12	Optimiser de la taille de l'image	En tant qu'Admin je veux réduire la taille de l'image	H	6
13	Mise en place d'une solution de pré-configuration des serveurs	En tant qu'Admin je veux avoir un environnement préconfiguré.	H	10

3.6.2 Planification des sprints

La réalisation de tout projet passe par l'établissement et surtout le respect d'un planning prévisionnel bien défini en accord avec le groupe de pilotage.

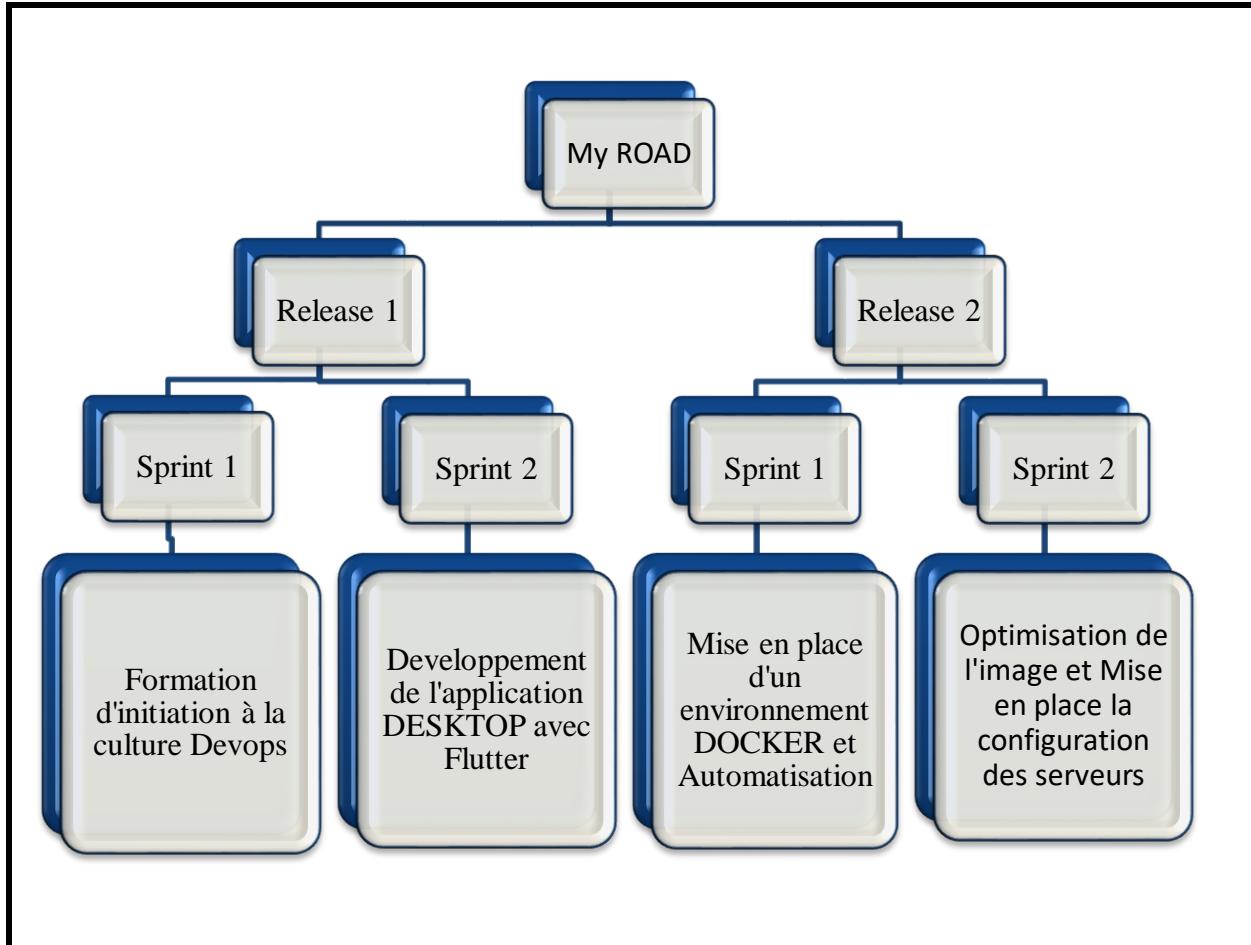


Figure 13: Planification des sprints

Diagramme de Gantt

Le diagramme de Gantt est un outil utilisé pour la gestion du projet. Il permet de représenter l'état d'avancement des différentes fonctionnalités qui constituent un projet.

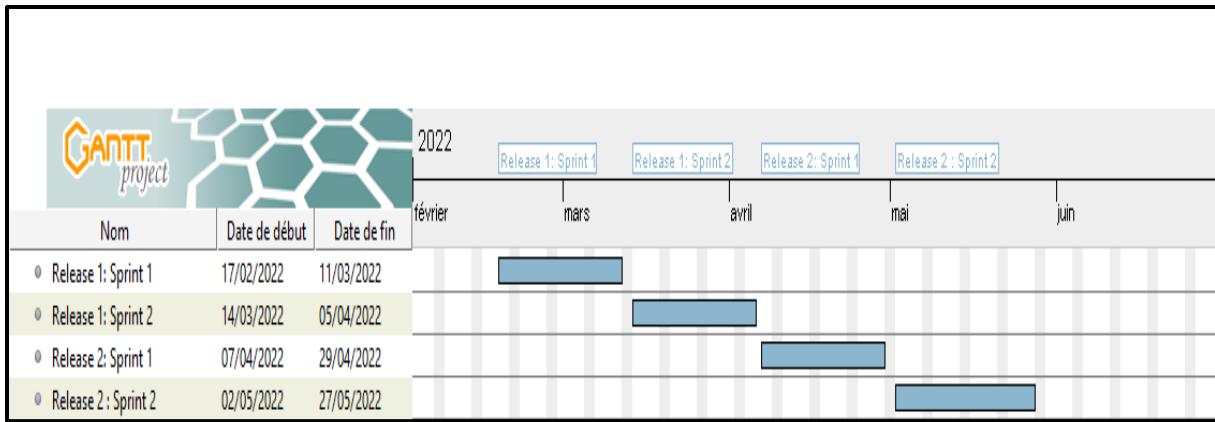


Figure 14: Diagramme de Gantt

3.6 Conclusion

A fin de ce chapitre, on a bien étudié nos besoins. On a bien présenté l'ensemble de fonctionnalités du futur système par modélisation et conception d'un diagramme de cas d'utilisations et diagramme de classe globale, ainsi qu'on a réalisé les prototypes et l'architecture matérielle et logiciels.

Chapitre 4 : Mise en œuvre de Release 1

Introduction

Dans ce chapitre, nous présentons la réalisation du premier release, en organisant le travail sur trois phases principales qui sont l'analyse, la conception, et la réalisation.

4.1 Sprint 1

4.1.1 Backlog du Sprint

Tableau 8: Backlog de sprint 1(Release1)

ID	Cas	USER STORY	Priorité	STORY POINT (j)
1	Rechercher et développer sur la culture, l'historique et les outils DevOps.	En tant qu'administrateur je veux rechercher et développer sur la culture, l'historique et les outils DevOps.	TH	3
2	Préparer une présentation PowerPoint.	En tant qu'administrateur je veux préparer une présentation PowerPoint.	TH	3

4.1.2 Exécution de sprint

4.1.2.1 Préparation à la formation

Dans la startup Accent l'équipe s'occupe de suivre plusieurs processus qui sont contrôlés par un responsable processus pour avoir une équipe bien organisée. Parmi les processus appliqués dans Accent nous avons la réalisation d'une formation d'initiation chaque fois qu'un nouveau membre avec une nouvelle technologie, peut faire une formation pour le partage avec l'équipe et donc il peut préparer un livrable et avoir l'opportunité de présenter la formation pour toute l'équipe.

Pour ce fait, j'ai commencé par la préparation d'un plan qui contient les étapes à suivre pour élaborer une formation riche et simple au même temps :

- Mesurer le degré de motivation des participants et leur niveau de connaissance en DevOps
- Déterminer le thème et le sujet de la formation.
- Faire l'inventaire des informations et de la documentation nécessaire.
- Agencer logiquement les contenus.
- Préparer la présentation et un livrable.

Suite à la survie de ce plan, j'ai fini la préparation de la formation qu'elle a la description suivante :

Titre : Initiation à la culture DevOps

Date : 10/03/2022

Nombre des participants : L'Equipe et l'Encadrant Mme « Chaima GHAZOUANI » avec le plan suivant :

- L'historique de DevOps
- Avant et après l'approche DevOps
- DevOps et Agile
- Les bonnes pratiques de DevOps
- Les bénéfices de DevOps
- Le huit de DevOps

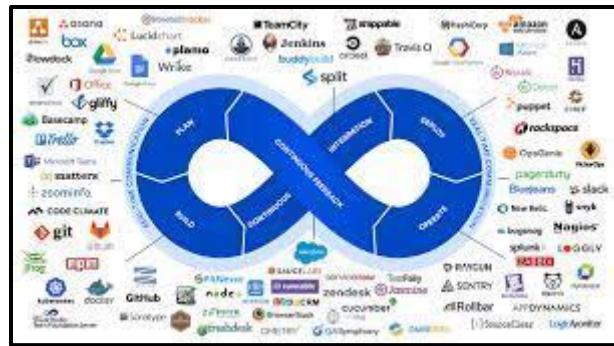


Figure 15: Le huit de DevOps

4.1.2.2 Conclusion de la formation

On ne peut plus nier que le monde du digital est en constante mutation et développement rapide. Les besoins et exigences évoluent sans cesse et le développement doit être de plus en plus rapide. Les équipes de développement logiciel et celles chargées du déploiement doivent réagir rapidement. C'est dans ce contexte que le terme DevOps, qui désigne une philosophie de travail. Une approche, qui favorise une meilleure communication entre les équipes chargées du développement (Dev) et l'équipe opérationnelle chargée de l'exploitation des systèmes (Ops). C'est un ensemble de bonnes pratiques, un assemblage de processus et d'outils visant à réduire le "time-to-market", autrement dit le temps entre le développement d'un nouveau produit ou d'une fonctionnalité et la livraison aux utilisateurs finaux, tout en garantissant la qualité des produits. C'est pour cela, nous avons préparé une étude complète pour appliquer l'approche DevOps dans la startup et dans le but d'assurer la transformation d'Accent de Dev à DevOps.

4.1.3 Dérroulement de sprint

4.1.3.1 Sprint Review

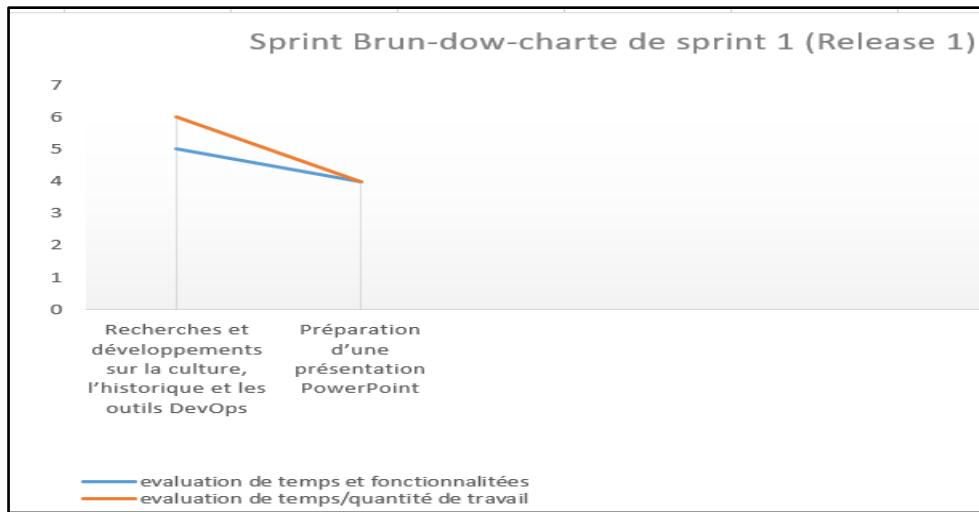


Figure 16: Sprint Burndown chart de sprint 1 (Release 1)

4.1.3.2 Sprint Rétrospective

À la fin du sprint, une réunion de présentation a été organisée. C'est l'occasion de présenter les réalisations réalisées lors du sprint.

Tableau 9: Tableau de validation de sprint 1 (Release 1)

Cas de test	Démarche de test	Comportement attendu	Résultat
Rechercher et développer sur la culture, l'historique et les outils DevOps	L'administrateur lance une recherche rapide à partir de mots-clés	Résultat de recherche est affiché.	Conforme
Préparer une présentation avec PowerPoint	. L'Admin prépare une présentation avec PowerPoint.	Peut-être il y'a une information obligatoire manquante.	Conforme

4.2 Sprint 2

4.2.1 Backlog du sprint

Tableau 10: Backlog de produit de sprint 2 (Release 1)

ID	Cas	USER STORY	Priorité	STORY POINT (j)
1	S'authentifier	En tant qu'administrateur je veux s'authentifier pour sécuriser l'application.	TH	3
2	S'inscrire	En tant qu'administrateur je veux m'inscrire pour bénéficier des services de l'application.	B	4
3	Gérer les véhicules	En tant qu'administrateur je veux gérer les véhicules.	H	5
4	Gérer les personnes	En tant qu'administrateur je veux gérer les personnes.	H	4
5	Gérer les localisations	En tant qu'administrateur je veux accéder au Maps et gérer localisation.	H	8
6	Consulter historique des trajets	En tant qu'administrateur je veux consulter l'historique des trajets	H	6
7	Accéder au Map	En tant qu'administrateur je veux accéder au Map		3

4.2.2 Analyse

4.2.2.1 Identification des acteurs et des cas d'utilisations

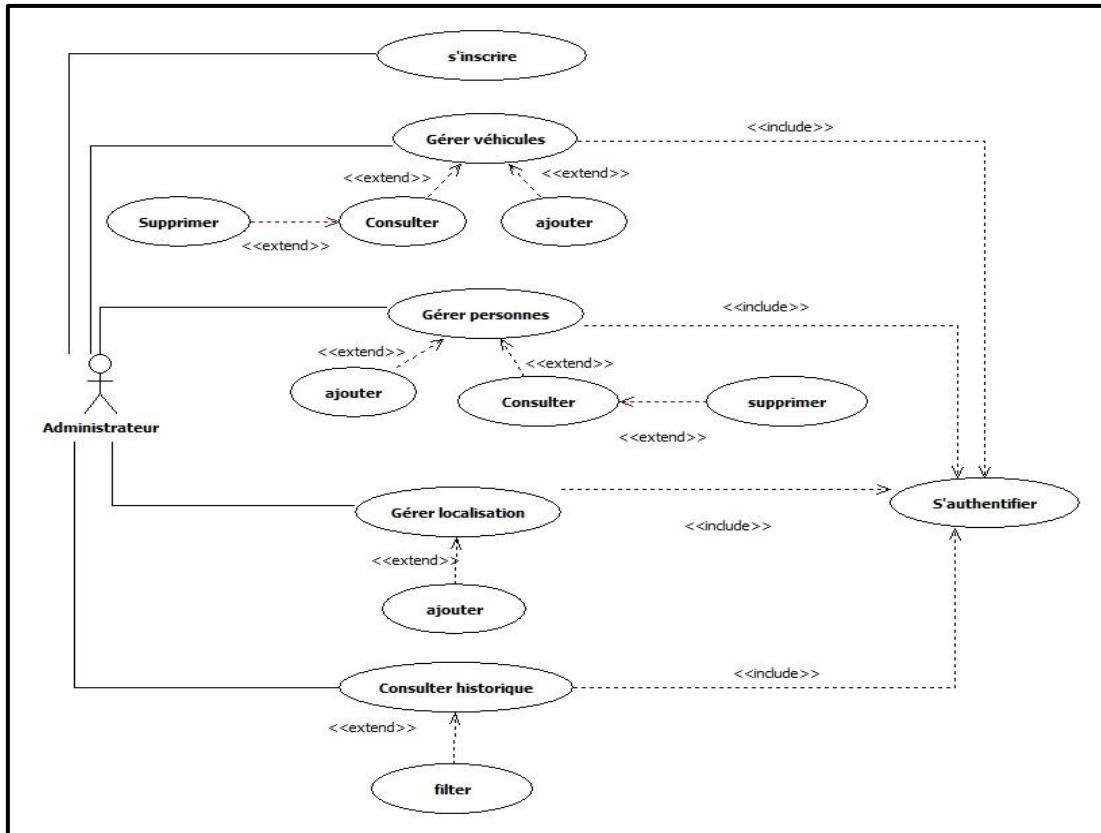


Figure 17: Diagramme de cas d'utilisations de sprint 2 (Release 1)

4.2.2.2 Descriptions des cas d'utilisations

- Description de cas d'utilisation «**s'authentifier**»

Tableau 11: Description de cas "s'authentifier"

Acteur	L'administrateur
Objectif	Lors de l'accès au site, l'administrateur doit s'authentifier pour accéder aux différentes fonctionnalités du site.
Pré-condition(s)	<ul style="list-style-type: none"> - L'administrateur doit être inscrit. - L'administrateur doit connaître ses identifiants.

	<ul style="list-style-type: none"> - L'administrateur doit accéder à la page d'accueil.
Post-condition(s)	L'administrateur accède à son espace personnel.
Scénario-Nominal	<ol style="list-style-type: none"> 1. L'administrateur remplit le formulaire 2. Le système vérifie les champs 3. Le système exécute les requêtes de connexion. 4. La base de données collecte les données 5. La base de données renvoi des résultats. 6. Le système redirige l'administrateur vers espace privé.
<i>Scénario(s) alternatif(s)</i>	<ol style="list-style-type: none"> 3. a /Le système renvoie un message d'erreur « veuillez vérifier la saisie 6. a / Le système renvoie un message d'erreur « les données saisies sont invalides »

- Description de cas d'utilisation « **ajouter véhicule** »

Tableau 12:Description de cas « Ajouter véhicule »

Acteur	L'administrateur
Objectif	L'administrateur ajoute un véhicule.
Pré-condition(s)	<ul style="list-style-type: none"> - L'administrateur doit être authentifié. - L'administrateur doit accéder à l'interface liste des véhicules. - L'administrateur choisit d'ajouter un véhicule.
Post-condition(s)	Véhicule est ajoutée avec succès.
Scénario-Nominal	<ol style="list-style-type: none"> 1. L'administrateur rempli le formulaire d'ajout. 2. L'administrateur valide la saisie 3. Le système vérifie les champs. 4. Le système ajoute les données à la base de données. 5. Le système doit afficher messages succès. 6. Le système réaffiche la liste des véhicules.

Scénario(s) alternatif(s)	3. a/ Le système affiche un message d'erreur.
--------------------------------------	---

- Description de cas d'utilisation « **ajouter une localisation** »

Tableau 13: Description de cas d'utilisation "ajouter une localisation"

Acteur	L'administrateur
Objectif	L'administrateur localise une personne/véhicule.
Pré-condition(s)	<ul style="list-style-type: none"> - L'administrateur doit être authentifié. - L'administrateur doit accéder à la liste des personnes.
Post-condition(s)	Localisation ajoutée avec succès.
Scénario-Nominal	<ol style="list-style-type: none"> 1. L'administrateur choisit une personne/véhicule à localiser. 2. Le système affiche le Map 3. L'administrateur clique sur un point sur le Map. 4. Le système affiche le trajet.
Scénario(s) alternatif(s)	Vide

- Description de cas d'utilisation « **supprimer une véhicule** »

Tableau 14: Description de cas d'utilisation "supprimer véhicule"

Acteur	L'administrateur
Objectif	L'administrateur supprime un véhicule.
Pré-condition(s)	<ul style="list-style-type: none"> - L'administrateur doit être authentifié. - L'administrateur doit accéder à la liste des véhicules.
Post-condition(s)	Véhicule supprimé avec succès

Scénario-Nominal	<ol style="list-style-type: none">1. L'administrateur sélectionne un véhicule.2. L'administrateur demande de supprimer le véhicule sélectionné.3. Le système demande une confirmation de suppression.4. L'administrateur confirme la suppression du véhicule.5. Le système supprime le véhicule de la base de données et affiche le message suivant : « véhicule supprimé avec succès ».
Scénario(s) alternatif(s)	4. a/ L'administrateur annule la suppression du véhicule.

4.2.3 Conception

4.2.3.1 Conception statique

- Diagramme de classe

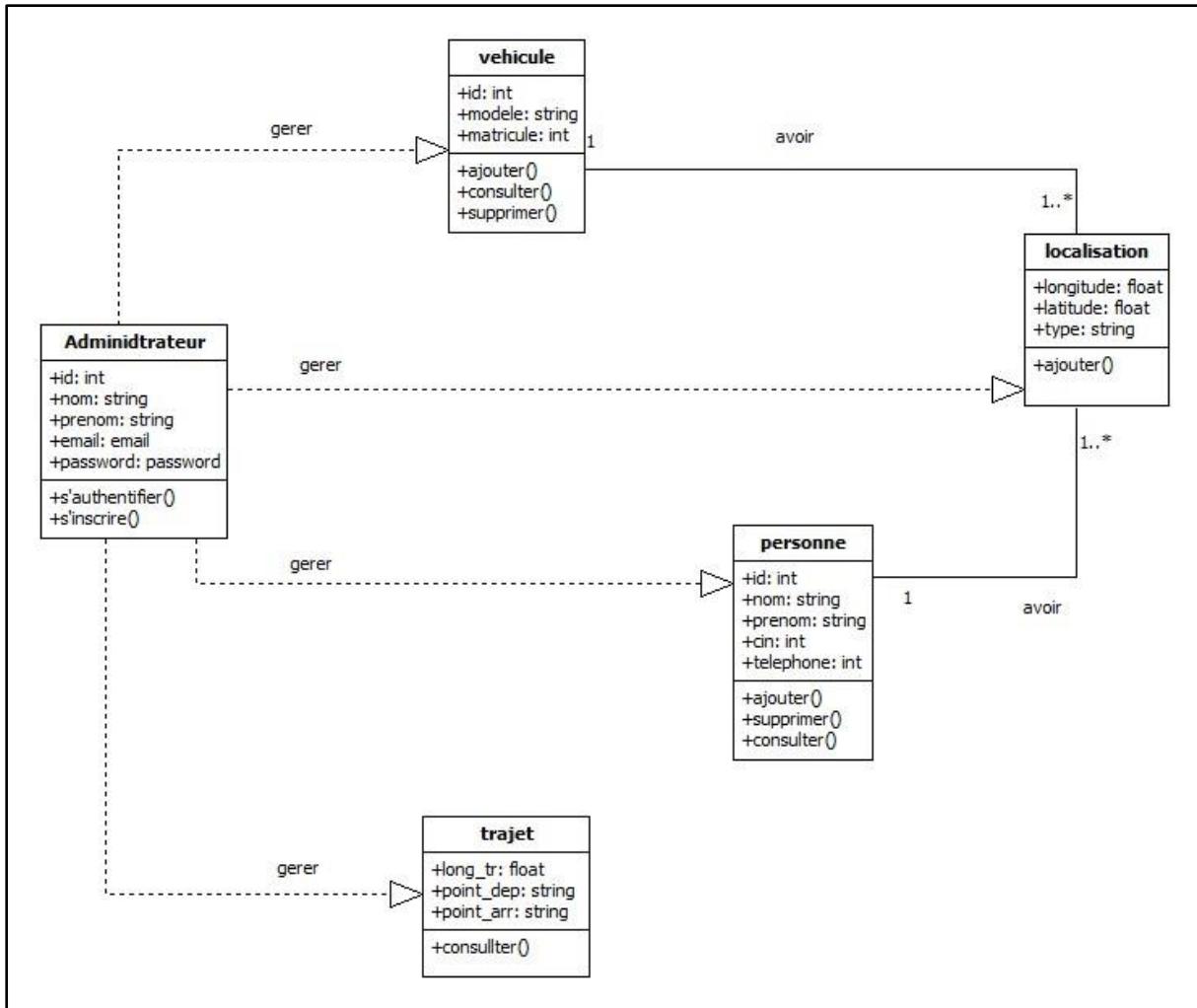


Figure 18: Diagramme de classe de sprint 2 (Release 1)

4.2.3.2 Conception dynamique

- Diagrammes de séquences
- Diagramme de séquence de cas « **S'authentifier** »

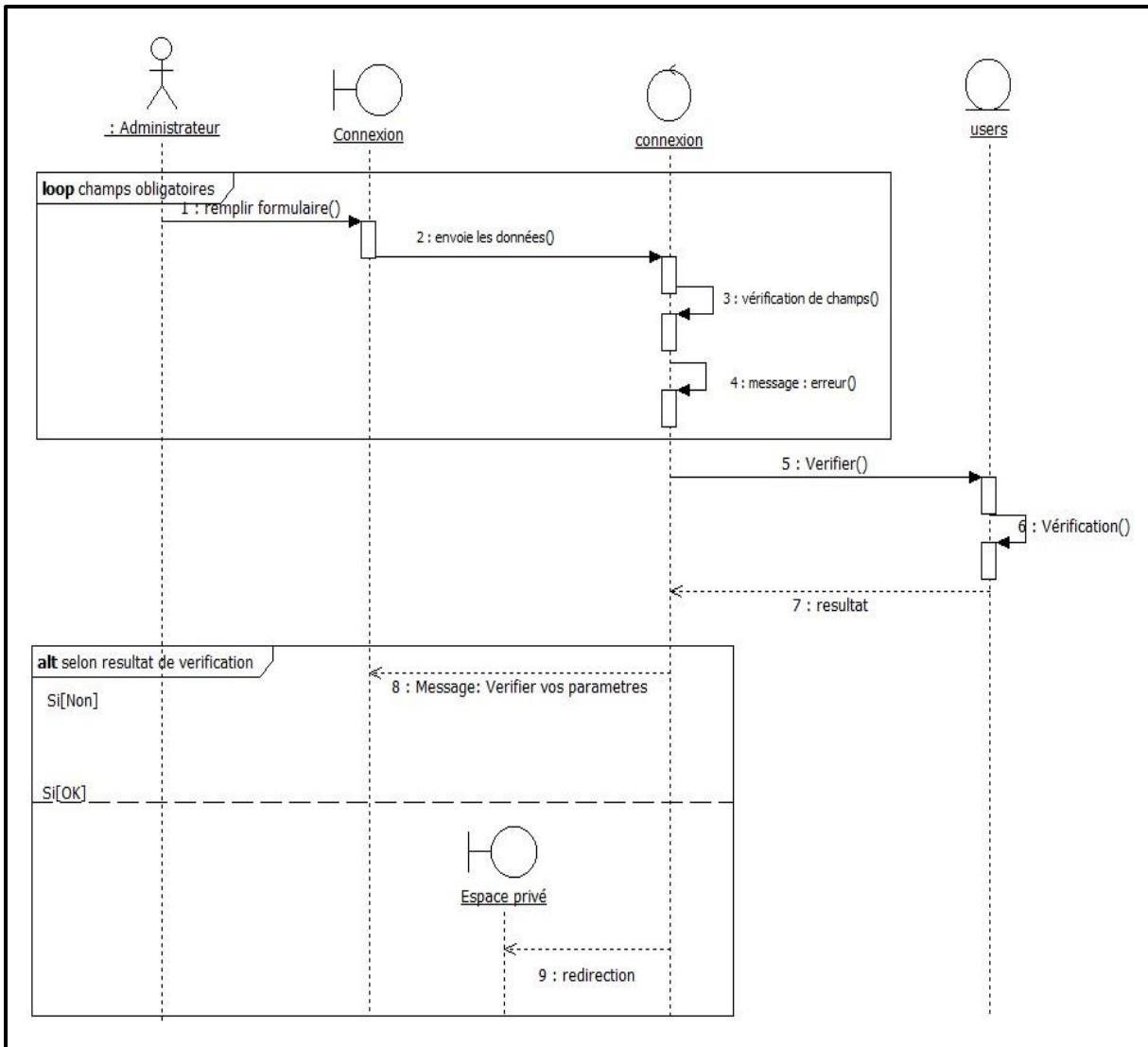


Figure 19: Diagramme de séquence s'authentifier"

- Diagramme de séquence de cas « **ajouter une personne** »

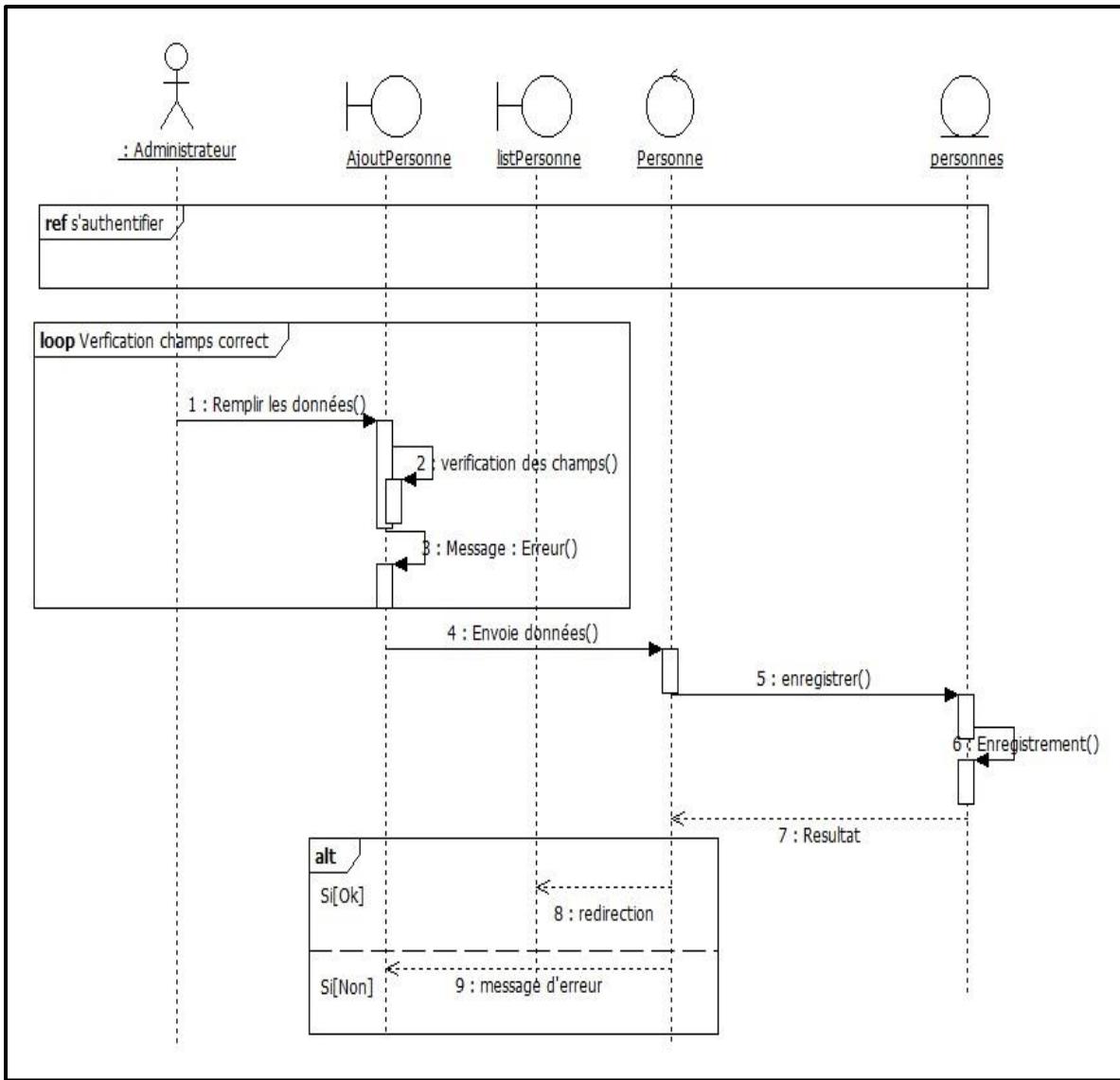


Figure 20: Diagramme de séquence "ajouter une personne"

- Diagramme de séquence de cas « **supprimer une personne** »

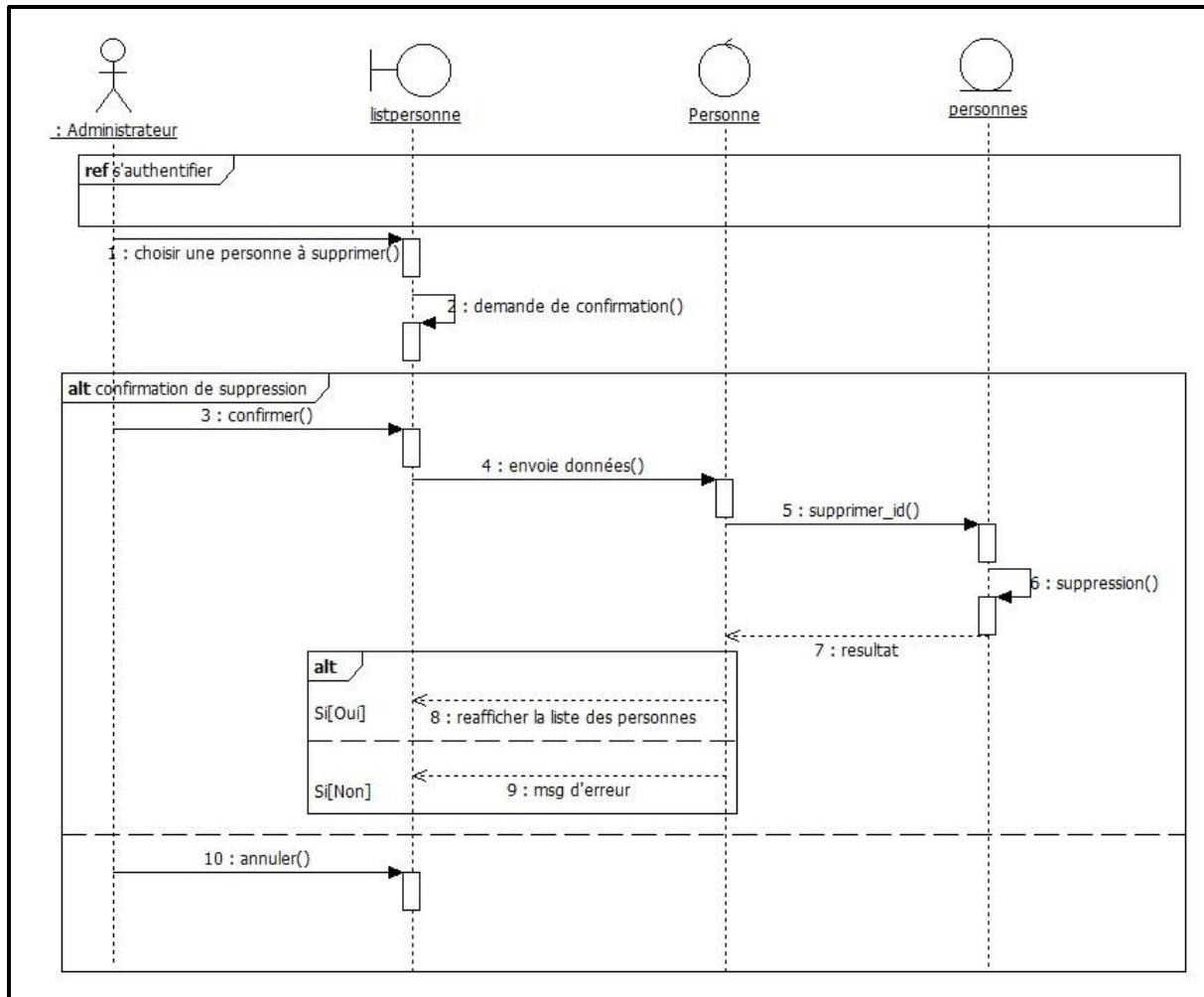


Figure 21: Diagramme de séquence de "supprimer une personne"

4.2.4 Réalisation

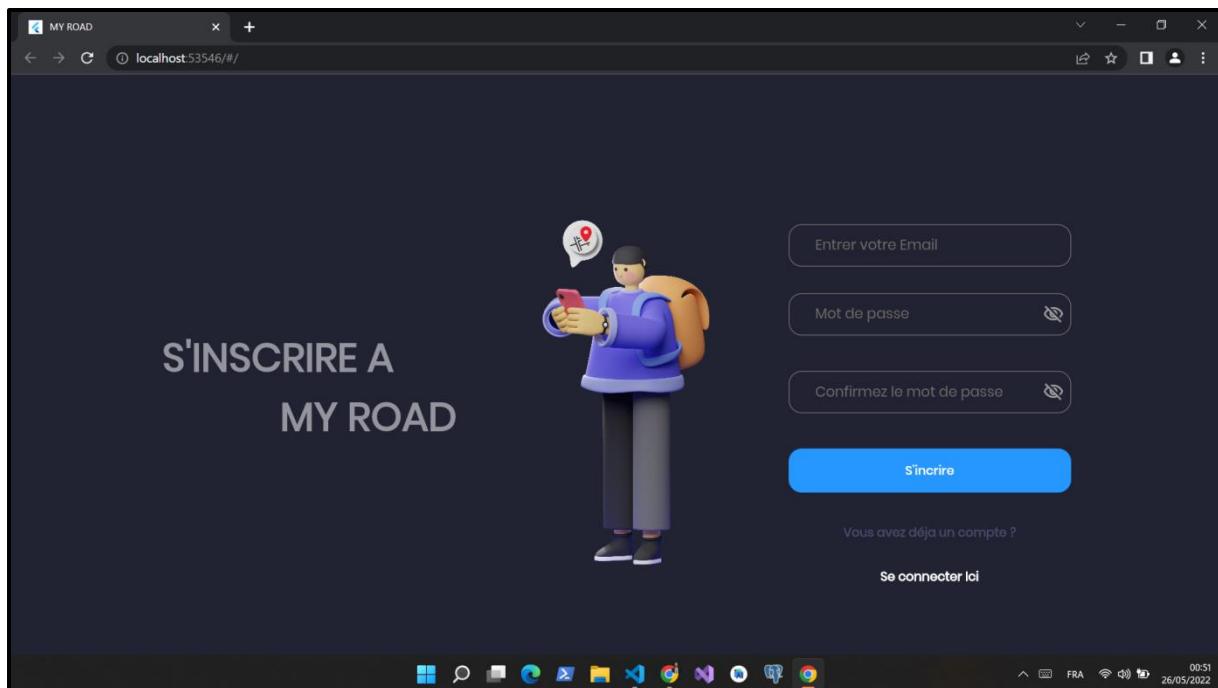


Figure 22: Interface d'inscription

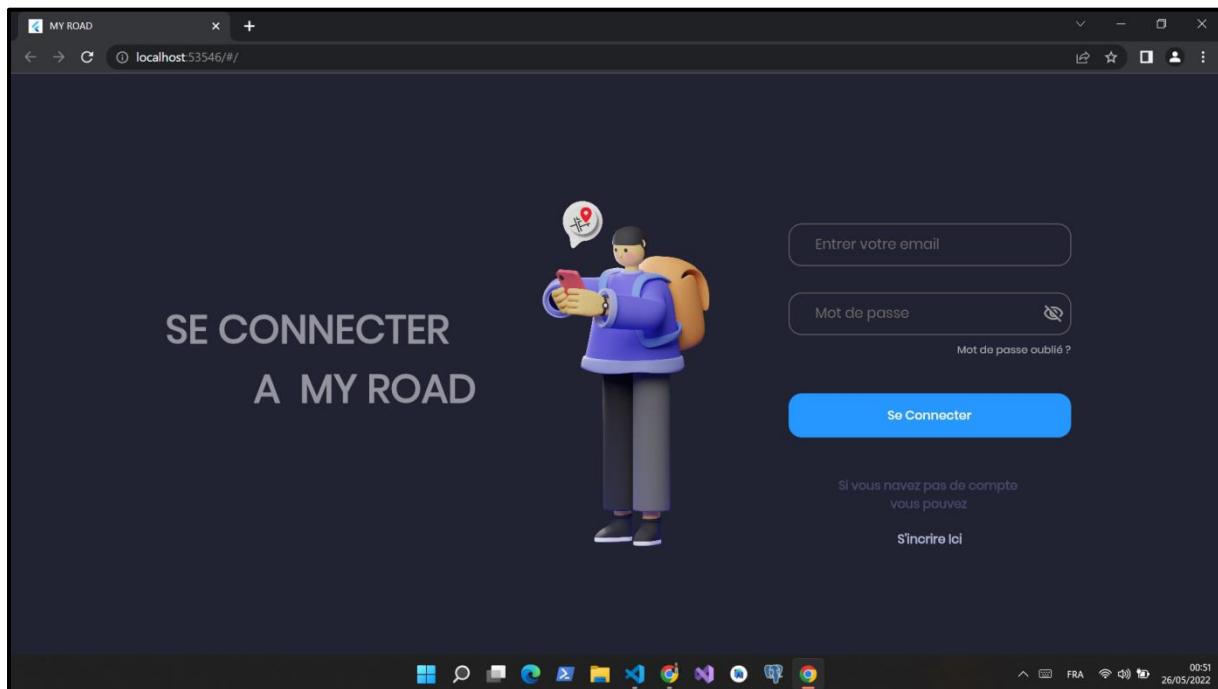


Figure 23: Interface d'authentification

Chapitre 4 : Mise en œuvre de release 1

Nom complet	CIN	Numéro de téléphone
Ahmed trabelsi	09526647	55478147
Oussema jemai	00587441	28774158
Oumayma mejri	09526992	22875441
Wiem bjeoui	12478859	99281596
Wassim mejri	05874123	50730037
Ali trabelsi	09587741	95478996

Figure 24: Interface liste des personnes

Etes-vous sûr de vouloir supprimer cette personne ?

Figure 25: Interface de suppression d'une personne

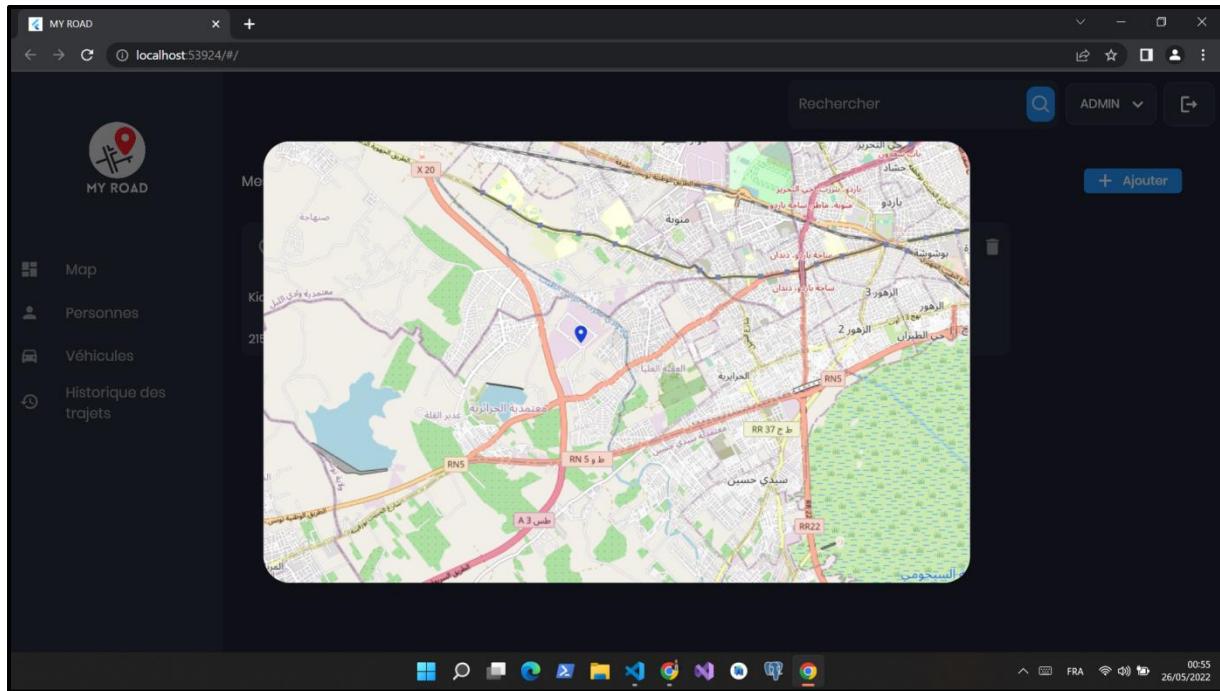


Figure 26: Interface d'ajout d'une localisation

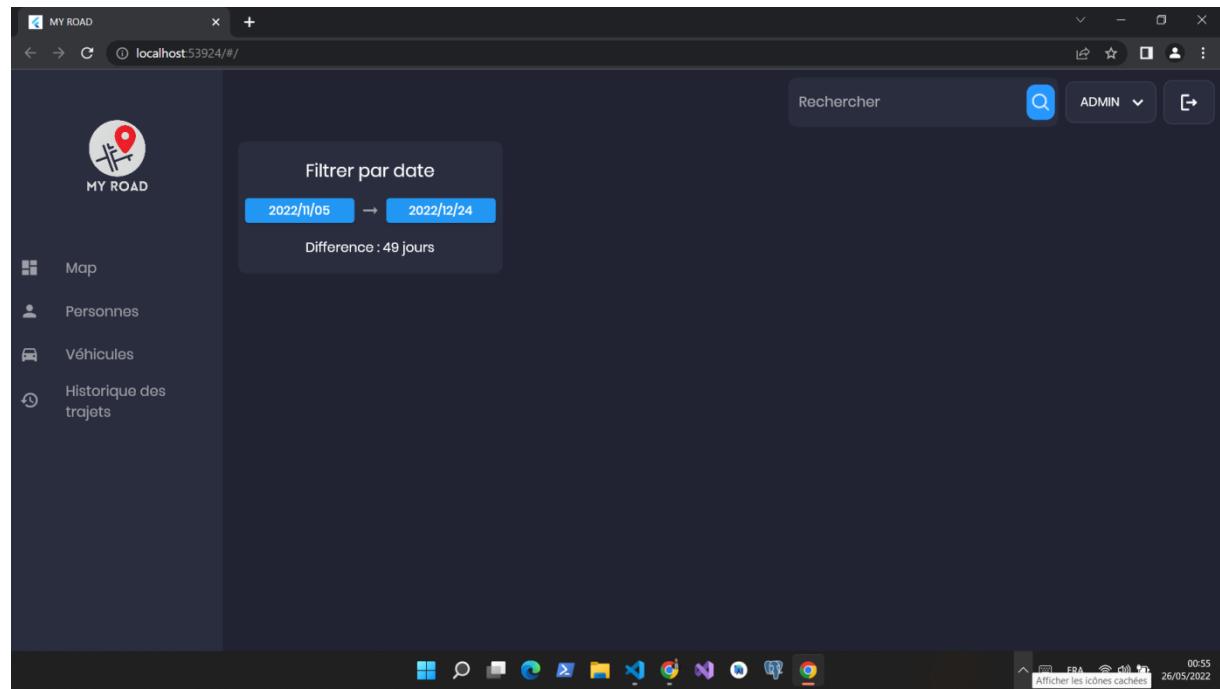


Figure 27: Interface d'historique des trajets (filtre par date)

4.2.5 Déroulement de sprint

4.2.5.1 Sprint Review

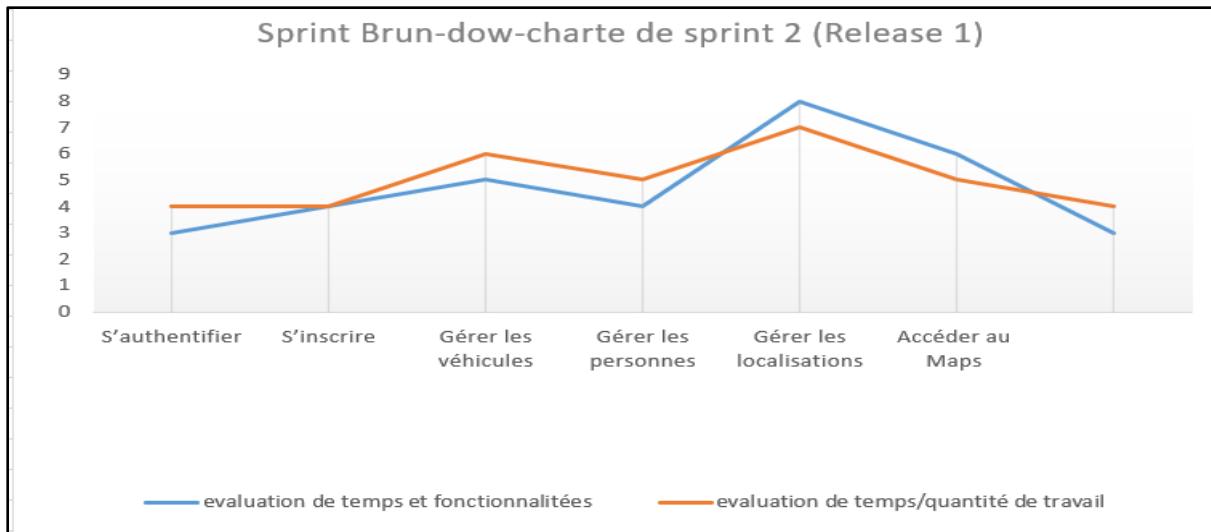


Figure 28: Sprint Brun-down-charte de sprint 2 (Release 1)

4.2.5.2 Sprint rétrospective

Tableau 15: Tableau de validation de sprint 2 (Release 1)

Cas de test	Démarche de test	Compartiment attendu	Résultat
S'inscrire	Le client de l'application va remplir un formulaire d'inscription.	Message d'erreur s'il y'a une information obligatoire manquante	Conforme
S'authentifier	Le client de l'application va remplir un formulaire d'authentification.	L'espace personnel est affiché.	Conforme
Gérer véhicules	L'administrateur va consulter la liste des véhicules, ainsi qu'il va ajouter ou supprimer un véhicule.	La gestion se réalise sans erreur.	Conforme
Gérer personnes	L'administrateur va consulter la liste des personnes, ainsi qu'il va ajouter ou supprimer une personne.	La gestion se réalise sans erreur.	Conforme

Gérer localisations	L'administrateur demande d'ajouter une localisation, et l'enregistre dans la base.	La gestion se réalise sans erreur.	Non Conforme
Afficher de Maps	L'administrateur va demander d'afficher le Maps.	Maps est affiché.	Conforme

Conclusion

Dans ce chapitre nous avons étudié les deux premiers sprints. Le chapitre suivant sera consacré aux deux sprints de déploiement.

Chapitre 5 : Mise en œuvre de Release 2

Introduction

Dans ce chapitre, nous présentons la réalisation du deuxième release, en organisant le travail sur trois phases principales qui sont l'analyse, l'architecture, et la réalisation.

5.1 Sprint 1

5.1.1 Backlog du Sprint

ID	Cas	USER STORY	Priorité	STORY POINT (j)
10	Mise en place d'un environnement conteneurisé avec Docker	En tant qu'Admin je veux préparer un environnement conteneurisé et construire une image Docker de la solution existante	H	10
11	Automatiser	En tant qu'Admin je veux mettre en place un pipeline CI/CD.	H	8

5.1.2 Architecture de l'environnement

5.1.2.1 Architecture docker

Docker utilise une architecture client-serveur. Le client Docker parle au démon Docker, qui est responsable du gros travail de construction, exploitation et distribution de conteneurs.

Le client et le démon Docker peut fonctionner sur le même système ou sur différents systèmes. Ils utilisent des API REST pour communiquer via des sockets UNIX ou à travers une interface réseaux.

- Le démon Docker : Un serveur appelé démon, engine ou docker écoute les requêtes de l'API Docker et gère les objets Docker tels que les images, les conteneurs, les réseaux et les volumes. Un démon peut également communiquer avec d'autres démons pour gérer les services Docker.
- Le client Docker : Le client Docker est le principal moyen par lequel les utilisateurs interagissent. Lorsque nous utilisons des commandes telles que docker run, le client envoie ces commandes à docker, qui les exécute en utilisant l'API Docker. Le client Docker peut communiquer avec plusieurs démons.
- REST-API : Une API REST permet de spécifier un ensemble d'interfaces qui permettent à d'autres programmes de communiquer et de donner des instructions au démon Docker. L'un de ces programmes est le terminal du système d'exploitation.
- Registre Docker : Un registre Docker est une bibliothèque d'images qui peut être publique ou privée. Il peut se trouver sur le même serveur que le démon Docker ou le client Docker ou bien sur un serveur totalement séparé. [32]

La figure 29 permet de mieux comprendre cette architecture.

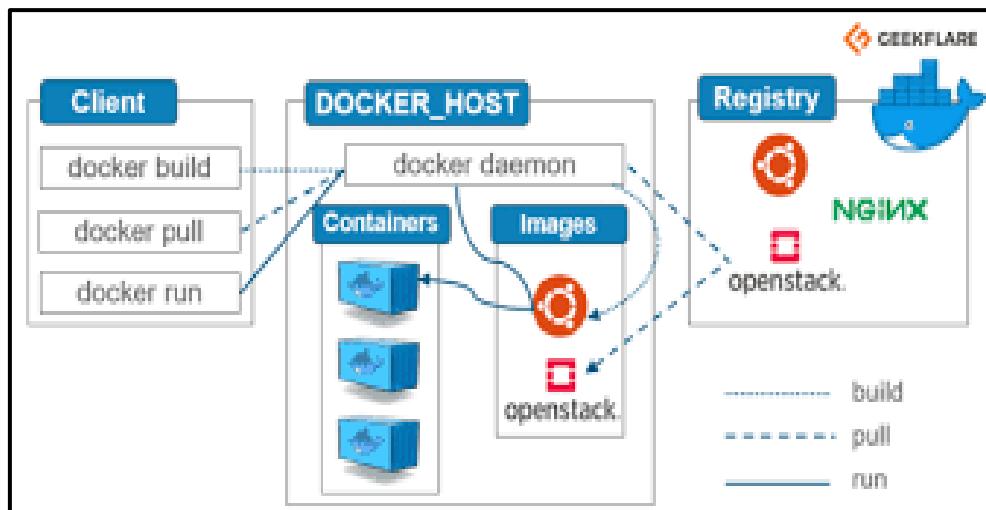


Figure 29: Architecture docker

5.1.2.2 Architecture conteneurisée

Nous sommes maintenant en disposition d'avoir une image globale sur l'architecture cible et de la mettre en place.

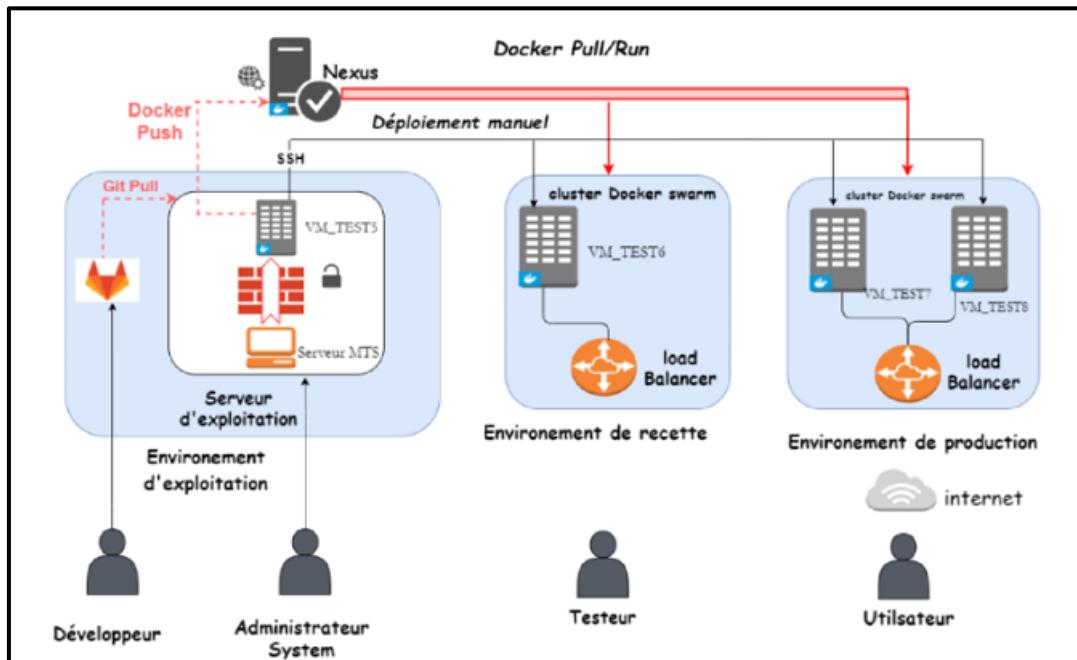


Figure 30: Architecture conteneurisé

La figure 29 présente l'architecture de l'environnement, que nous avons mis en place avec Docker. Cet environnement facilite la création, le déploiement et l'exécution d'applications grâce aux commandes Docker. De point de vue technique, l'architecture de base de notre solution a subi le changement suivant :

* **Cluster Docker swarm:** Le Cluster définit un groupe de ressources, telles que des serveurs ou des ordinateurs. Ce groupe se comporte comme un seul et même système. Il en résulte ainsi une disponibilité élevée, et parfois, des fonctions de traitement parallèle et d'équilibrage de la charge.

Swarm est l'orchestrateur développé par Docker. Il est uniquement dédié à la gestion de conteneurs Docker.

* **Nexus :** C'est un composant qui permet le stockage des images docker et leur récupération. Dans cette phase l'administrateur système aura la possibilité de créer une image et de faire un push-sur Nexus et un « pull - vers les machines de recette et de production. [33]

5.1.2.3 Architecture Jenkins

Dédié aux DevOps, Jenkins est un outil d'intégration continue open source (sous licence MIT) développé en Java. A chaque modification de code d'une application dans le gestionnaire de configuration, Jenkins se charge automatiquement de la recompiler, et de la tester. Pour cette seconde étape, Jenkins intègre le framework de test open source JUnit. En cas d'erreur détectée, Jenkins alerte le développeur afin qu'il résolve le problème. Un processus évidemment des plus avantageux dans le cadre d'un projet de développement.

Fork de l'outil Hudson, Jenkins s'adosse à un serveur de servlets comme Apache Tomcat ou peut reposer sur son propre serveur web embarqué. Accessible via un navigateur web, il est compatible avec les systèmes de gestion de versions les plus populaires comme Git ou Subversion. De manière standard, il supporte les pipelines d'intégration continue (CI) basés sur les outils de build Apache Ant et Apache Maven.

Les plugins Jenkins les plus populaires :

- Via son Update Center, Jenkins propose 1 500 plugins permettant d'étendre son environnement d'intégration continue. Parmi les plus populaires figurent notamment :
- **Dashboard View Plugin** qui est conçu pour suivre le statut des tâches,
- **Monitoring Plugin** qui mesure la performance des jobs,
- **Kubernetes Plugin** qui gère le déploiement des agents Jenkins sur une infrastructure Kubernetes,
- **Multijob Plugin** qui est taillé pour orchestrer l'exécution de tâches complexes de manière séquentielle,
- **GitHub API** qui planifie et déclenche des builds en s'appuyant sur du code extrait de GitHub,
- **Git Client** fournit une API Git pour les plugins Jenkins. [34]

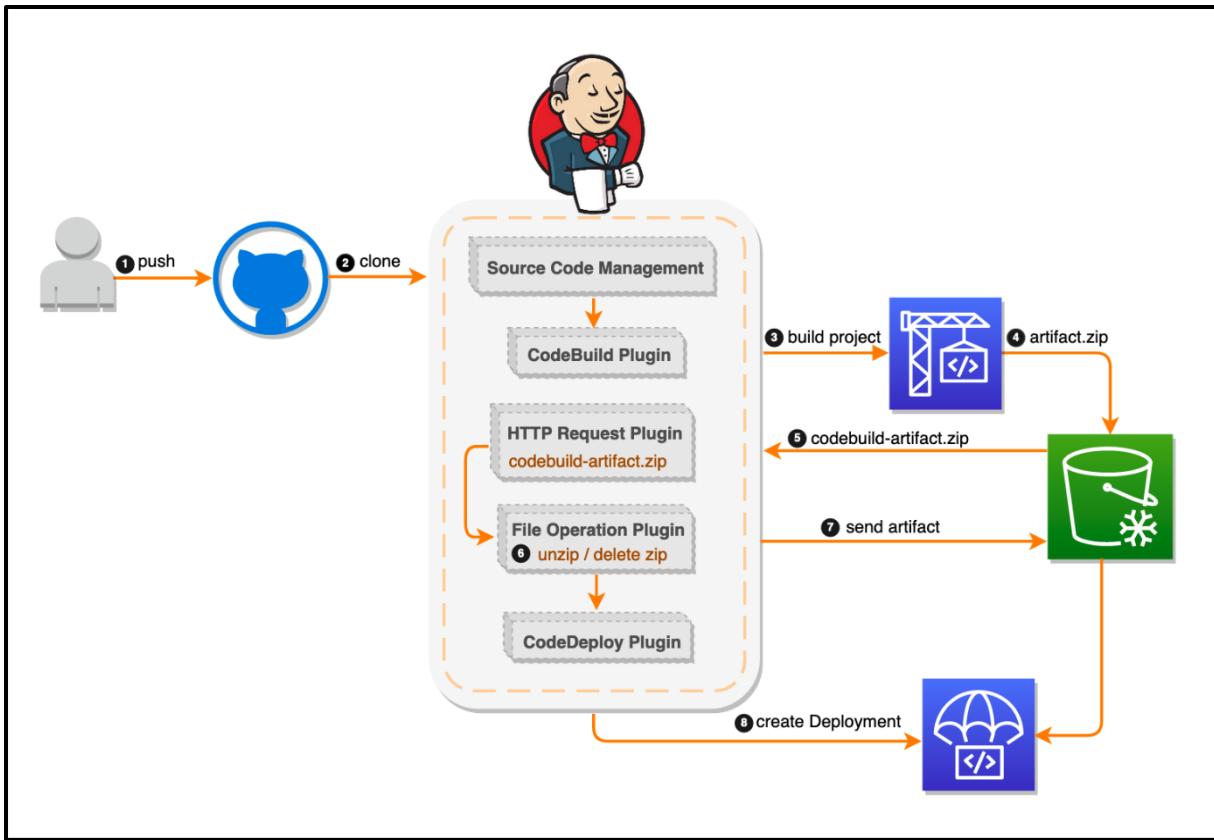


Figure 31: Architecture Jenkins

5.1.3 Réalisation

5.1.3.1 Préparation de l'environnement

La première étape de la conteneurisation consiste à concevoir les machines VM-TEST à l'aide de Docker Machine. Ainsi l'environnement de travail crée contient tous les outils nécessaires pour le bon fonctionnement de Docker. Il existe trois façons différentes d'installer la plateforme de conteneurs Docker :

- Installation manuelle via le package DEB.
- Installation à partir du dépôt Docker.
- Installation à partir du Repository Ubuntu.

Nous devons s'assurer avant tout des mises à jour de l'index des paquets de notre système d'exploitation. Une fois docker est installé, le démon Docker démarre automatiquement.

Afin de faciliter le fonctionnement des conteneurs, il faut obligatoirement avoir recours à une solution d'orchestration. De ce fait, nous avons choisi Docker-swarm comme un moteur d'exécution et qui permet à Docker d'offrir un "mode swarm". Ce mode donne la possibilité de créer des clusters de machines exécutant des conteneurs Docker, qui fonctionnent ensemble

comme une seule machine. Nous pouvons ainsi exécuter des commandes Docker sur les machines composant le cluster, comme s'il s'agissait d'une seule et même machine. Le cluster est contrôlé depuis une machine maîtresse, appelée Node Manager et il garantit la disponibilité et les hautes performances de l'application en la répartissant sur le nombre d'hôtes Docker à l'intérieur d'un cluster.

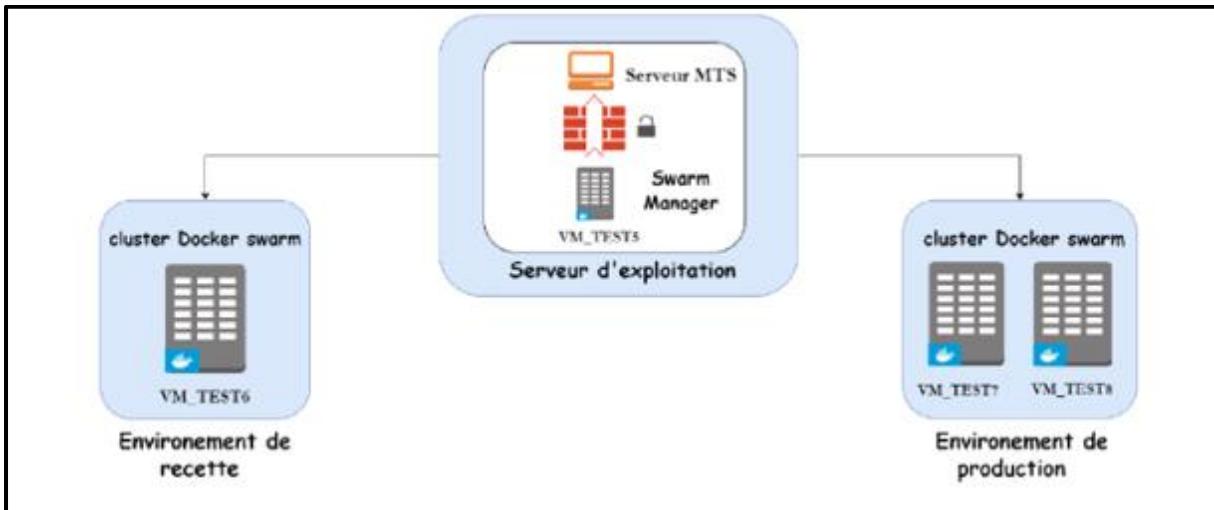
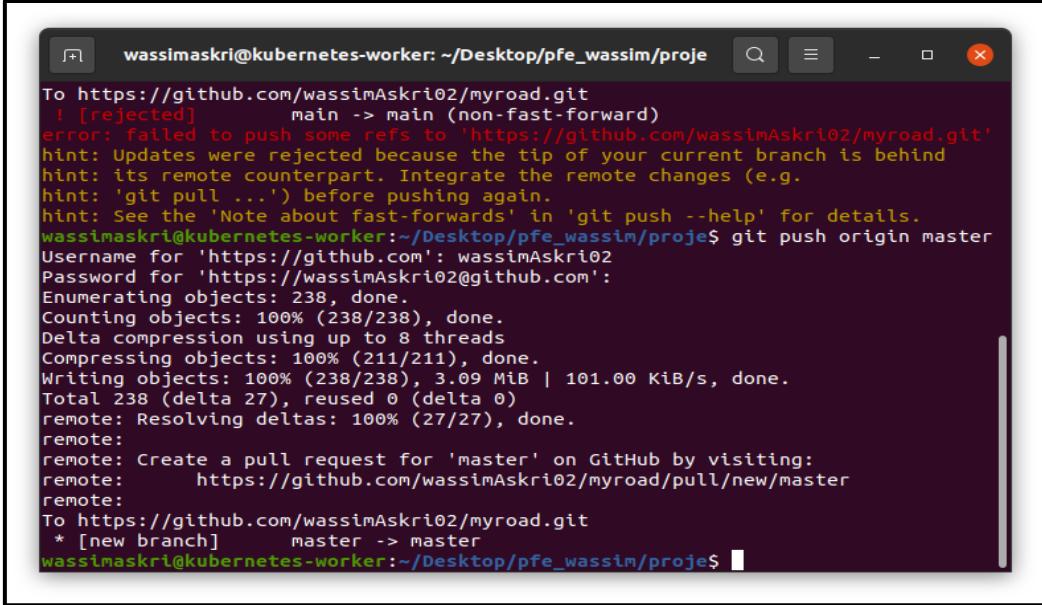


Figure 32: Notion de clustering

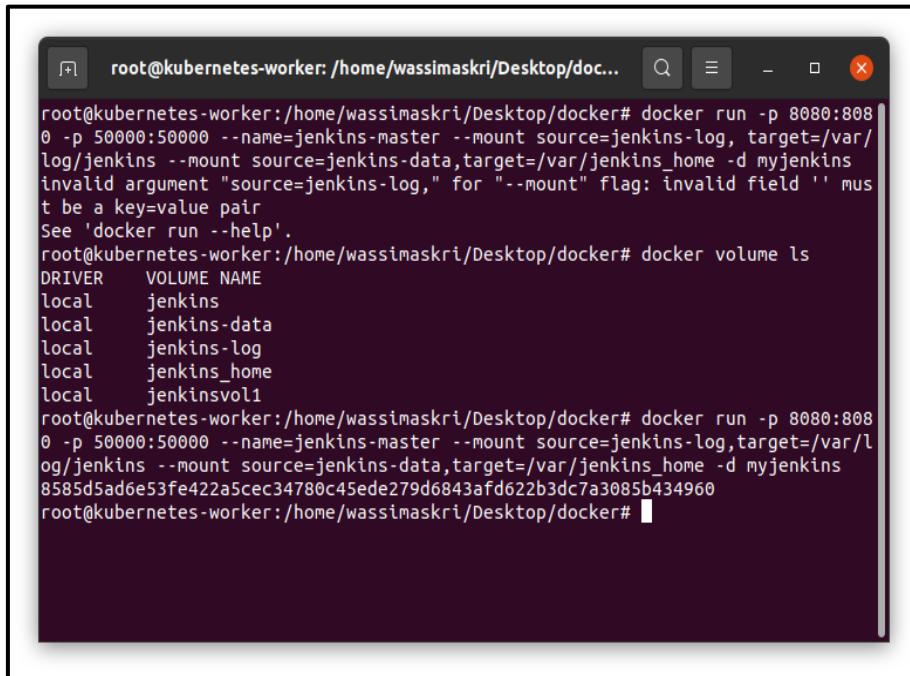
5.1.3.2 Capture d'écrans de la réalisation

Nous illustrons la réalisation de partie CI/CD à partir des captures d'écran qui décrivent les étapes à suivre pour créer un environnement automatisé avec la technologie Jenkins.



```
wassimaskri@kubernetes-worker: ~/Desktop/pfe_wassim/proje
To https://github.com/wassimAskri02/myroad.git
! [rejected]      main -> main (non-fast-forward)
error: failed to push some refs to 'https://github.com/wassimAskri02/myroad.git'
hint: Updates were rejected because the tip of your current branch is behind
hint: its remote counterpart. Integrate the remote changes (e.g.
hint: 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
wassimaskri@kubernetes-worker:~/Desktop/pfe_wassim/proje$ git push origin master
Username for 'https://github.com': wassimAskri02
Password for 'https://wassimAskri02@github.com':
Enumerating objects: 238, done.
Counting objects: 100% (238/238), done.
Delta compression using up to 8 threads
Compressing objects: 100% (211/211), done.
Writing objects: 100% (238/238), 3.09 MiB | 101.00 KiB/s, done.
Total 238 (delta 27), reused 0 (delta 0)
remote: Resolving deltas: 100% (27/27), done.
remote:
remote: Create a pull request for 'master' on GitHub by visiting:
remote:     https://github.com/wassimAskri02/myroad/pull/new/master
remote:
To https://github.com/wassimAskri02/myroad.git
 * [new branch]    master -> master
wassimaskri@kubernetes-worker:~/Desktop/pfe_wassim/proje$
```

Figure 33: git push code master



```
root@kubernetes-worker: /home/wassimaskri/Desktop/doc...
root@kubernetes-worker:/home/wassimaskri/Desktop/docker# docker run -p 8080:8080
0 -p 50000:50000 --name=jenkins-master --mount source=jenkins-log, target=/var/
log/jenkins --mount source=jenkins-data,target=/var/jenkins_home -d myjenkins
invalid argument "source=jenkins-log," for "--mount" flag: invalid field '' mus
t be a key=value pair
See 'docker run --help'.
root@kubernetes-worker:/home/wassimaskri/Desktop/docker# docker volume ls
DRIVER VOLUME NAME
local jenkins
local jenkins-data
local jenkins-log
local jenkins_home
local jenkinsvol1
root@kubernetes-worker:/home/wassimaskri/Desktop/docker# docker run -p 8080:808
0 -p 50000:50000 --name=jenkins-master --mount source=jenkins-log,target=/var/l
og/jenkins --mount source=jenkins-data,target=/var/jenkins_home -d myjenkins
8585d5ad6e53fe422a5cec34780c45ede279d6843af622b3dc7a3085b434960
root@kubernetes-worker:/home/wassimaskri/Desktop/docker#
```

Figure 34: Install docker-Jenkins container with data log volume

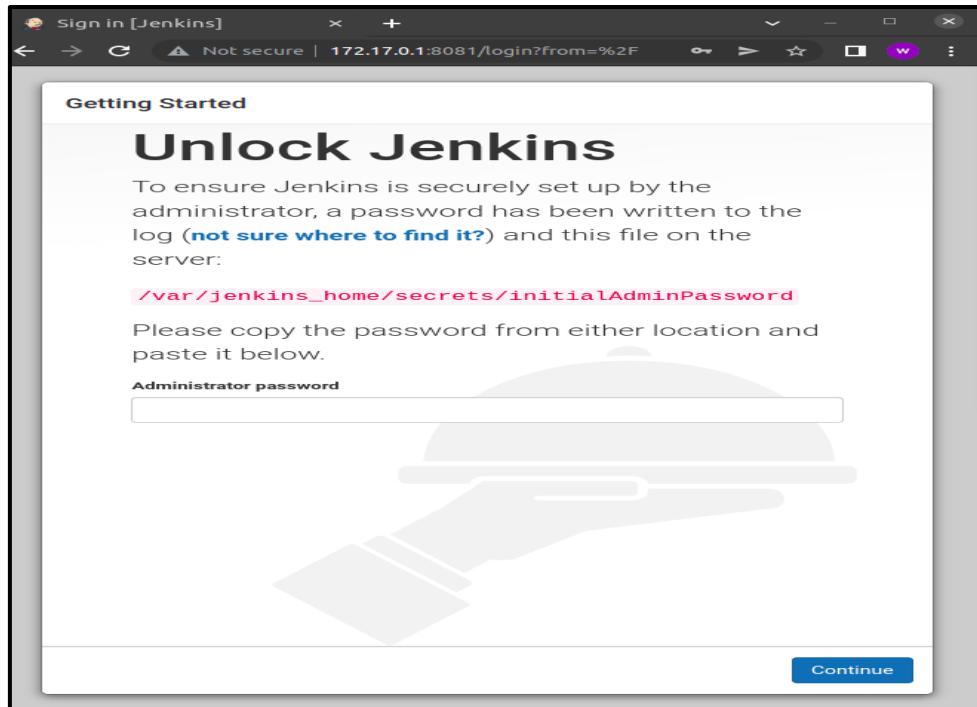


Figure 35: unlock Jenkins cat

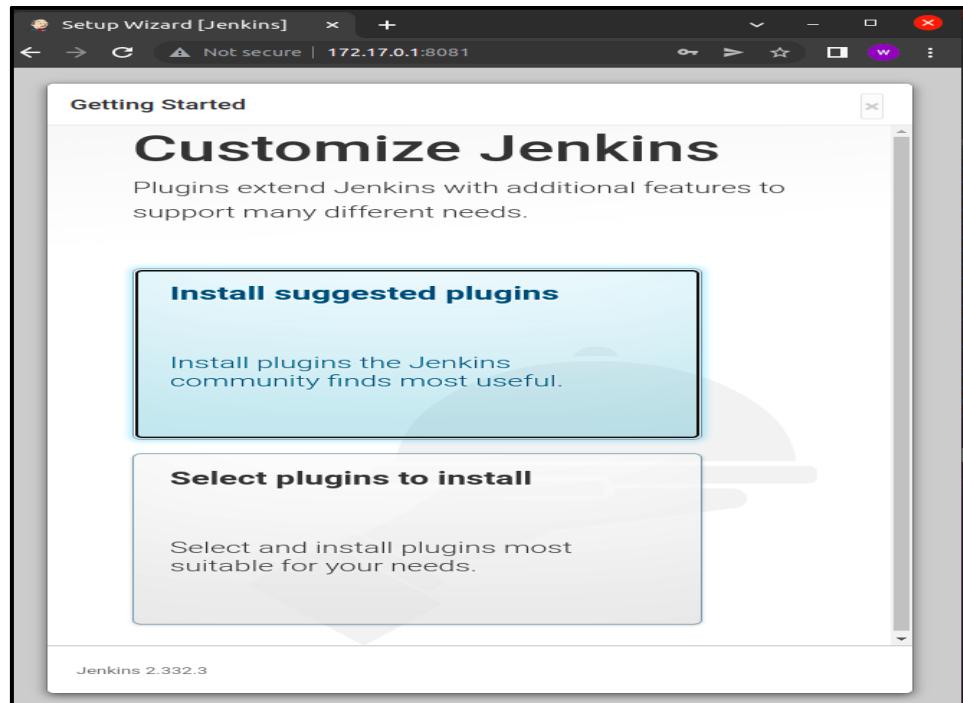


Figure 36: setup Jenkins server

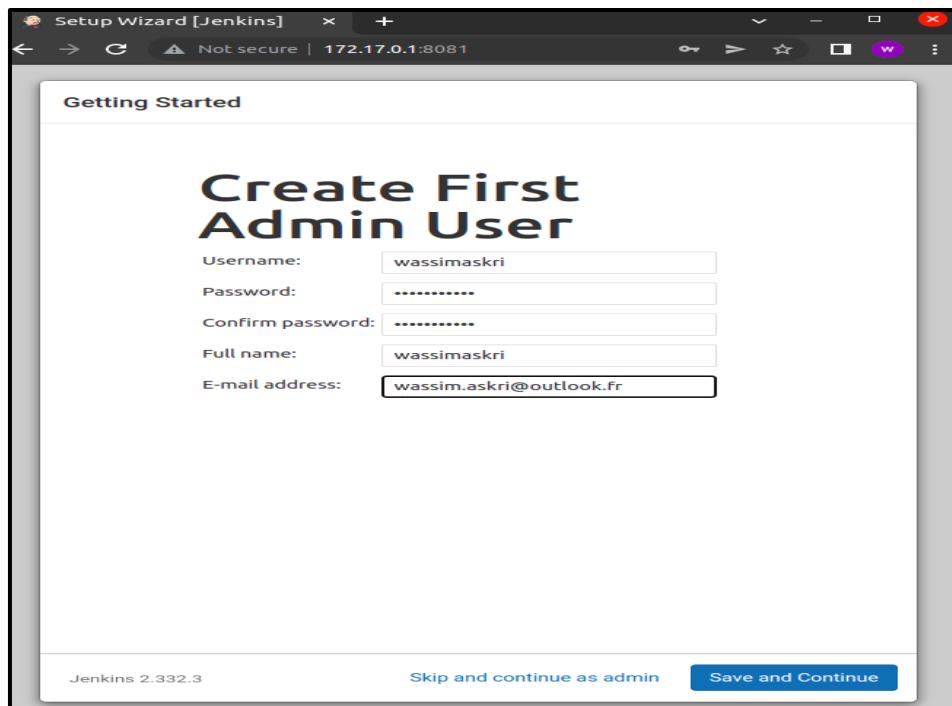


Figure 37: create admin user

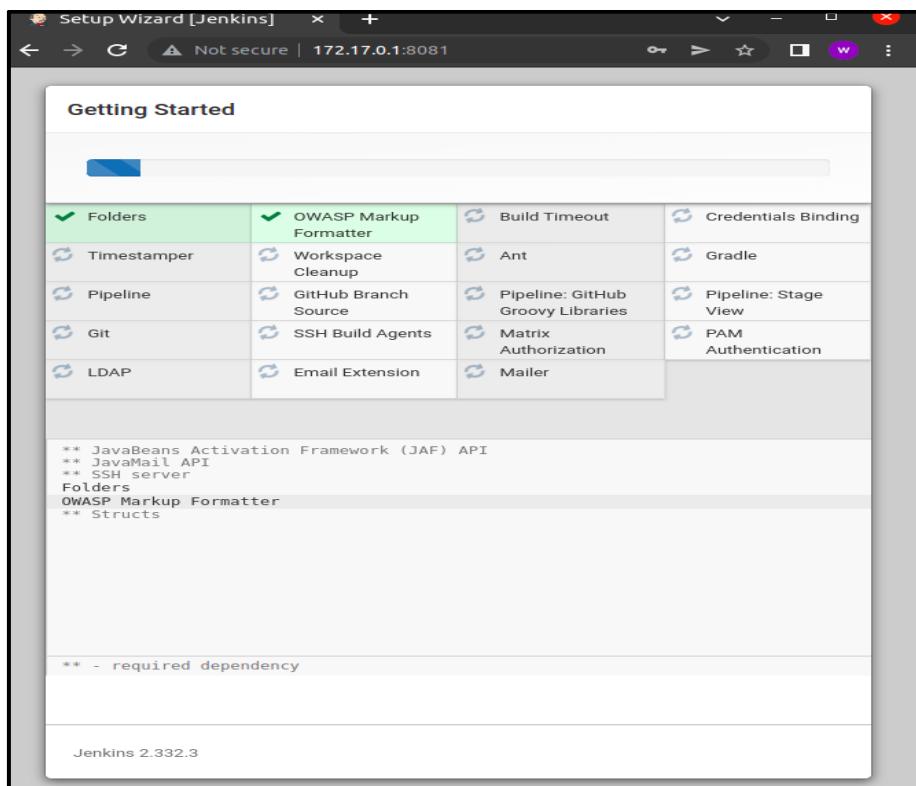


Figure 38: getting start Jenkins

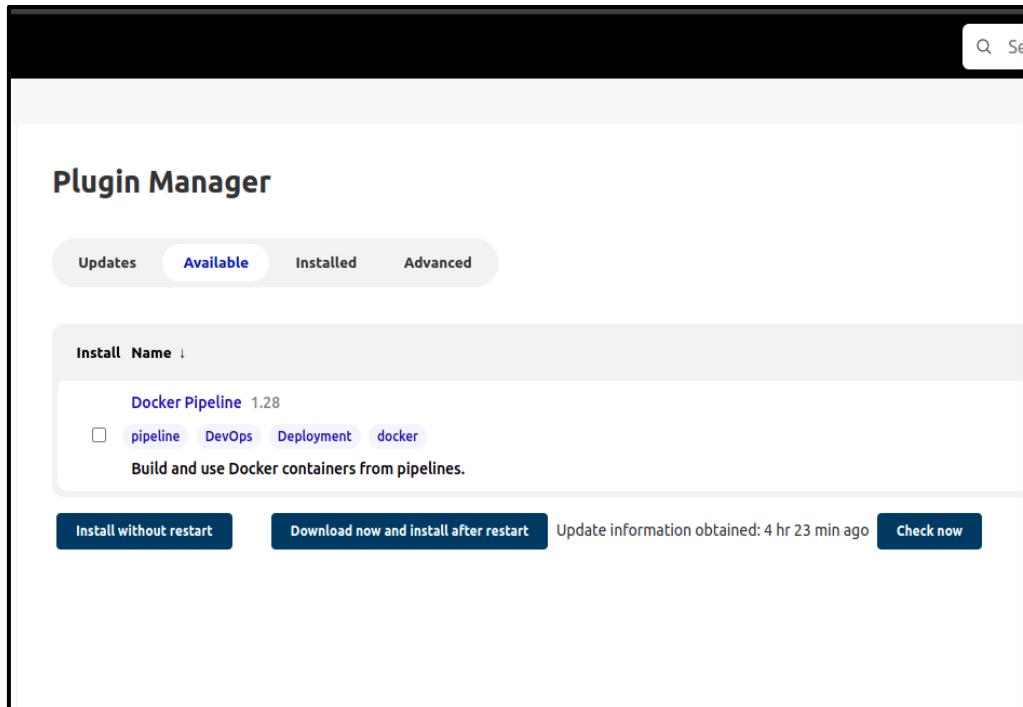


Figure 39: install plugin manager docker pipeline

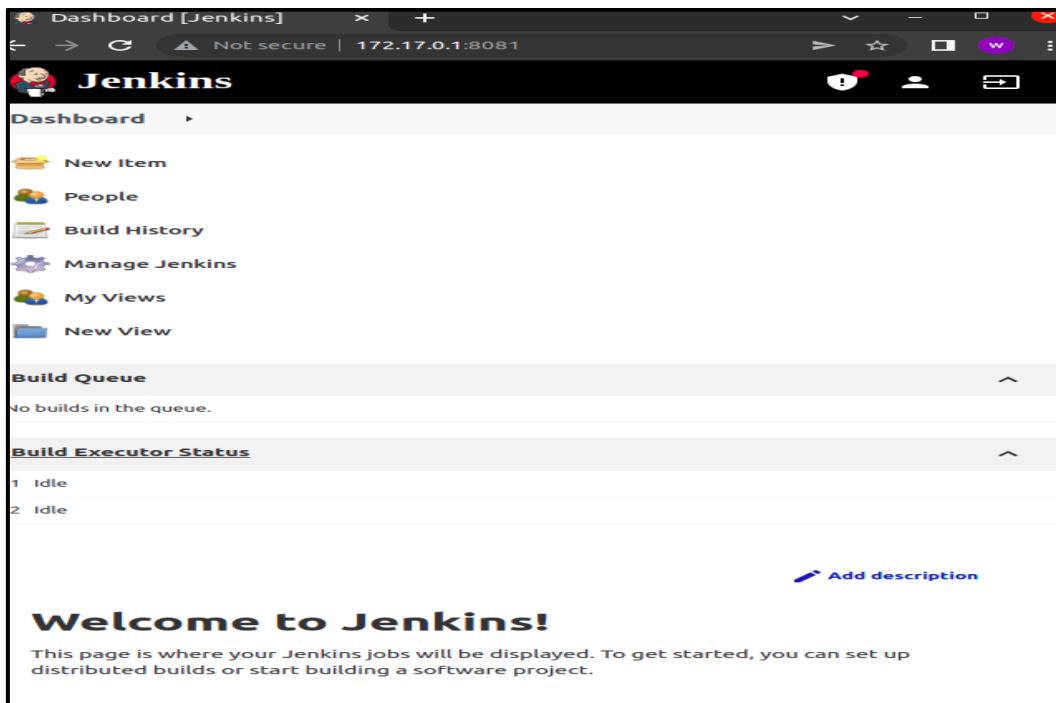


Figure 40: Dashboard Jenkins

Chapitre 5 : Mise en œuvre de release 2

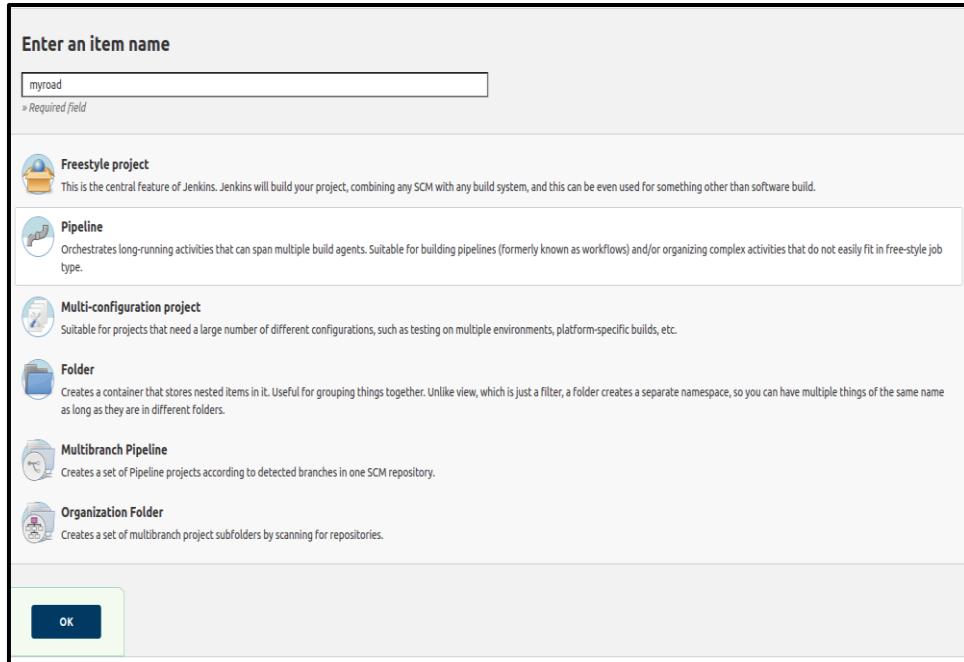


Figure 41: select new pipeline to application « myroad »

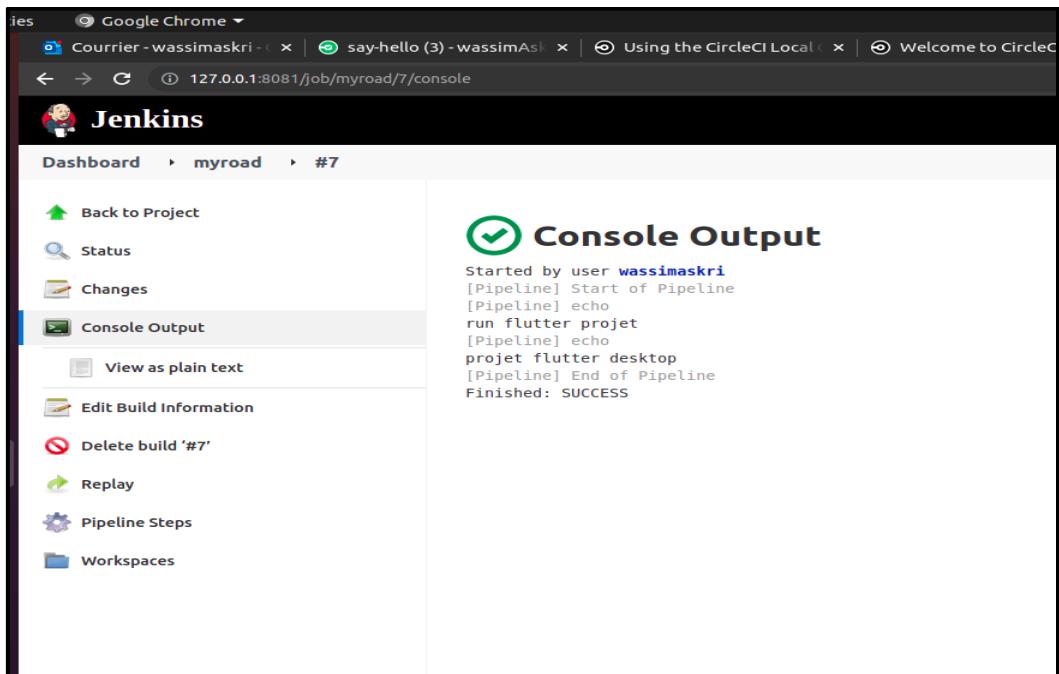


Figure 42: build console run output CI

5.1.3 Déroulement de sprint

5.1.3.1 Sprint Review

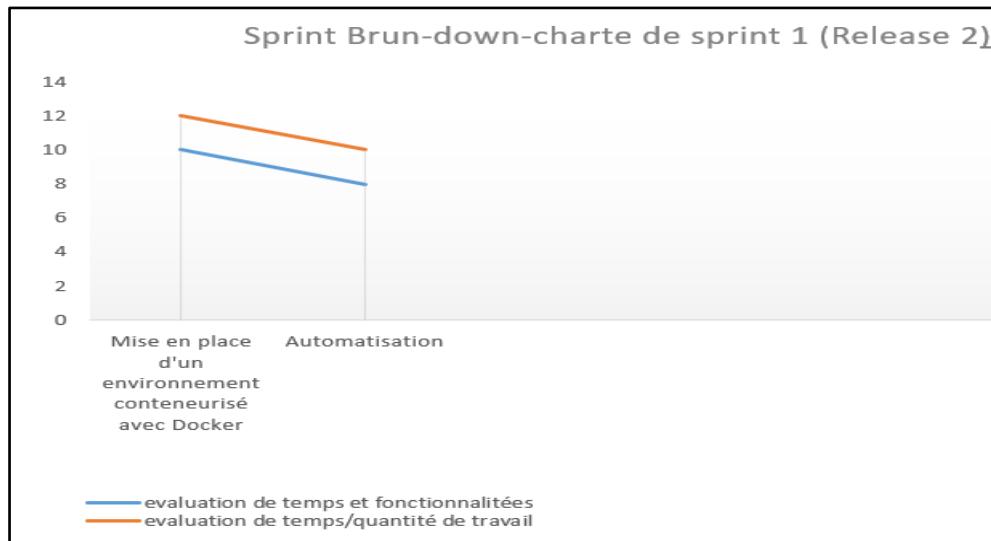


Figure 43: Sprint Burn-down chart de sprint 1 (Release 2)

5.1.3.2 Sprint rétrospective

Tableau 16: Tableau de validation de sprint 1 (Release 2)

Cas de test	Comportiment attendu	Résultat
Mise en place d'un environnement conteneurisé avec Docker	Préparation d'environnement avec succès. Construction d'image docker réalisé.	Conforme
Automatiser	Mise en place d'un pipeline CI/CD (Avec Jenkins) réalisé.	Conforme

5.2 Sprint 2

5.2.1 Backlog du sprint

ID	Cas	USER STORY	Priorité	STORY POINT (j)
12	Optimiser de la taille de l'image	En tant qu'Admin je veux réduire la taille de l'image	H	6
13	Mise en place d'une solution de pré-configuration des serveurs	En tant qu'Admin je veux avoir un environnement préconfiguré.	H	10

5.2.2 Architecture de l'environnement

5.2.2.1 Architecture optimisée

Étant donné que l'optimisation de la taille d'une image Docker prend du sens dans le cadre de l'intégration continue, il faut cependant aussi être attentif à l'impact sur le pipeline. L'idéal étant de trouver une solution qui soit un compromis entre les avantages apportés par une taille optimisée d'image Docker et les modifications que cela imposerait à la définition du pipeline.

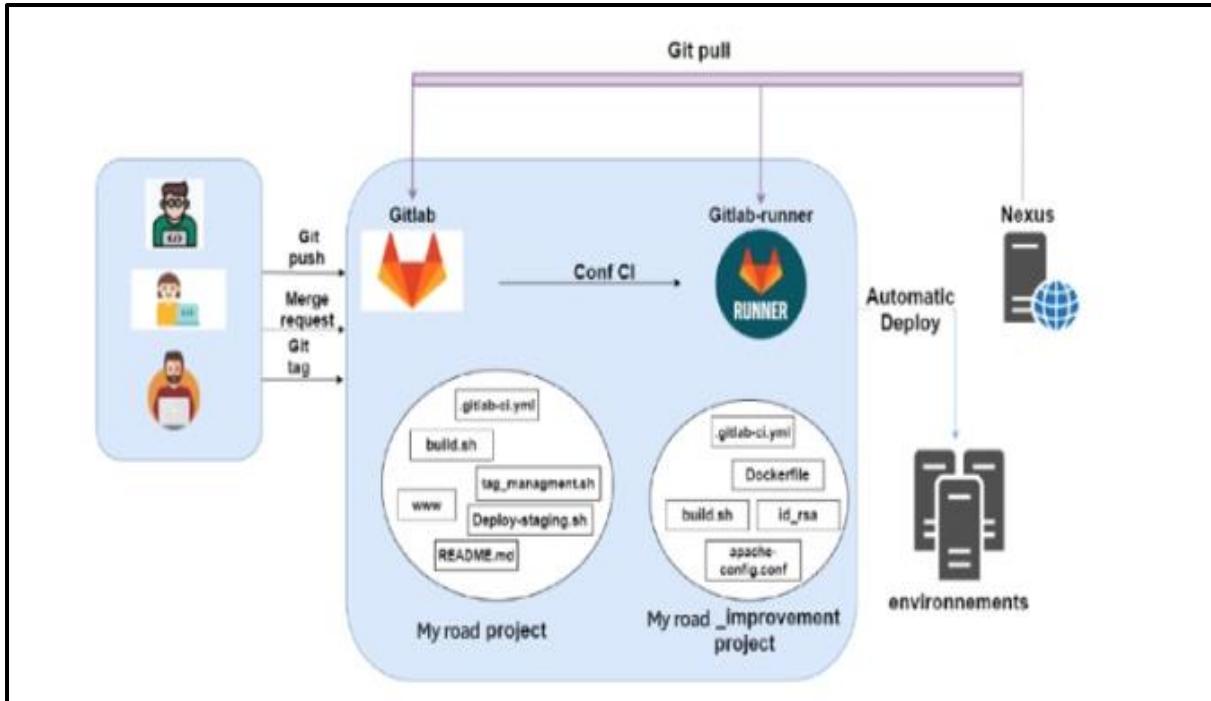


Figure 44: Architecture optimisée

La figure 44 présente l'architecture de l'environnement qui résume l'étude évoquée précédemment. Cet environnement permet de réduire la taille de l'image afin de minimiser le stockage et de restreindre le temps de réalisation.

De ce fait l'équipe des développeurs n'aura pas besoin de refaire le build de la création d'une nouvelle image à la suite de chaque modification. Ceci lui permet d'optimiser la taille et le nombre des images déployées. [35]

5.2.2.2 Architecture d'Ansible

Ansible repose sur la connexion d'un nœud maître à un ensemble des machines hôtes ou esclaves. Cette connexion est effectuée via SSH et en y poussant de petits programmes, appelés modules. Ces modules sont définis dans un fichier nommé le « Playbook.yml» ce fichier permet à un utilisateur d'Ansible d'élaborer des instructions sous forme de commandes ou les regrouper dans des scénarios réutilisables.

En contrepartie, le nœud manager se base sur un fichier inventaire » qui génère une liste de périphériques à cibler et sur lesquels les modules Ansible doivent être exécutés.

Afin d'améliorer la structure du Playbook, Ansible met en disposition un système de rôle.

Ces rôles sont une caractéristique robuste d'Ansible qui facilitent la réutilisation, favorisent d'avantage la modularisation de la configuration et simplifient l'écriture des Playbook » complexes en le divisant logiquement en composants réutilisables afin de profiter par la même occasion de les rendre plus facile à lire.

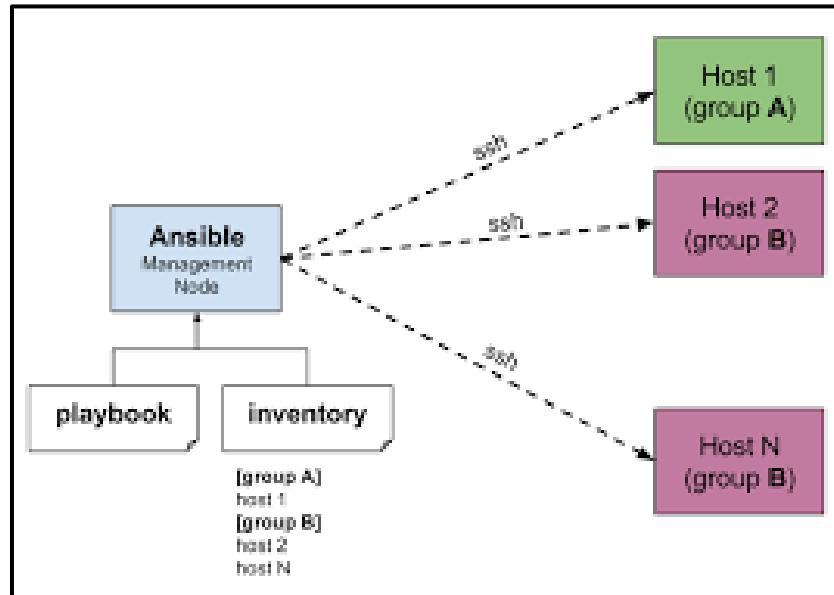


Figure 45: Architecture Ansible

5.2.2.3 Architecture générale

Avant de mettre en œuvre les tâches de ce sprint, nous devons concevoir l'architecture cible de notre environnement afin de clarifier le travail à faire.

L'un des objectifs de ce projet est d'avoir un ou plusieurs centres de commande à partir desquels nous pouvons envoyer des commandes sur des machines distantes afin d'exécuter des instructions séquencées via des playbooks. La figure 46 Présente l'architecture globale de notre solution adoptée.

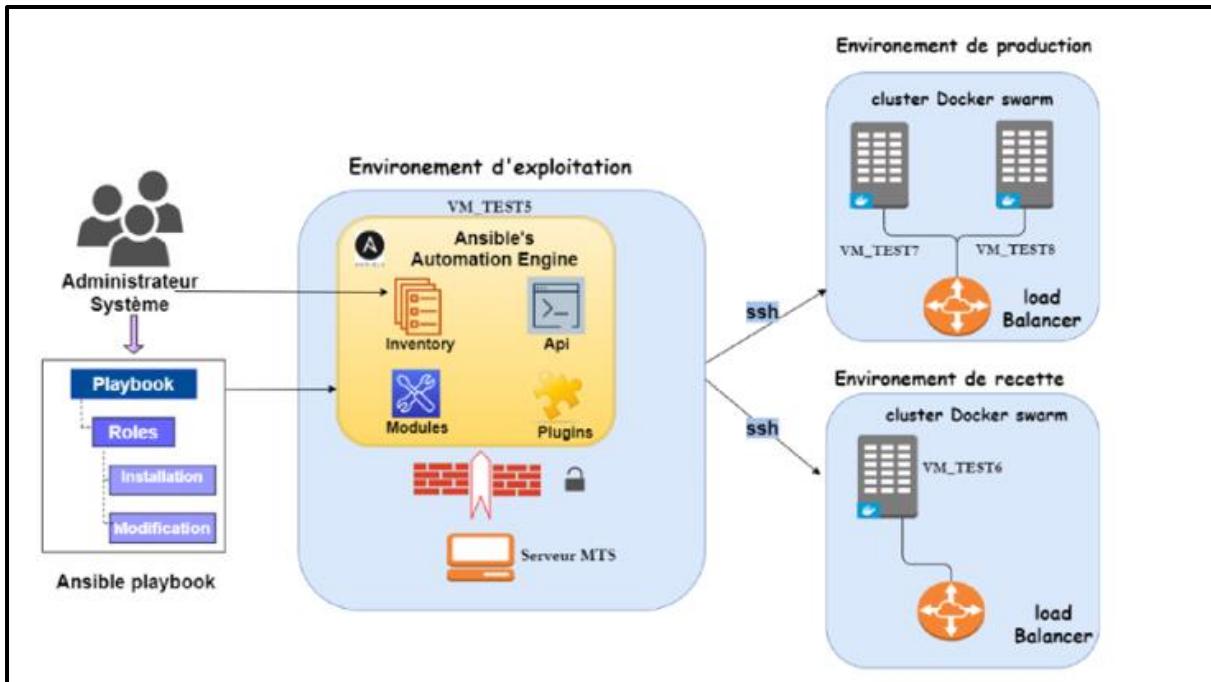


Figure 46: Architecture générale

Cette architecture repose sur deux éléments de base un playbook et un moteur d'automatisation Ansible.

Les playbooks peuvent non seulement déclarer des configurations, mais ils peuvent également orchestrer les étapes de toute tâche commandée manuellement dans le but de les exécuter d'une manière synchrone et asynchrone depuis la machine de contrôle VM-TEST5 vers les VM-TEST6, 7 et 8.

Afin d'éviter d'écrire encore et encore les mêmes playbooks, on peut utiliser les rôles Ansible qui seront ensuite intégrés aux playbooks.

Dans notre cas, nous avons regroupé l'ensemble des tâches en deux rôles, d'installation et de modification afin de faciliter la réutilisation et le partage avec les autres utilisateurs.

Ansible dispose aussi d'un moteur d'automatisation qui permet aux développeurs d'exécuter directement un playbook déployé sur les hôtes. Il existe plusieurs composants dans le moteur d'automatisation Ansible :

- ❖ **L'inventaire** : Il fait partie du moteur d'automatisation et représente l'ensemble des machines hôtes dans un seul fichier simple, avec leurs adresses IP, bases de données et serveurs

- ❖ **L'API Ansible** : aide à créer l'API nécessaire pour l'interaction des modules de bout en bout.
- ❖ **Les modules** : Ils sont directement exécutés à l'aide de playbooks.
- ❖ **Les plugins** : représentent un morceau de code qui étend les fonctionnalités de base d'Ansible. Il existe de nombreux plugins utiles, et nous pouvons également écrire nos propres plugins.

→ Pour cela Ansible nous aidera à créer un environnement plus polyvalent et flexible en incorporant les versions et les outils mis à niveau selon les exigences du système.

5.2.3 Réalisation

5.2.3.1 Configuration des machines

La première étape de la configuration consiste à créer un nouveau dossier nommé « www » dans chacune des machines sa fin de déplacer manuellement le code source pour qu'il soit déployé directement dans les environnements de recette et de production en adoptant la méthode de <<rsync >>. Cette méthode utilise un ensemble de paramètres permettant de faire des filtres et des tries afin de limiter la copie des modifications effectuées sur le code source.

5.2.3.2 Découpage de projet myroad

Une fois la configuration est faite avec succès, nous mettons l'accent sur l'optimisation du processus de l'intégration continue.

Notre projet met en évidence l'ensemble des scripts suivants

- ❖ **.gitlab-ci.yml** : C'est le fichier qui permet le déclenchement du pipeline.
- ❖ **Dockerfile** : C'est le fichier qu'on a modifié dans l'étape précédente, sur lequel se base notre image Docker.
- ❖ **Apache-config.conf** : C'est le fichier de configuration du serveur web apache. Ce fichier est nécessaire pour le Dockerfile.
- ❖ **Build.sh** : C'est un script qui permet la création de notre image vide.
- ❖ **Entrypoint.sh** : C'est un script qui inclut le chemin vers le service déployé dans le container.
- ❖ **Id-rsa** : C'est le fichier qui assure l'authentification par clé privée
- ❖ **Jenkins file** : est un fichier texte au format Groovy qui décrit un pipeline Jenkins.

5.2.3.3 Mise à jour de fichier. Gitlab-ci.yml

Dans la partie suivante, nous avons reconfiguré les deux fichiers. Gitlab-ci.yml afin de les adapter à la nouvelle modification de projet Myroad.

Le fichier gitlab-ci.yml qui contient les variables, les stages, le cache, des jobs :

* **Les variables** : Permet de déclarer des variables d'environnement.

* **Le cache** : Permet de spécifier une liste de fichiers et de répertoires à mettre en cache tout le long de notre pipeline. Une fois le pipeline est terminé le cache sera détruit. Plusieurs sous-directives sont possibles :

- **Paths:** C'est une option obligatoire, elle permet de spécifier la liste de fichiers et répertoires à mettre en cache.
- **Key:** Cette option est facultative, elle permet de définir une clé pour la liste de fichiers et de répertoires.

* **Les stages:** Cette directive permet de grouper des jobs en étapes. Par exemple dans notre cas on a trois étapes :

- La première étape permet d'importer toutes les modifications de code source afin de les déployer sur l'image stocké dans Nexus dans le but de les exécutées dans l'environnement de staging ou de recette.
- La seconde identifie un tag unique pour chaque version de l'image créée pour qu'elle soit référencée.
- La dernière permet de faire la copie des modifications sur l'environnement de production après la validation, dans l'environnement de test.

5.2.3.4 Préparation de l'environnement Ansible

Pour le déploiement d'Ansible, l'environnement matériel a son importance. D'après ce qui est évoqué auparavant, Ansible est un outil d'automatisation sans agent qui doit s'installer sur un nœud de contrôle afin de gérer à distance les différents hôtes. Alors, pour le mettre en place il nous faut deux types de nœuds :

- **Un nœud de contrôle Ansible** : C'est la machine que nous utiliserons pour se connecter aux hôtes Ansible et pour assurer le contrôle via SSH.
- **Un ou plusieurs hôtes Ansible** : C'est toute machine automatisée par le nœud de contrôle Ansible.

La première étape consiste à installer Ansible sur le nœud de contrôle. Il existe trois façons différentes d'installer l'outil d'automatisation Ansible :

- L'installation via les sources officielles (Archives ou Gat) maintenues par Red Hat
- L'installation via les packages logiciels sur un système Linux

5.2.3.5 Configuration du rôle de d'installation

Dans le but d'installer plusieurs services, on a choisi de créer un rôle d'installation qui en globe les différents services et packages à installer.

5.2.3.6 Configuration du rôle de modification

Nous passons à la configuration des installations à partir du rôle "modification" qui permet d'éditer les fichiers, créer des répertoires et modifier des paramètres spécifiques.

5.2.3.7 Configuration du fichier Hosts

La partie qui suit, met en évidence le développement du fichier hosts.

Une fois les hôtes configurés, on peut tester la connexion grâce au module Ping afin de s'assurer que la configuration fonctionne correctement.

5.2.3.5 Configuration du fichier Playbook.yml

La dernière partie de cette configuration consiste à déclarer les rôles déjà configurés dans un playbook principale pour les exécuter dans un ordre bien définie et de faire le parcours des rôles d'installation et de modification sur les hôtes qui sont présentés dans le fichier hosts.

Finalement, l'exécution de notre fichier Playbook.yml par la commande « ansible-playbook -i hosts Playbook.yml >>.

5.2.4 Déroulement de sprint

5.2.4.1 Sprint Review

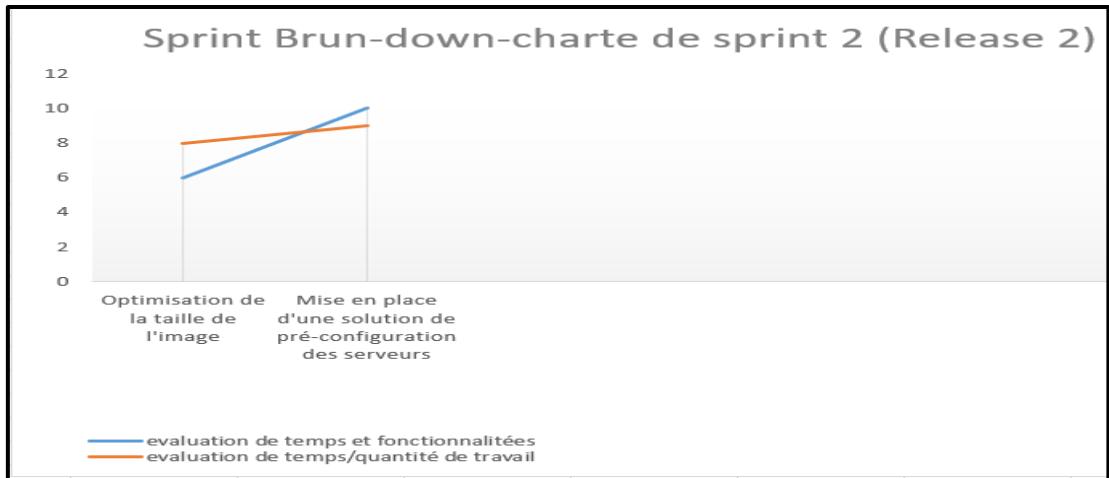


Figure 47: Sprint Burn-down-charte de sprint 2 (Release 2)

5.2.4.2 Sprint rétrospective

Tableau 17: Tableau de validation de sprint 2 (Release 2)

Cas de test	Compartiment attendu	Résultat
Optimisation de la taille d'image	Optimisation avec succès.	Conforme
Mise en place d'une solution de pré-configuration des serveurs.	Serveurs préconfigurés	Conforme

Conclusion

Dans ce chapitre nous avons terminé la dernière partie de notre projet, dans laquelle nous avons réalisé l'optimisation de la taille d'image et nous avons mise en place une solution préconfigurée des serveurs.

Conclusion générale et perspectives

Ce projet représente le fruit pour les années des études. Notre projet été proposé dans le cadre d’élaboration d’un stage au sein de société Accent située à Tunis.

Sur le plan professionnel, ce stage est une véritable occasion de mettre en pratique nos connaissances acquises et à raffiner nos compétences de conception et méthode de travail en milieu professionnel.

Notre projet consiste à développer une application desktop qui vise principalement à gérer des localisations et visualiser des trajets.

La réalisation de ce projet nécessite une la mise en œuvre de plusieurs technologies afin d’obtenir une application très performante et compacte.

Ce travail nous a fourni également un grand apport au niveau de plusieurs niveaux. Sur le plan technique, nous avons appris à maîtriser le fameux Framework « Flutter ».

Nous avons également eu l’opportunité de maîtriser le déploiement avec les technologies DevOps.

Le sujet compte également des perspectives importantes ainsi que des pistes de développement et de recherche qui pourraient être utilisées telles que :

L’amélioration de la sécurité de l’image déployée afin d’éviter les attaques et les vulnérabilités, maintenir la disponibilité de notre application web à un niveau optimal en utilisant l’autoscaling. L’hébergement vers le Cloud qui nous permettra d’éviter les tâches onéreuses de maintenance en interne.

Pour conclure, ce stage a été très enrichissant techniquement et professionnellement ce qui nous a permis de réaliser notre projet de fin d’études dans les meilleures conditions.

Webographie

Num	Lien	Consulté le :
[1]	https://www.weencar.tn	16 février 2022
[2]	https://www.mycartracks.com	16 février 2022
[3]	https://www.techno-science.net/definition/11786.html	14 mars 2022
[4]	https://anydesk.com/fr	14 mars 2022
[5]	https://air.imag.fr/index.php/StarUML	14 mars 2022
[6]	https://framalibre.org/content/visual-studio-code	23 mars 2021
[7]	https://www.frandroid.com/android/535194_quest-ce-que-flutter-loutil-permettant-de-creer-des-applications-android-et-ios	24 mars 2022
[8]	https://soat.developpez.com/tutoriels/dart/dart-manuel-reference/	24 mars 2022
[9]	https://aqueduct.io	25 mars 2022
[10]	https://supabase.com	25 mars 2022
[11]	https://www.lemagit.fr/definition/PostgreSQL	25 mars 2022
[12]	https://www.oreilly.com/library/view/devopsanddns/9781492049241/ch01	02 avril 2022
[13]	https://www.guru99.com/jenkin-continuous-integration.html	02 avril 2022
[14]	https://www.journaldunet.fr/web-tech/guide-de-l-entreprise-digitale/1443814-gitlab-definition-api-pricing-ci-cd	02 avril 2022
[15]	https://devopssec.fr/article/avantages-fonctionnement-ansible	02 avril 2022
[16]	https://www.lemagit.fr/definition/Ansible	05 avril 2022
[17]	https://aws.amazon.com/fr/devops/what-is-devops/	05 avril 2022
[18]	https://www.1min30.com/gestion-de-projet/les-avantages-devops-1287497989	06 avril 2022
[19]	https://www.redhat.com/fr/topics/automation/what-is-deployment-automation	06 avril 2022
[20]	https://www.techno-science.net/definition/731.html	06 avril 2022
[21]	https://docs.microsoft.com/en-us/devops/develop/git/what-is-version-control	07 avril 2022
[22]	https://docs.microsoft.com/fr-fr/azure/devops/repos/git/branch-policies-overview?view=azure-devops	07 avril 2022

[23]	https://www.lemagit.fr/definition/Orchestration-du-cloud	07 avril 2022
[24]	https://mobiskill.fr/blog/conseils-emploi-tech/automatisation-du-devops-quels-sont-les-avantages/	08 avril 2022
[25]	https://www.axido.fr/la-virtualisation-comment-ca-fonctionne/	08 avril 2022
[26]	https://codalis.ch/conteneurisation-docker/	09 avril 2022
[27]	https://geekflare.com/container-orchestration-software/	10 avril 2022
[28]	https://web.maths.unsw.edu.au/~lafaye/CCM/uml/uml-use-cases.htm#:~:text=Un%20acteur%2C%20au%20sens%20UML,à%20droite%20les%20acteurs%20secondaires	17 février 2022
[29]	https://agiliste.fr/introduction-methodes-agiles	17 février 2022
[30]	https://www.journaldunet.fr/web-tech/guide-de-l-entreprise-digitale/1443838-methode-agile-definition-comparatif-et-avantages/	17 février 2022
[31]	https://agiliste.fr/guide-de-demarrage-scrum/	17 février 2022
[32]	https://www.aquasec.com/cloud-native-academy/docker-container/docker-architecture/	23 avril 2022
[33]	https://www.nextinpath.com/article/48913/docker-et-conteneurisation-par-exemple	23 avril 2022
[34]	https://www.tutorialandexample.com/jenkins-architecture	23 avril 2022
[35]	https://www.softfluent.fr/blog/devops-docker-net-core-optimiser-taille-image-docker/	23 avril 2022

RESUME

Ce travail s'inscrit dans le cadre de projet de fin d'études en Licence en technologie de l'informatique à l'institut supérieur des études technologique de Jendouba (ISETJ). On a effectué un stage au sein de l'entreprise industrielle (ACCENT) qui a pour objectif de mettre en place une application desktop qui permet la géo-localisation des personnes et des véhicules et de réaliser un système de déploiement automatique de notre solution basé sur la méthode DevOps. Un ensemble de technologies, des langages de développement desktop, ainsi que des outils informatiques ont été utilisés pour aboutir au déploiement que nous avons proposé.

Mots clés: Flutter, Dart, Aqueduct, Docker, Jenkins, GitLab, Ansible...

ABSTRACT

This work is part of the end-of-study project in Computer Technology at the Higher Institute of Technological Studies of Jendouba (ISETJ). We carried out an internship within the industrial company (ACCENT) which aims to set up a desktop application that allows the geo-location of people and vehicles and to create an automatic deployment system for our solution based on the DevOps method. A set of technologies, desktop development languages, as well as IT tools were used to achieve the deployment we proposed.

Keywords: Flutter, Dart, Aqueduct, Docker, Jenkins, GitLab, Ansible...