



Université Sultan Moulay Slimane
Ecole Nationale des Sciences Appliquées
-Khouribga-



Projet de fin d'étude
En vue de l'obtention du diplôme

INGENIEUR D'ETAT

Filière : Génie Informatique

Présenté par
Bahaa Eddine ELBAGHAZAOUI

Contribution à la réalisation de la plateform Prospektor

Soutenu le (date de soutenance), devant le jury :

NOM 1	Établissement	Président
NOM 2	Établissement	Examinateur
NOM 3	Établissement	Examinateur
Abdelhaq ELABI	Groupe OCP	Encadrant Externe
Mohamed AMNAI	ENSA-K	Encadrant Interne

Résumé

Dans le cadre de ce Projet de Fin d'études, nous étions responsables de la conception des différentes fonctionnalités d'une solution moderne Prospektor. Cette solution vient dans un contexte de digitalisation du processus de gestion des anomalies dans les mines du groupe OCP lors de l'extraction de phosphates en se basant sur une architecture orientée API. Notre contribution dans ce projet a commencé par une analyse de l'existant pour préciser le point de départ, ensuite nous avons commencé à implémenter les fonctionnalités par Epics dans une approche Agile.

Ce stage de fin d'étude nous a été très bénéfique, il nous a aidé à projeter l'ensemble des connaissances acquises durant notre formation d'ingénieur à l'ENSA de Khouribga, ainsi que l'implication dans les différentes phases liées à un projet de développement logiciel. Une forte valeur ajoutée réside dans l'apprentissage des nouveaux concepts qui sont actuellement la tendance du domaine du génie logiciel à savoir l'utilisation d'une architecture orientée APIs basé sur les web service qui tient en compte le contexte actuel de la technologie ainsi que l'adoption d'une méthodologie agile qui nous a permis une marge de flexibilité pour innover et livrer plus de valeur. Encore plus, cette expérience de stage nous a aidé à intégrer le monde de l'entreprise, l'interaction et le travail au sein d'une équipe qui ont rendu ce projet une expérience professionnelle très réussie.

La solution Prospektor n'est pas encore terminée en terme de développement vu la voluminosité de l'ensemble des fonctionnalités. Il reste encore à développer des fonctionnalités en relation avec le reste des Epics de la RoadMap ainsi que les tâches liées aux analytiques.

Mots clés : Prospektor, Orienté API, Aproche Agile.

Abstract

As part of the graduation project, we were responsible for designing the various functionalities of a modern prospektor solution. This solution is part of the digitalization of the OCP's anomaly management process, it's based on an API oriented architecture. Our contribution in this project started with an analysis of the existing tools to specify the departure point, then we started to implement the functionalities by Epics in an Agile approach.

This graduation internship was very beneficial, it helped us to project all the knowledge acquired during our engineering studies at ENSA Khouribga, as well as the involvement in the different steps related to a software development project. A strong added value lies in learning new concepts that are currently the trend of the software engineering field namely the use of an API oriented architecture based on web services that considers the current context of technology, as well as the adoption of an agile methodology that has given us a margin of flexibility to innovate and deliver more value. Even more, this internship experience helped us to integrate the business world, interaction and working within a team that made this project a very successful work experience.

The Prospektor solution is not yet finished in its development due to the volume of all the functionalities. It still remains to develop features related to the rest of the RoadMap Epics as well as the tasks related to analytics.

Keywords : Prospektor, API oriented architecture, Agile methodology.

Dédicaces

C'est avec plaisir que je dédie ce modeste travail

À mon père, décédé trop tôt, qui m'a toujours poussé et motivé dans mes études.
Puisse Dieu, le tout puissant, l'avoir en sa sainte miséricorde.

À ma mère .Aucun hommage ne pourrait être à la hauteur de l'amour dont elle ne
cesse de me combler. Que dieu la procure bonne santé et longue vie.

À ma sœur Azhar et à toute ma famille pour leur amour, leur tendresse et leur
encouragement.

Au membres de digital factory, Bakr, Amine, Ayoub, Driss, Karima, Abdelhalim,
Meryem, Ghita, Imad et Zaynab pour leur aide et leur servabilité.

À tous ceux que j'aime.

Bref, à tous ceux qui ont contribué à la réalisation de ce travail. Que chacun y
trouve l'expression de ma profonde gratitude.

Remerciement

Je présente mes remerciements les plus sincères à Monsieur **Mohamed AMNAI** d'avoir accepté de m'encadrer, de m'avoir soutenu et assisté tout au long de mon stage et de m'avoir fait part de ses remarques pertinentes et de ses idées constructives. Je tiens à lui exprimer toute mon admiration et ma reconnaissance.

Je tiens à remercier également Monsieur **Abdelhaq EL AIBI** et tous les membres de la digital factory pour leur encadrement, leurs précieux conseils tout au long de mon stage, ainsi que pour les remarques éclairées qu'ils m'ont prodiguées tout au long de ce stage.

Par la même occasion, j'adresse mes remerciements à tous mes enseignants pour leurs efforts qui ont guidé mes pas et enrichi mes travaux tout au long de mes études universitaires, et à ceux qui ont perfectionné mes connaissances théoriques et pratiques durant la période de la formation.

Table des matières

1 Contexte général	12
1.1 Introduction	12
1.2 Présentation de l'organisme d'accueil	12
1.2.1 Office chérifien des phosphates	12
1.2.2 L'importance de la digitalisation	14
1.2.3 La Digital Factory de l'OCP	15
1.3 Présentation générale du projet	16
1.3.1 Présentation du projet de fin d'études	16
1.3.2 Exigences du projet	17
1.3.3 Objectifs du Projet	18
1.4 Conduite du projet	18
1.4.1 Organisation du projet	18
1.5 Méthodologie de travail	19
1.5.1 Méthodologie Scrum	19
1.5.2 Outils de suivi	21
1.6 Conclusion	23
2 Analyse et spécification des besoins	24
2.1 Introduction	24
2.2 Analyse des besoins	24
2.2.1 Identification des acteurs	24
2.2.2 Les besoins fonctionnels	24
2.2.3 Les besoins non fonctionnels	25
2.2.4 Diagramme des cas d'utilisation	26
2.2.5 Description des cas d'utilisation	28

2.3	Planning du projet	30
2.3.1	Minimum Viable Product	30
2.3.2	Post Minimum Viable Product	31
2.4	Conclusion	32
3	Conception générale du projet	33
3.1	Introduction	33
3.2	Conception générale	33
3.2.1	Architecture globale du système	33
3.2.2	Architecture applicative	34
3.3	Entity relationship diagram	39
3.4	Conclusion	40
4	Implémentation de la solution	41
4.1	Introduction	41
4.2	Environnement logiciel	41
4.3	Architecture technique du système	42
4.3.1	Couche BackEnd	42
4.3.2	Couche FrontEnd	46
4.3.3	Couche Base de données	48
4.3.4	Outils de DevOps	48
4.4	Implémentation & tests	50
4.4.1	Sprint modèle : Créer un chantier	51
4.4.2	Captures d'écran du résultat courant	53
4.4.3	Tests de validation	56
4.4.4	Déploiement de projet	59
4.5	Conclusion	60

Table des figures

1.1	Structure de la Digital Office de l'OCP	14
1.2	les buts majeurs de la Digital Factory de l'OCP	15
1.3	Les cinq étapes d'extraction du phosphate	16
1.4	Chaîne cinétique d'extraction de phosphate	17
1.5	Équipe de projet	19
1.6	Méthode SCRUM	20
1.7	Capture de l'outil JIRA	22
1.8	Capture de l'outil Slack	22
2.1	Diagramme de cas d'utilisation globale pour le prospecteur	27
2.2	Diagramme de cas d'utilisation globale pour l'exploitant	27
2.3	Diagramme de cas d'utilisation globale pour l'administrateur	28
2.4	RoadMap de la version MVP	31
2.5	RoadMap de la version Post MVP	31
3.1	Architecture du système	34
3.2	Architecture 3-tiers	35
3.3	Architecture MVC	36
3.4	Architecture applicative backend	36
3.5	Architecture Applicative backoffice	37
3.6	Architecture Applicative frontoffice	38
3.7	Diagramme de Crow's Foot	40
4.1	Architecture technique du système	42
4.2	Documentation de API Prospektor	44
4.3	Authentification avec JWT	45
4.4	Prospektor avec Postman	45

4.5	Prospektor avec MinIo	48
4.6	Containers Vs Machines virtuelles	49
4.7	Kubernetes	50
4.8	Ecran d'accueil	51
4.9	Écran de chantier	52
4.10	Outil Android Studio	52
4.11	Prospecteur WorkFlow	53
4.12	Exploitant WorkFlow	55
4.13	Test unitaires de la partie frontoffice	57
4.14	SonarQube	57
4.15	EFK Stack	59
4.16	GitLab	59
4.17	Processus de déploiement	60
4.18	Pipeline de Prospektor	60

Liste des tableaux

3.1	Cardinalité du Crow's Foot	39
4.1	Environnement logiciel	41
4.2	Bibliothèques & FrameWorks de test	56
4.3	Configuration SonarQube	58

Table d'abréviations

AD Active Directory. [33](#)

API Application programming interface. [1](#), [7](#), [38](#), [41](#), [43–45](#), [61](#)

DF Digital Factory. [39](#), [42](#), [57](#), [59](#)

ERD Entity relationship diagram. [39](#)

HTTP HyperText Transfer Protocol. [39](#), [42](#)

JWT JSON Web Token. [44](#)

LDAP Lightweight Directory Access Protocol. [34](#)

MVC Model View Controller. [35](#)

MVP Model View Presenter. [37](#)

MVP Minimum Viable Product. [7](#), [11](#), [31](#), [52](#), [56](#)

NAS Networked Attached Storage. [33](#)

NFS Network File System. [33](#), [34](#), [39](#)

OCP Office Chérifien des Phosphates. [1](#), [11](#), [16](#), [32](#), [39](#), [46](#), [61](#)

PFE Projet de Fin d'Étude. [11](#)

REST Representational state transfer. [38](#), [42](#)

SMS Short Message Service. [39](#), [61](#)

SQL Structured Query Language. [48](#)

Introduction générale

La gestion des activités métiers devient de plus en plus un défi majeur pour les sociétés, un défi qui est aujourd’hui un point déterminant en termes d’optimisation des processus métiers ainsi que l’amélioration de leur visibilité et de leur gestion. Les entreprises manufacturières changent de stratégie au fur et à mesure de l’évolution des marchés. Soumises à de fortes pressions concurrentielles au cours des dernières décennies, les industries se sont orientées vers la digitalisation.

Dans ce cadre-là s’inscrit le sujet de mon **PFE** au sein de l’**OCP**, dont le but est de concevoir et implémenter une solution informatique avec une architecture moderne pour digitaliser le processus métier. Dans notre situation est gestionné les anomalies dans les sites du groupe **OCP** lors de l’extraction de phosphates.

Le point de départ de notre projet est de faire une analyse profonde pour réaliser la première version **MVP** (Une réalisation qu’on peut la mettre en face des clients pour commencer à valider nos hypothèses), après on va faire des améliorations correspondants à nos besoins. L’équipe travaille avec une méthodologie Scrum selon les Epics tracés dans la RoadMap du projet.

Le présent rapport décrit l’ensemble du travail réalisé dans le cadre de ce projet, il contient quatre chapitres. Le premier chapitre contient une description du contexte général du projet notamment la présentation de la Digital Factory de l’**OCP** ainsi que la motivation et les objectifs du projet. Le deuxième chapitre présente une analyse de besoins fonctionnels et non fonctionnels. Par la suite le troisième chapitre mettra l’accent sur l’ensemble des éléments de l’étude conceptuelle. Enfin, le chapitre quatre présentera les résultats de l’implémentation.

Contexte général

1.1 Introduction

Ce chapitre abordera comme sujet la situation du contexte général du projet, sur un niveau organisationnel en présentant l'organisme d'accueil, et sur un niveau contextuel qui reflète la motivation et les objectifs du projet ainsi que la méthodologie de travail durant son déroulement.

1.2 Présentation de l'organisme d'accueil

1.2.1 Office chérifien des phosphates

L'Office Chérifien des Phosphates à sa création, le Groupe OCP, depuis 1975, a évolué sur le plan juridique, pour devenir en 2008 une société anonyme dénommée « OCP S.A ».

D'une activité d'extraction et de traitement de la roche à ses débuts, OCP s'est positionné au fil du temps sur tous les maillons de la chaîne de valeur, de la production d'engrais à celle d'acide phosphorique, en passant par les produits dérivés. L'OCP trouve, depuis sa création, les ressources de sa croissance continue et de son leadership dans sa stratégie industrielle. Celle-ci est rythmée par une montée en puissance régulière de l'outil de production, par une politique ambitieuse de partenariats durables et servie par une politique financière efficace.

Ces partenariats touchent aussi bien des accords de livraison à moyen et à long terme que la construction d'unités de production sous forme de jointventures, basées au Maroc et à l'étranger. Aujourd'hui, OCP compte douze filiales et joint-ventures ainsi que quatre bureaux de représentations dans le monde.

Depuis sa création, OCP est passé de quelques centaines de personnes à près de 23 000 collaborateurs et 46 milliards de DH de chiffre d'affaires en 2013.

Afin de mener à bien la transformation digitale du Groupe OCP, l'entité « Digital Office » s'organise autour des principes managériaux suivants :

- Flexibilité et agilité dans l'allocation des ressources humaines dans une logique d'efficacité et ce, à travers une structuration en pools permettant l'agilité dans le staffing, la réduction des niveaux hiérarchiques, ainsi que le renforcement de la collaboration et de l'autonomie.
- Ouverture forte vers les métiers, dans une logique de démarche centrées utilisateur, par la mise en place d'interfaces métiers pour capturer et challenger leurs besoins dans un objectif de création de valeur .
- Stimulation et accompagnement à l'innovation par le déploiement de capacités d'innovation technologique et d'acculturation digitale .
- Synergies entre les entités Digitales et celles du Système d'Information pour concilier les besoins d'agilité avec les enjeux de stabilité, de sécurisation et de pérennisation du socle SI, assurant par là-même le succès de la transformation.

En application de ces principes, l'entité « Digital Office » est structurée comme suit :

- Une « Digital Factory » avec des antennes sur les différents sites du Groupe, en charge de livrer les initiatives digitales de la feuille de route, dans des cycles d'itération courts, à travers des méthodes de fonctionnement agile autour d'équipes cross-fonctionnelles .
- Une entité « Systèmes d'Information », en charge d'assurer la réalisation des projets SI, le bon fonctionnement des infrastructures SI et télécoms du Groupe, ainsi que l'assistance et le support aux utilisateurs du Groupe, dans le respect des exigences de qualité et des meilleurs standards en la matière .
- Une entité « Data Management » en charge d'élaborer une stratégie et un modèle de gouvernance de la Data, de définir l'architecture Data sein du Groupe, et de mettre en œuvre les initiatives data y afférentes .
- Une entité « Data Planning & PMO », en charge de la construction et de la mise à jour de la roadmap digitale du Groupe et du suivi de son exécution .
- Une Entité « Business Architecture », en charge de conseiller et challenger les métiers sur les solutions digitales pertinentes pour adresser leurs besoins, les consolider et suivre leur mise en œuvre .
- Un pool « Digital Innovation & Change Officers », en charge d'identifier et

mener des projets d'innovation en matière de Digital (veille technologique, prototypage, Open Innovation ...), ainsi que de la promotion d'une culture digitale et de nouveaux modes de travail au sein du Groupe .

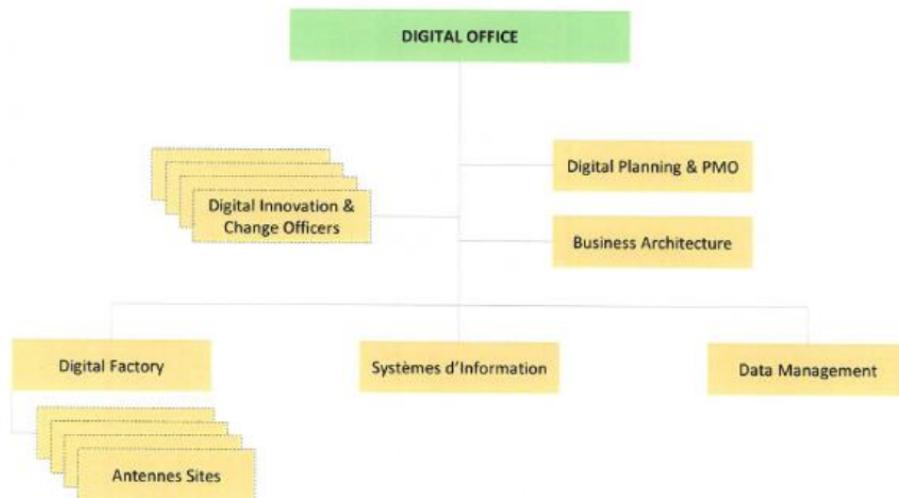


FIGURE 1.1 – Structure de la Digital Office de l'OCP

1.2.2 L'importance de la digitalisation

Le digital est un incubateur de nouveaux modes de fonctionnement à l'échelle du Groupe. Il favorise l'intrapreneuriat, la prise d'initiative et réinvente la manière d'interagir avec l'écosystème du Groupe. Le digital représente aussi un gisement considérable en termes d'innovations et de développement industriel et favorise l'émergence de nouveaux talents au sein du Groupe OCP et de son écosystème. Le challenge de cette transformation digitale réside dans la diffusion d'une culture digitale auprès de l'ensemble des collaborateurs et la conception de solutions innovantes pour enrichir et supporter les différents métiers d'OCP.

1.2.2.1 Objectifs

Renforcer l'efficacité opérationnelle :anticiper les changements et agir en temps réel à travers la promotion de l'Advanced Analytics , augmenter les capacités de production, réduire les coûts, et construire une Supply Chain agile, intégrée et adaptée à un marché dynamique.

Se connecter aux agriculteurs et aux clients :être plus proches de leurs besoins, enjeux et problématiques et leur proposer des expériences (fully digitized).

Développer de nouveaux produits et services :augmenter la flexibilité pour améliorer l'existant et créer de nouvelles offres.

Explorer de nouvelles voies de croissance :introduire des méthodologies de gestion de projets plus agiles, rapides et fluides, à travers des cycles de conception et d'innovation plus courts.

1.2.3 La Digital Factory de l'OCP

La vision digitale du Groupe OCP consiste à devenir un organisme de digitalisation phare de l'industrie dans la région, ainsi que et favoriser l'innovation et les moyens numériques de travailler dans l'ensemble de l'organisation.

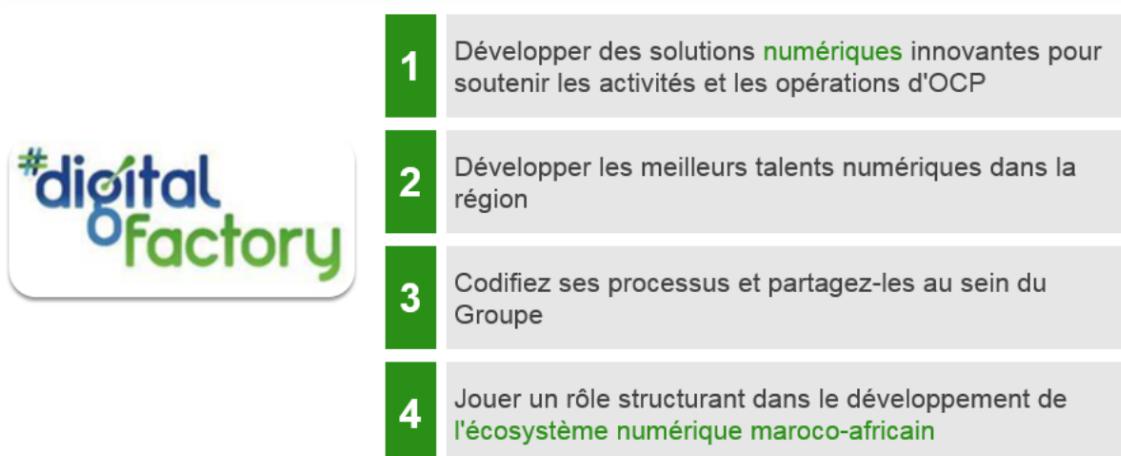


FIGURE 1.2 – les buts majeurs de la Digital Factory de l'OCP

Le Groupe OCP a construit sa nouvelle entité ◁ Digital Factory ▷ qui sera à la pointe de l'innovation et de la transformation numérique de toute l'organisation et au-delà.

La Digital Factory rassemble les compétences, les processus, la culture, et les points d'entrées afin de mener la transformation digital et de livrer des produits digitaux. Elle vise à :

- Construire la capacité interne à posséder la définition et la livraison de produits numériques
- Réduire le ◁ time to market ▷
- Augmenter la qualité de la livraison
- Mettre l'accent sur l'impact et les résultats
- Mettre à l'échelle une culture transformative

- Fournir un plan pour l'avenir du travail qui dynamise l'entreprise et encourage les employés
- Créer un vortex pour l'innovation et la créativité qui attire les meilleurs talents de l'intérieur et de l'extérieur de l'organisation

1.3 Présentation générale du projet

La Digital Factory de l'[OCP](#) a entamé une nouvelle phase de digitalisation avancée relative au processus de gestion des anomalies. Ceci permettra aux employées d'une part d'avoir en temps réel, l'ensemble des informations relatives à chaque étape du processus. Depuis l'alert du problème jusqu'à leur résolution et d'autre part, offrira de la transparence pour être plus performant et économiser en temps et ressources.

L'objectif générale du projet est de réaliser une application web & mobile qui sera utiliser par trois types des employés :

- **Prospecteur** : dont le rôle est d'assurer la qualité / quantité de phosphates en auditant les sites d'extraction.
- **Exploitant** : dont le rôle est d'assurer la continuité de l'exploitation (machines, pilotes ...).
- **Administrateur** : dont le rôle est de gérer l'application.

1.3.1 Présentation du projet de fin d'études

L'extraction du phosphate dans les mines de Khouribga suivent les cinq étapes suivantes :

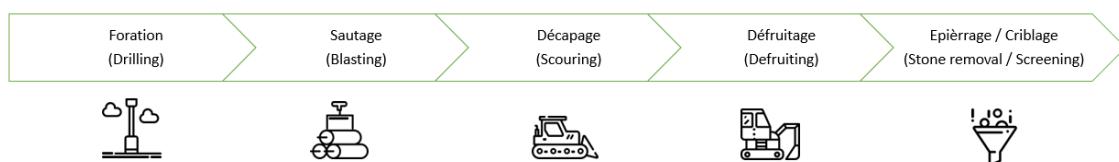


FIGURE 1.3 – Les cinq étapes d'extraction du phosphate

Les prospecteurs fournissent des informations géologiques aux responsables des opérations (Exploitant) avant le début du premier forage, puis suivent l'évolution jusqu'à l'extraction du phosphate.

Après le décapage, la première couche du phosphate est visible et le prospecteur peut prélever un échantillon pour déterminer la qualité du phosphate (HT : Haute

Teneur, BT : Basse Teneur ...). Disons que le premier échantillon prélevé était HT, mais après avoir analysé et stocké le phosphate, nous avons fait un autre test et la qualité était BT.

Pour éviter cette situation, le travail de Prospector consiste à assurer la qualité / quantité du phosphate extrait avant de le stocker et à éliminer toute anomalie en vérifiant et en vérifiant toutes les 5 étapes de l'extraction du phosphate.

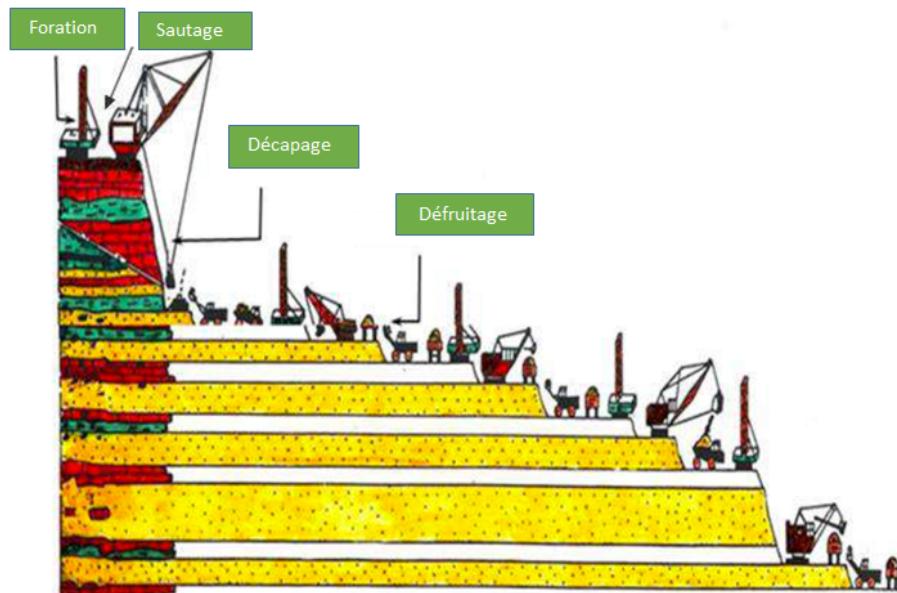


FIGURE 1.4 – Chaîne cinétique d'extraction de phosphate

Les prospecteurs n'ont aucun support numérique lors de leurs inspections quotidiennes. Ils donnent toujours des instructions à Machine Conductors et à Operations Manager (Exploitant) verbalement, sans autre système de suivi que les appels téléphoniques ou les conversations directes.

Comment pouvons-nous aider les prospecteurs à relever et à garder trace des anomalies au cours de leur inspection quotidienne pour éviter les souillures au phosphate ?

1.3.2 Exigences du projet

Après une examination au terrain, les utilisateurs de notre application ont des exigences nécessaires qu'on doit les respecter :

1.3.2.1 Besoins des prospecteurs

- S'assurer que la qualité du phosphate est conforme.

- S'assurer que le phosphate a été complètement exploité
- Communiquer facilement avec les exploitants de la mine
- Mesurer et suivre les anomalies remontées lors de sa tournée.

1.3.2.2 Besoins des exploitants

- Terminer son shift sans incident.
- Diriger son équipe avec plus d'efficacité
- Libérer le chantier le plus rapidement possible
- Veillez au respect des consignes des Prospecteurs
- S'assurer que toutes les machines sont fonctionnelles

1.3.3 Objectifs du Projet

Les exigences initiales indiquent que la solution sera une application mobile aidant les prospecteurs et les exploitants à effectuer leur inspection quotidienne.

- Répétition des consignes et anomalies
- Aucun moyen de suivi des anomalies remontées
- Disponibilité de l'Exploitant sur chantier
- Moyens de communications limités entre le prospecteur et l'exploitant
- Le téléphone ne capte pas toujours dans les mines

À l'aide d'une série d'invites, nous pouvons résoudre le problème initial et définir une feuille de route claire et cohérente qui nous aidera à développer un MVP.

En les aidant à se sentir en sécurité et à être productifs.

1.4 Conduite du projet

1.4.1 Organisation du projet

Après avoir passé une période de documentation et d'intégration au sein de la Digital Factory de l'OCP, j'ai intégré une équipe qui se constitue de 6 personnes tout en respectant la structure d'une équipe dans le cadre Scrum. La figure suivante illustre l'architecture de notre équipe :

- Product Owner (PO),
- 1 équipe de développement constituée de 3 développeurs,
- 1 Scrum Master (SM).

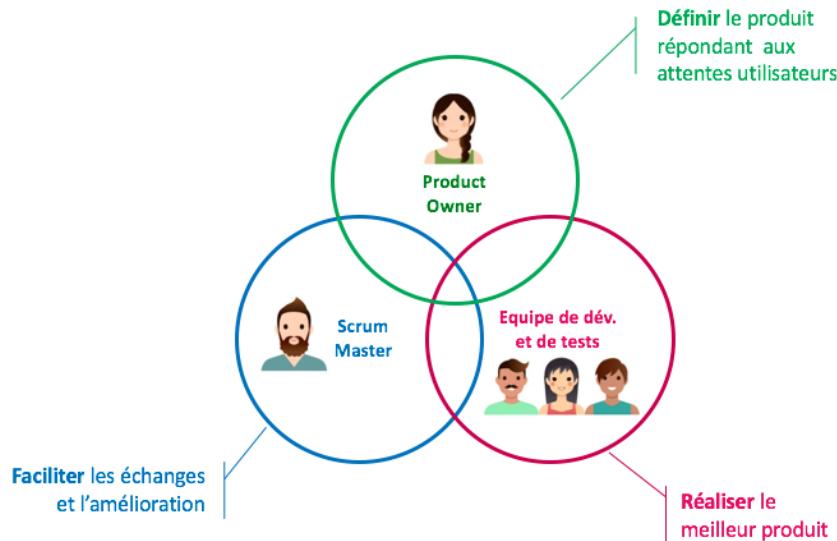


FIGURE 1.5 – Équipe de projet

Notre équipe Scrum est auto-organisées et pluridisciplinaire. C'est à dire nous choisissons la meilleure façon d'accomplir notre travail, au lieu d'être dirigées par des personnes externes à l'équipe.

1.5 Méthodologie de travail

1.5.1 Méthodologie Scrum

Afin de dérouler le projet dans des conditions standards qui s'inspirent des méthodologies de travail les plus efficaces et dans le cadre de mon équipe de travail nous avons choisi la méthodologie Scrum pour la gestion du projet.

Le choix de cette méthodologie est dû à plusieurs raisons. Dans un premier temps cette méthodologie favorise la productivité au sein d'une équipe de développement en travaillant sur des objectifs prioritaires et à court terme et en suivant un développement itératif et incrémental avec une planification évolutive basée sur la division de l'ensemble des tâches selon des unités qu'on appelle sprint. Aussi le choix de cette méthodologie tiens en compte la tendance du contexte de développement dans le monde qui réside dans l'adaptation de l'esprit agile durant la réalisation des projets, un esprit ou Scrum est l'une de ses méthodologies pilier.

Le choix de Scrum comme méthodologie présente plusieurs avantages liés à la productivité ainsi que l'organisation du travail en se basant sur des rôles définis, ce qui n'existe pas dans les méthodologies traditionnelles, qui représentent plusieurs

inconvénients, ceux de la rigidité ainsi que la génération massive de la documentation et la difficulté d'introduction des changements. Ceci justifie le choix de Scrum comme méthodologie de développement qu'on résume selon une vision d'ensemble dans la figure suivante :

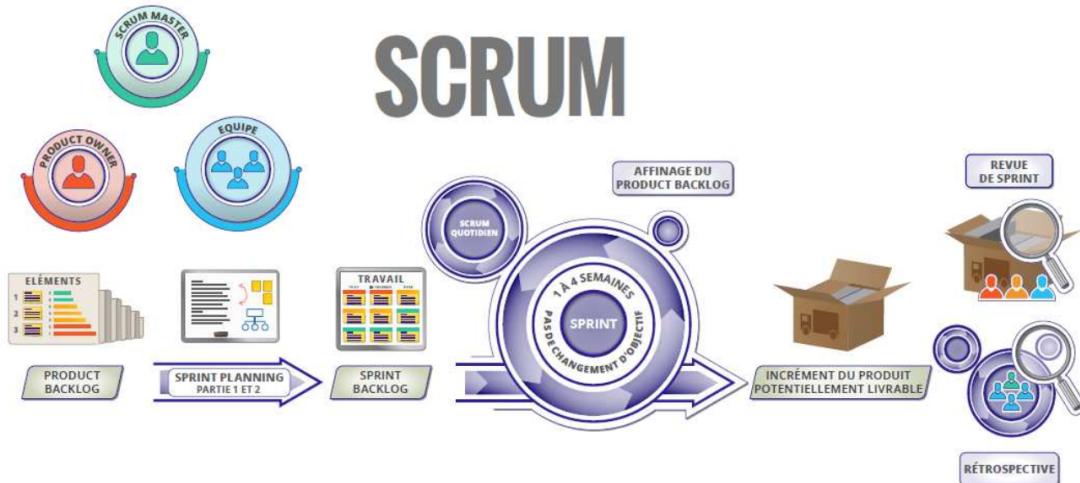


FIGURE 1.6 – Méthode SCRUM

Comme le montre la figure la répartition des tâches se fait selon des rôles bien précis. Dans notre cas, l'équipe se constitue de 8 personnes ayant les profils suivants :

Le PO(Product Owner) : c'est un expert métier qui définit dans un premier temps les spécifications fonctionnelles. Aussi, il établit la priorité des fonctionnalités à développer ou corriger et valide les fonctionnalités développées. Bref, le PO joue le rôle du client.

Le Scrum Master : c'est expert Agile qui s'assure que les principes et les valeurs de Scrum sont respectés. Il assure la communication au sein de l'équipe ainsi que la recherche d'amélioration de la productivité et du savoir-faire de notre équipe.

UX Designer : il fait le design de chaque user story selon l'expérience d'utilisateur (user experience). L'UX design a plus un rôle de conception de produit qu'un rôle de conception graphique.

L'équipe de développement : constituée de 6 personnes qui s'occupent du développement des user stories ainsi que la réalisation des tests unitaires et des tests d'intégration.

1.5.2 Outils de suivi

1.5.2.1 JIRA Atlassian

Le framework agile Scrum repose entre autres sur le principe de (transparence). Certaines informations doivent donc être accessibles par tous, comme la tâche en cours de chacun, son état d'avancement, et l'objectif actuel de l'équipe. D'où l'importance que ces informations soient visibles en permanence.

C'est le tableau Scrum qui va jouer ce rôle. Il permet d'organiser le backlog, les tâches du sprint en cours et leur état d'avancement. Les tableaux Scrum peuvent être aussi simples qu'un tableau blanc et des posts-its, ou peuvent revêtir un format plus élaboré avec des logiciels spécialisés disposant de graphiques et de fonctionnalités de gestion des tâches plus avancées.

Pour notre tableau Scrum, nous utilisons JIRA Atlassian. Notre tableau est divisé en 5 listes qui correspondent au flux de travail des tâches :

- **À Faire** : quand je planifie mon sprint, je déplace les tâches du backlog vers cette liste.
- **En Cours** : contient les tâches en cours de développement et de réalisation.
- **Terminé** : la tâche est complète dans la phase du développement mais en attente de la validation fonctionnelle par le PO
- **Approuvé** : une fois la tâche est approuvée fonctionnellement par le PO on peut la placer dans la colonne APPROUVÉ.
- **Bloqué** : j'utilise cette liste lorsque la finalisation d'une tâche dépend d'un facteur externe (par exemple, je dois réaliser un achat et obtenir l'aval de mon PO), en spécifiant les raisons du blocage dans un commentaire.

The screenshot shows a JIRA board with five columns: A FAIRE (22), EN COURS (5), TERMINÉ (34), APPROUVE (33), and BLOQUÉ. Each column contains several cards representing user stories or tasks. The cards include details like story ID, assignee, and labels like 'GESTION ANOMALIES' or 'GESTION DES TOURNÉES'. The 'TERMINÉ' column has a large number of cards, while the others have fewer.

FIGURE 1.7 – Capture de l’outil JIRA

1.5.2.2 Slack

Slack est une plateforme de communication collaborative sur ordinateur et smartphone. Chaque entreprise peut créer un groupe privé sur Slack, et y inviter tout ou partie de ses employés, qui peuvent ainsi discuter entre eux.

Avec les conversations instantanées classées par (chaînes), la communication se fluidifie, il est possible d’interagir en temps réel, tout en s’adressant uniquement aux personnes concernées, au contraire de l’e-mail.

L’autre atout de l’application : toutes les interconnexions qu’elle permet avec d’autres logiciels. L’outil Slack est pratique pour recevoir une notification dès chaque modification dans un document Drive.

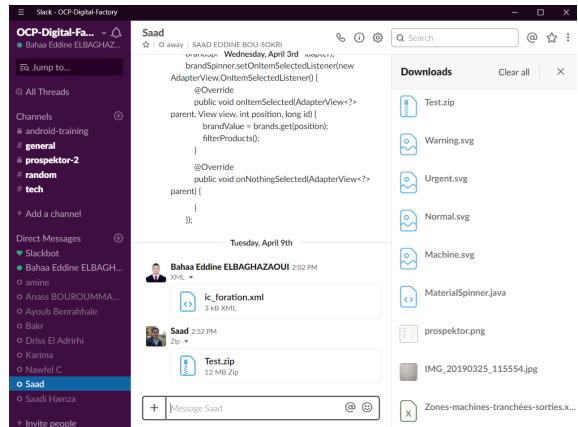


FIGURE 1.8 – Capture de l’outil Slack

1.6 Conclusion

Dans ce chapitre nous avons présenté l'ensemble des éléments qui permettent la situation de notre Projet de Fin d'études dans son contexte organisationnel ainsi que la démarche de gestion du projet qui organise son déroulement et les outils utilisés. Par la suite dans le chapitre suivant on va mettre l'accent sur l'étape de l'analyse et spécification des besoins qui permettra la collection des différents besoins afin de concevoir une solution qui répondra aux exigences exprimées

Chapitre 2

Analyse et spécification des besoins

2.1 Introduction

Ce chapitre est consacré à l'analyse et à la spécification des besoins fonctionnels et non fonctionnels de la solution qui est une étape primordiale pour la réalisation de notre projet.

2.2 Analyse des besoins

Dans cette partie, nous présenterons les besoins fonctionnels et non fonctionnels identifiés après la sélection des besoins.

2.2.1 Identification des acteurs

Dans le cas de notre projet on considère trois acteurs :

- **Le prospecteur** : Il a comme mission principale de gérer les anomalies de la plateforme.
- **L'exploitant** : Il permet de répondre sur les anomalies qui ne sont pas encore résolu.
- **L'administrateur** : Il permet de gérer les autres utilisateurs (prospecteurs, exploitants), les machines ,les géolocalisation

2.2.2 Les besoins fonctionnels

Au cours de cette étape, nous allons extraire les différentes fonctionnalités offertes par notre projet.

- L'application **prospektor** doit permettre à chaque prospecteur de :
 - Créer des anomalies lors de leur exploitation.
 - Ajouter des attachements aux anomalies qui a créée.
 - suivre et consulter les anomalies qui a créer.
 - consulter la liste des attachements liées par anomalies.
 - Notifier les exploitants au cas d'un dérangement fatal.
- L'application **prospektor** doit permettre à chaque exploitant de :
 - suivre et consulter les anomalies par étape, date & criticité.
 - consulter les attachements liées par anomalies.
 - Ajouter des attachements aux anomalies qui ne sont pas encore résolu.
 - Faire des interventions au cas d'une perturbation.
- L'application **prospektor** doit permettre à chaque administrateur de :
 - Gérer :
 - Prospecteurs
 - Exploitants
 - Géolocalisation (Mine, Zone, Tranchée, Sortie)
 - Machines
 - Consulter :
 - Tournées
 - Chantiers
 - Erreurs

2.2.3 Les besoins non fonctionnels

Outre les fonctions citées ci-dessus, l'application doit assurer en certaine mesure les caractéristiques suivantes :

- **L'efficacité** : L'efficacité de l'application doit permettre l'accomplissement de la tâche avec le minimum de manipulation. Ceci doit être garanti pour que l'application puisse s'intégrer facilement dans l'environnement où elle va être déployée.
- **La sécurité** : Les différents comptes utilisés par les utilisateurs doivent être sécurisés et vérifiés pour éviter les faux comptes et les fausses informations.
- **La fiabilité** : Touche à l'aspect qualité des données et persistance des informations dans l'application ainsi que la vitesse de chargement des interfaces.
- La performance : le temps de réponse de la plateforme doit être rapide.
- **La maintenabilité** : La solution doit être stable face aux changements, ainsi

qu'un fort niveau de testabilité assuré par les tests fonctionnels.

- **La scalabilité** : la solution doit d'être extensible en termes de la charge des requêtes traitées.
- **L'évolutivité** : possibilité d'ajout des nouvelles fonctionnalités au cours du temps selon le besoin des fournisseurs.
- **Le déploiement intelligent** : l'introduction des nouveaux changements ne doit pas impacter les modules existants, d'où le besoin d'une démarche de déploiement intelligente de chaque module.
- **La portabilité** : facilité de passage d'un environnement de développement et tests vers un environnement de pré-production ou un environnement de production.

2.2.4 Diagramme des cas d'utilisation

Dans cette phase, on va traiter les différents cas d'utilisation et identifier les différentes interactions entre le système et les acteurs identifiés.

la figure suivante représente le diagramme de cas d'utilisation globale de l'acteur prospecteur :

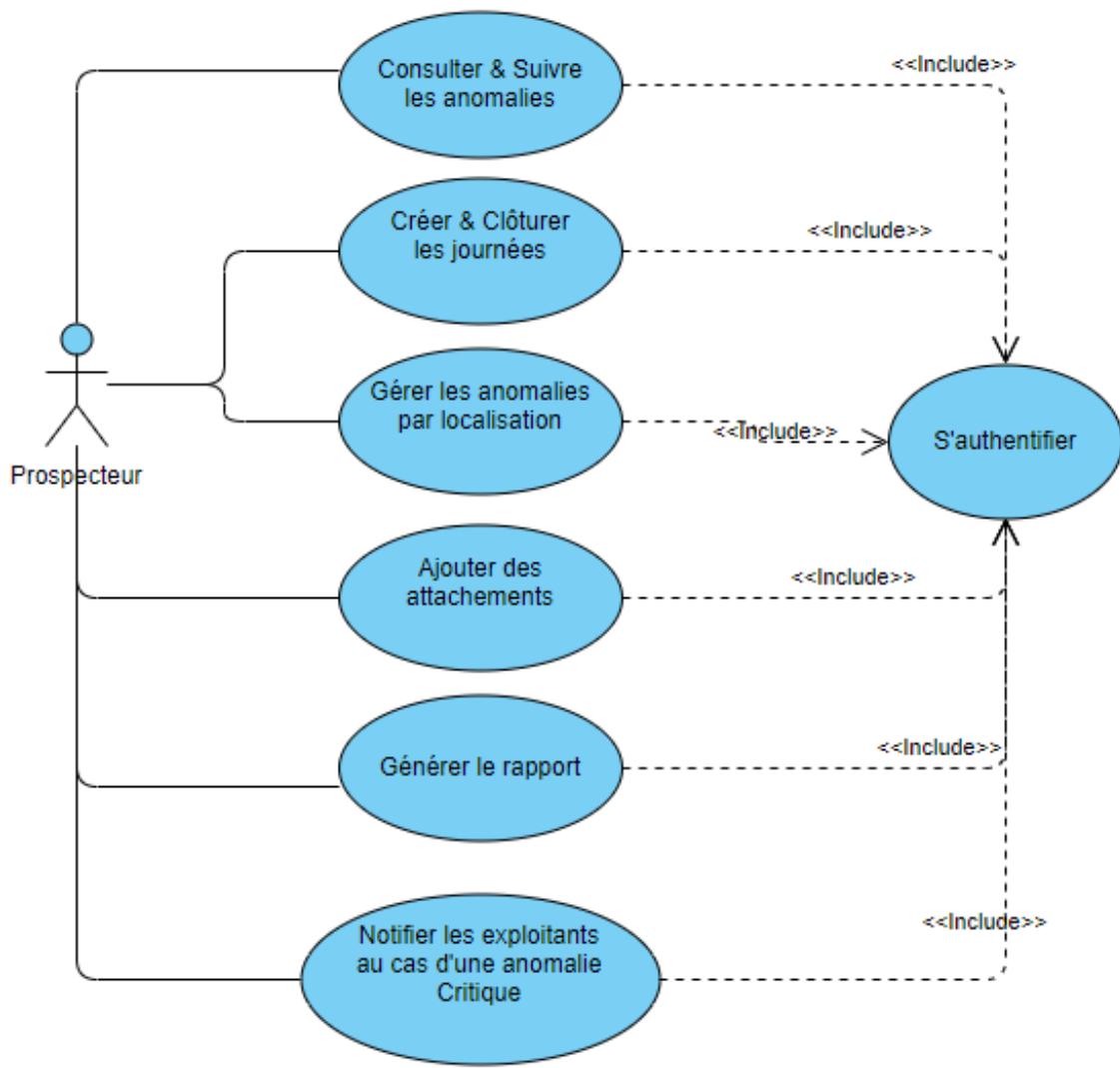


FIGURE 2.1 – Diagramme de cas d'utilisation globale pour le prospecteur

Cette figure représente le diagramme de cas d'utilisation globale de l'acteur exploitant :

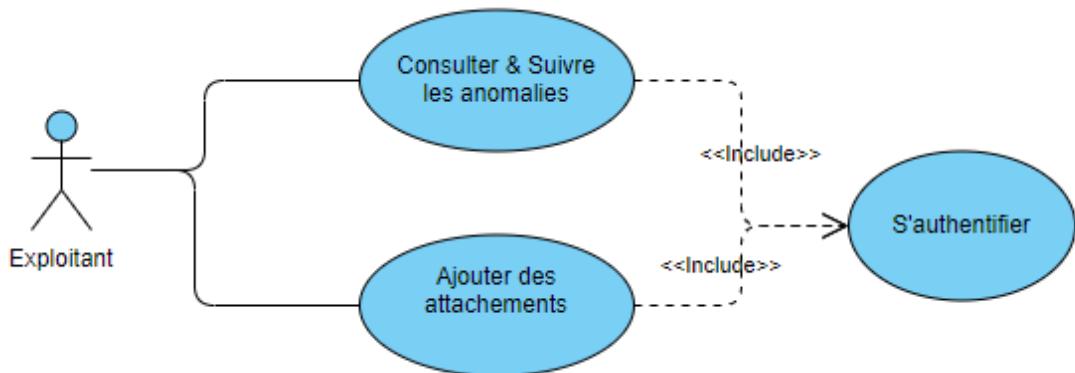


FIGURE 2.2 – Diagramme de cas d'utilisation globale pour l'exploitant

Cette figure représente le diagramme de cas d'utilisation globale de l'acteur administrateur :

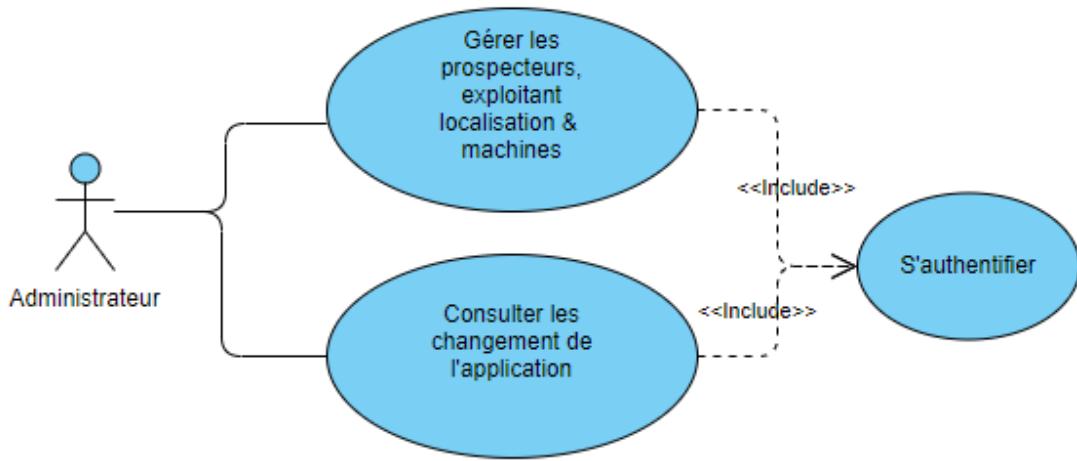


FIGURE 2.3 – Diagramme de cas d'utilisation globale pour l'administrateur

2.2.5 Description des cas d'utilisation

Prospecteur : Les grandes étapes pour un prospecteur lors de l'utilisation de l'application prospektor.

- Ouvrir l'application mobile prospektor
- S'authentifier
 - Consulter les anomalies en cours de traitement
 - Consulter les demandes d'intervention
 - Consulter l'historique des anomalies
 - Consulter le rapport des tournées
 - Démarrer & Continuer une tournée
 - spécifier la localisation du prospection
 - Remplir la check-list des éléments à auditer
 - Création d'une anomalie au cas d'un dérangement
 - Ajouter des attachments (audio & photos) si nécessaire
 - Continuer la tournée vers un autre chantier
 - Obtenir le rapport complet de la tournée

Les exceptions pour un prospecteur lors de l'utilisation de l'application prospektor sont :

- Email ou mot de passe sont incorrecte.
- Créer une anomalie sans préciser leur localisation.

- Créer une anomalie sans description.
- Générer le rapport sans terminer toutes les chantiers.
- ...

Exploitant : Les grandes étapes pour un exploitant lors de l'utilisation de l'application prospektor.

- Ouvrir l'application mobile prospektor
- S'authentifier
 - Consulter les anomalies en cours
 - Consulter les anomalies à traiter
 - Consulter l'historique des anomalies

Les exceptions pour un exploitant lors de l'utilisation de l'application prospektor sont :

- Email ou mot de passe sont incorrecte.
- Repondre à une anomalie sans description.
- Repondre à une anomalie déjà clôturer.
- ...

Administrateur : Les grandes étapes pour un administrateur lors de l'utilisation de l'application prospektor.

- Ouvrir l'application web prospektor
- S'authentifier
 - Consulter les utilisateurs (prospecteur & exploitant).
 - Ajouter un utilisateur.
 - Supprimer un utilisateur.
 - Modifier un utilisateur.
 - Désactivé un utilisateur.
 - Consulter les détails d'un utilisateur.
 - Gérer les localisations (Mine, Zone, Tranchée, Sortie).
 - Consulter les machines.
 - Ajouter une machine.
 - Supprimer une machine.
 - Modifier une machine.
 - Désactivé une machine.
 - Consulter les détails d'une machine.
 - Consulter les tournées.
 - Consulter les chantiers par zones ou par étape.
 - Consulter les anomalies par criticité.

— ...

Les exceptions pour un administrateur lors de l'utilisation de l'application prospektor sont :

- Email ou mot de passe sont incorrecte
- Ajouter un utilisateur qui n'existe pas dans le group ocp
- Ajouter une machine sans preciser leur localisation
- ...

2.3 Planning du projet

On a drésse le RoadMap qui va être réaliser entre Mars et Juin sur deux parties. Chaque partie va prendre 8 semaines de travail.

Pour notre projet prospektor, On a définie

- 1 Sprint = 1 Semaine
- Daily Meeting & check à 10 :00 AM
- Sprint planning chaque mercredi matin

2.3.1 Minimum Viable Product

Cette version permet d'offrir une consultation complète pour le développement de notre produit.

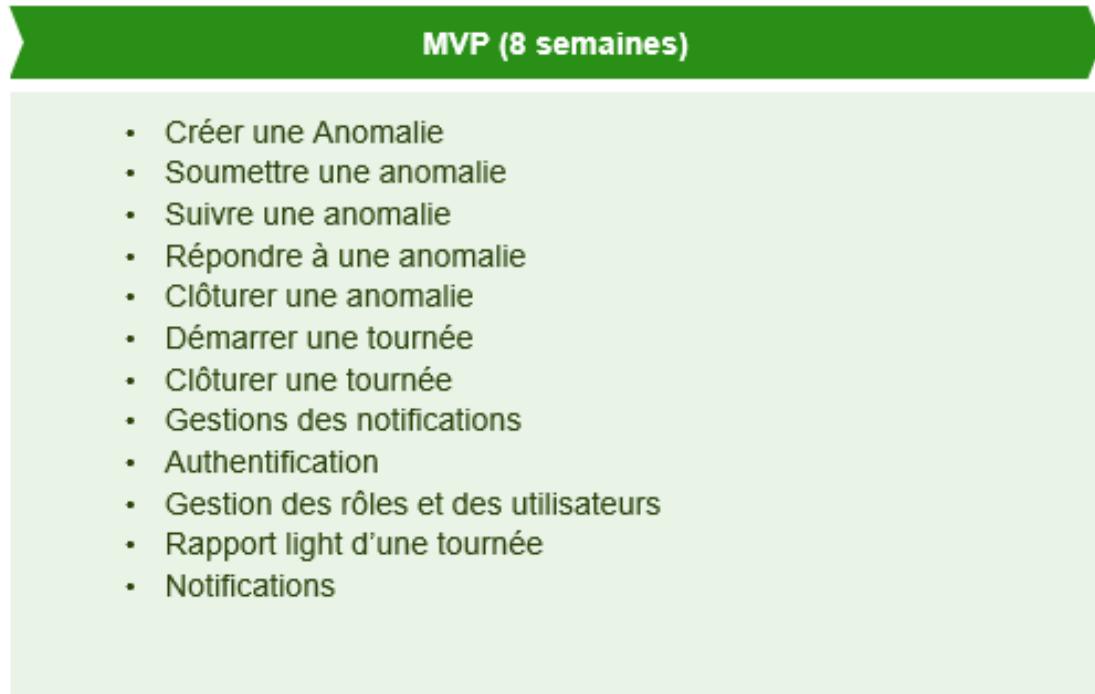


FIGURE 2.4 – RoadMap de la version MVP

Cette version nous permet de gérer les anomalies lors d'une exploitation, notifier les exploitants, stocker les attachments ainsi avoir un rapport sur chaque tournée.

2.3.2 Post Minimum Viable Product

Cette version est celle qui suit la première version.

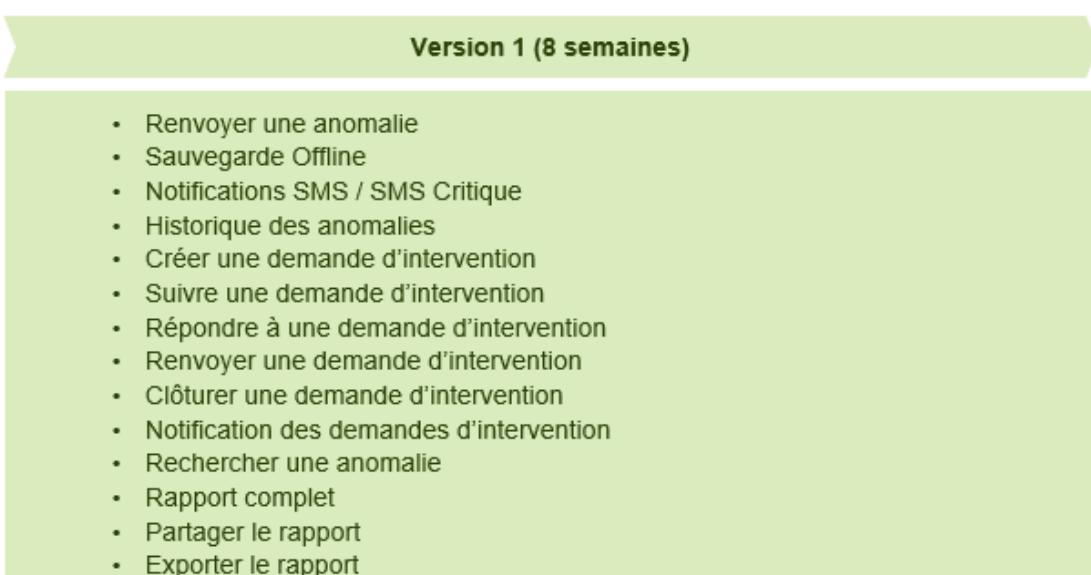


FIGURE 2.5 – RoadMap de la version Post MVP

Cette version nous permet d'utiliser notre application dans les mines d'OCP, Stocker les données localement, synchroniser l'application et exporter les rapports des tournées pour les utiliser dans autres projets.

2.4 Conclusion

Dans ce chapitre nous avons détecté des besoins fonctionnels et non fonctionnels qui sont un complément de l'existant. Par la suite nous avons poussé l'analyse des besoins vers les diagrammes de cas d'utilisation afin de visualiser les différentes fonctionnalités de la solution.

Chapitre 3

Conception générale du projet

3.1 Introduction

Ce chapitre est consacré à la conception générale du projet. Après la définition des besoins fonctionnels et non fonctionnels, nous allons passer à expliquer l'architecture globale du système qui représente une vue générale de notre solution.

3.2 Conception générale

Cette partie se base sur la conception globale de l'architecture fonctionnelle de notre projet. Effectivement, elle explique les composants de notre système ainsi la communication entre eux.

3.2.1 Architecture globale du système

L'architecture de notre système est composée par :

- **F5 VPN** : utilise le protocole Secure Sockets Layer, une technologie d'authentification et de cryptage intégrée à chaque navigateur Web, pour créer une connexion sécurisée et cryptée sur un réseau moins sécurisé, comme Internet.
- **MINIO** : vous permet d'utiliser un seul **NFS** (comme **NFS**, GlusterFS et d'autres systèmes de fichiers distribués) en tant que système de stockage pour plusieurs serveurs MinIO. La synchronisation entre les serveurs MinIO est prise en charge par la conception.
- **AD** est un système basé sur une base de données qui fournit une authentification, un annuaire, une stratégie et d'autres services dans un environnement

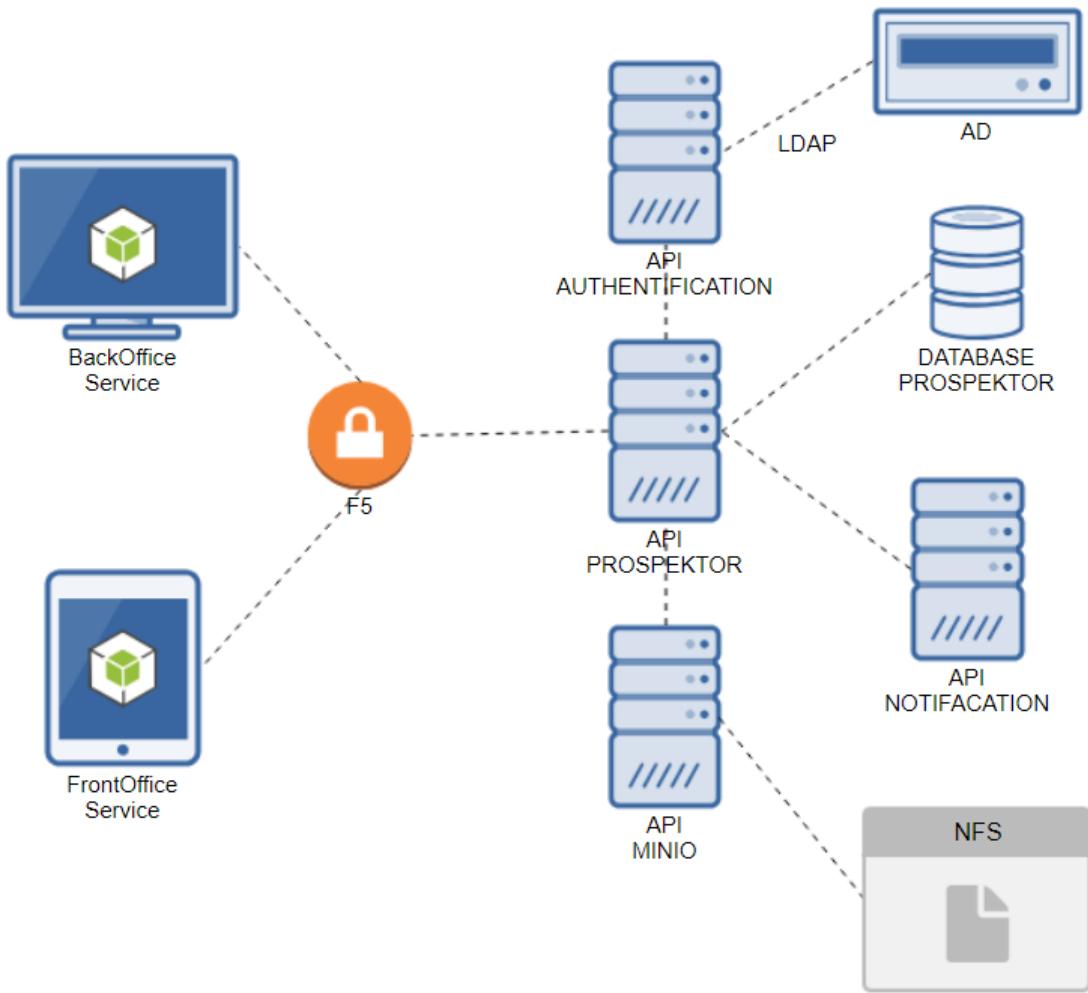


FIGURE 3.1 – Architecture du système

Windows.

- **LDAP** est un protocole d’application permettant d’interroger et de modifier des éléments dans des fournisseurs de services d’annuaire tels qu’Active Directory, qui prend en charge une forme LDAP.
- **NFS**, littéralement système de fichiers en réseau, est à l’origine un protocole qui permet à un ordinateur d’accéder via un réseau à des fichiers distants.

3.2.2 Architecture applicative

D’après l’architecture globale du système, on constat qu’elle est composé par deux parties :

- **Partie backend** : qui se représente par l’API prospektor, est la partie serveur ou partie logique métier.

- **Partie frontend** : qui représente la partie client ou partie interface.
Dans notre projet, elle se décompose par deux parties :
 - Une application de **back-office** comprend le logiciel utilisé par une organisation pour administrer des opérations qui ne sont liées à aucun effort de vente directe et à des interfaces qui ne sont pas vues par les consommateurs.
 - En revanche, une application de type **front-office** serait une interface client facilitant la vente ou le traitement d'une transaction.

3.2.2.1 Cas général pour le backend

Pour les services Backend le choix de l'architecture applicative suit une décomposition en trois couches (3-tiers) :

- Couche DAO : c'est la couche d'accès aux données persistant dans la SGBD.
- Couche Services : implémente la logique métier du service, et les différentes règles de gestion qui représentent les fonctionnalités du service.
- Couche API : il permet l'encapsulation des différents services dans une API RESTfull et fournit les contrôleurs l'accès à ces services.

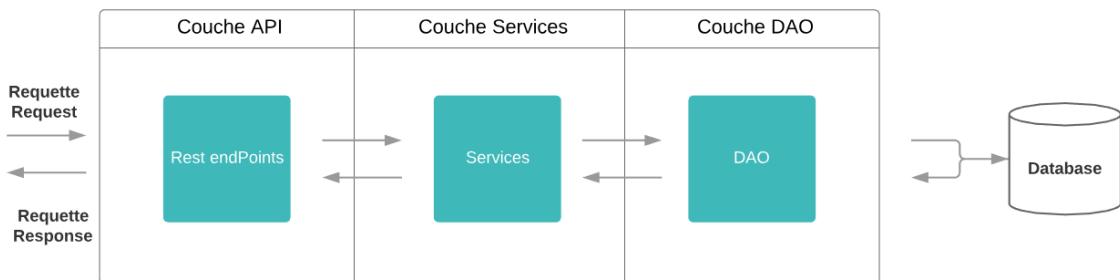


FIGURE 3.2 – Architecture 3-tiers

On a aussi implémenté l'architecture **MVC** qui signifie Modèle, Vue et Contrôleur. MVC sépare les applications en trois composants :

- **Modèle** : Représentations de modèle sous forme de données et de logique applicative. Il contient les données de l'application. Les objets de modèle récupèrent et stockent l'état du modèle dans une base de données.
- **View** : View est une interface utilisateur. Visualisez les données d'affichage en utilisant le modèle pour l'utilisateur et leur permettre également de modifier les données.
- **Contrôleur** : le contrôleur gère la demande de l'utilisateur. En règle générale, l'utilisateur interagit avec View, ce qui déclenche à son tour la demande

d'URL appropriée. Cette demande sera gérée par un contrôleur. Le contrôleur rend le approprié afficher avec les données du modèle en réponse.

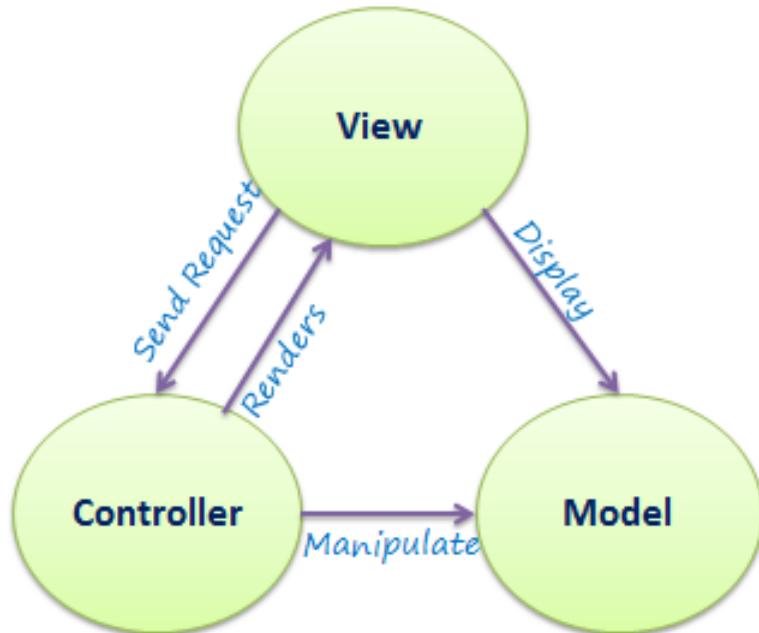


FIGURE 3.3 – Architecture MVC

On obtient comme architecture final pour le coté backend la figure suivante :

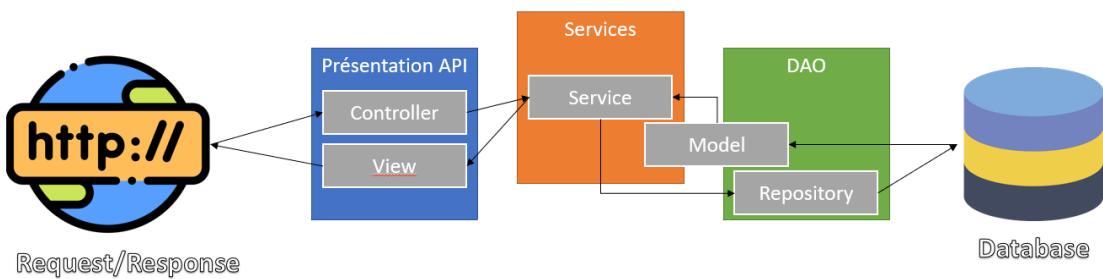


FIGURE 3.4 – Architecture applicative backend

3.2.2.2 Cas général pour le backoffice

L'architecture qu'on va utiliser agit comme un conteneur d'état et facilite la gestion du flux de données de notre application. Il a été introduit en 2015 lors de la conférence [ReactEurope](#) de [Dan Abramov](#). Elle ressemble à l'architecture Flux et a beaucoup en commun avec elle.

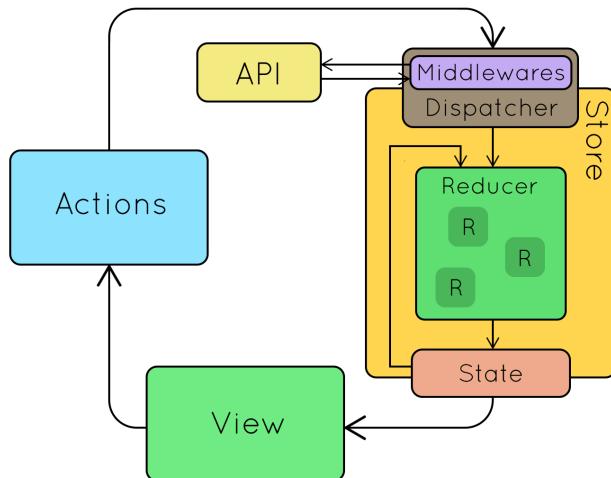


FIGURE 3.5 – Architecture Applicative backoffice

Nous avons les composants de vue qui envoient une action. La même action peut être envoyée par une autre partie de notre système. Cette action est envoyée non pas à un hub central, mais directement au store. Nous disons "store" et non "stores" car il n'y en a qu'un global. La logique qui a décidé comment nos données changent vit dans des fonctions pures appelées reducers. Une fois que le store reçoit une action, il demande aux reducers la nouvelle version de l'état en envoyant l'état actuel et l'action en question. Ensuite, de manière immuable, le reducer doit retourner le nouvel état. Le store continue à partir de là et met à jour son état interne. Enfin, le composant câblé au store est rendu à nouveau.

3.2.2.3 Cas général pour le frontoffice

Le modèle de conception architecturale **MVP** est un modèle de conception assez connu des développeurs Android. Il vous permet de découpler la logique métier de la logique de vue (Activité / Fragment) en introduisant un intermédiaire appelé Présentateur.

Comme son nom l'indique, **MVP** est divisé en trois couches différentes.

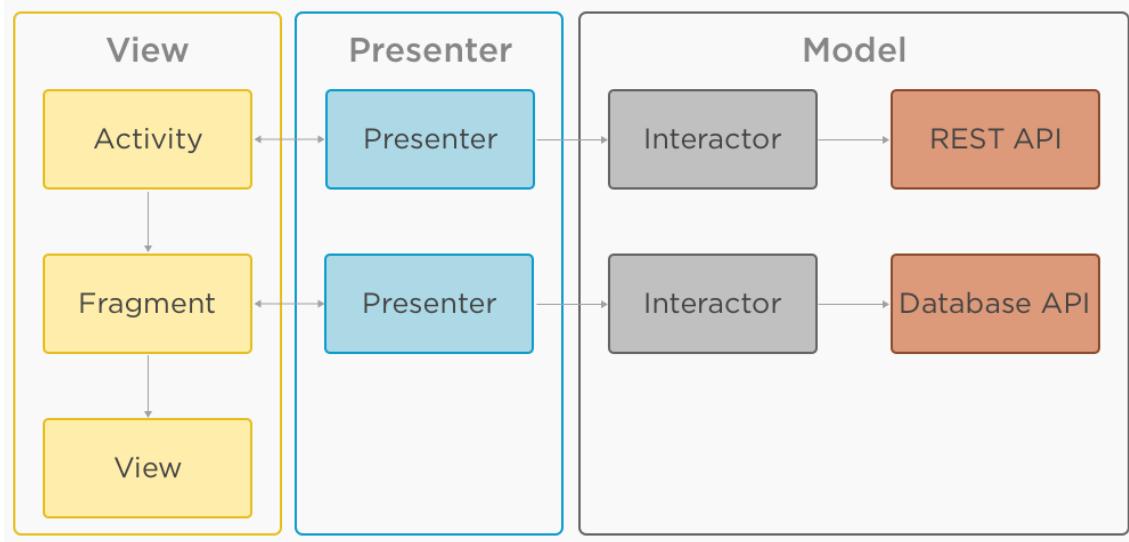


FIGURE 3.6 – Architecture Applicative frontoffice

- **Modèle** - Comme mentionné ci-dessous, où est stockée votre logique métier et votre application de données ? Sous Android, le rôle d'un modèle est généralement joué par l'[API](#) ou l'[API REST](#).
Il est non seulement responsable du stockage des données de l'application, mais également de composants contenant des responsabilités pour la génération, l'exposition et la récupération des données.
En général, toutes ces fonctionnalités sont exécutées en arrière-plan car elles peuvent bloquer le thread d'interface utilisateur.
- **View** - View est essentiellement une interface d'utilisateur passive qui est responsable du routage de l'action de l'utilisateur vers le présentateur.
En général, l'affichage n'est pas visible pour votre modèle, à l'exception du POJOS et des entités de l'application. Pour faire plus simplement, les vues ne communiquent pas directement avec les modèles. Cependant, ils parlent aux présentateurs.
- **Presenter** - Le présentateur est l'intermédiaire ou le médiateur entre View et Model.
En termes généraux, Presenter interroge le modèle et met à jour la vue tout en répondant aux interactions de l'utilisateur.
Il surveille la façon dont ils sont, et ils ne peuvent pas le gérer.

3.2.2.4 Les services externes

Dans notre projet, on est besoin de trois services externes qui sont déjà réalisé par l'entité DF. La communication avec ces services se base sur les requêtes HTTP & leurs utilités sont :

- **Service d'authentification** : permet de vérifier l'identité d'un utilisateur, est ce qu'il est appartient au groupe OCP.
- **Service Minio** : permet de stocker les attachements des anomalies. On va utiliser le protocole NFS lors du stockage.
- **Service Notification** : permet d'envoyer des messages SMS.

3.3 Entity relationship diagram

Un certain nombre de techniques de modélisation de données sont utilisées aujourd'hui. L'un des plus courants est le diagramme de relation d'entité (ERD). Plusieurs notations ERD sont disponibles. Nous avons utilisé la notation du Crow's Foot.

La **cardinalité** et la **modalité** sont les indicateurs des règles de gestion entourant une relation. La cardinalité fait référence au nombre maximum de fois qu'une instance d'une entité peut être associée à des instances de l'entité associée. La modalité fait référence au nombre minimum de fois qu'une instance d'une entité peut être associée à une instance de l'entité associée.

Symbole	cardinalité
	zéro ou plus
	1 ou plus
	1 et seulement 1
	zéro ou 1

TABLE 3.1 – Cardinalité du Crow's Foot

Le diagramme du crow's foot de notre projet prospektor est représenté dans le figure ci-dessous :

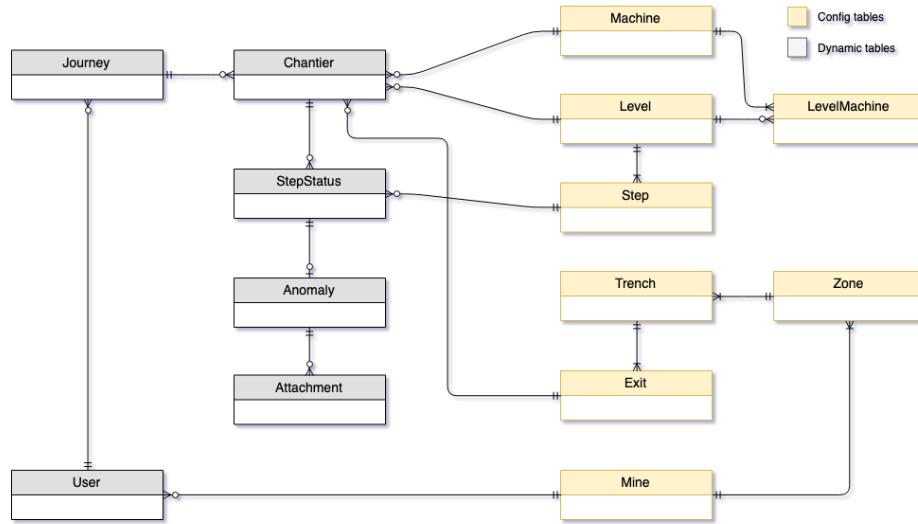


FIGURE 3.7 – Diagramme de Crow's Foot

Cette diapositive représente le diagramme des entités qui sont utilisées dans notre projet. Elles sont composées à deux types :

- **Statique** : représenté par la couleur jaune, contient des informations statiques sur les entités de géolocalisation (Mine, Zone, Tranchée, Sorties), les machines existantes et les phases d'extraction de phosphates.
- **Dynamique** : représenté par la couleur gris, contient des informations dynamiques et changeables, par exemple : les utilisateurs, les anomalies, les attachements, les chantiers visites ainsi que les tournées de chaque exploitants.

Dans la figure ci-dessous

3.4 Conclusion

Dans ce chapitre nous avons entamé la conception générale du projet en décrivant l'architecture du projet de manière détaillée. Pour les technologies utilisées et les outils de développements, on va les présenter dans le prochain chapitre.

Chapitre 4

Implémentation de la solution

4.1 Introduction

Ce chapitre aborde comme sujet les choix technologiques et les outils pour l'implémentation de notre solution ainsi que les captures d'écran des différents fenêtres réaliser et les tests de validation effectués sur les modules développés.

4.2 Environnement logiciel

L'environnement logiciel utilisé pour la réalisation du Prospektor sont :

Outil	Description
JDK 1.8	Java Development Kit (JDK) désigne un ensemble de bibliothèques logicielles de base du langage de programmation Java.
NodeJS 10.16.0 LTS	est un moteur d'exécution JavaScript basé sur le moteur JavaScript V8 de Chrome.
IntelliJ IDEA	Environnement de développement intégré (IDE).
Visual Studio Code	Environnement de développement côté frontend.
Android Studio	Environnement de développement pour développer des applications mobiles Android.
GitBash	C'est une ligne de commande dans laquelle on peut exécuter les commandes git.
Zeplin	C'est un outil de design des fonctionnalités. Permet de collaborer entre les designers et les développeurs frontends.
Postman	Est actuellement l'un des outils les plus populaires utilisés dans les tests d' API .

TABLE 4.1 – Environnement logiciel

4.3 Architecture technique du système

Notre système se base en totalité sur l'architecture orienté service et plus précisément sur l'architecture **REST**. C'est un style d'architecture pour la conception d'applications faiblement couplées sur **HTTP**, souvent utilisé dans le développement de services Web.

Une étude qui a été faite par l'entité **DF**, ils sont convaincu d'adapter quelques technologies & outils dans tous leur projets.

La figure suivante représente les technologies & les bibliothèques essentiels utilisés dans notre projet (Prospektor).

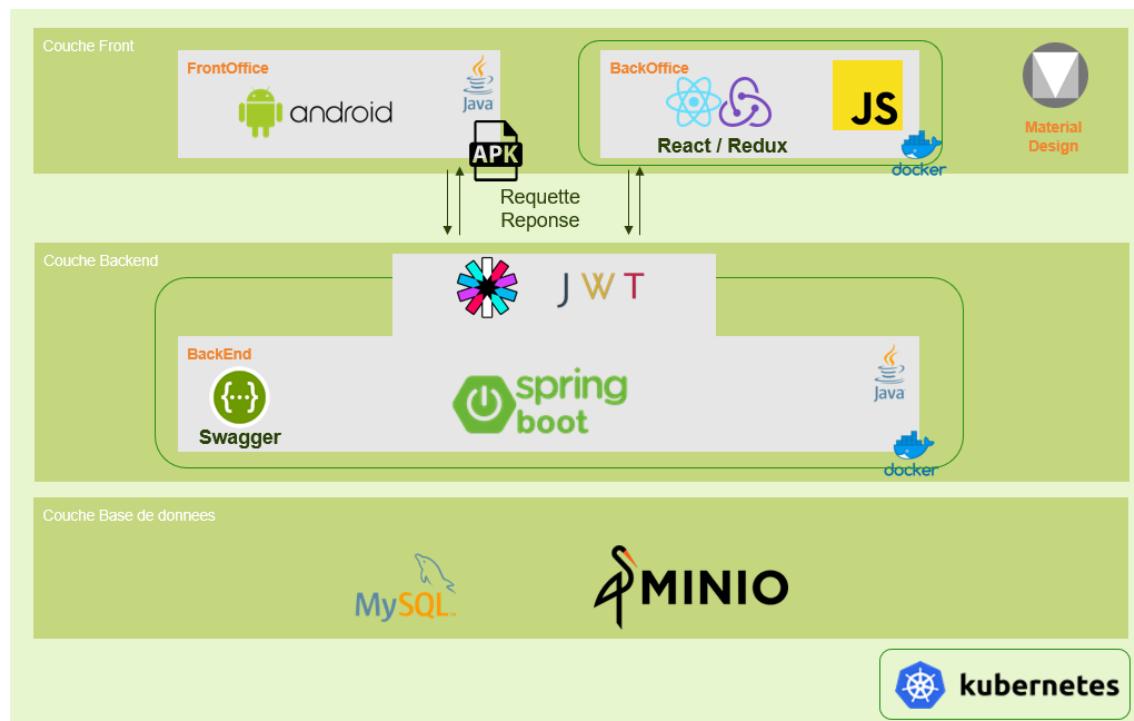


FIGURE 4.1 – Architecture technique du système

Maintenant, on va expliquer l'utilité des technologies utilisées dans notre implémentation.

4.3.1 Couche BackEnd

Pour la partie backend, on va utiliser les technologies suivantes :

- **SpringBoot** : est un framework Java open source utilisé pour créer un Micro Service. Il est facile de créer des stand-alone applications. Spring Boot contient une prise en charge complète de l'infrastructure pour le dévelop-

pement d'un micro-service et vous permet de développer des applications d'entreprise.

Pourquoi Spring Boot est un excellent choix pour Prospektor :

- Spring Boot est basé sur le framework Spring.
- Les modules centraux de Spring sont stables pendant une longue période et la plupart des modifications sont compatibles avec les versions antérieures.
- Basé sur la machine virtuelle Java (JVM).
- Spring Boot et les principaux modules Spring de l'écosystème sont des logiciels open source. Cependant, il est fortement entretenu et soutenu par Pivotal, une société proposant des plates-formes et des outils permettant de créer de meilleurs logiciels.
- Son utilisation est gratuite.
- Exécutez-le sur notre propre serveur, machines virtuelles, conteneurs ou hôte sur Heroku, AWS ou similaire. C'est à nous de décider.
- Avec Spring, nous pouvons facilement connecter notre application à des bases de données relationnelles, à des bases de données NoSQL ou à des services de file d'attente.
- Prend en charge Oracle, PostgreSQL, MySQL, MongoDB, Redis, Solr, ElasticSearch, Rabbit MQ, ActiveMQ et bien d'autres encore.
- Avec Spring Boot, nous pouvons développer des applications Web rendues typiques côté serveur, RESTful et d'autres [API](#) Web, ou même créer des travaux par lots et des applications de ligne de commande standard.
- Les développeurs adorent programmer avec Spring Boot. Ils sont plus productifs, bénéficient des avantages de l'écosystème Spring et de la tranquillité d'esprit des systèmes de production en fonctionnement.
- **Swagger** : nous permet de décrire la structure de nos [API](#) afin que les machines puissent les lire. La capacité des [API](#) à décrire leur propre structure facilite aux développeurs frontend la communication avec nos serveurs.

FIGURE 4.2 – Documentation de API Prospektor

Pour sécuriser notre serveur, on a utilisé la normalisation [JWT](#).

[JWT](#) : est une normalisation permettant d'utiliser des jetons pour s'authentifier sur le Web en général. Il est robuste et peut contenir beaucoup d'informations. Comme tout autre jeton, [JWT](#) peut être utilisé pour transmettre l'identité d'utilisateurs authentifiés entre un fournisseur d'identité et un fournisseur de services. Il peut également contenir toutes les revendications de l'utilisateur, telles que les données d'autorisation. Le fournisseur de services n'a donc pas besoin d'entrer dans la base de données ou dans des systèmes externes pour vérifier les rôles et autorisations des utilisateurs pour chaque demande. Ces données sont extraites du jeton.

Choisir [JWT](#) pour sécuriser nos points de terminaison d'[API](#) est un excellent choix car il garantit un échange sans état de jetons entre le client et le serveur, est compact et sécurisé pour les URL. Avec [JWT](#), il est inutile de stocker les jetons d'accès dans une base de données (bien que vous puissiez toujours le faire et même en avoir besoin en fonction du cas d'utilisation) ou de nous soucier des sessions bloquées, cela rend la construction de redondance dans notre application d'entreprise plus rentable au moins aussi rentable.

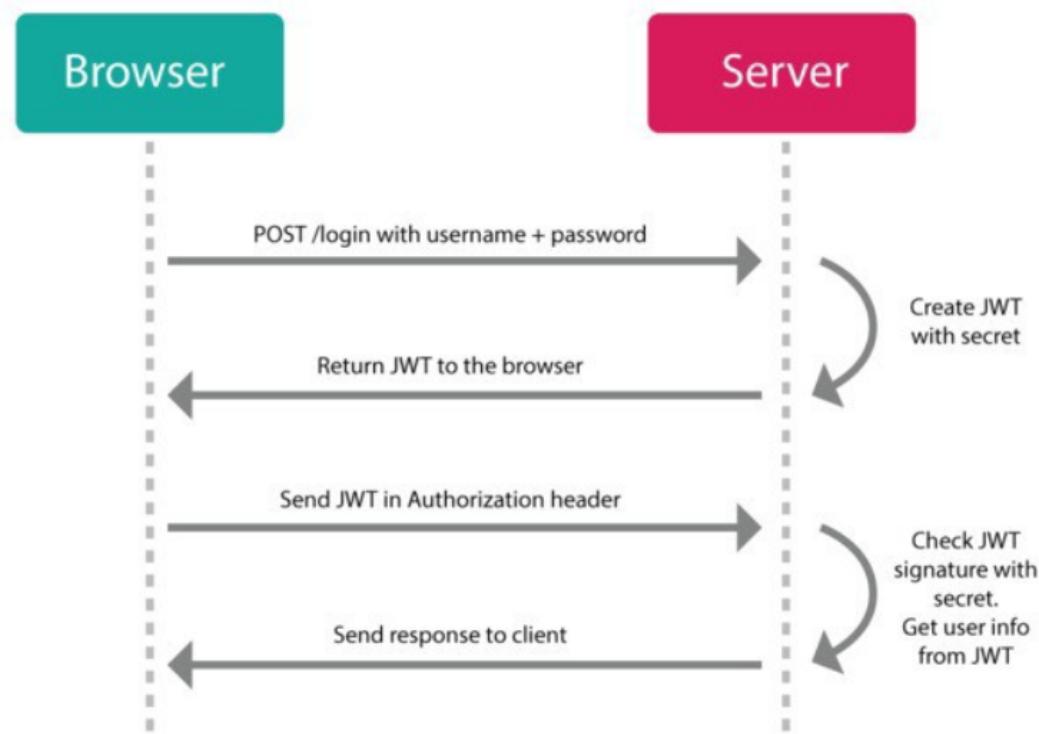


FIGURE 4.3 – Authentification avec JWT

On a pris Postman comme outil pour tester notre API et vérifier le bon fonctionnement de la partie backend.

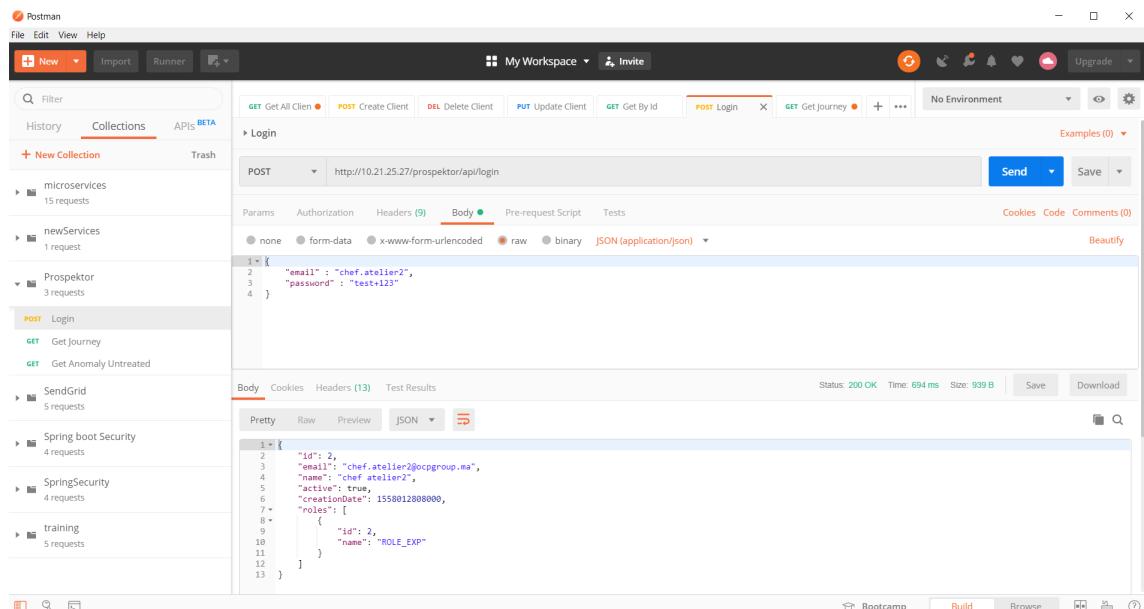


FIGURE 4.4 – Prospektor avec Postman

4.3.2 Couche FrontEnd

On a suivi **Material Design** dans toutes les interfaces de notre application (est un ensemble de règles de design proposées par **Google** et qui s'appliquent à l'interface graphique des logiciels et applications).

Cette couche décompose à deux parties :

4.3.2.1 Couche FrontOffice

Pour la partie frontOffice, on a une application mobile développé par android native, utilisant le langage JAVA pour une tablette spécifique utiliser dans les chantiers du groupe **OCP**.

pourquoi android native ?

- Meilleure rapidité, fiabilité et dotée d'une meilleure réactivité ainsi qu'une résolution supérieure ce qui assure une meilleure expérience utilisateur.
- Elle permet un accès plus facile à toutes les fonctionnalités du téléphone, de l'accéléromètre en passant par la caméra et même le micro.
- Les notifications push, uniquement disponibles sur les apps native. Ces notifications nous permettent d'alerter nos utilisateurs et d'attirer leur attention chaque fois que nous le souhaitons, que ce soit pour du nouveau contenu ou une offre promotionnelle.
- Ne requiert pas forcément internet pour fonctionner, ce qui est un réel avantage. Dans notre cas, il existe des zones très peu couvertes par le réseau internet, et permettre à nos utilisateurs d'accéder à l'app sans connexion web est un très gros point fort à ne pas négliger.

4.3.2.2 Couche BackOffice

- **ReactJS** : est essentiellement une bibliothèque JavaScript open-source qui est utilisée pour créer des interfaces utilisateur spécifiquement pour les applications à page unique. React nous permet également de créer des composants d'interface utilisateur réutilisables.
- **Redux** : Bibliothèque complémentaire à React qui permet de conserver facilement les données (State) et les événements (Actions) .Redux isole l'objet d'état des composants.
- **Redux-saga** : est une bibliothèque de middleware redux conçue pour simplifier la gestion des effets secondaires de votre application redux. Pour ce faire,

il exploite une fonctionnalité de l'ES6 appelée Generators, qui nous permet d'écrire un code asynchrone qui a l'air synchrone et qui est très facile à tester.
Pourquoi intégrer ReactJS dans Prospektor ?

1. Le contenu est référençable

Grâce à l'utilisation d'un serveur Node, le code va pouvoir être généré côté client ET côté serveur à la différence des autres frameworks JS traditionnels (Backbone.js, AngularJS, Ember.js, etc.) qui de manière native exécutent le code seulement côté client.

2. ReactJS est très rapide

ReactJS crée son propre DOM virtuel où sont rattachés nos composants. Cette approche nous donne énormément de flexibilité et des performances exceptionnelles, car ReactJS calcule quel changement dans le DOM a besoin d'être fait, et change juste LA PARTIE qui a besoin d'être mise à jour. De cette façon, ReactJS évite des opérations coûteuses dans le DOM.

3. Les composants sont le futur du développement web

ReactJS a pris le concept de Shadow DOM et du framework PolymerJS et l'a poussé à un niveau supérieur. React.js n'utilise pas Shadow DOM à la place il nous donne l'habileté de créer nos propres composants que nous pourrons réutiliser plus tard, combiner, et/ou inclure dans le cœur de notre contenu. Cette fonctionnalité à elle seule est un gage de productivité de par la facilité à définir et manipuler nos propres composants.

4. L'intelligibilité

ReactJS produit du code (propre) (simple à lire), sa lecture permet de déterminer immédiatement quelles sont les fonctionnalités de notre application. Ce qui est essentiel pour la maintenance et l'expansion de notre projet dans le temps.

5. Le Javascript plus simple à écrire

ReactJS utilise une syntaxe spéciale appelé JSX, qui permet de mixer l'HTML et le Javascript. Ce n'est pas obligatoire, nous pouvons toujours écrire notre app ReactJS en Javascript natif, mais nous suggérons très fortement d'essayer cette nouvelle syntaxe car elle nous permet d'écrire nos composants très facilement. Être capable de mettre une touche de HTML dans nos fonctions de rendu sans avoir à concaténer nos chaînes et après quelque temps cela devient très naturel.

4.3.3 Couche Base de données

- **MySQL** : est un système de gestion de base de données relationnelle open source basé sur le langage **SQL**. Leur utilité est faire stocker les données textuel de notre application.
- **MINIO** : est un serveur de stockage d'objets haute performance compatible avec les API Amazon S3. Pour faire stocker les attachements de l'application (images & audios).

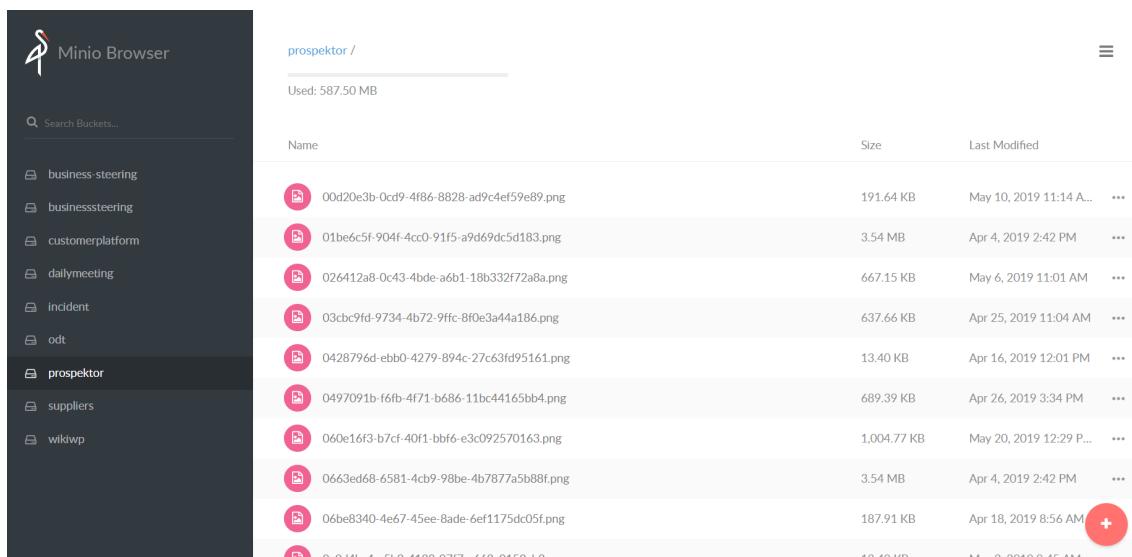


FIGURE 4.5 – Prospektor avec MinIo

4.3.4 Outils de DevOps

- **Docker** : est un outil conçu pour faciliter la création, le déploiement et l'exécution d'applications à l'aide de conteneurs. Les conteneurs permettent à un développeur de conditionner une application avec toutes les pièces dont il a besoin, telles que des bibliothèques et autres dépendances, et de l'expédier dans un package unique.

Quand on parle de conteneurisation, on le compare très souvent aux machines virtuelles. Jetons un coup d'œil à l'image suivante pour voir la différence principale :

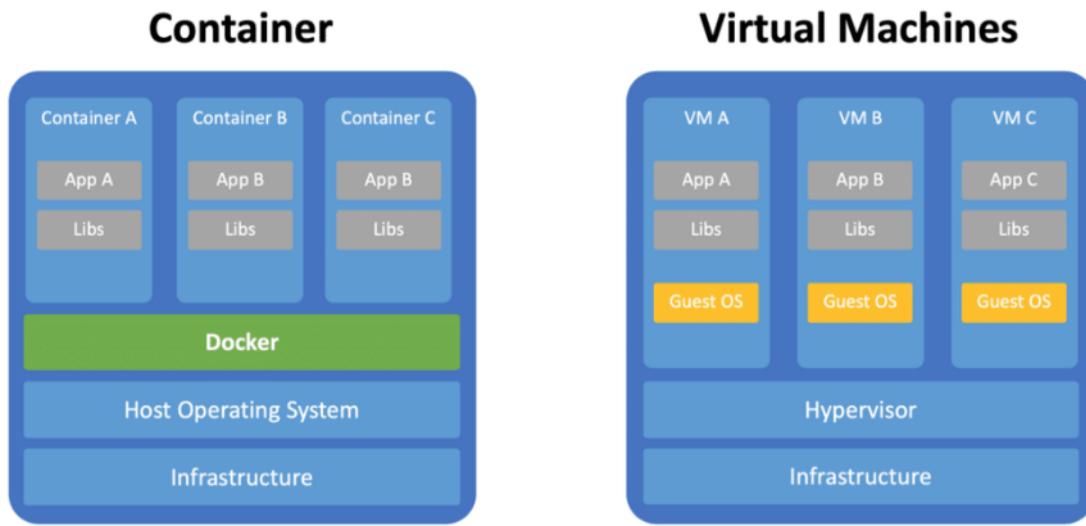


FIGURE 4.6 – Containers Vs Machines virtuelles

La plate-forme de conteneur Docker est toujours exécutée sur le système d’exploitation hôte. Les conteneurs contiennent les fichiers binaires, les bibliothèques et l’application elle-même. Les conteneurs ne contiennent pas de système d’exploitation invité garantissant leur légèreté.

En revanche, les machines virtuelles s’exécutent sur un hyperviseur (responsable de l’exécution des machines virtuelles) et incluent leur propre système d’exploitation invité. Cela augmente considérablement la taille des machines virtuelles, rend la configuration de machines virtuelles plus complexe et nécessite plus de ressources pour exécuter chaque machine virtuelle.

- **Kubernetes** : est un système d’orchestration de conteneur open-source permettant d’automatiser le déploiement, la mise à l’échelle et la gestion des applications.

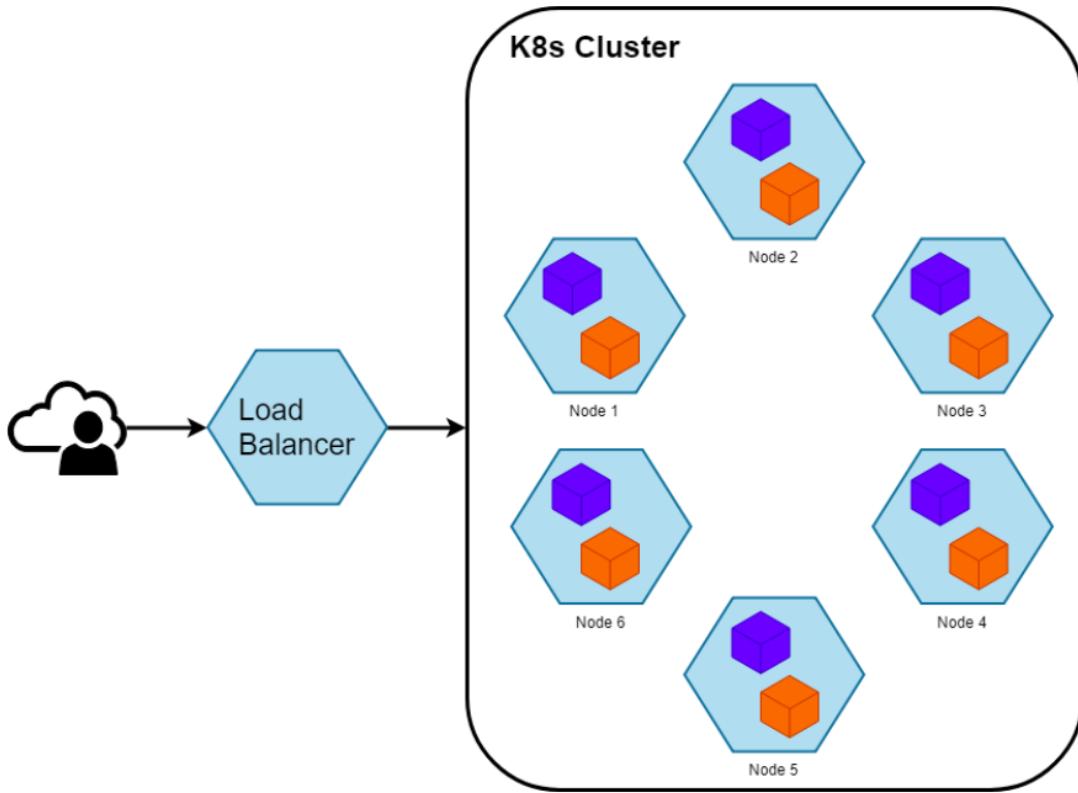


FIGURE 4.7 – Kubernetes

Dans notre figure, nous avons un utilisateur qui se connecte à un équilibrEUR (load balancer). L'équilibrEUR de charge redirige la demande vers un nœud situé à l'intérieur du cluster kubernetes. Le nœud traite ensuite la demande et renvoie une réponse.

Travailler avec Kubernetes (K8) comprend cinq étapes principales :

1. Développer une application.
2. Containerize notre application.
3. Créez un cluster Kubernetes.
4. Déployez notre conteneur sur le cluster.
5. Exposer et redimensionner le cluster.

4.4 Implémentation & tests

Nous avons réparti le travail en plusieurs itérations (Sprints). Le sprint est un bloc de temps (1 semaine) durant lequel un incrément du produit sera réalisé. Tous les sprints ont la même durée et ne chevauchent jamais. Tout au long de cette partie, on

va traiter le sprint 5 "Créer un chantier" comme modèle pour tous les autres sprints, puis on va donner des captures d'écran de la résultat courant du notre projet avec les tests de validation.

4.4.1 Sprint modèle : Créer un chantier

4.4.1.1 Description générale du sprint

Au cours de ce sprint nous sommes focalisés sur la création d'un chantier. Dans un premier temps, on va afficher une écran qui permet au utilisateur de remplir les éléments nécessaire pour créer un chantier. Cette sprint se base sur la partie frontOffice (concernant l'utilisateur de l'application mobile précisément le prospecteur).

4.4.1.2 Description détaille du sprint

Après l'authentification, le prospecteur passe à l'écran d'accueil. Puis, il peut suivre les démarches suivantes :

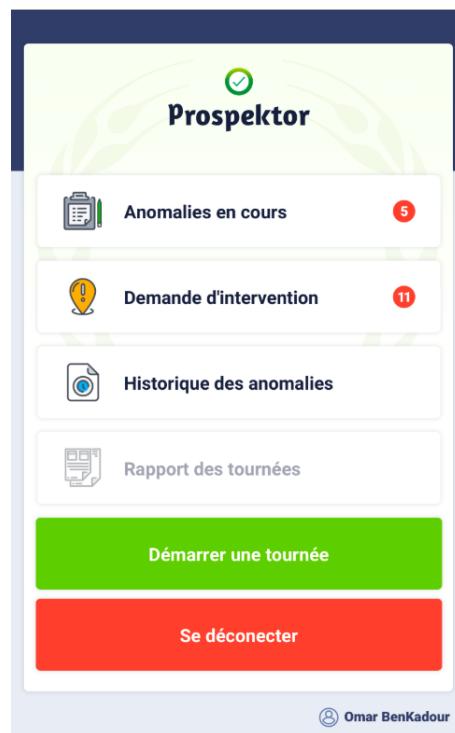


FIGURE 4.8 – Ecran d'accueil

- Après le prospecteur clique sur le button (Démarrer une tournée).
- la tablette envoie au serveur une requête pour générer un chantier vide et recevoir leur identifiant.

- Passant à l'autre écran qui contient les détails d'un chantier.
- Avant l'affichage de l'écran, la tablette demande au serveur toutes les informations sur les machines, zones, trenchées & sorties d'une mine spécifique.
- Après la réception de ces informations, l'écran sera afficher et le prospecteur peut remplir les champs de chantier d'une façon ordonnée.

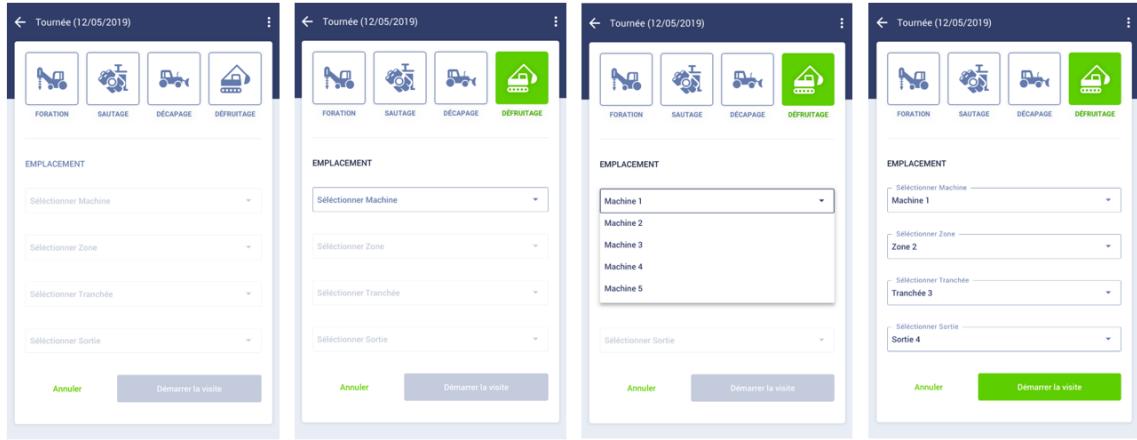


FIGURE 4.9 – Écran de chantier

- Enfin, une button pour la confirmation du chantier.
- une requête sera envoyé au serveur qui contient tout les informations sur ce chantier.

On a développer les dernières écran par android studio. respectant l'architecture **MVP** qu'on a déjà expliqué.

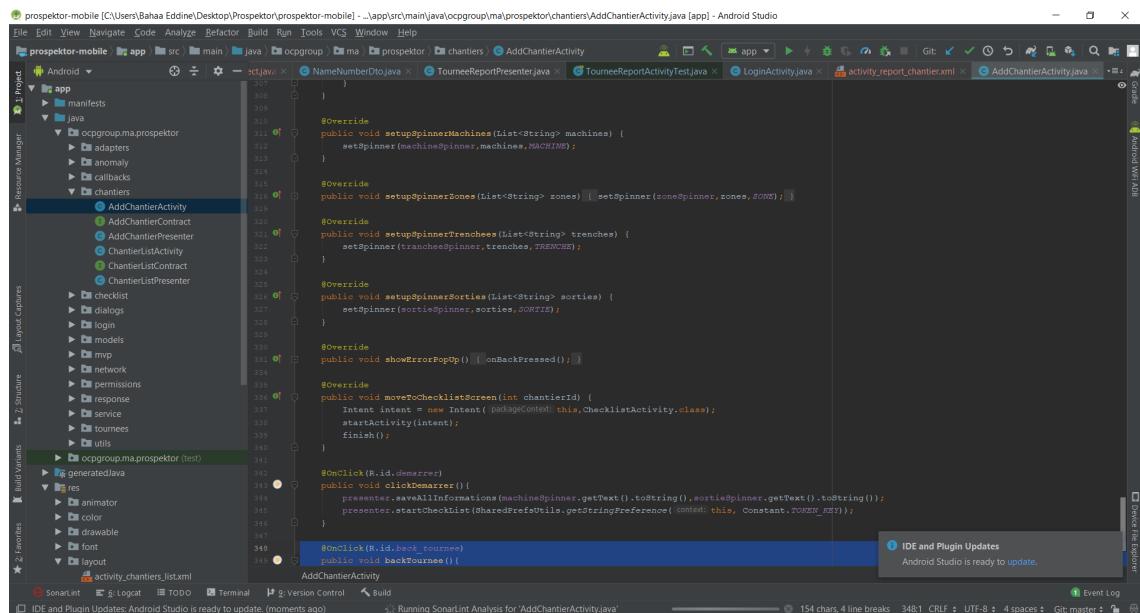


FIGURE 4.10 – Outil Android Studio

4.4.2 Captures d'écran du résultat courant

Pour la résultat courant de notre application. On va concentrer sur la partie frontoffice, c'est la partie importante dans notre projet qui suive les procédures suivantes :

4.4.2.1 Prospecteur WorkFlow

Lors d'authentification comme prospecteur, l'utilisateur peut suivre les écrans suivantes :

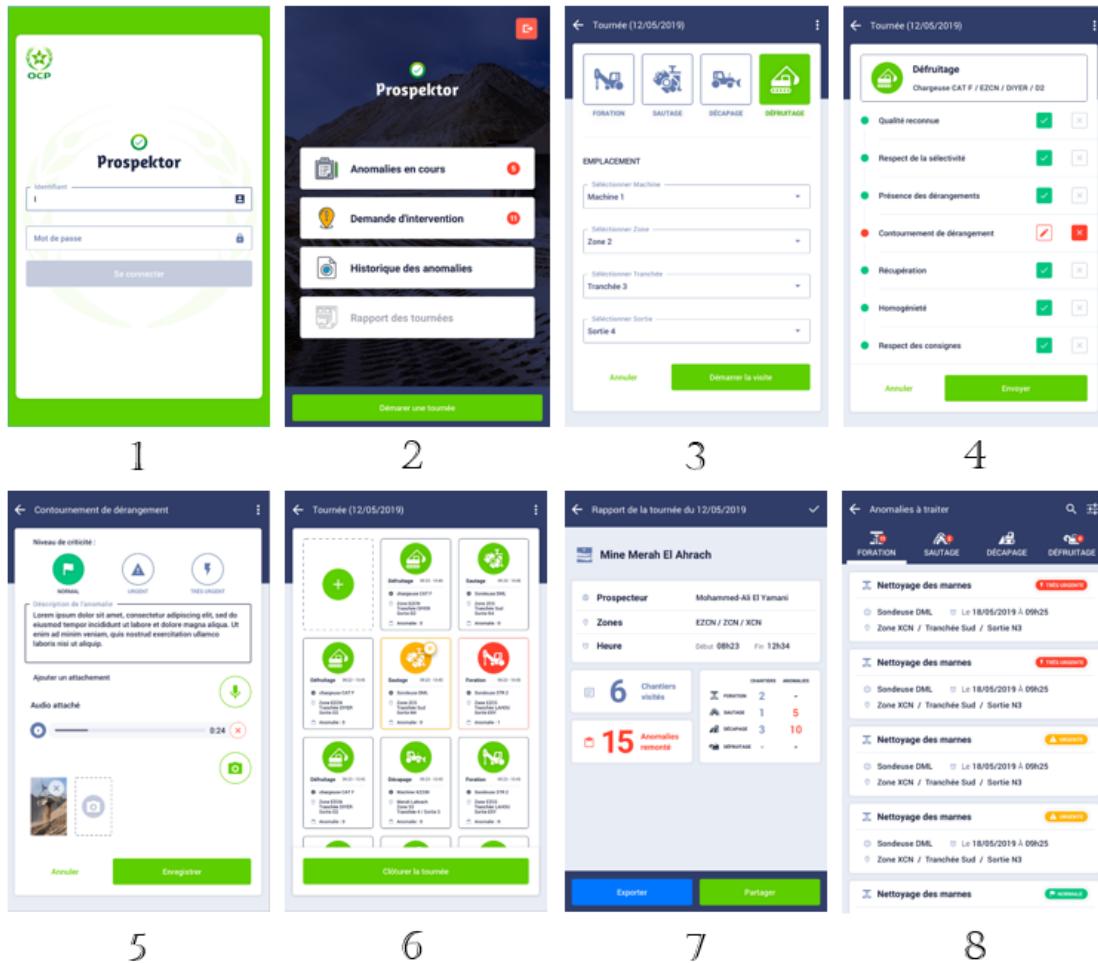


FIGURE 4.11 – Prospecteur WorkFlow

- 1. Authentification** : Insertion du nom d'utilisateur & mot de passe.
- 2. Page d'accueil** : Pouvoir de choisir l'un des options proposés ou de démarrer une nouvelle tournée.
- 3. Création d'un chantier** : Remplir les détails d'un nouveau chantier.

4. **Checklist d'un chantier** : Vérification des réglés d'un chantier par phase lors d'extraction des phosphates.
5. **Création d'une anomalie & Ajouter des attachements** : Créer une anomalie par leur criticité & ajouter des attachements (photo ,audio) pour plus clarifier le problème.
6. **liste des chantiers** : L'affichage détaillés de tous les chantiers par la tournée courante.
7. **Rapport d'une tournée** : Avoir le rapport de la tournée après leur clôture-
tion.
8. **liste des anomalies** : L'affichage détaillés de toutes les anomalies.

4.4.2.2 Exploitant WorkFlow

Lors d'authentification comme exploitant, l'utilisateur peut suivre les écrans suivantes :

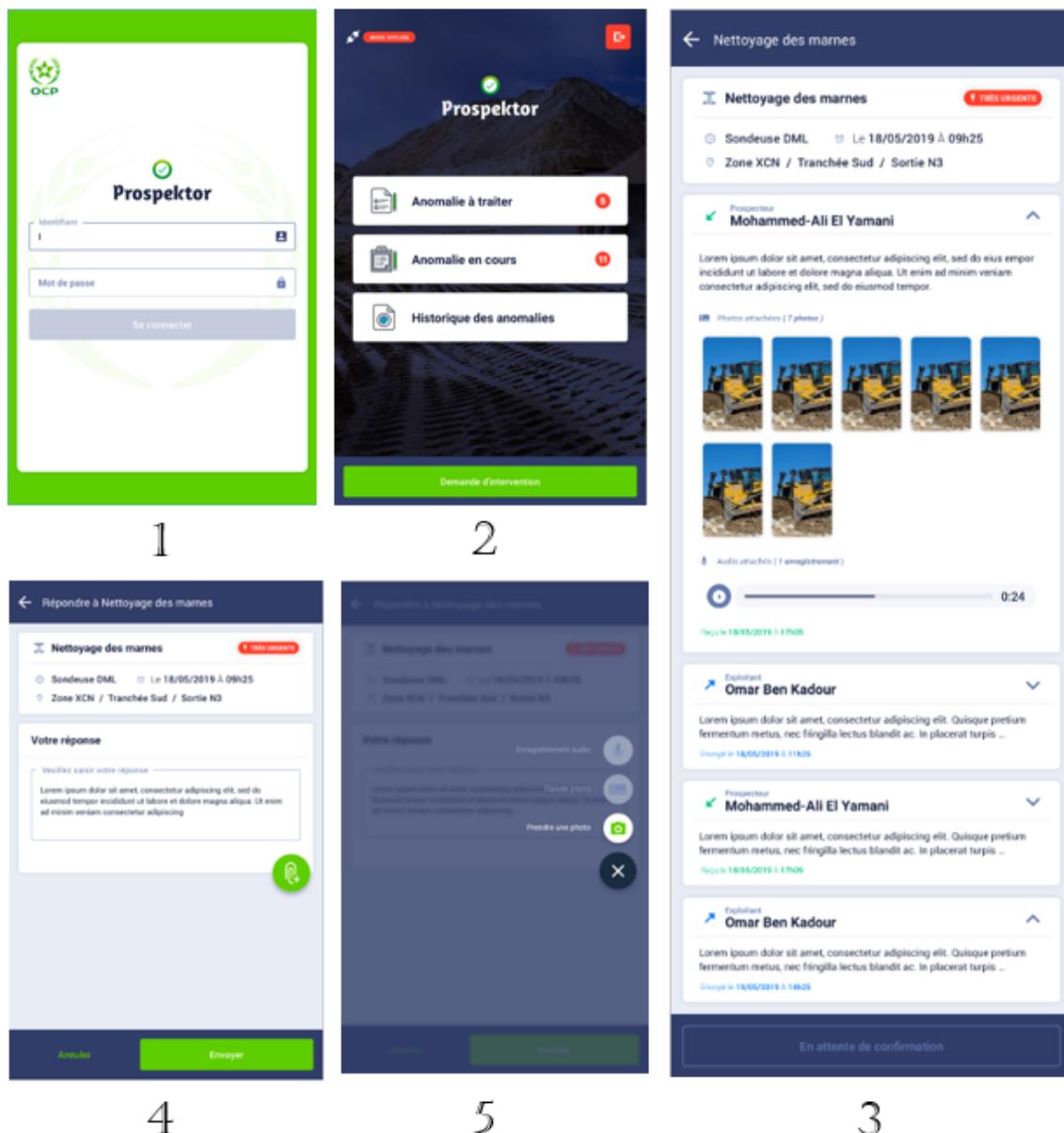


FIGURE 4.12 – Exploitant WorkFlow

- Authentification :** Insertion du nom d'utilisateur & mot de passe.
- Page d'accueil :** Pouvoir de choisir l'un des options proposés.
- liste des anomalies créer par les prospecteurs :** L'affichage détaillées de toutes les anomalies ainsi par leur créateur (prospecteur).
- Répondre à une anomalie & Ajouter des attachments :** Après l'examen d'une anomalie citée dans la liste, le prospecteur peut répondre sur cette anomalie ainsi ajouter des attachments (photo ,audio) pour plus clarifier leur intervention (solution ou bien clarification du problème).

4.4.3 Tests de validation

Les tests nous permettent de vérifier les fonctionnalités de notre système et avoir une bonne qualité de notre produit. On a utilisé quelques frameworks et bibliothèque pour tester notre application.

Biblio & framework	Description
	JUnit est un framework de tests unitaires pour langage de programmation Java.
	Mockito peut être utilisé avec JUnit. Mockito vous permet de créer et de configurer des objets fictifs.
	Robolectric est une framework de test unitaire qui permet de tester les applications Android sur la machine virtuelle sans émulateur ni périphérique.
	Jest est une bibliothèque pour tester le code JavaScript. Il s'agit d'un projet open source géré par Facebook et particulièrement adapté aux tests de code JavaScript.

TABLE 4.2 – Bibliothèques & FrameWorks de test

Au cours de la réalisation de la partie frontoffice, on a crée plus que 170 test unitaire pour le partie **MVP** (test unitaire est une procédure permettant de vérifier le bon fonctionnement d'une partie précise d'un logiciel ou d'une portion d'un programme) pour valider notre programme et respecter les métriques de test.

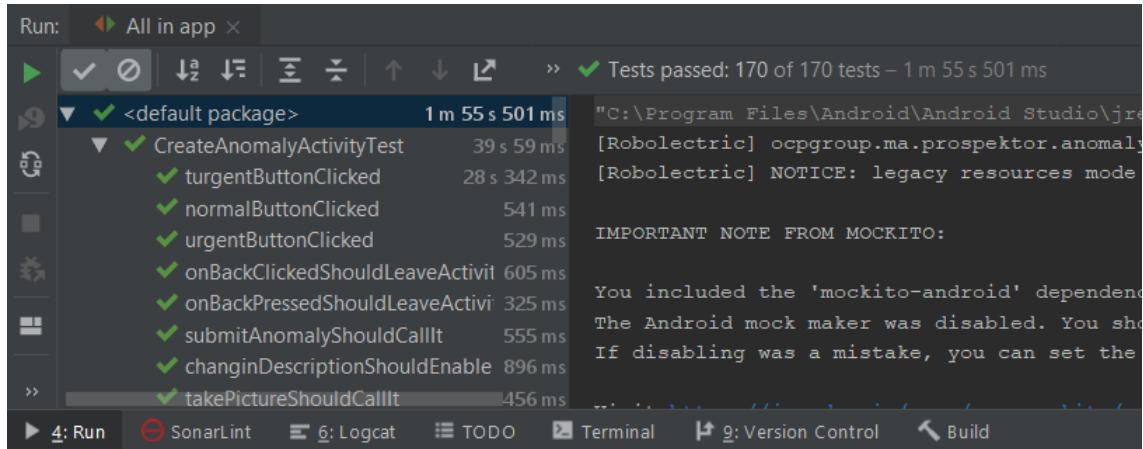


FIGURE 4.13 – Test unitaires de la partie frontoffice

Pour vérifier nos tests et mesurer la qualité du code source de façon continue de notre projet. On a utilisé les outils suivantes :

- **SonarQube** : est une plate-forme open-source développée par SonarSource pour l'inspection continue de la qualité du code afin d'effectuer des révisions automatiques avec analyse statique du code afin de détecter les bugs, les odeurs de code et les vulnérabilités de sécurité.

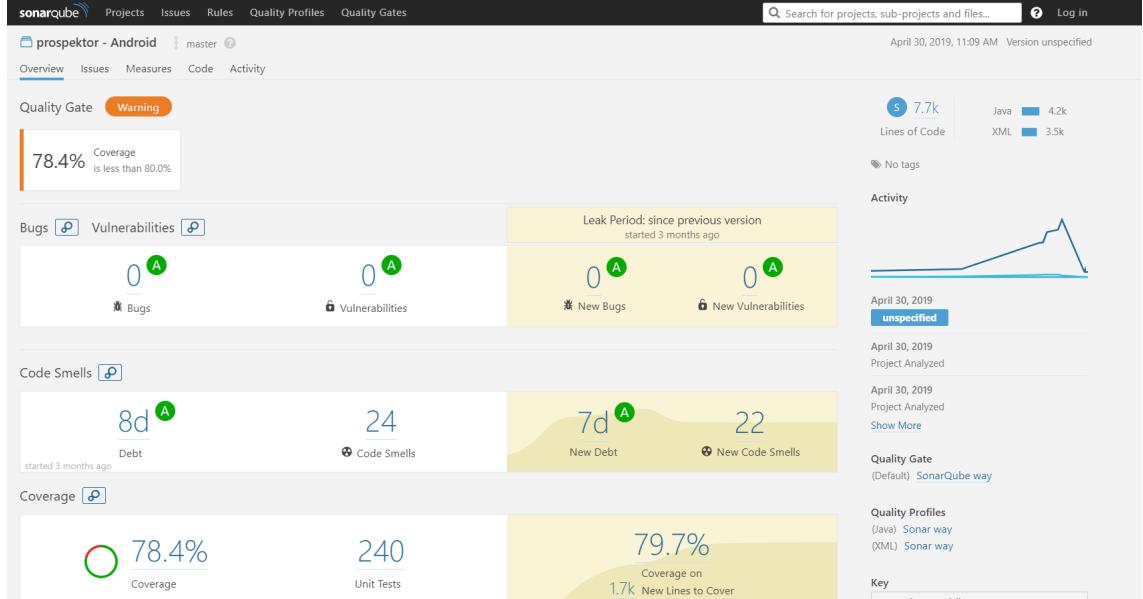


FIGURE 4.14 – SonarQube

La DF a déjà configurer sonarqube pour respecter les métriques suivantes :

Metric	Période de fuite	Opérateur	Attention	Erreur
Bugs	No	is greater than		0
Code Smells	No	is greater than	50	100
Coverage	No	is less than	80.0%	30.0%
Duplicated Lines on New Code (%)	Always	is greater than		3.0%
Maintainability Rating on New Code	Always	is worse than		A
Reliability Rating on New Code	Always	is worse than		A
Security Rating on New Code	Always	is worse than		A

TABLE 4.3 – Configuration SonarQube

Remarque : On donne la note (Rate) suivant les règles suivantes :

- $\leq 5\%$ la note est A
- entre 6 to 10% la note est B
- entre 11 to 20% la note est C
- entre 21 to 50% la note est D
- $> 50\%$ la note est E
- **EFK Stack** : permet de créer une pile puissante pour collecter, stocker et visualiser les données dans un emplacement centralisé. En effet, Fluentd, Elasticsearch et Kibana sont également connus sous le nom de EFK stack. Fluentd transfère les journaux des instances individuelles du cluster vers un système de journalisation centralisé, où ils sont combinés pour générer des rapports de niveau supérieur à l'aide d'ElasticSearch et de Kibana.

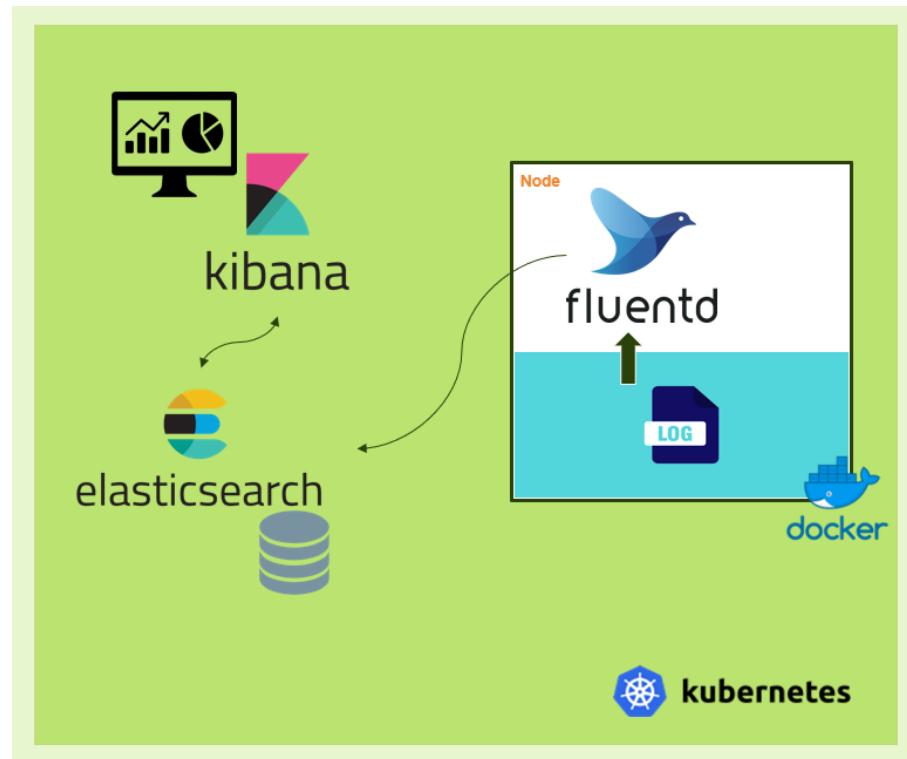


FIGURE 4.15 – EFK Stack

4.4.4 Déploiement de projet

L'entité DF utilise GitLab pour offrir un emplacement pour le stockage de code en ligne et le développement collaboratif de projets logiciels volumineux.

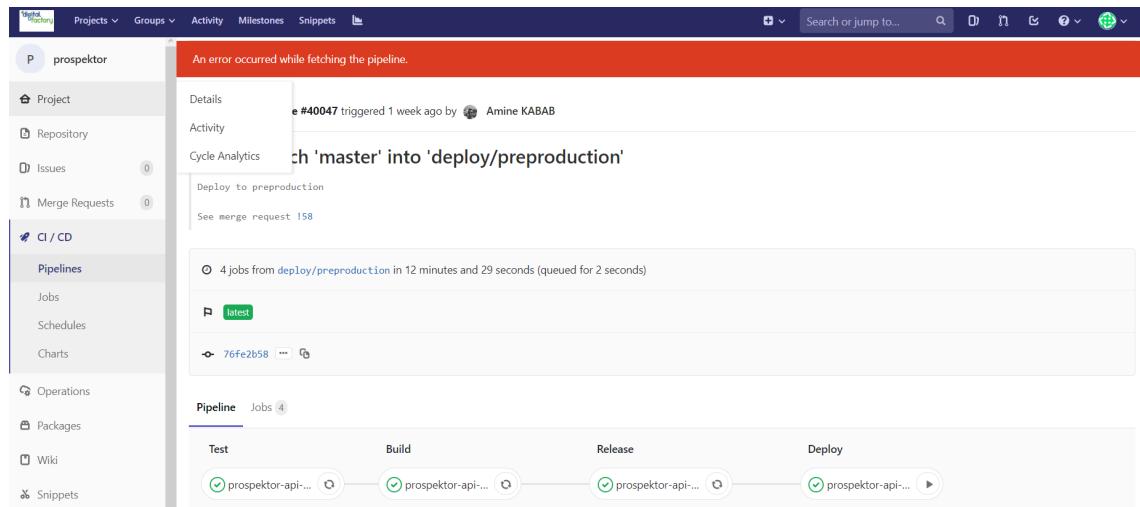


FIGURE 4.16 – GitLab

Le processus de déploiement que nous avons mis en place est le suivant :

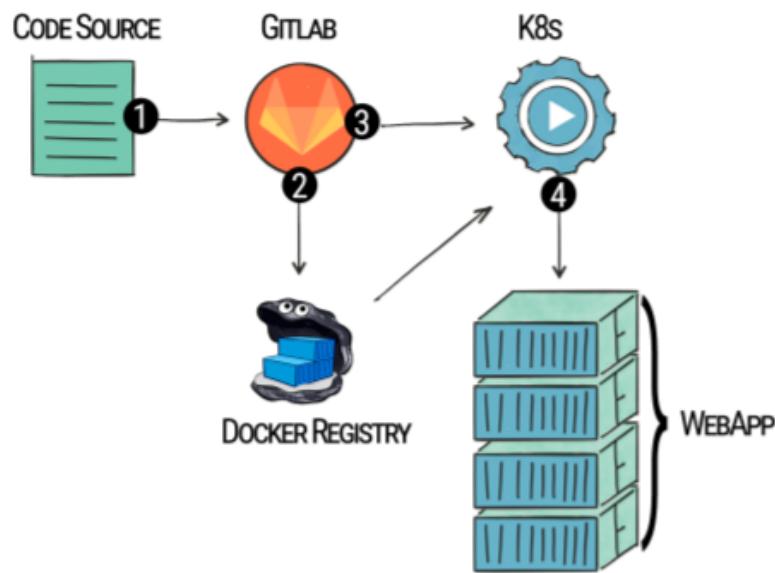


FIGURE 4.17 – Processus de déploiement

1. dépôt du code source après la validation des tests & le lancement normale de l'application dans gitlab.
2. création automatique d'une image Docker et stockage dans un registre,
3. création ou modification d'un déploiement d'application dans Kubernetes,
4. déploiement de l'application par Kubernetes qui crée les containers.

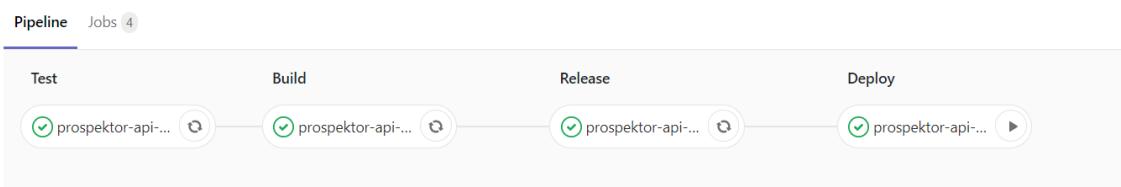


FIGURE 4.18 – Pipeline de Prospektor

Ce procédure est configurer dans un fichier YAML ".gitlabci.yml".

4.5 Conclusion

À la fin de ce chapitre, nous avons réussi à expliquer l'architecture techniques de notre system, l'état d'avancement de notre projet ainsi les outils et les bibliothèques pour tester notre code.

Conclusion générale

Ce mémoire présente le bilan du travail effectué durant la période de mon stage de Fin d'Etudes au sein de la Digital Factory de l'Office Chérifien des Phosphates. Le but de mon stage est de contribuer à la conception ainsi que le développement d'une solution moderne pour le gestionnement et le suivi complet des anomalies dans les mines du groupe [OCP](#).

La naissance de ce projet est due au besoin de la centralisation des données ainsi minimisation le temps de réponse pour résoudre les anomalies lors de l'extraction de phosphates. En effet, la plateforme Prospektor présente une résolution à cette problématique en se basant sur une vision architecturale qui répond aux besoins de modularité ainsi que la scalabilité.

Durant l'ensemble des activités menées dans le cadre de ce projet, nous avons passé par l'ensemble des phases d'un projet logiciel ainsi que nous avons utilisé plusieurs technologies notamment Spring Boot comme framework d'implémentation de [API Backend](#) et Android Native & ReactJs comme framework d'implémentation de la logique Frontend. Le projet s'est déroulé en adoptant une méthodologie agile basée essentiellement sur Scrum comme processus de développement afin de maximiser la livraison de valeur dans un minimum de temps.

Durant ce travail, on a pu réaliser plusieurs fonctionnalités, à savoir le suivi complet des anomalies dans les sites, leurs attachements et les commentaires des utilisateurs sur ces anomalies. Toutefois, il reste plusieurs fonctionnalités à réaliser, à savoir stocker les données localement en cas d'absence de connexion internet, envoyer des notifications par [SMS](#) ainsi les analytiques et le moteur de recherche qui doivent être livrées les mois qui suivent.

Bibliographie

- [1] Hansy : *Consultant technique indépendant, architecte de solution et développeur full-stack.* <https://medium.com/@hantsy>
- [2] Tarun Sharma. <https://medium.com/@tkssharma>
- [3] Spring Boot framework. <https://spring.io/projects/spring-boot/>
- [4] Android Developer. <https://developer.android.com/>
- [5] ReactJs. <https://reactjs.org/tutorial/tutorial.html/>
- [6] Redux. <https://redux.js.org/basics/basic-tutorial/>
- [7] Saga. <https://redux-saga.js.org/docs/introduction/BeginnerTutorial.html>
- [8] Stackoverflow. <https://stackoverflow.com/>
- [9] Tutorialspoint. <https://www.tutorialspoint.com/>
- [10] Github. <https://github.com/>
- [11] Wikipedia. <https://www.wikipedia.org/>

