République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Abderrahmane Mira – Bejaïa

Faculté des Sciences Exactes

Département d'Informatique

Master 2, Option Génie Logiciel

Projet de fin de cycle

Thème

Conception et Réalisation d'un module pour la gestion de Production dans le cadre de l'ERP de l'entreprise VMS

Cas d'étude : « VMS Industrie »

Réalisé par :

TAHAKOURT Maya

Mémoire encadré par :

Mr. OUZEGGANE Redouane

Années Universitaire 2020/2021

Remerciements

Mes remerciements s'adressent en premier lieu à mon encadreur Mr. Redouane OUZEGGANE pour le temps qu'il m'a consacré pour la réalisation de ce travail, malgré ses multiples occupations ; Je vous remercie d'avoir pris soin de me conseiller, m'orienter que ça soit dans la rédaction du mémoire ou la réalisation de l'application.

Je tiens à exprimer mes plus vifs remerciements à Mr. Ghiles HITACHI, chef de projets informatiques chez VMS Industrie et à toute son équipe qui m'a accueilli, intégrée et formée tout au long de cette expérience professionnelle avec beaucoup de patience et de pédagogie, j'espère que vous trouverez ici le témoignage de ma gratitude.

Mes remerciements vont aussi à tous les membres de jury qui nous ont fait l'honneur d'accepter d'examiner ce travail et de l'enrichir.

Pour finir, je reste reconnaissante envers toutes les personnes qui m'ont tendu la main, qui ont été là pour moi et qui ont contribué de près ou de loin à la réalisation de ce travail.

Dédicaces

Je dédie ce travail :

A mes chers parents qui m'ont beaucoup encouragée et cela du début à la fin et m'ont permis d'atteindre mes objectifs

A mes sœurs pour leur amour et leur soutien

A toute ma famille qui a toujours été là pour moi

Table des Matieres

Liste des Figures

Liste des Tableaux

Liste des Abreviations

AJAX	Asynchronous JavaScript And XML
API	Application Programming Interface
BD	Base de Données
CRUD	Create Read Update Delete
CSS	Cascade Style Sheet
DAO	Data Access Object
DI	Demande d'intervention
DMR	Direction Maintenance et réalisation « DMR »
DOM	Document Object Model
EJB	Entreprise Java Bean
ERDP	l'entreprise de raffinage des produits pétroliers
GPL	Gaz de Pétrole Liquéfié
HTML	Hyper Text Markup Languag
HTTP	Hyper Text Transfer Protocol
IDE	Integrated Development Environment
IHM	Interface Homme Machine
ING	Service information de gestion
JEE	Java Edition Entreprise
JPA	Java Persistence Annotation
JS	JavaScript
JSP	Java Server Pages
MVC	Modele Vue Controleur
MySQL	My Structured Query Language
OT	Ordres de travaux
SGBD	Système de Gestion de Base de Données
UML	Unifed Modeling Language
XAMPPX	(across) Apache MariaDB Perl Php

Introduction Generale

Introduction Generale

Dans un contexte très changeant associé à l'évolution technologique et à la multiplication du volume des données, les entreprises d'aujourd'hui petites ou grandes sont confrontées à plusieurs problèmes notamment : la compétitivité du marché, l'exigence des clients, l'augmentation de la demande de production, etc. Dans un tel environnement, la compétitivité des entreprises dépend de leur capacité à s'adapter et à innover, que ça soit dans leur structure organisationnelle ou leur mode de production et d'échange avec les clients. Dans leur recherche de compétitivité elles rencontrent des difficultés du fait que les informations sont limitées d'un service à un autre, et peinent à récolter ces informations en temps réel. Ceci vient du fait que la plupart de ces organisations fonctionnent en sous-systèmes, elles sont alors en quelque sorte divisée en sous parties indépendantes chacune avec ses propres données qui ne sont pas accessibles par les autres. Pour contrer ce problème l'ERP représente un enjeu qui permet d'intégrer tous les processus de gestion en un système d'information unique et cohérent.

La vacation des ERPs (*Entreprise Ressource Planning*) ou PGI (*Progiciel de Gestion Intégré*) dans une entreprise est d'homogénéiser et partager le même système d'information en facilitant la communication en interne et en externe afin de couvrir un large périmètre d'achats et de ventes, d'éviter les redondances d'informations qui représente un vrai problème au sein des organisation du fait qu'elles génèrent des incohérences dans le système d'informations, d'avoir les informations du système en temps réel, et de minimiser les couts grâce à la synchronisation des traitements, etc.

Pour l'entreprise VMS compagnie, la gestion de l'ensemble de ses activités se fait manuellement ou par le biais de quelques applications informatiques. L'entreprise souhaite gérer la totalité de ses activités autour d'un même système d'information à l'aide d'un ERP.

Pour cela et dans le cadre de notre mémoire de fin d'étude, nous nous sommes portés sur le développement d'une solution intégrée pour l'entreprise VMS, notre projet consiste à concevoir et développer un module de gestion de production pour permettre une gestion plus efficace du processus de production de l'entreprise, et en parallèle afin de mener à bien notre travail, nous avons suivi un stage au sein de l'entreprise VMS dont l'activité dans son ensemble est de produire des motocycles.

Notre plan est structuré en quatre chapitres :

- Le premier chapitre, comportera la description des principales technologies avec lesquelles nous travailleront.
- Le deuxième chapitre, présentera le contexte de l'étude et la démarche que nous avons menée.
- Le troisième chapitre, englobera la capture des besoins, l'analyse et la conception détaillée des besoins de l'entreprise.
- Le quatrième chapitre, qui représente le dernier chapitre, nous y présenterons la phase réalisation et intégration de la solution, et décrierons les choix technologiques utilisés pour la mise en œuvre.

Enfin, nous clôturerons ce mémoire avec une conclusion générale et des perspectives futures.

Chapitre I

technologies web

Chapitre I

Technologies web

I.1- Introduction

De nos jours, les développeurs Full-stack métrisant les deux blocs du développement web à savoir le coté Front-end orienté client et le coté Back-end orienté serveur sont demandés dans quasiment toutes les entreprises et cela dû au fait qu'ils sont des développeurs multifonctions et peuvent s'adapter et effectuer n'importe quelle tâche demandée.

Dans notre projet de développement, nous allons jouer le rôle de développeur full-stack et réaliser une application web complète. Pour cela, nous avons décidé d'utiliser deux technologies qui sont actuellement très célèbres. Pour ce qui est du coté client, nous adopterons *React JS* qui est l'une des bibliothèques les plus utilisées. En ce qui concerne le coté serveur cette fois-ci, le célèbre framework Spring sera utilisé pour réaliser l'api qui se chargera d'envoyer les données à l'interface.

Dans ce chapitre, nous allons passer en revue dans un premier temps certains concepts React afin de bien comprendre le fonctionnement de cette bibliothèque qui a tellement fait parler d'elle, et nous enchainerons avec la présentation du framework Java Spring.

I.2- React JS

React de Facebook a changé notre façon de penser les applications Web et le développement d'interface utilisateur [2]. Grace à sa flexibilité, et sa performance qu'il doit notamment à sa fonctionnalité connue sous le nom de DOM virtuel, de plus en plus de célèbres entreprises adoptent cette fameuse bibliothèque dans le développement de leurs interfaces et de nos jours React ne cesse de gagner du terrain face à ses redoutable concurrents, on site Vue Js ou Angular.

I.2.1- Definition

React est une bibliothèque JavaScript qui vous oblige à penser en termes de composants. Ce modèle de pensée s'adapte bien aux interfaces utilisateur. [4]

Développé en 2013 par Facebook, React facilite le développement d'applications coté front-end et même coté back-end, en effet react est isomorphisme et peut être utilisé du coté server en moteur de template avec NodeJS ou Go. De ce fait React apporte une approche très intéressante pour le développement d'applications. D'ailleurs plusieurs applications ont désormais intégré cette bibliothèque à leurs projets, on retrouve notamment : Instagram, Gmail, Youtube ou encore Netflix.

Avec la variante React native, cette bibliothèque permet également le développement d'applications mobiles native sous IOS ou encore Android. Et donc d'avoir une seule application sous différents périphériques.

I.2.2- React et le DOM

La manipulation du DOM des applications web s'avère être primordiale, et cela du fait que nos applications web ne sont pas statiques, elles ont un état représenté par l'interface utilisateur (UI) qu'un navigateur Web rend, et cet état peut être changé lorsqu'un événement se produit, on parle ici de deux types d'événements qui sont sois un :

- Evénement utilisateur : lorsqu'un utilisateur tape, clique, fait défiler, redimensionne, etc.
- Evénements serveur : lorsqu'une application reçoit des données ou une erreur d'un serveur, entre autres

Lorsqu'un de ces événements précédemment cités se produit, nous mettons à jour les données dont dépend notre application, et ces données représentent un état de notre modèle de données. À son tour, lorsqu'un état de notre modèle de données change, nous pourrions vouloir refléter ce changement en mettant à jour un état de notre interface utilisateur. Nous cherchons en quelques sortes un moyen de synchroniser deux états différents : l'état de l'interface utilisateur et l'état du modèle de données.

React propose une solution appelée DOM virtuel. Le DOM virtuel est une représentation rapide en mémoire du DOM réel, et c'est une abstraction qui nous permet de traiter JavaScript et DOM comme s'ils étaient réactifs. A chaque fois que l'état du modèle de données change, le DOM virtuel et React redonnent une interface utilisateur à une représentation DOM virtuelle.

React calcule ensuite la différence entre les deux représentations DOM virtuelles : la représentation DOM virtuelle précédente qui a été calculée avant la modification des données et la représentation DOM virtuelle actuelle qui a été calculée après la modification des données. Cette différence entre les deux représentations du DOM virtuel est ce qui doit en fait être changé dans le vrai DOM.

React ne met à jour que ce qui doit être mis à jour dans le vrai DOM.

Cette approche donne beaucoup de flexibilité et de performances exceptionnelles, car React calcul quel changement dans le DOM a besoin d'être fait et change que la partie qui a besoin d'être mise à jour. De cette façon React évite des opérations couteuses dans le DOM. [4]

I.2.2- Single Page Application(SPA)

Une *single-page application* (SPA) est une application qui charge une unique page HTML et toutes les ressources nécessaires (telles que du JavaScript et des CSS) requises pour le fonctionnement de l'application. Aucune interaction sur la page ou les pages ultérieures ne nécessitera un nouvel aller-retour avec le serveur, ce qui signifie que la page n'a pas besoin d'être rafraîchie. [6]

React peut être utilisé pour réaliser des application SPA, mais peut également être utilisé pour modifier des parties de pages de sites déjà existantes dans le but d'améliorer ces parties et ceci sans à avoir à réécrire le code existant.

I.2.2- Node JS

Node.js est une plate-forme qui nous permet d'écrire des applications côté serveur avec un langage côté client que nous connaissons tous : JavaScript. [4]

Le fait que nodeJS utilise un modèle d'Entrée/Sorties non bloquant contrôlé par les événements est un gros avantage pour créer des applications en temps réel et qui recueil beaucoup de données, cela veut dire qu'on pourra manipuler directement les flux entrant dès leur arrivé.

NodeJs est très riche, il comprend une panoplie de modules. Un module étant une application nodeJS, il peut être réutilisé dans différents projets selon notre besoin. Actuellement on dénote plus de 120 000 modules que contient NodeJS, et tous sont manipulés par un gestionnaire de modules appelé « npm » qui vient avec l'installation de nodeJS.

On dit du contenu de React qu'il est référençable, ce qui veut dire qu'il peut être utilisé avec node.js et générer du code du coté client comme du coté serveur.

Cette caractéristique lui permet de se démarquer des autres bibliothèques ou frameworks.

I.2.2- Create React App

Create React App est un environnement qui permet de créer une nouvelle application web en React en deux trois opérations, et facilite ainsi aux débutants ou même aux plus expérimentés de démarrer un projet React en un temps record. [7]

Après avoir installé Node (>= 8.10) et de npm (>= 5.6) sur l'ordinateur on lance sur le terminal les commandes suivantes :

```
npx create-react-app mon-app
```

Cette commande crée un dossier « mon-app » contenant :

- Trois dossiers « node_modules », « src » et « public »
- Deux fichiers json « package.json » et « package-lock.json »

```
cd mon-app
npm start
```

Ces commandes nous permettent d'entrer dans le dossier « mon-app » et lancer notre application React dans le navigateur.

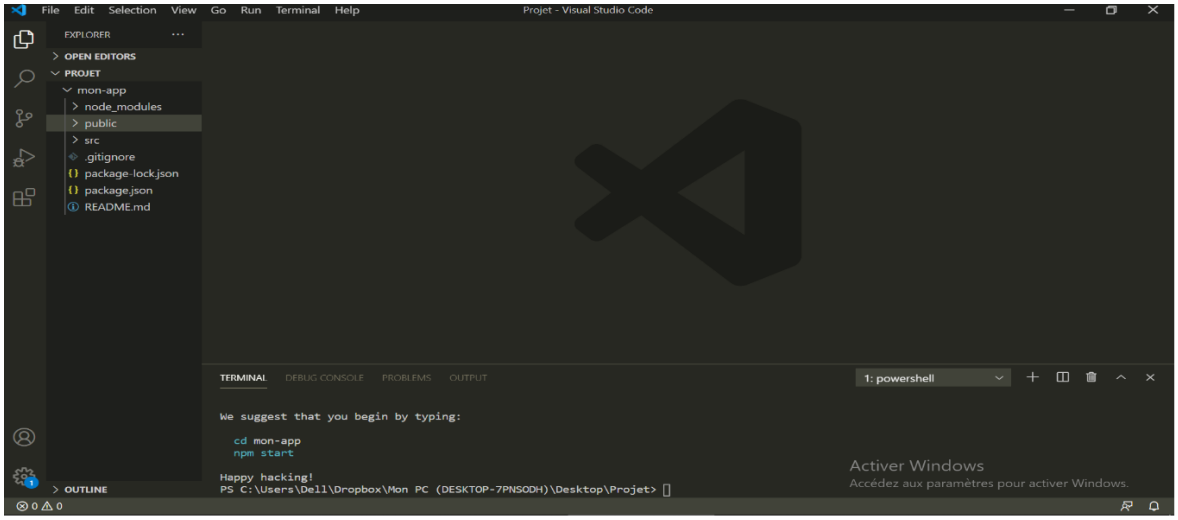


Figure 1 : Nouvelle application créée avec « Create React App »

I.2.2- Composants React

React est dit orienté composants, cette caractéristique qu'il a popularisée depuis 2013 nous permet de réaliser des applications avec un code source maintenable et réutilisable, et accélérer le cycle de développement. [8]

Les composants React sont de petits morceaux de code réutilisables qui renvoient un élément à afficher dans la page, ces composants peuvent être sois des éléments retournés par une fonction javaScript ou bien retourné par une fonction d'une classe ES6, ces fonctions peuvent prendre en entrée ce que l'on appelle des *Props* et renvoyer par la suite le résultat voulus.

I.2.2- React et JSX

Le JSX est une extension syntaxique de javaScript, souvent confondu avec du html ou du XML, ce langage décrit l'interface utilisateur en produisant des composants React, et en les retranscrivant dans le DOM.

Du fait que React ne fonctionne qu'avec du javascript, Facebook a créé le JSX qui ressemble beaucoup au langage de balisage html afin de faciliter la création des vue aux utilisateurs.

Grace au JSX, React nous permet de manipuler les DOM virtuel React (Grappes) d'une manière facile et pratique, d'autant plus JSX est considéré par la plupart des gens comme étant facile, intuitif et très lisible, cependant il n'est pas obligatoire d'utiliser le JSX dans React, l'utilisateur a le libre choix de coder ou pas avec ce langage.

I.2- Framework Spring

Spring est le framework Java le plus populaire pour la création d'applications pour entreprises. Le framework Spring fournit un écosystème de projet riche pour répondre aux besoins des applications modernes, telles que la sécurité, l'accès simplifié au stockage de données relationnelles et NoSQL, le traitement par lots, l'intégration avec les sites de réseaux sociaux et le traitement massif de flux de données. Étant donné que Spring est un framework très flexible et personnalisable, il existe généralement plusieurs façons de configurer les applications.

I.2.1- Composants du framework Spring

L'écosysteme Spring est divisé en plusieurs compartiments riches les uns comme les autres ce qui constitue la force de ce framework, dans ce qui suit nous allons énumérer quelques-uns de ces composants :

- Spring Core

  Cette composante est le fondement de l'écosystème Spring. Il contient le "core container" (qui permet l'injection des dépendances), mais il contient également

Spring MVC qui permet faire du web et du Data Access qui fournit les éléments de base pour communiquer avec la base de données.

- Spring Data

Ce composant permet la communication avec plusieurs types de bases de données. Il communique avec la base de données uniquement en implémentant des interfaces qui ont une certaine annotation (@Repository).

- Spring Security

Dans tout développement informatique, la sécurité est un point primordial et incontestable pour bon fonctionnement du logiciel et la protection des données. C'est pourquoi ce composant est l'un des composants les plus critiques de Spring Framework, bien qu'il soit également l'un des plus complexes. Il vous permet de gérer l'authentification, l'autorisation et la sécurité des API.

- Spring Cloud

L'architecture microservices est une architecture très convoitisée actuellement. Nombreux frameworks fournissent ce type de service, et Spring framwork aussi et il est procuré par Spring Cloud

- Spring Boot

Ce compartiment est très particulier car il joue le rôle de jonction entre les différents autres compartiments et permet de les implémenter facilement. Ce composant est très puissant car il supervise :

- L'autoconfiguration automatique de Spring
- Les starters de dépendances
- Les endpoints Actuator

Ci-dessous nous allons illustrer les composants précédemment cités de l'écosystème Spring.

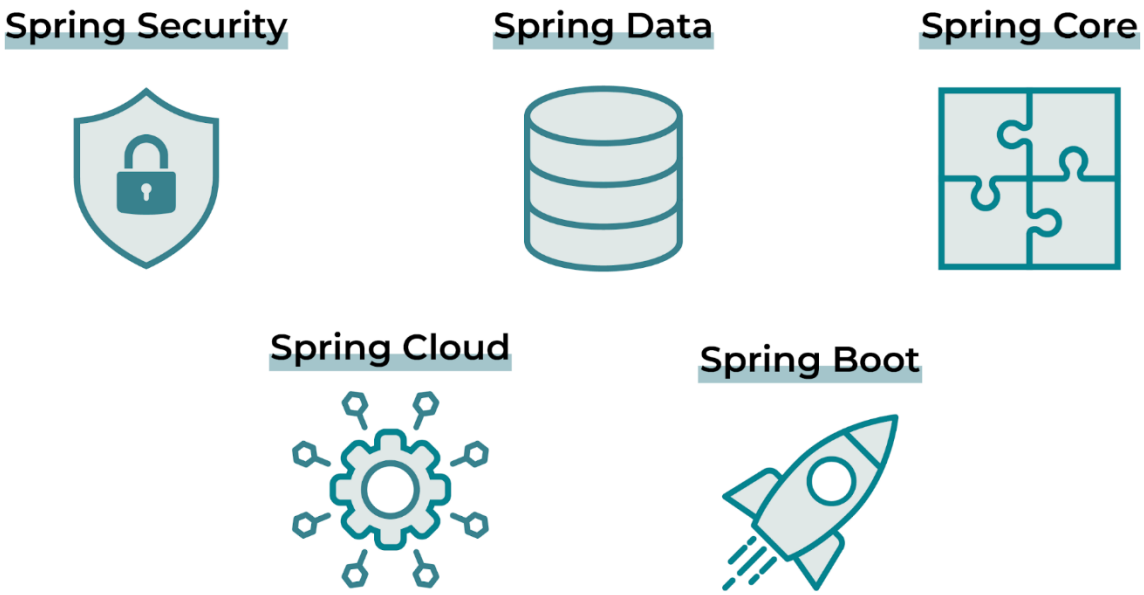


Figure 2 : Composants de Spring Framework

I.2.2- Spring Boot

Spring Boot est un framework avisé suivant l'approche "Convention Over Configuration", qui permet de créer rapidement et facilement des applications basées sur Spring. L'objectif principal de Spring Boot est de créer rapidement des applications basées sur Spring sans obliger les développeurs à écrire la même configuration passe-partout encore et encore.

I.2.2.1- Pourquoi utiliser Spring Boot ?

Spring boot nous offre plusieurs avantages pour une meilleure expérience du framework Spring.

Il Facilite à l'utilisateur le développement d'applications Spring et réduit de manière considérable le temps de programmation, il nous évite alors de réécrire le même code à chaque fois et de réinventer la roue. Ceci permet de laisser plus de temps à la créativité et d'améliorer la productivité.

Spring Boot nous évite les configurations complexes XML de Spring et permet de démarrer directement son projet sans perdre de temps à trouver la configuration adéquate.

Il nous procure une panoplie de starters qui assemble les meilleures combinaisons de dépendances déjà prêtes à l'utilisation. Et permet alors de commencer son projet de la manière la plus facile.

La gestion des propriétés au sein de Spring Boot se fait en toute simplicité. Le contenue du fichier « application. propriété » qui est un élément clé pour la gestion de propriété du projet doit être exécuter par certaine classes seulement, grâce à Spring boot, on est pas dans l'obligation de spécifier les classes en question une par une.

I.2.2.2- Auto configuration

Spring boot configure automatiquement l'application selon les dépendances ajoutées au projet en ajoutant l'annotation **@EnableAutoConfiguration** ou **@SpringBootApplication** dans la class « main », et cela permet de gagner du temps en se focalisant seulement sur le code métier, néanmoins, la configuration par défaut peut être modifier selon les besoins.

Le point d'entrée de l'application est la classe contenant l'annotation **@SpringBootApplication** et la méthode main.

I.2.2.3- Starter

La gestion des dépendances s'avère être une tâche difficile dans la réalisation de projets et surtout ceux de grande envergure, problème auquel Spring boot apporte une solution en fournissant un ensemble de dépendances Maven qu'on appelle « Starter » et qui facilite la tâche aux développeurs. Ces derniers peuvent alors ajouter au fur et à mesure les dépendances dont ils ont besoin.

Un starter est en réalité un fichier XML appelé « pom.xml » ou un « gadle build file » qui définit et gère un ensemble de dépendances avec leurs versions. Par exemple, le starter « data-jpa » inclut les dépendances « spring-data-jpa », « JSR-303 » et « hibernate ». Il est notamment possible aux développeurs de créer leur propre starter.

I.2.2.4- Annotations

Une annotation est une sorte d'étiquette qu'on ajoute à une classe, un attribut ou une méthode Java afin de leur attribuer un rôle ou un comportement. Elle est de la forme « @ + nom de l'annotation » et se situe au-dessus de l'élément auquel on le lui a attribué. Par exemple, @Autowired permet d'injecter des Beans pour les utiliser dans la classe, ou bien l'annotation @Service qui elle prévient Spring Boot que la classe en question joue le rôle de la couche Service etc...

I.2.2.5- Création de projet avec Spring Boot

Après avoir fait le tour de Spring Boot et avoir vu ce qu'il peut nous apporter et ce dont il est capable, nous passons à la création du projet en question.

Voici ci-dessous le plan d'action pour tout projet Spring Boot.

- Générer la structure minimale :

La création de projet avec Spring Boot peut se faire de deux manières, la première se fait en utilisant « Spring initializr » et la deuxième avec « Spring Tool Suit »

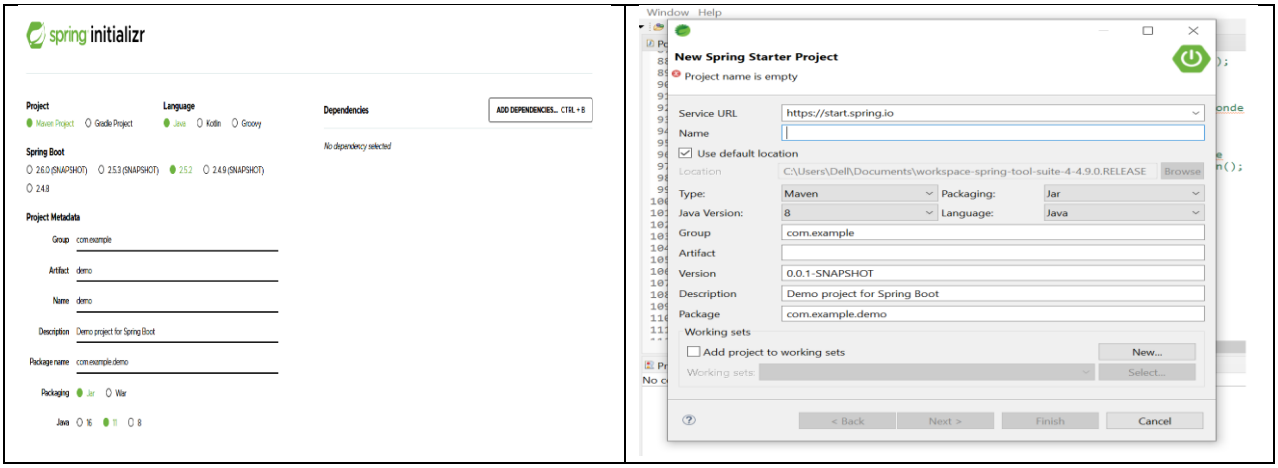


Figure 3 : Création de projet avec Spring Boot

Dans le cas où l'on ajoute aucune dépendance, Spring Boot ajoute automatiquement le starter « **spring-boot-starter** » qui contient toutes les fonctionnalités de base

- Structurer et configurer le projet

Apres la création du projet, on se retrouve avec une structure de projet minimale qui comprends ce qui suit :

- Les ressources :
  - src/main/java : regroupant les packages et les classes java
  - src/main/resources : contient les fichiers ressource (fichier de propriété ou templates html, par exemple)
  - src/test/java : contient des classes java de tests.

- Les fichiers en relation Maven, on site : « pom.xml » « mvnw » et « mvnw.cmd »
- Les dépendances : on retrouve ici les dépendances Maven et la JRE.

En plus de ces répertoires et packages, on retrouve également le fichier des propriétés du projet qu'on appelle « application.properties » et deux autres classes java, l'une d'entre elles étant la classe principale contenant le main et l'annotation @SpringBootApplication et l'autre se chargeant de vérifier le lancement du contexte Spring.

Ci-dessous une image de la structure minimale d'un projet créé avec Spring Tool Suit.

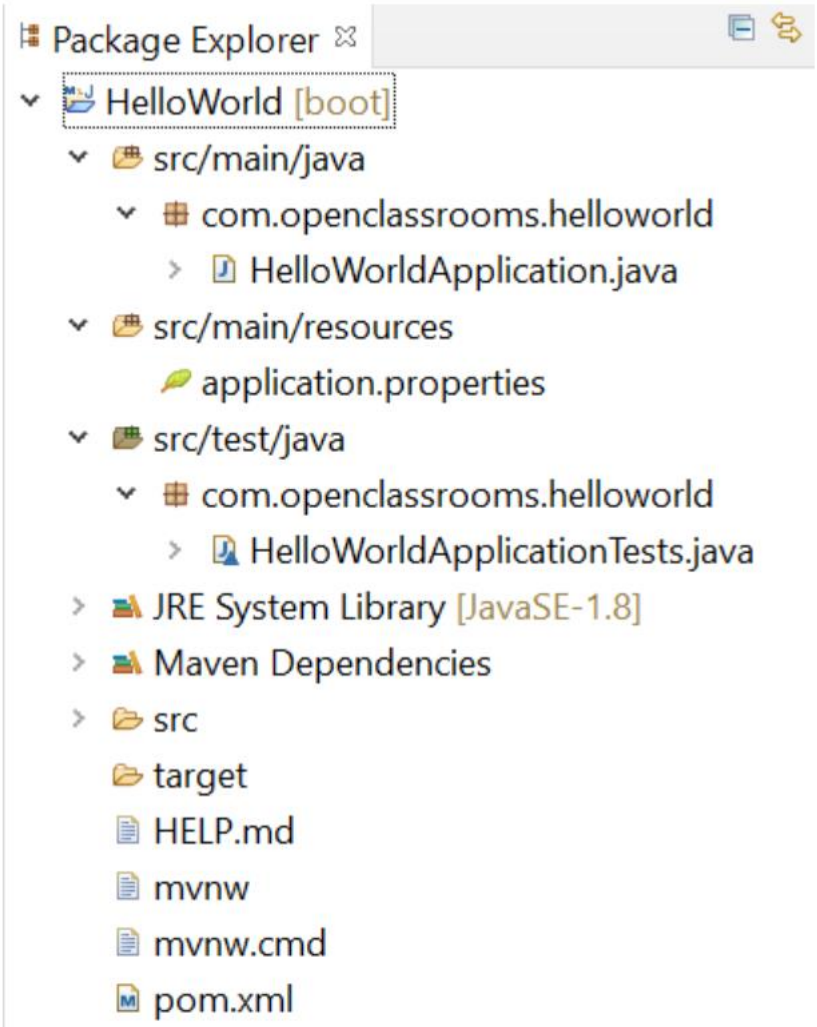


Figure 4 : Structure minimale d'un projet Spring

- Ecrire le code

Une fois que Spring Boot a créé, structuré et configuré le projet, le champ est libre au développeur d'organiser son code comme il le souhaite, car même si Spring Boot nous fournit une structure minimale avec laquelle on peut démarrer elle n'est pas suffisante.

La plupart du temps, Spring est utilisé dans le développement d'application web, de ce fait il est recommandé d'adopter une architecture standard qui correspondant à la majorité des projets, et qui est très fréquemment utilisée. Cette architecture composée de plusieurs couches possédant différents rôles nous aide à structurer au mieux le code. Dans ce qui suit, nous allons énumérer ces différentes couches.

- Couche Controller : cette couche est en contact directe avec le client et réceptionne les requêtes http que ce dernier lui envoie (GET, POST, UPDATE, DELETE, PUT). Les classes java de cette couche porte l'annotation **@RestController**.
- Couche Service : cette couche se charge d'implémenter la logique et le traitement métier qui diffère selon les besoins de l'application. Les classes java de cette couche porte l'annotation **@Service**.
- Couche Repository : cette couche est représentée sous forme d'une interface java, elle se charge de l'interaction avec les données externes (base de données, par exemple). Les classes java de cette couche porte l'annotation **@Repository**.
- Couche Model : cette couche comporte l'implémentation des objets métier qui seront par la suite utilisés par les autres couches afin d'exécuter des traitements. Les classes java de cette couche porte l'annotation **@Entity**.

Nous allons illustrer ci-dessous les différentes couches de l'architecture précédemment cité.

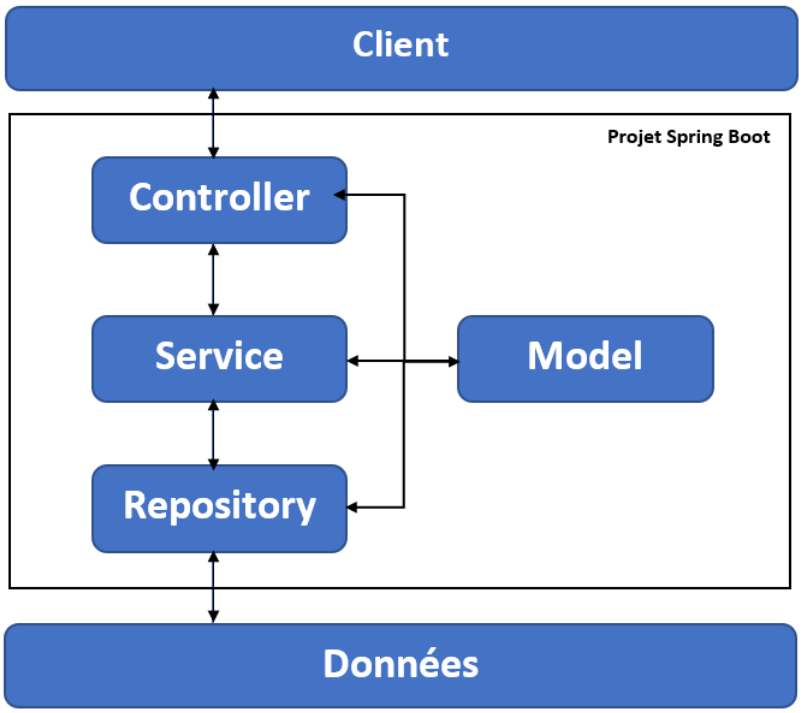


Figure 5 : Architecture d'un projet Spring

- Tester et déployer

Les tests de Spring Boot se font en ajoutant l'annotation **@SpringBootTest, cette annotation permet d'initialiser le contexte Spring. Quant au déploiement il peut se faire de trois manières :**

- A travers l'IDE avec **Run As> Spring Boot App**
- Avec le Goal Maven **spring-boot:run**
- **En exécutant le JAR** grâce à la commande java -jar.

I.3- Conclusion

Dans ce chapitre nous avons pu faire le tour des deux principales technologies avec lesquelles nous réaliserons notre projet. Nous avons vu que Réact est une bibliothèque qui permet de réaliser des interfaces des plus modernes et que combiné à Spring Boot cela pourrait nous amener à créer une application complète, moderne et ceci en l'espace d'un temps très court.

Chapitre II

Presentation de l'organisme d'accueil et le recueil des besoins

Chapitre II

Presentation de l'organisme d'accueil et le recueil des besoins

II.1- Introduction

Toute analyse doit commencer par l'étude de ce qui existe. L'analyse de l'existant ou étude préalable, constitue une étape fondamentale dans l'analyse des besoins.

Dans ce chapitre, nous présenterons dans un premier temps, l'organisme SARL VMS, qui nous a accueillis dans le cadre du stage pratique, le déroulement de son activité de façon générale, ainsi que les problèmes vécus quotidiennement par ses responsables et décideurs.

Pour finir, nous expliquerons la solution que nous proposons afin de remédier aux problèmes énoncés, tout en spécifiant la méthodologie de notre travail, et les objectifs de notre projet.

II.2- Présentation de l'organisme d'accueil

La SARL VMS INDUSTRIE est une entreprise de fabrication et de montage de cycles et motocycle créée en 2014, et est la seule entreprise de production de motocycles en Algérie.

Le volet formation est l'un des objectifs majeurs de l'entreprise qui a signé une convention avec l'université de BEJAIA et les centres de formations professionnels pour faciliter l'intégration de ses futurs employés.

VMS comportait en 2020, 250 employés dont 40% sont des cadres (Ingénieurs, techniciens supérieurs et licenciés) et dont la moyenne d'âge est de 30 ans, et cet effectif augmente progressivement. Profitant de leurs savoir-faire et longue

expérience, l'entreprise fait des partenariats de transfert technologique et paie à ses employés des formations au sein des centres de ses partenaires étrangers.

VMS INDUSTRIE offre une large gamme de produits visant à satisfaire les clients. Et enregistre une croissance continue comme en témoigne l'évolution de son activité qui double d'année en année. Cette croissance est également accompagnée du développement de son réseau de distribution qui a atteint actuellement 38 wilayas du pays, et continu de s'étendre en fonction de l'augmentation de la production. Ce réseau de distribution assure aussi le service après-vente qui procure des pièces de rechanges d'origine et les équipements de sécurité adéquats.

Sur le plan marketing et développement, VMS assistent à toutes les foires liées à l'automobile (toutes les foires de la production nationales, la foire de la sous-traitance, de la pèche et de l'aquaculture, salon de la journée nationale de l'innovation, salon de l'emploi, salon AUTOWEST, salon international du savoir-faire et de sous-traitance industrielle MIDEST LYON et AFRICALLIA forum d'affaires qui s'est tenu à ABIDJAN COTE D'IVOIRE).

II.2.1- Historique

Les associés de la SARL VMS INDUSTRIE ont entamé les démarches pour la création de cette activité de fabrication et montage de cycles et motocycles à partir de l'année 2012 avec la construction du bâtiment au niveau d'IFRI OUZELLAGUEN W. de BEJAIA qui a servi de première unité de Montage. La création officielle de la SARL VMS INDUSTRIE eu lieu le 24/07/2014 et le démarrage de l'activité le 21/01/2015 avec un capital social de départ de 79 710 000 DA.

Au cours de l'année 2015 un terrain a été acquis au niveau de la zone d'activité de LARBAA commune de TOUDJA pour la réalisation d'une unité de production aux normes industrielles et aux capacités de production plus importantes. Cette unité est mise en exploitation en octobre 2018.

Un projet est en voie de finalisation à la zone d'activité de GUELLAL W. de SETIF, ou il y aura le montage de deux types de motocycles et à la fabrication d'intrants destinés à la production de motocycles.

II.3- Champs d'étude

Vu les problèmes de gestion auxquels l'entreprise VMS INDUSTRIE est confrontée, elle a été contrainte à trouver des alternatives à la façon dont elle exerce ses activités internes et a décidé d'automatiser son entreprise en développant un ERP capable de structurer, unifier et gérer au mieux toutes ses ressources.

L'étendu de notre étude se limite au service production, ce dernier étant une partie très importante de l'organisation, sa bonne gestion relève de l'obligation. La gestion du processus de production sera donc le thème de notre projet.

Lors de notre première visite de VMS INDUSTRIE, nous nous sommes rendus à leur unité de production afin de mieux comprendre l'environnement et le processus, là-bas nous avons pu voir de près l'activité de cette unité et avons suivi du début à la fin l'enchainement des activités, grâce à ça nous pouvons résumer ce processus en plusieurs étapes comme suit :

1) L'approvisionnement du stock : lorsqu'on se rend compte que le stock est en voie de rupture en matière première, on rédige un bon de commande au magasinier afin de le réapprovisionner.

   Dans le cas où l'on a besoin d'un composant en petite quantité dans l'immédiat (par exemple une visse ou autre), le service après-vente dépanne la chaine de production en lui procurant les pièces manquantes. Et cela se fait en rédigeant un bon de commande au service après-vente par l'unité de production.
2) La chaine de production : le composant traverse deux phases dans la chaine de production, la phase « production » et la phase « contrôle ».

Dans la phase production, il existe deux types de postes, premièrement, on retrouve les postes de montages, ou l'employé a pour tâche d'assembler des pièces ou des ensembles de pièces montés préalablement afin de produire un nouveau composant qui sera envoyé au prochain poste. Deuxièmement les postes qui gravent les numéros des composants, où l'on attribue et colle aux pièces des identifiants.

La phase contrôle de la chaine de production s'effectue en deux fois :

- Contrôle technique : ou l'on démarre le produit (motocycle) pour vérifier qu'il est fonctionnel et qu'il ne comporte pas de problèmes (électriques ou autre).
- Contrôle qualité : vérifie que tous les composants du produit y sont, et qu'il n'y a pas eu de manque dû à un oubli. Par la suite, des accessoires et autres composants (batterie et rétroviseurs) sont ajoutés au produit finit.

Ces deux contrôles détermineront si le produit est fini ou semi-fini, et pourra ainsi être transféré au dépôt.

Dans le cas où le produit est semi-fini, qui veut dire qu'il n'est pas prêt à être vendu ceci dû à une pièce manquante ou à un quelconque problème, le responsable de chaine doit le justifier.

Les tâches des employés de la chaine de production ne sont pas fixes, car ils peuvent changer de postes en fonction des jours.

3) Contrôle pré-distribution : Avant la distribution, on doit enregistrer les produits qui seront sorti pour être livrés, et cela se fait manuellement avant le chargement.

II.4- L'organigramme de VMS Industrie

L'illustration ci-dessous représente l'organigramme de l'entreprise VMS Industrie

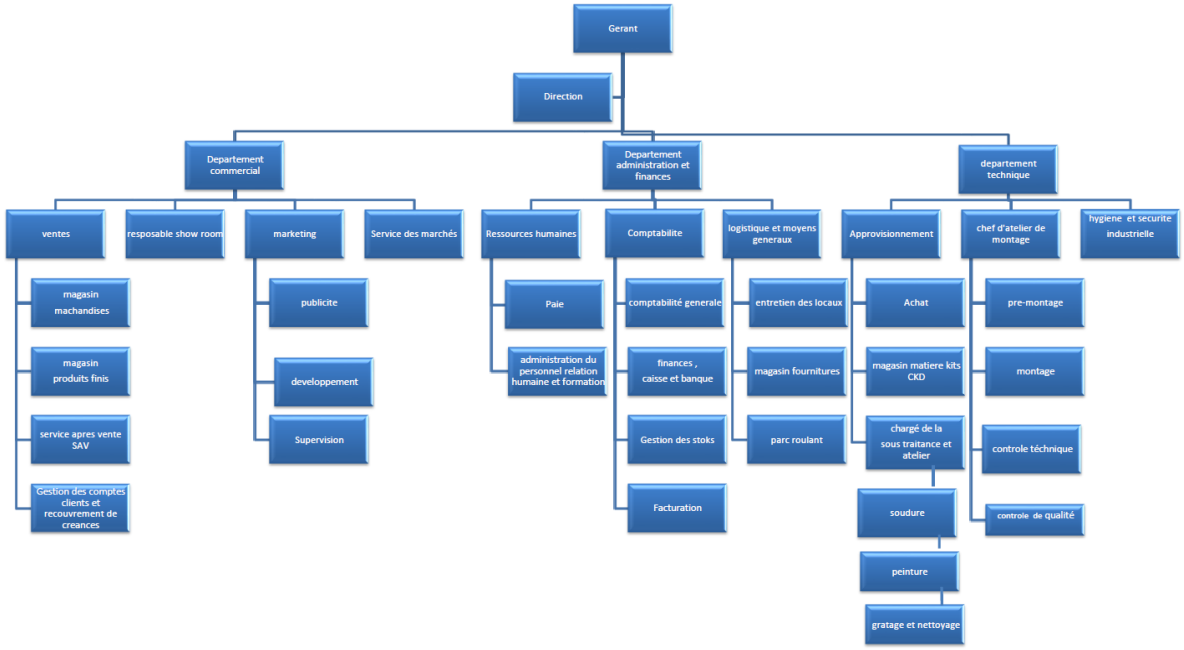


Figure 6 : Organigramme de VMS Industrie

II.5- Problématique

Lors de notre découverte de l'unité de production de VMS INDISTRIE dans le cadre de notre stage pratique, nous avons constaté les difficultés et la complexité de la tâche qui est la gestion de la production. Nous avons pu alors énumérer les problèmes rencontrés au niveau de cette unité :

- L'inexistence de système informatique automatisant ce service.
- Nombre important d'employés qui y travaillent, et par la suite, impossibilité de retrouver les éléments qui commettent des erreurs de montage, ralentissent la chaine de production ou l'inverse.
- Risque de manquer d'approvisionnement sans s'en rendre compte.
- Risque de distribution des mauvais produits suite à des erreurs de chargement.
- Risque de recevoir de fausses doléances des clients n'ayant pas reçus leurs accessoires (Casque de sécurité) ou pièces (rétroviseurs, batterie etc.).

II.6- Solution proposée

L'objectif de notre travail est de réaliser un module parmi les modules de gestion de L'ERP de VMS INDUSTRIE, qui sera celui de la gestion de la production, qui veut dire l'automatisation des étapes de la chaine de production et l'enregistrement les éléments qui y participe.

Notre application vise à atteindre les objectifs suivants :

- L'automatisation du service production.
- L'intégration du système d'information.
- Une meilleure cohérence et un plus grand partage de données.
- Limiter les erreurs du quotidien.

II.7- Méthodologie de travail

Afin de réaliser notre projet, nous allons suivre la méthode 2TUP (2Track Unified Process), qui apporte la solution aux contraintes de changement continuel imposés au système d'information de l'entreprise. Cette méthode est divisée en deux parties : l'axe fonctionnels et l'axe technique.

Dans la partie conception, nous utiliserons UML, du fait qu'il est le plus adapté pour les projets informatiques orienté objet, et que ces diagrammes sont plus lisibles et compréhensibles.

II.8- Cahier des charges

La réalisation du cahier des charges est l'une des étapes primordiales des projets informatiques, car il résume le premier contact entre le concepteur et le client. Il est divisé en deux parties, les besoins fonctionnels, et les besoins non fonctionnels.

II.8.1 Besoins fonctionnels

Les besoins fonctionnels représentent les fonctionnalités du futur système. Nous pouvons les résumer comme suit :

- Gestion des chaines de production

- Gestion des employés
- Gestion des postes de production
- Gestion des tâches

Le terme « Gestion » dans les quatre points précédents regroupe les actions : d'ajout modification et suppression de l'élément en question.

- Gestion des postes de production

  Le futur système enregistrera pour chaque poste d'une chaine de production, les personnes qui ont effectués les tâches attribuées et leur chef de poste.

- Gestion des codes QR

  Le système permettra de générer des codes QR pour chaque produit finis ou semi-finis manipulé par la chaine de production, et on pourra par la suite les imprimer les ou les scanner, tout en gardant trace des postes s'étant chargé d'effectuer le montage ou autres étapes par lesquelles est passé le produit final.

- Gestion des erreurs

  Une notification sera générée dans le cas où il y a eu des retards dans la chaine de production, et détermine le poste ayant causé le retard. Il permettra aussi de signaler une anomalie sur le produit.

- Gestion des composants « matière première »

  Le système devra prévenir le manque de matières premières à l'approche de l'épuisement du stock, une notification sera générée afin qu'on puisse rédiger un bon de commande au magasin. Une fois le stock réapprovisionné, l'application de notifiera également.

  Dans le cas où l'on demande des articles au service après-vente, le système là aussi sera en mesure de notifier l'arrivé des articles.

- Gestion des accessoires

  Pour chaque produit (motocycle) on doit pouvoir scanner et enregistrer ses accessoires avant la distribution.

- Gestion de la pré-distribution

  Chaque produit fini prêt à être distribué, doit être scanné et enregistré deux fois, avant et après le chargement.

II.8.2 Besoins non fonctionnels

Les besoins non fonctionnels sont les besoins liés à la sécurité et à l'utilisation, ils permettent d'améliorer la qualité logicielle du système, Ce dernier doit répondre aux exigences suivantes :

- Sécurité : la sécurité d'un système est un point très important dans sa réalisation. Notre système devra alors être accédé par les utilisateurs avec un identifiant et un mot de passe.
- Interface : Les interfaces de notre système doivent répondre à plusieurs critères :
  - Conviviale : L'interface doit être intuitive et simple d'utilisation.
  - Ergonomique : L'interface doit avoir des couleurs selon le thème de l'entreprise.
  - Compatible : L'interface doit s'intégrer dans l'activité réelle des utilisateurs.
- Performance : La performance veut dire la rapidité du système et temps de réponse court.

II.9-Conclusion

Dans ce chapitre nous avons présenté l'organisme d'accueil, soulevé la problématique et proposé la solution optimale afin d'améliorer la gestion des différentes taches de cette entreprise et cela à travers les besoins fonctionnels et non

fonctionnels. Cela nous a permis de mieux comprendre le fonctionnement de l'entreprise, afin de pouvoir passer au chapitre suivant qui lui comportera une analyse plus profonde des besoins.

Chapitre III

Analyse des besoins

Chapitre III

Analyse des besoins

III.1- Introduction

Dans ce chapitre, nous allons énoncer dans un premier temps les différents acteurs de notre système, leurs rôles ainsi que leurs interactions avec celui-ci. Par la suite nous présenterons le diagramme de cas d'utilisation qui résumera les besoins de l'entreprise, et nous terminerons par une description textuelle des cas d'utilisation et les diagrammes de séquence système.

III.2- Identification des acteurs

Un acteur est l'idéalisation d'un rôle joué par une personne externe, un processus ou une chose qui interagit avec un système. [1]
En ce qui concerne notre système, nous citons les acteurs suivants :

1. Responsable de production : Son rôle est de superviser le service de façon générale, il consulte le taux de production et doit rendre des comptes à son supérieur. C'est lui qui désigne les chefs de chaine de production.
2. Chef de chaine : il est responsable de la chaine de production, et doit la superviser. Il crée les postes et leurs affecte des tâches, des employés et des chefs de poste. Les bons de commande sont créés par cet utilisateur pour être validé par la suite par l'ordonnanceur.
3. Chef de poste : il supervise le poste auquel il est affecté. Il valide les tâches effectuées par son poste et transfert le produit au prochain poste.
4. Ordonnanceur : cet acteur à pour rôle de valider les bons de commande générés par le chef de chaine et réceptionner les bons de livraison.
5. Administrateur : il aura pour mission de gérer les comptes utilisateurs.

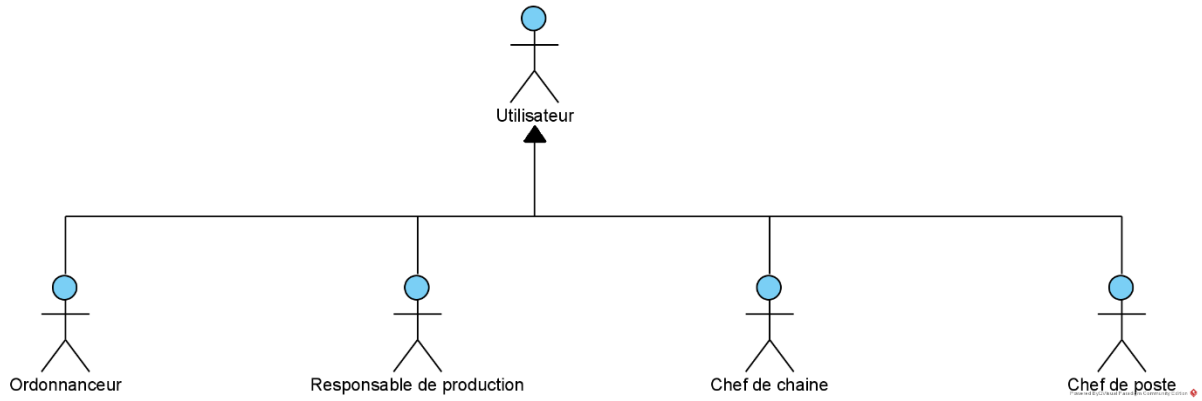


Figure 7 : Acteurs et héritage entre acteurs

III.3- Diagramme de contexte

Le diagramme de contexte présente le système sous forme d'une boite noire et les différents acteurs avec qui il interagit.

En se basant sur les acteurs énoncés précédemment et leurs interactions avec notre futur système, nous vous présenterons ci-dessous le diagramme de contexte de notre application.

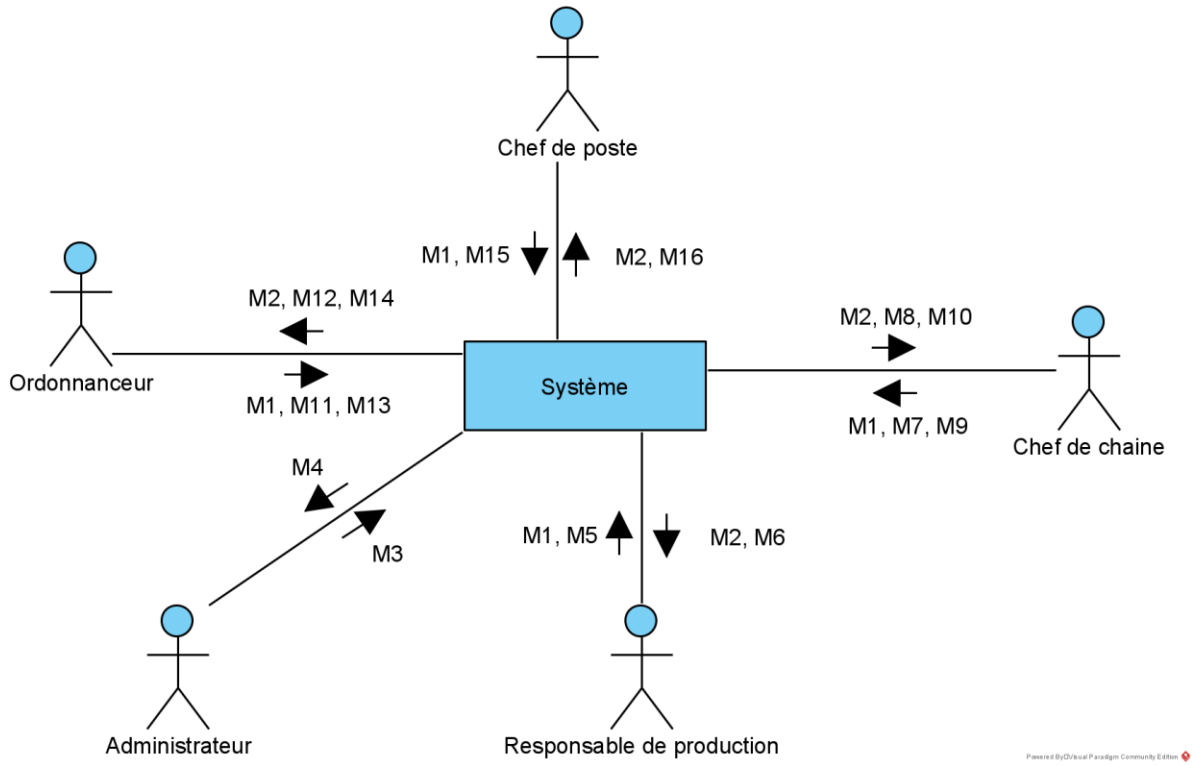


Figure 8 : Diagramme de contexte

III.3.1- identifications des messages

Les différents acteurs de notre système peuvent échanger des messages entrants ou bien *sortants comme suit :*

Tableau 1 : Identification des messages échangés.

N°	Message Acteur -> Système	N°	Message Système -> Acteur
M1	Demande d'authentification	M2	Affichage de l'interface d'accueil correspondant à chaque acteur
M3	Créer compte	M4	Création compte
M5	Créer chef de chaine	M6	Création chef de chaine
M7	Créer poste avec chef de poste	M8	Création poste avec chef de poste
M9	Créer bon de commande	M10	Création bon de commande
M11	Lister bon de livraison	M12	Affichage des bons de livraison
M13	Générer bon de réception	M14	Génération du bon de réception
M15	Exécuter une tâche	M16	Exécution de la tâche et transférer le produit au poste suivant

III.4- Identification des cas d'utilisation

Dans ce qui suit, nous allons présenter les différents cas d'utilisation pour chaque acteur du système. Pour mieux comprendre ces cas d'utilisation.

Tableau 2 : Identification des cas d'utilisation

N°	Cas d'utilisation		Acteur
1	S'authentifier		Utilisateur
2	Lister comptes utilisateur	Ajouter	Administrateur
		Modifier	
		Supprimer	
3	Lister chaines de production	Ajouter	Responsable de production
		Modifier	
		Supprimer	
4	Lister les produits semi-finis	Justifier produits semi-finis	
		Transfert dépôt	
5	Lister les postes de	Modifier	Chef de chaine

	production	Ajouter	Ajouter tache	
			Modifier tache	
			Supprimer tache	
			Ajouter employé	
			Modifier employé	
			Supprimer employé	
			Ajouter chef de poste	
			Modifier chef de poste	
			Supprimer chef de poste	
		Supprimer		
	Lister les bons de commande	Ajouter		
		Modifier		
		Supprimer		
6	Créer produit	Ajouter numéro de châssis		Chef de poste
	Transformer produit	Ajouter moteur		
		Ajouter articles		
		Ajouter couleur		
		Ajouter anomalie		
		Scanner code barre		
		Scanner code QR		
		Générer code QR		
7	Lister les bons de commande	Valider bon de commande		Ordonnanceur
	Lister les bons de livraison	Générer bons de réception		

III.4.1- Diagramme de cas d'utilisation

Un diagramme de cas d'utilisation capture le comportement d'un système, d'un sous-système, d'une classe ou d'un composant tel qu'un utilisateur extérieur le voit. Il scinde la fonctionnalité du système en unités cohérentes, les cas d'utilisation, ayant un sens pour les acteurs. Les cas d'utilisation permettent d'exprimer le besoin des utilisateurs d'un système, ils sont donc une vision orientée utilisateur de ce besoin au contraire d'une vision informatique. [1]

Ci-dessous le diagramme de cas d'utilisation de notre système.

1) Diagramme de cas d'utilisation de l'acteur « Administrateur »
2) Diagramme de cas d'utilisation de l'acteur « Responsable de production »

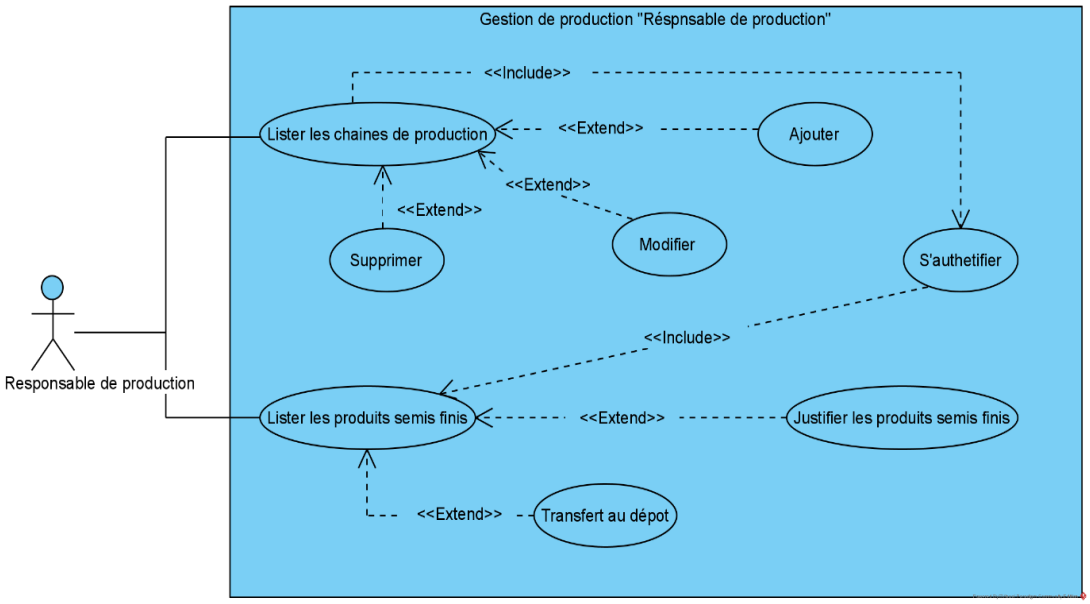


Figure 9 : Diagramme de cas d'utilisation " Responsable de production "

3) Diagramme de cas d'utilisation de l'acteur « Chef de chaine »

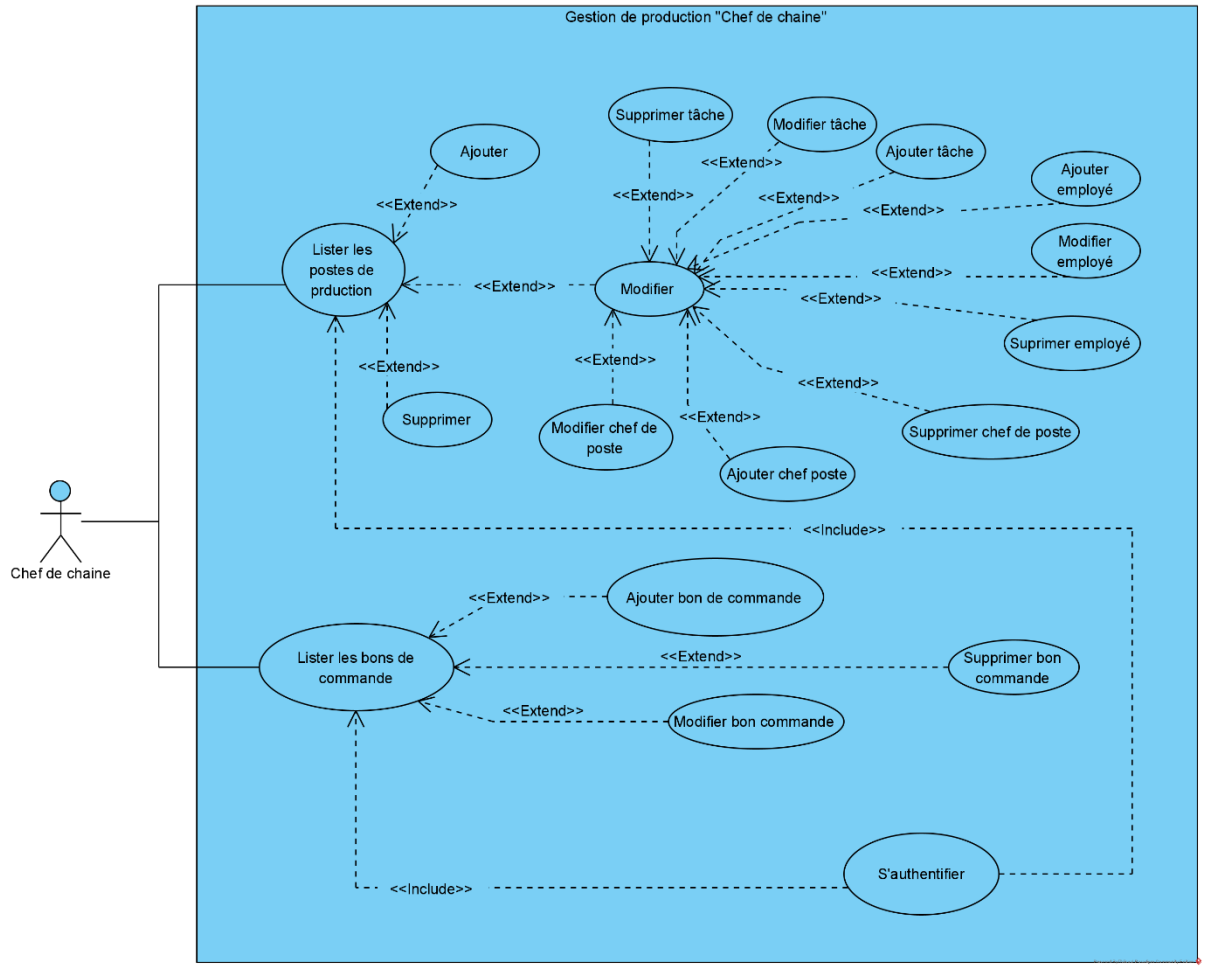


Figure 10 : Diagramme de cas d'utilisation " Chef de chaine "

4) Diagramme de cas d'utilisation de l'acteur « Chef de poste »

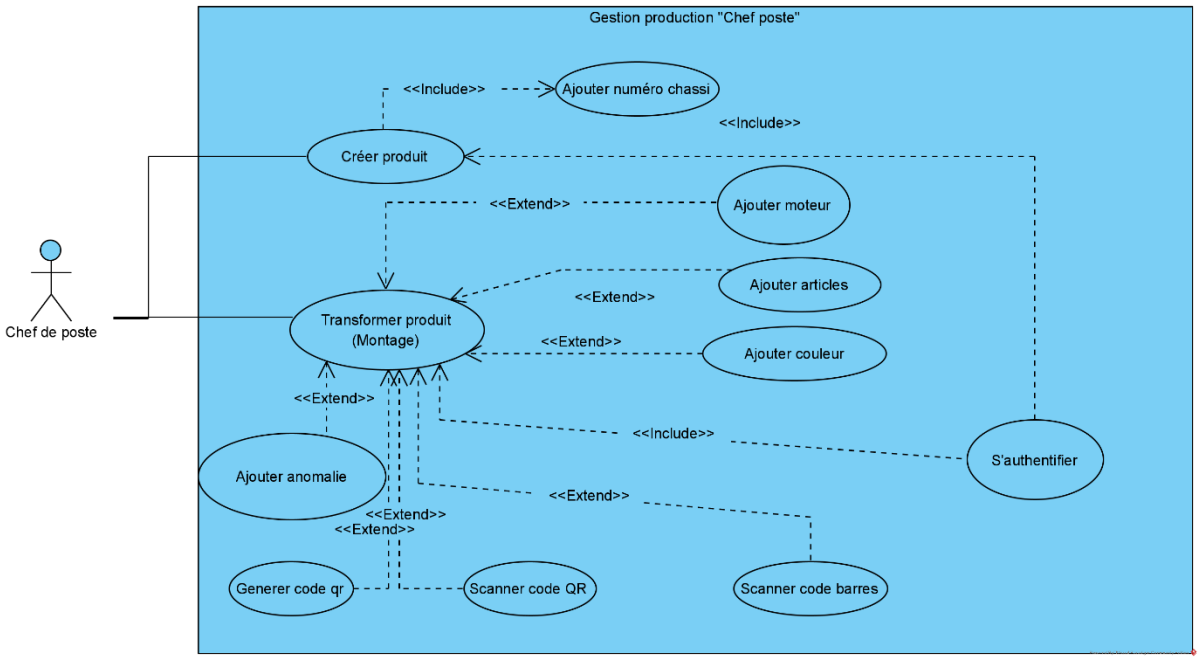


Figure 11 : Diagramme de cas d'utilisation " Chef de poste "

5) Diagramme de cas d'utilisation de l'acteur « Ordonnanceur »

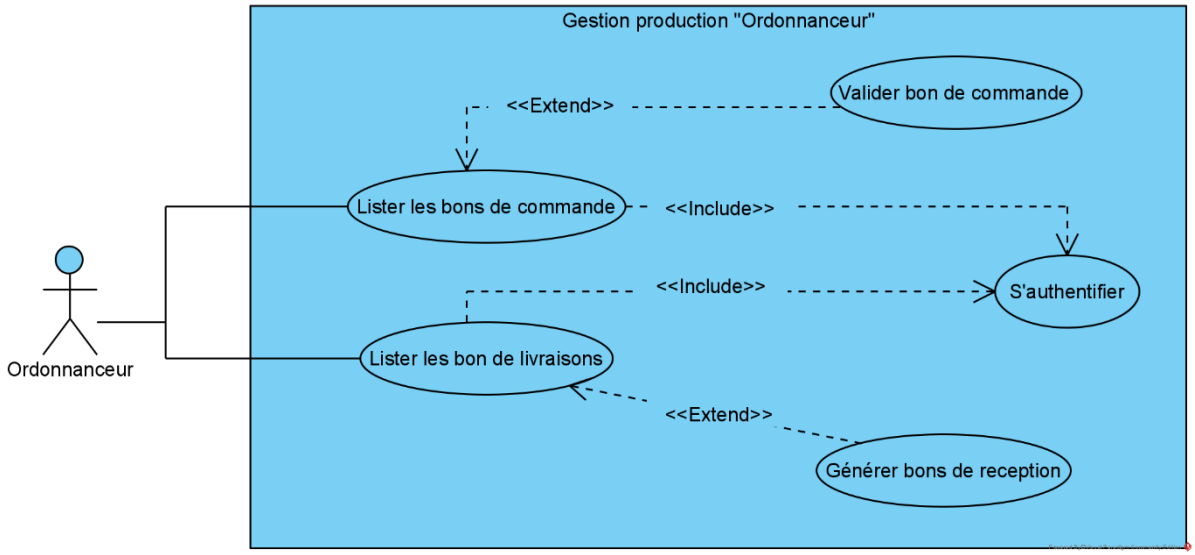


Figure 12 : Diagramme de cas d'utilisation " Ordonnanceur "

III.5- Description textuelle des cas d'utilisation

Le diagramme de cas d'utilisation décrit les fonctionnalités du système de façon globale, et n'expose pas de façon détaillée l'interaction acteur-système au sein de l'application. Pour y remédier, dans ce qui suit nous rédigerons la description

textuelle de chaque cas d'utilisation pour une meilleure compréhension de ces derniers.

1) Description textuelle du cas d'utilisation « S'authentifier »

Tableau 3 : Description textuelle du cas d'utilisation « S'authentifier »

Cas d'utilisation « S'authentifier »	
But	Permet d'identifier l'utilisateur et d'avoir son rôle dans l'application (Privilèges de l'utilisateur).
Acteur	Utilisateur
Description	L'utilisateur doit fournir un identifiant et un mot de passe. Le système utilise l'identifiant et le mot de passe pour vérifier si ses informations sont correctes. Dans le cas où les informations sont correctes le système affiche l'interface adéquate de l'utilisateur. Dans le cas contraire un message d'erreur sera affiché.
Précondition	/
Scénario nominal	Ce cas d'utilisation est déclenché lorsqu'un utilisateur veut accéder au système. S'authentifier : ce cas permet à l'utilisateur d'accéder au système selon l'enchaînement suivant : 1- Le système affiche l'interface de l'authentification 2- L'utilisateur saisit son identifiant et son mot de passe et valide l'opération. 3- Le système effectue des vérifications. -Si l'utilisateur a omis l'identifiant ou le mot de passe alors il faut exécuter [Exception01 : Champs Vides] -Si l'identifiant et/ou mot de passe ne sont pas corrects alors il faut exécuter [Exception02 : Champs Incorrects] 4- Dans le cas de succès, le système affiche l'interface d'accueil correspondante à l'utilisateur.
Scénario alternatif	/
Scénario d'exception	Exception 1 : Le système notifie une erreur à l'utilisateur lui indiquant qu'il a oublié, un ou plusieurs champs à saisir (identifiant ou mot de passe), et l'invite à compléter les champs manquants. Exception 2 : Le système indique à l'utilisateur qu'une erreur est détectée liée à son identifiant et/ou à son mot de passe, il l'invite à

	ressaisir son identifiant et/ou son mot de passe.

2) Description textuelle du cas d'utilisation « Créer poste production »

Tableau 4 : Description textuelle du cas d'utilisation « Créer poste production »

Cas d'utilisation « Créer poste production »	
But	Permet de créer les postes de la chaine de production.
Acteur	Chef de chaine
Description	Le chef de chaine attribue au poste un numéro, des employés, des tâches et un chef de poste. Le système utilisera les valeurs des champs pour créer le poste de production.
Précondition	S'authentifier
Scénario nominal	Ce cas d'utilisation est déclenché lorsque le chef de chaine valide le formulaire lui étant affiché. Ce cas d'utilisation permet au chef de chaine de créer un poste selon l'enchaînement suivant : 1- Le chef de chaine s'authentifie. 2- Le système lui affiche son interface 3- Le chef de chaine ouvre l'interface des postes de production et lui affiche les postes existants. 4- L'utilisateur sélectionne le bouton créer un poste. 5- Le système lui ouvre un pop-up avec un formulaire pour lui permettre d'entrer les informations du nouveau poste. 6- Le chef de poste remplie les champs et valide -Si le chef de poste n'a pas rempli tous les champs le système exécutera : [Exception01 : Champs Vides] - Sinon, le système met à jour le produit, et enregistre le poste et la date et l'heure de mise à jour.
Scénario alternatif	/
Scénario d'exception	Exception 1 : Le système notifie une erreur à l'utilisateur lui indiquant qu'il a oublié, un ou plusieurs champs à saisir, et l'invite à compléter les champs manquants.

3) Description textuelle du cas d'utilisation « Transformer produit »

Tableau 5 : Description textuelle du cas d'utilisation « Transformer produit »

Cas d'utilisation « Transformer produit »	
But	Permet à chaque poste d'exécuter sa ou ses tâches.
Acteur	Chef de poste
Description	Le chef de poste doit fournir soit le numéro de châssis, le numéro de moteur ou bien les articles ajoutés au produit et ceci selon les tâches attribuées au poste. Le système utilisera les valeurs des champs pour mettre à jour le produit tout au long de la chaine de production.
Précondition	S'authentifier
Scénario nominal	Ce cas d'utilisation est déclenché lorsque le chef de poste valide le formulaire lui étant affiché. Ce cas d'utilisation permet au chef de poste de valider ses taches selon l'enchaînement suivant : 1- Le chef de poste s'authentifie. 2- Le système lui affiche son interface. 3- Le chef de poste ouvre l'interface du poste et lui affiche les produits parvenants du poste précédent. 4- L'utilisateur sélectionne un produit. 5- Le système lui ouvre un pop-up pour lui permettre d'entrer les informations des champs de saisies. 6- Le chef de poste remplie les champs manuellement, ou bien en scannant les codes QR et valide -Si le chef de poste n'a pas rempli tous les champs de saisie alors le système exécutera : [Exception01 : Champs Vides] - Sinon, le système met à jour le produit, et enregistre le poste et la date et l'heure de mise à jour.
Scénario alternatif	/
Scénario d'exception	Exception 1 : Le système notifie une erreur à l'utilisateur lui indiquant qu'il a oublié, un ou plusieurs champs à saisir, et l'invite à compléter les champs manquants.

4) Description textuelle du cas d'utilisation « Générer bon de réception »

Tableau 6 : Description textuelle du cas d'utilisation « Générer bon de réception »

Cas d'utilisation « Générer bon de réception »	
But	Permet de créer des bons de réception à partir des bon de livraisons.
Acteur	Ordonnanceur.
Description	L'Ordonnanceur liste les bon de livraison générés par le service « Magasin ». L'ordonnanceur vérifie sur place si les produit sont bien arrivés et le confirme en générant à partir de ces bons de livraison des bons de réception.
Précondition	S'authentifier
Scénario nominal	Ce cas d'utilisation permet à l'ordonnanceur de créer un bon de réception à partir du bon de livraison selon l'enchaînement suivant : 1- l'ordonnanceur s'authentifie. 2- Le système lui affiche son interface 3- l'ordonnanceur ouvre l'interface des bons de livraisons. 4- Le système lui liste les bons de livraisons existants. 4- l'ordonnanceur sélectionne un bon de livraison. 5- Le système lui ouvre un pop-up avec un formulaire pour lui permettre d'entrer les informations du bon de réception. 6- l'ordonnanceur remplie les champs et valide. -Si l'ordonnanceur n'a pas rempli tous les champs le système exécutera : [Exception01 : Champs Vides] - Sinon, le système met à jour le produit, et enregistre le poste et la date et l'heure de mise à jour.
Scénario alternatif	/
Scénario d'exception	Exception 1 : Le système notifie une erreur à l'utilisateur lui indiquant qu'il a oublié, un ou plusieurs champs à saisir, et l'invite à compléter les champs manquants.

III.6- Diagrammes de séquence système

Un diagramme de séquence est un diagramme d'interaction qui expose en détail la façon dont les opérations sont effectuées : quels messages sont envoyés et quand ils le sont. [5]

Dans ce qui suit nous allons résumer le scénario nominal de la description textuelle des cas d'utilisations précédemment énoncé en diagrammes de séquence système. Ces derniers nous permettront de détailler les interactions entre les acteurs et le système, qui lui sera représenté comme étant une boite noire.

1) Diagramme de séquence système « S'authentifier »

Lorsque l'utilisateur arrive sur la page d'accueil de notre application, il ne pourra rien faire d'autre que de s'authentifier. Il clique alors sur le bouton « Se connecter » et l'interface d'authentification s'affiche. Il introduit son identifiant et mot de passe pour accéder à sa section. Le système contrôle dans un premier temps la validité des champs de saisie, si un ou tous les champs sont vides, un message sera affiché à l'utilisateur lui expliquant qu'il y a des champs manquants. Sinon, son interface d'accueil s'affiche.

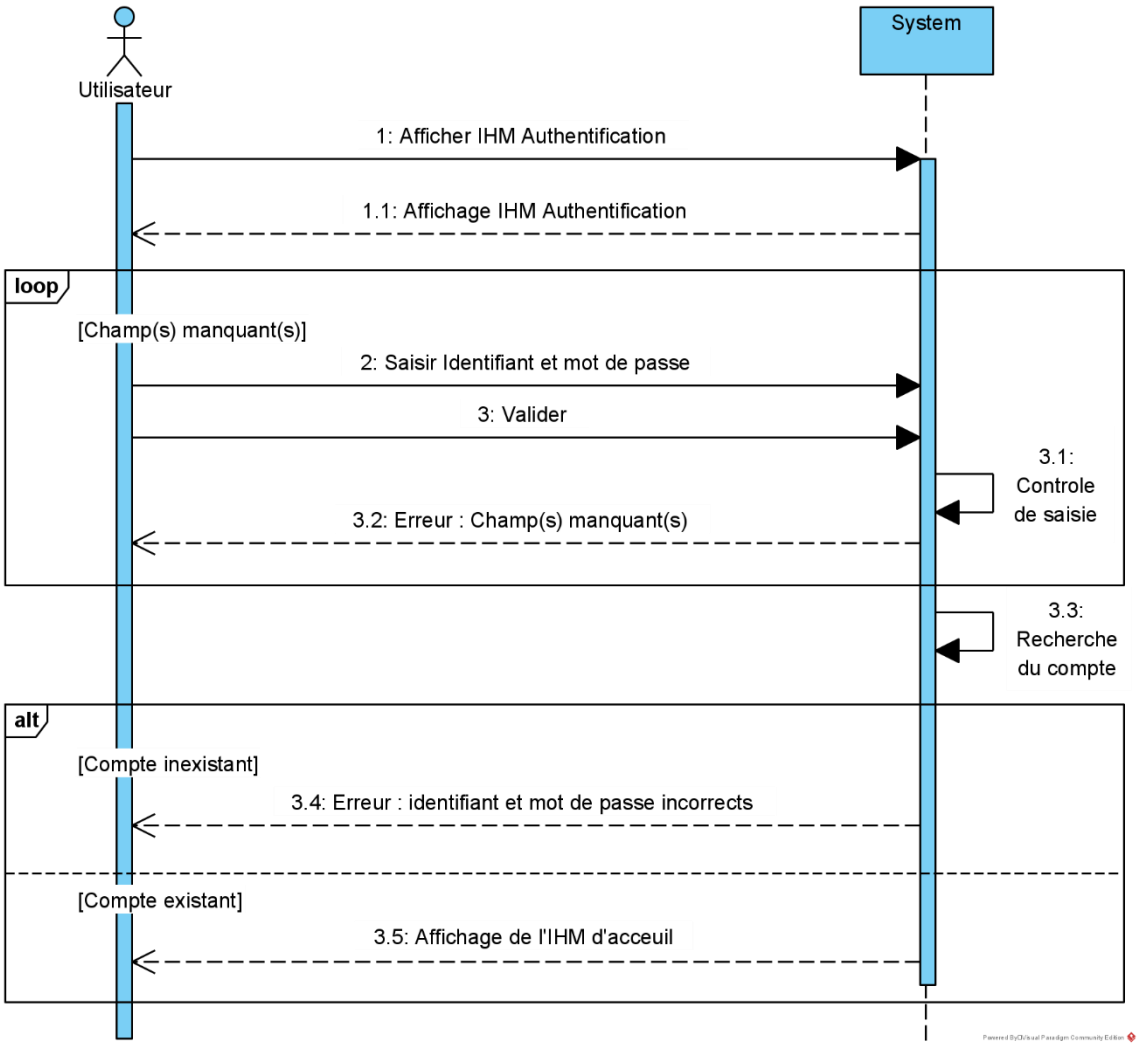


Figure 13 : Diagramme de séquence système « S'authentifier »

2) Diagramme de séquence système « Créer poste »

Ce cas d'utilisation concerne le chef de chaine, une fois ce dernier authentifié, il sélectionne l'interface des postes. L'interface s'affiche avec tous les postes existants. Le chef de poste sélectionne le bouton d'ajout et le formulaire d'ajout apparait. Une fois les champs de saisie remplis et le formulaire validé. Le système contrôle si tous les champs ont bien été remplis et créé le poste.

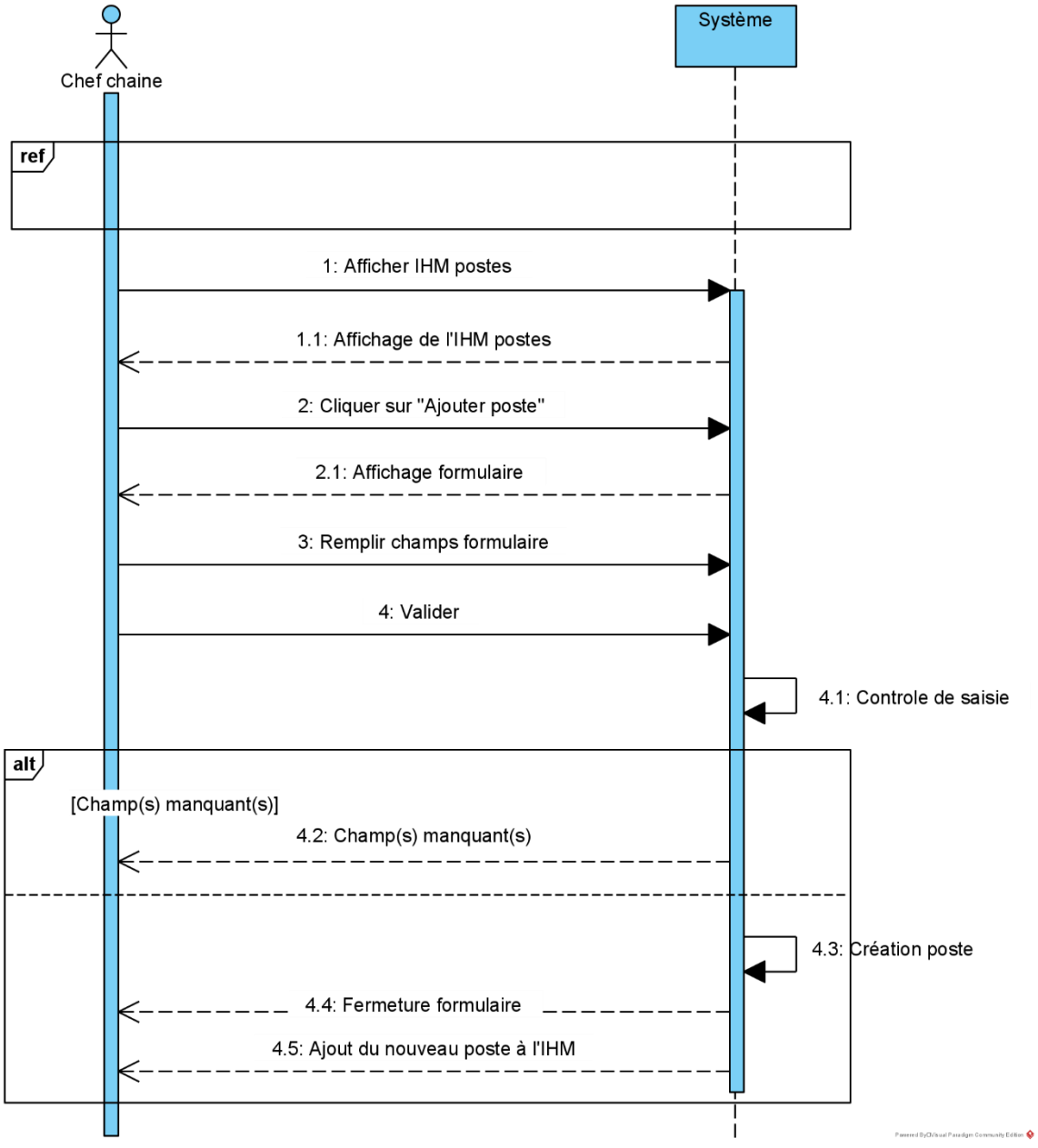


Figure 14 : Diagramme de séquence système « Créer poste »

3) Diagramme de séquence système « Transformer produit »

La transformation du produit dans la chaine de production est effectuée par les chefs de poste. Une fois authentifié, le chef de poste accède à l'interface de montage où apparait la liste des employés du poste et des taches à faire, ainsi que les produits parvenant du poste précédant. Il sélectionne l'un de ces derniers, le modifie, et valide les taches du poste sur ce produit. Une fois la modification validé, le nouveau produit s'affiche dans l'interface du poste suivant etc…

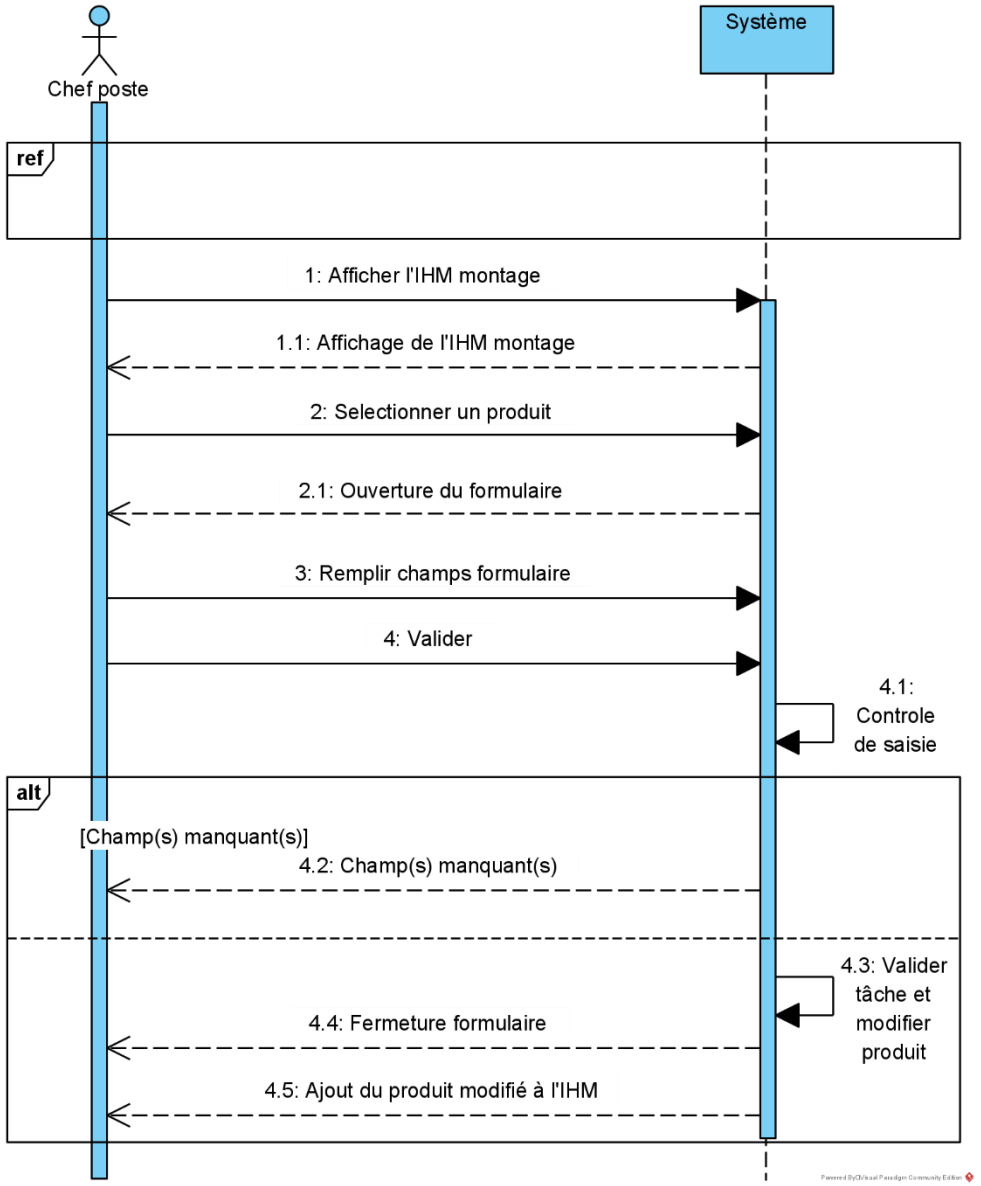


Figure 15 : Diagramme de séquence système « Transformer produit »

4) Diagramme de séquence système « Générer bon de réception »

Ce cas d'utilisation est réalisé par l'ordonnanceur lorsqu'il reçoit des bons de livraison. Apres l'authentification, l'ordonnanceur accède à son interface d'accueil, sélectionne l'interface des bons de livraison et tous les bons

de livraison apparaissent. Une fois là-bas, il sélectionne un bon de livraison et génère son bon de réception. Un formulaire s'ouvre et l'ordonnanceur introduit les informations du bon de réception et enregistre. Le système créé alors ce bon de réception en fonction des informations introduite et celles du bon de livraison (numéro du bon de livraison, lignes de commande).

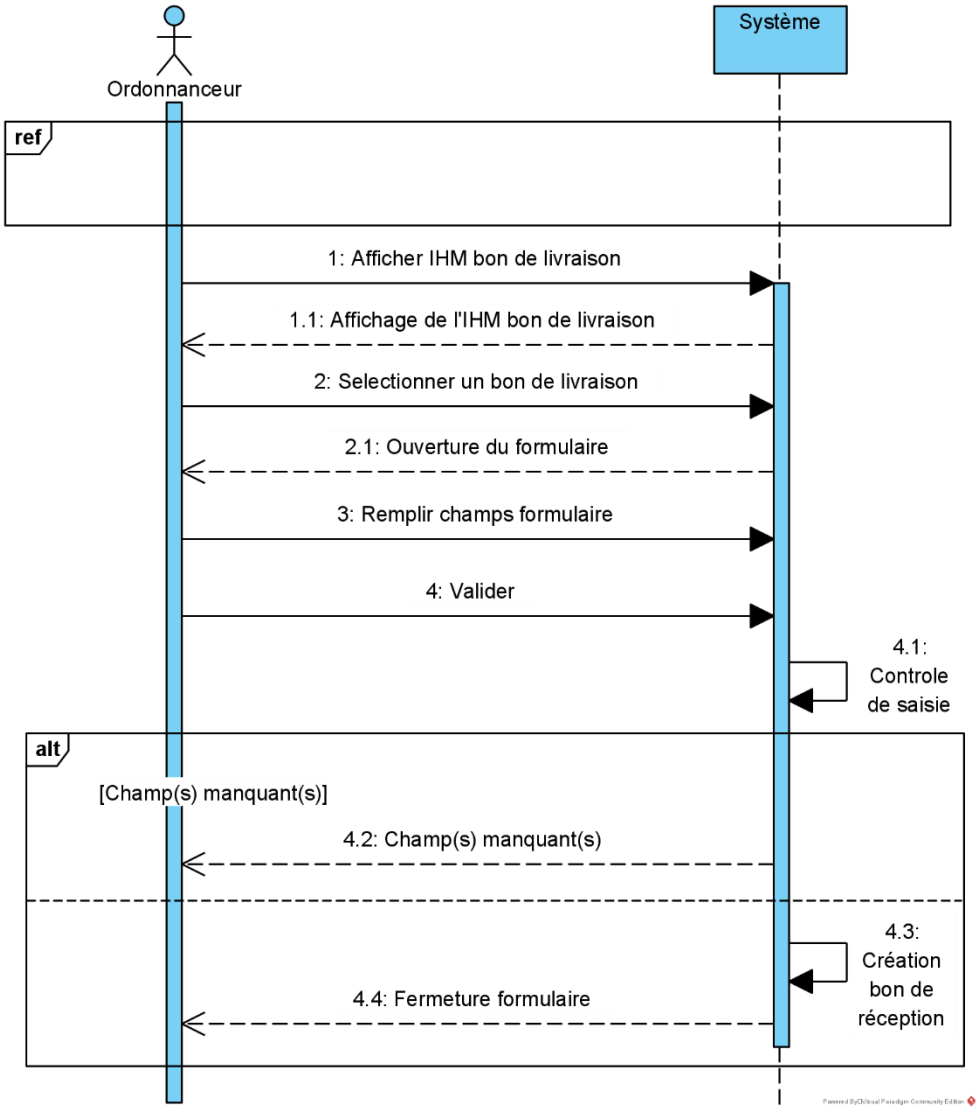


Figure 16 : Diagramme de séquence système « Générer bon de réception »

III.7-Conclusion

Dans ce chapitre nous avons exprimé et détaillé les besoins de l'organisme d'accueil et cela à travers le diagramme de cas d'utilisation et les diagrammes de sequence système. Cette étape nous a permis de mieux comprendre les besoin afin d'entamer la conception de notre système. Nous pouvons à present passer au chapitre suivant qui explique la conception de l'application.

Chapitre IV

Conception et Elaboration du Schema Relationnel.

Chapitre IV

Conception et elaboration du schema relationnel

IV.1- Introduction

Apres l'analyse des besoins, nous arrivons à l'étape de la conception qui constitue une étape primordiale dans la réalisation de projets informatiques, car elle fournit une méthodologie et notations qui aident à la conception de logiciels de qualité.

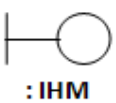
Dans ce chapitre, nous vous présenterons la conception de notre application via les diagrammes de séquences d'interactions, le diagramme de classes et enfin le modèle relationnel.

IV.2- Diagrammes d'interactions

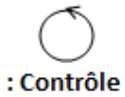
Le diagramme de séquence représente la succession chronologique des opérations réalisées par un acteur. Il indique les objets que l'acteur va manipuler et les opérations qui font passer d'un objet à l'autre. [1]

Dans ce diagramme nous allons nous servir de trois types de classes :

- **Classes d'interface (boundary) :** Des classes qui permettent l'interaction entre l'application et ses utilisateurs. Pour chaque cas d'utilisation, il y a au moins une classe d'interface. Ce type de classe est schématisé comme suit :



- **Classes de Contrôle (Control) :** Ce sont des classes qui contiennent les traitements et la cinématique de l'application. Elles font la transition entre les classes d'interface et les classes entités. Elles sont schématisées comme suit :



- **Classes entités (entity) :** Elles représentent les objets métiers, et ce sont très souvent des entités persistantes, c'est-à-dire qui vont garder leurs informations (données) après l'exécution d'un cas d'utilisation particulier. En général, elles sont enregistrées dans une base de données. Leurs schématisation se fait grâce à ce stéréotype :

IV.2.1- Diagrammes de séquences d'interaction des cas d'utilisation

IV.2.1.1- Diagramme de séquences d'interaction de cas d'utilisation « S'authentifier »

L'authentification garantie la sécurité de l'application. Chaque utilisateur devra se s'authentifier afin d'exécuter ses tâches.

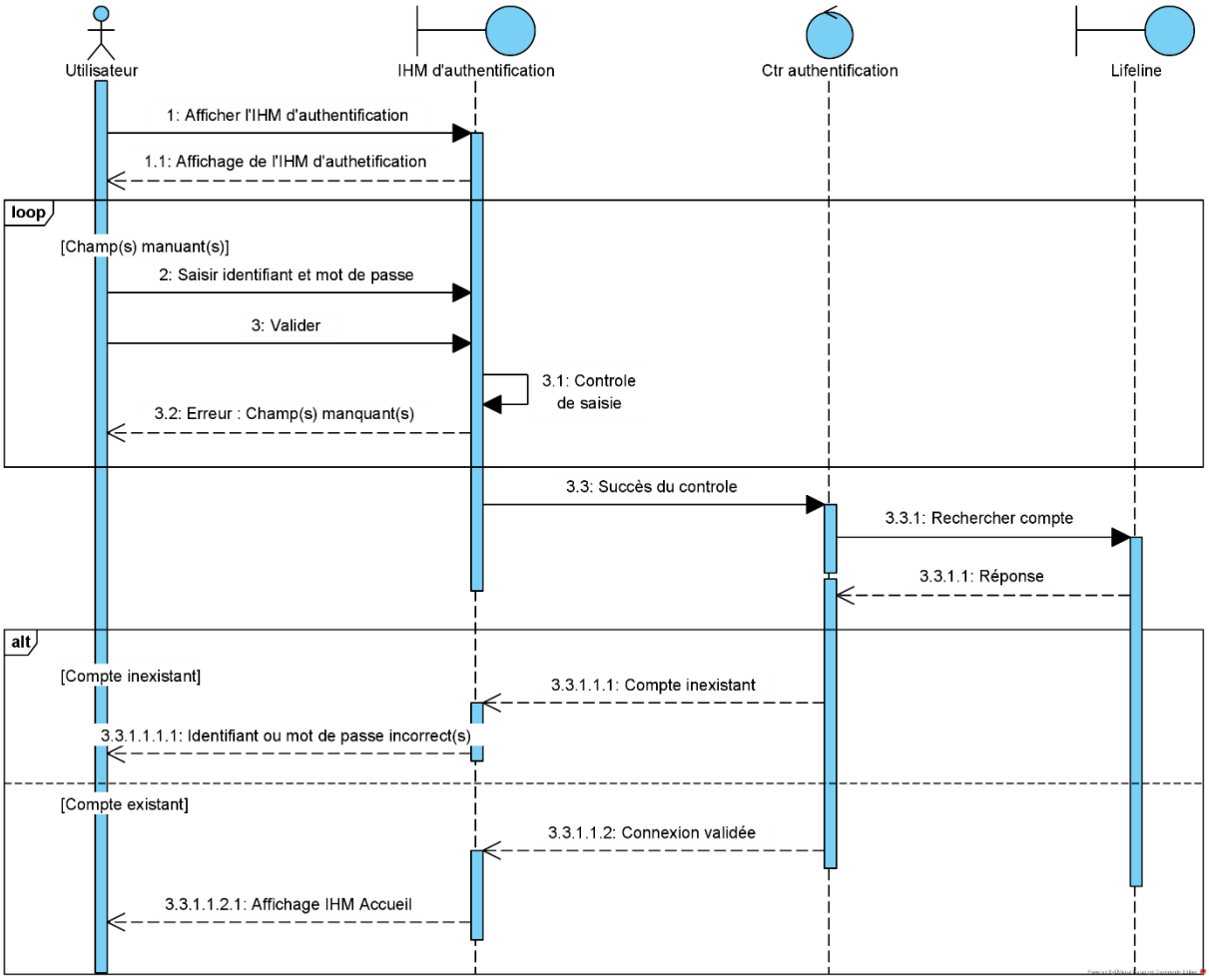


Figure 17 : Diagramme de séquences d'interaction de cas d'utilisation « S'authentifier »

IV.2.1.2- Diagramme de séquences d'interaction de cas d'utilisation « Créer poste »

Apres s'être authentifier, le chef de poste pourra effectuer l'ajout d'un poste.

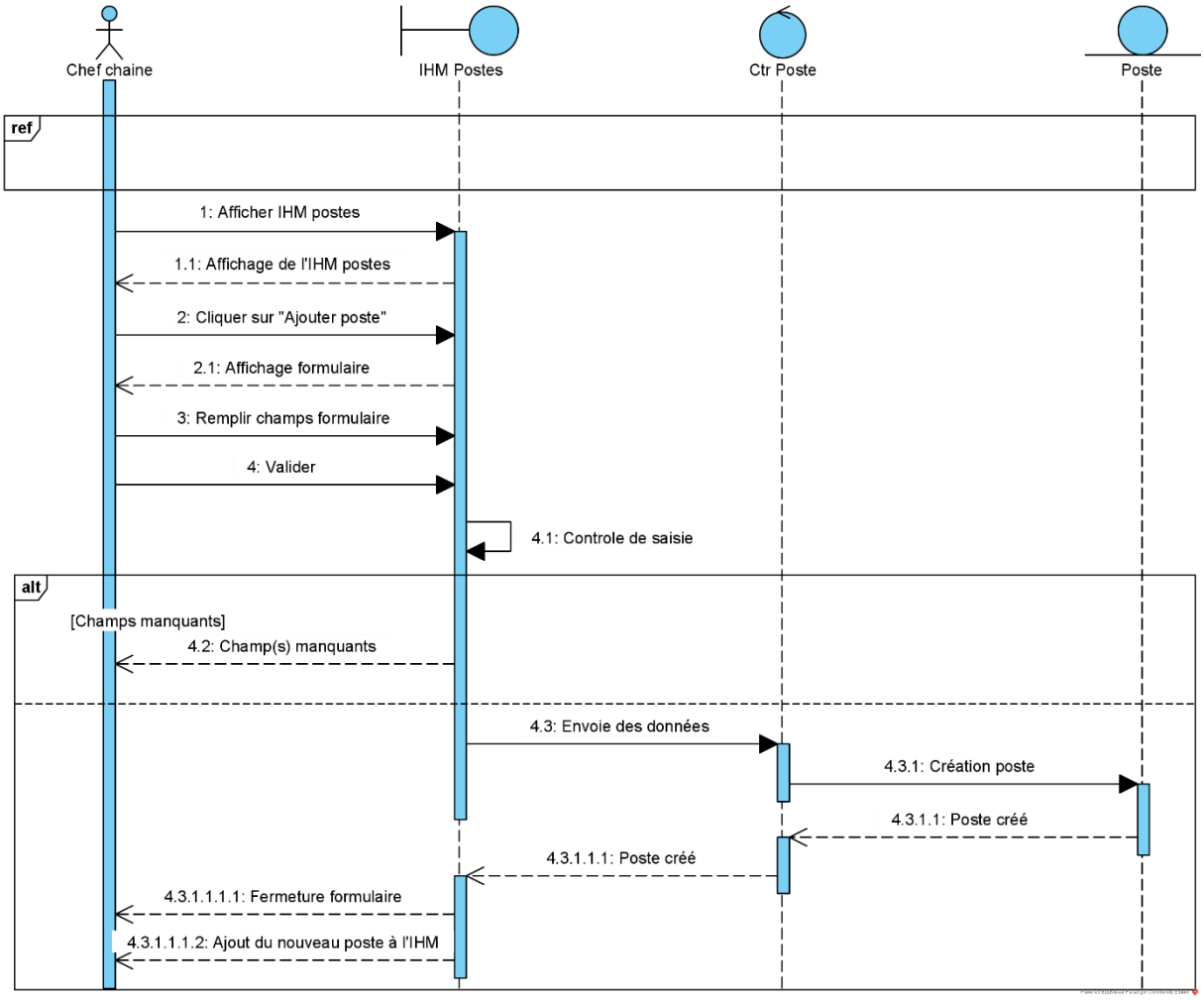


Figure 18 : Diagramme de séquences d'interaction de cas d'utilisation « Créer poste »

IV.2.1.3- Diagramme de séquences d'interactions de cas d'utilisation « Transformer produit »

Apres l'authentification le chef de poste exécute les taches préalablement affectées à son poste par le chef de chaine.

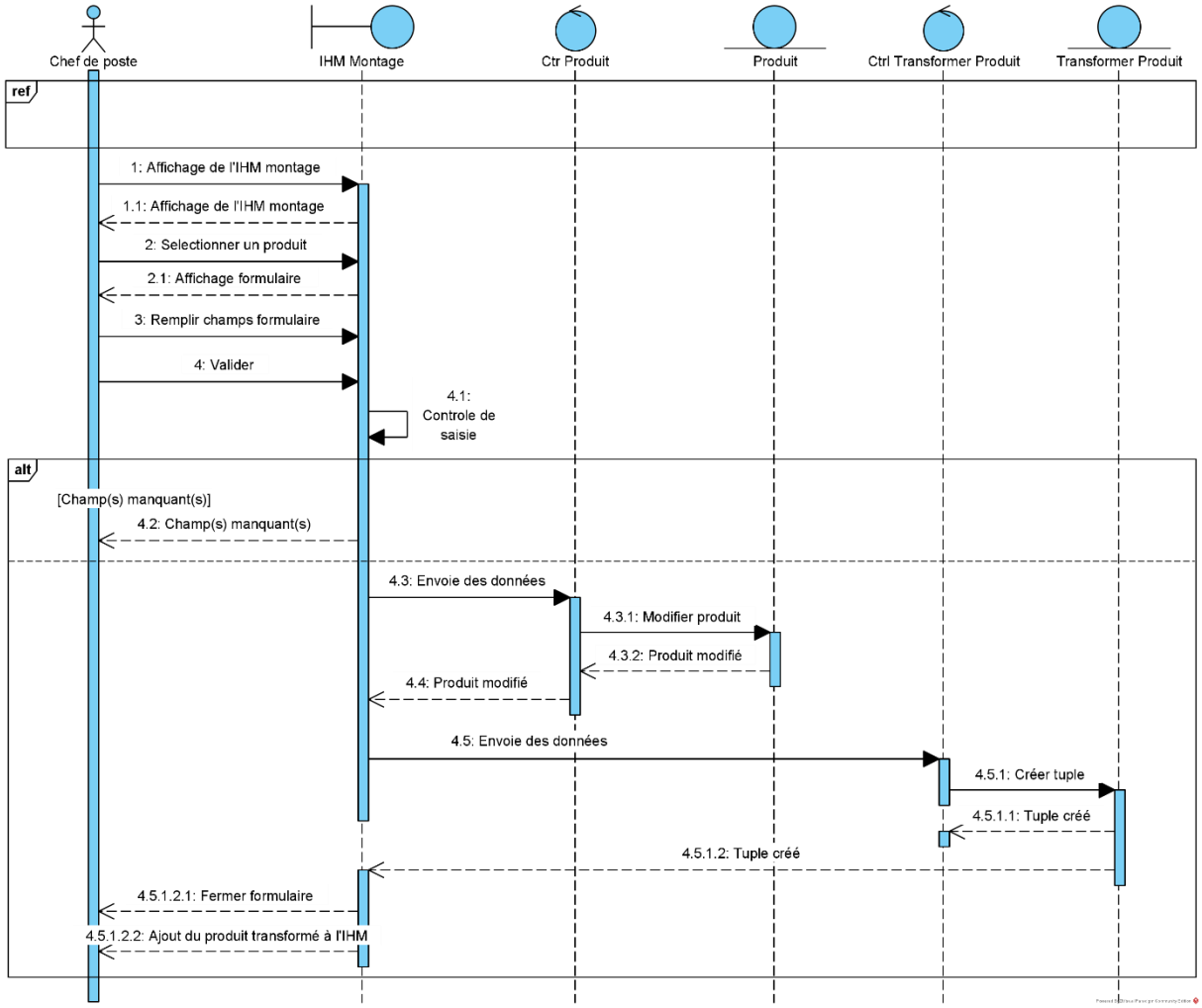


Figure 19 : Diagramme de séquences d'interactions de cas d'utilisation « Transformer produit »

IV.2.1.4- Diagramme de séquences d'interactions « Générer bon de réception »

Apres s'être authentifié l'ordonnanceur peut générer les bons de réception à partir des bons de livraison

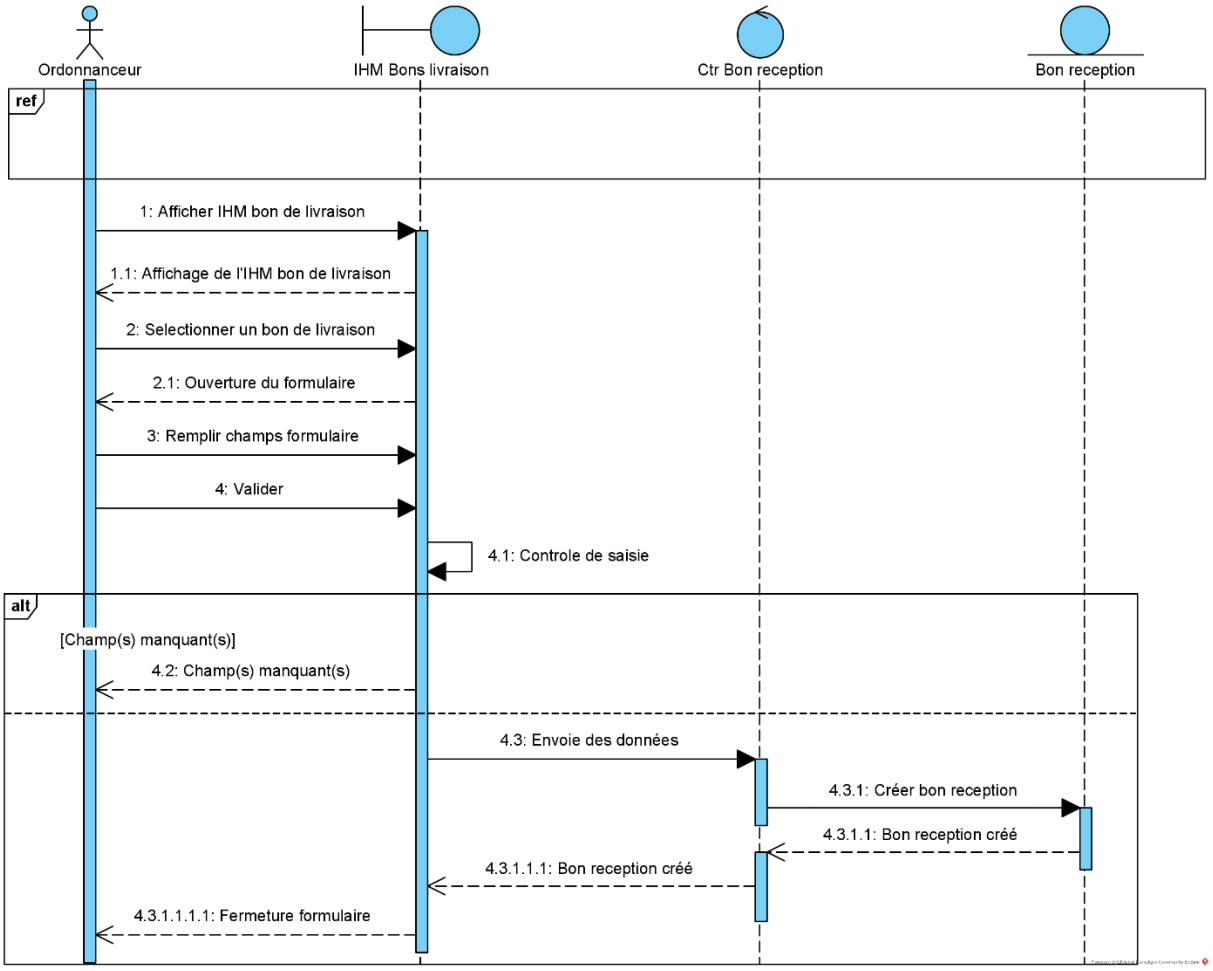


Figure 20 : Diagramme de séquences d'interactions « Générer bon de réception »

IV.3- Diagramme de classe de domaine

Le diagramme de classes est généralement considéré comme le plus important dans un développement orienté objet. Il représente l'architecture conceptuelle du système : il décrit les classes que le système utilise, ainsi que leurs liens, que ceux-ci représentent un emboîtage conceptuel (héritage) ou une relation organique (agrégation). [1]

Ci-dessous nous illustrant le diagramme de classes de notre application.

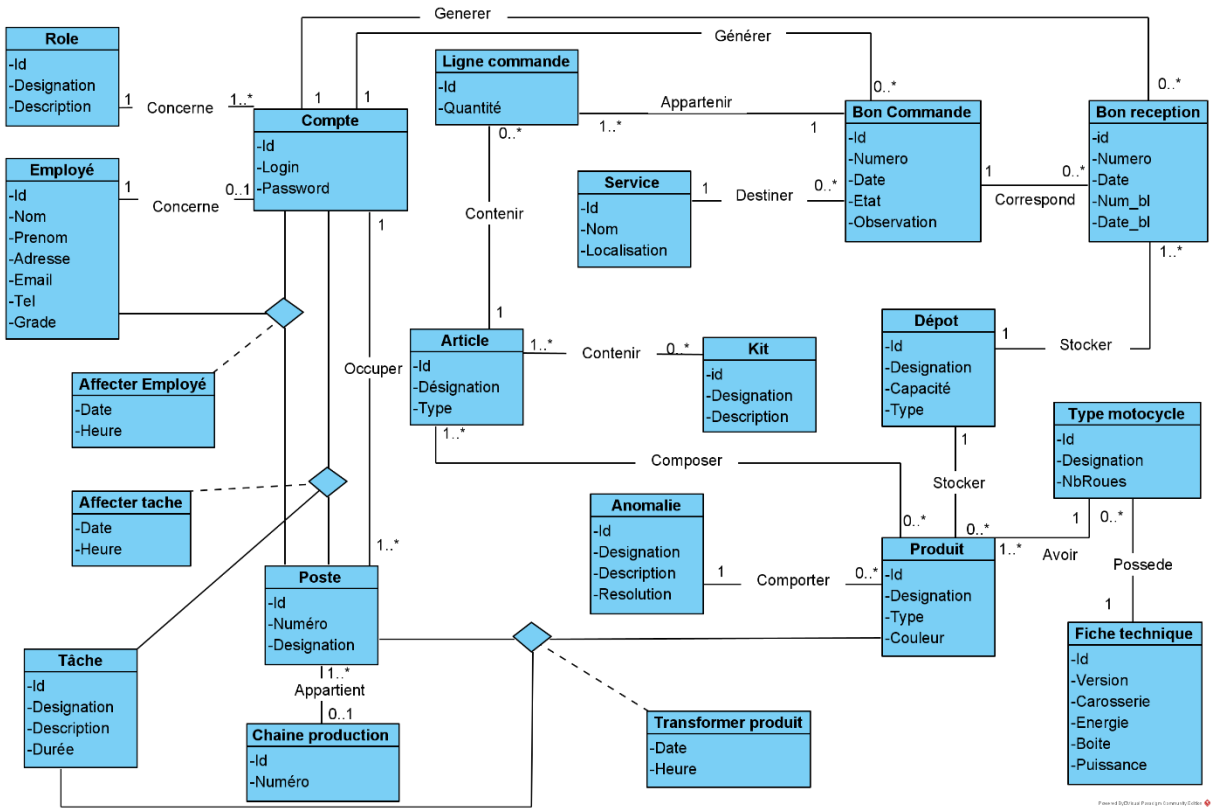


Figure 21 : Diagramme de classe de domaine

IV.3.1- Description détaillée des attributs de classes

Nous détaillerons dans ce qui suit chaque attribut de classe du diagramme de classes précédemment illustrées. Cela nous permettra de mieux comprendre ce dernier.

Ci-dessous se trouve la description des attributs qu'on a schématisés sous forme d'un tableau.

Tableau 7 : Description des classes du diagramme de classes du domaine

Classe	**Responsabilité**	**Attributs**		
		Désignation	**Signification**	**Type**
Compte	Classe qui enregistre les informations de connexion des utilisateurs.	Id	Identifiant	Numérique
		Login	Identifiant d'authentification	Chaine de caractères
		Password	Mot de passe d'authentification	Chaine de caractères
Role	Classe qui enregistre les rôles des utilisateurs.	Id	Identifiant	Numérique
		Désignation	Appellation du rôle	Chaine de caractères
		Description	Détails du rôle.	Chaine de caractères
Employé	Classe qui enregistre les informations des employés de l'entreprise.	Id	Identifiant	Numérique
		Nom	Nom de l'employé	Chaine de caractères
		Prénom	Prénom de l'employé	Chaine de caractères
		Adresse	Adresse de l'employé	Chaine de caractères
		Email	Adresse mail de l'employé	Chaine de caractères
		Tel	Numéro de téléphone de l'employé	Chaine de caractères
		Grade	Grade de l'employé au sein de l'entreprise.	Chaine de caractères
Poste	Classe qui enregistre les postes de production.	Id	Identifiant	Numérique
		Numéro		Numérique
		Désignation	Appellation du	Chaine de

			poste	caractères
Chaine production	Classe qui enregistre les chaines de production.	Id	Identifiant	Numérique
		Numéro		Numérique
Tache	Classe qui enregistre les taches des postes de production.	Id	Identifiant	Numérique
		Désignation	Appellation de la tache	Chaine de caractères
		Description	Détails de la tâche.	Chaine de caractères
		Durée	Durée de réalisation de la tâche	Numérique
Produit	Classe qui enregistre les produits de la chaine de production.	Id	Identifiant	Numérique
		Désignation	Appellation du produit.	Chaine de caractères
		Type	Le produit peut être fini ou semi-fini.	Enumération
		Couleur	Couleur du produit	Enumération
Article	Classe qui enregistre les articles (matière première).	Id	Identifiant	Numérique
		Désignation	Appellation de l'article	Chaine de caractères
		Type	L'article peut être matière première ou issu du service après-vente	Enumération
Kit	Classe qui enregistre les kits d'article.	Id	Identifiant	Numérique
		Désignation	Appellation du kit	Chaine de caractères
		Description	Détails du kit.	Chaine de caractères
Bon Commande	Classe qui enregistre les bons de commande.	Id	Identifiant	Numérique
		Numéro	Chaque bon de commande possède un numéro	Numérique

		Date	Date de création du bon de commande	Date
		Etat	Le bon de commande peut être « En cours » de livraison ou « Réceptionné »	Enumérations
		Observation	Commentaire à propos du bon de commande.	Chaine de caractères
Bon de réception	Classe qui enregistre les bons de réception.	Id	Identifiant	Numérique
		Numéro	Chaque bon de réception possède un numéro	Numérique
		NumBl	Chaque bon de livraison possède un numéro	Numérique
		DateBl	Date de création du bon de livraison	Date
		Observation	Commentaire à propos du bon de réception.	Chaine de caractères
Ligne commande	Classe qui enregistre les lignes de commande des bons de commande.	Id	Identifiant	Numérique
		Quantité	Quantité d'articles de la ligna de commande	Numérique
Service	Classe qui enregistre les services avec lesquels communique le service production.	Id	Identifiant	Numérique
		Nom	Appellation du service.	Chaine de caractères
		Localisation	Localisation du service	Chaine de caractères
Dépôt	Classe qui enregistre les dépôts du service	Id	Identifiant	Numérique
		Désignation	Appellation du dépôt.	Chaine de caractères

	production.	Capacité	Capacité du dépôt	Numérique
		Type	Un dépôt peut être soit dépôt de « Matières première » ou de « Produits finis » ou bien de « Produits semi-finis »	Enumération
Anomalie	Classe qui enregistre les anomalies les plus fréquentes de la production.	Id	Identifiant	Numérique
		Désignation	Appellation de l'anomalie.	Chaine de caractères
		Description	Détails de l'anomalie.	Chaine de caractères
		Résolution	Détails de comment résoudre l'anomalie.	Chaine de caractères
Type motocycle	Classe qui enregistre les types de motocycle que produit le service.	Id	Identifiant	Numérique
		Désignation	Appellation du type de motocycle.	Chaine de caractères
		NbreRoues	Nombre de roues	Numérique
Fiche technique	Classe qui enregistre les fiches techniques de chaque type de motocycle.	Id	Identifiant	Numérique
		Version	Version de motocycle	Numérique
		Carrosserie	Carrosserie de motocycle	Chaine de caractères
		Energie	Energie de motocycle	Chaine de caractères
		Boite	Boite de motocycle	Chaine de caractères

IV.3.2- Description détaillée des attributs des classes d'associations

Le tableau ci-dessous représente les classes d'associations et leurs attributs :

Tableau 8 : Descriptions des classes d'associations et leurs attributs.

Classes	Responsabilité	Attributs		
		Désignation	**Signification**	**Type**
Transformer produit	Classe qui enregistre toutes les transformations des produits.	IdProduit	Identifiant	Numérique
		IdPoste	Identifiant	Numérique
		IdTache	Identifiant	Numérique
		Date	Date de la transformation du produit	Date
		Heure	Heure de la transformation du produit	Date
Affecter Tache	Classe qui enregistre les affectations de taches aux postes.	IdCompte	Identifiant	Numérique
		IdPoste	Identifiant	Numérique
		IdTache	Identifiant	Numérique
		Date	Date d'affectation de la tache	Date
		Heure	Heure d'affectation de la tache	Date
Affecter employé	Classe qui enregistre les affectations d'employés aux postes.	IdCompte	Identifiant	Numérique
		IdPoste	Identifiant	Numérique
		IdEmploye	Identifiant	Numérique
		Date	Date d'affectation de l'employé	Date
		Heure	Heure d'affectation de l'employé	Date

IV.4- Schéma relationnel

Le schéma relationnel est le modèle logique de données, ce modèle décrit de façon abstraite comment sont représentées les données du diagramme de classe dans une base de données. Pour décrire une relation, nous allons indiquer tout simplement son nom, suivi du nom de ses attributs entre parenthèses. L'identifiant d'une relation est composé d'un ou plusieurs attributs qui forment la clé primaire. Une relation peut faire référence à une autre en utilisant une clé étrangère, qui correspond à la clé primaire de la relation référencée.

IV.4.1- Règles de passage au modèle relationnel

Les règles de passage au modèle relationnel sont :

- **Relation (1..*) :** il faut ajouter un attribut de type clé étrangère dans la relation fils de l'association .L'attribut aura le nom de la clé primaire de la relation père de l'association.
- **Relation (1..1) :** il faut ajouter une relation qui prend les deux clés primaires des classes mère comme clé étrangère.
- **Relation d'héritage :** Trois décompositions sont possibles pour traduire une association d'héritage en fonction des contraintes existantes :

– Décomposition par distinction : il faut transformer chaque sous-classe en une relation. La clé primaire de la classe mère, migre dans la (les) relation(s) issue(s) de la (des) sous-classe(s) et devient à la fois clé primaire et clé étrangère,

– Décomposition descendante (push-down) : Il faut faire migrer tous les attributs de la classe mère dans la (les) relation(s) issue(s) de la (des) sous classe(s),

– Décomposition ascendante (push up) : Dans ce cas on supprime les relations issues des sous classes et faire migré tous les attributs dans la relation issue de la classe mère.

IV.4.2- Le passage au modèle relationnel

Dans ce qui suit nous allons traduire notre diagramme de classes en schéma relationnel tout en respectant les règles de passage précédemment énumérés.

Employe (idEmploye, nom, prénom, grade, tel, adresse, email)

Role (idRole, désignation, description)

Compte (idCompte, login, password, idEmploye#, idRole#)

Tache (idTache, désignation, description, durée)

Poste (idPoste, numéro, désignation, idCompte#, idChaineProd#)

ChaineProduction (idChaineProd, numéro)

Produit (idProduit, désignation, type, couleur, idTypeMoto#, idDepot#, idAnomalie#)

TypeMotocycle (idTypeMoto, désignation, nbreRoues, idFicheTech#)

FicheTechnique (idFicheTech, version, carrosserie, énergie, boite)

Anomalie (idAnomalie, désignation, description, résolution)

BonCommande (idBc, numéro, date, état, observation, idCompte#, idService#)

BonReception (idBr, numéro, date, numBl, dateBl, idBc#, idDepot#)

LigneCommande (idLc, quantité, idArticle#, idBc#)

Article (idArticle, désignation, type)

Kit (idKit, désignation, description)

Service (idService, nom, localisation)

Depot (idDepot, désignation, capacité, type)

Contenir (#idArticle, #idKit)

Composer (#idArticle, #idProduit)

AffecterTache (#idCompte, #idPoste, #idTache, date, heure)

AffecterEmploye (#idCompte, #idPoste, #idEmploye, date, heure)

TransformerProduit (#idPoste, #idProduit, #idTache, date, heure)

IV.5- Conclusion

Dans ce chapitre nous avons présenté la partie conception de notre projet tout en l'illustrant avec les diagrammes de séquences d'interactions, de classe du domaine et bien évidemment avec le schéma relationnel. Tout ceci pour nous préparer à la prochaine étape qui consiste à réaliser l'application en question.

Dans le prochain chapitre, nous verrons en détaille la partie réalisation et toute la procédure et tous les outils qui nous a mené à la création de notre application.

Chapitre V

Realisation

Chapitre V

Realisation

V.1- Introduction

Dans ce chapitre, nous entamons la partie pratique, ou nous allons présenter l'environnement et les outils de développement utilisé dans le développement. Par la suite, nous illustrerons l'architecture de l'application. Et nous clôturerons ce chapitre en exposant quelques interfaces de celle-ci.

V.2- Langage et environnement de développement

Dans cette section, nous allons énumérer les différentes technologies qui sont utilisées pour développer notre système.

V.2.1- Les langages utilisés

- **Html (HyperText Markup Language)**

HTML signifie « *HyperText Markup Language* » qu'on peut traduire par « langage de balises pour l'hypertexte ». Il est utilisé afin de créer et de représenter le contenu d'une page web et sa structure. D'autres technologies sont utilisées avec HTML pour décrire la présentation d'une page (CSS) et/ou ses fonctionnalités interactives (JavaScript).

- **CSS (Cascading Style Sheets)**

CSS (Cascading Style Sheets) est un langage spéci_que au web, fréquemment employé comme complément du langage HTML, et dont la fonction et de formé des feuilles de style charger de la mise en formes des documentent web. Il gère l'esthétique (couleurs, typographie) et diverses fonctionnalités.

- **JavaScript**

JavaScript est un langage dynamique multi-paradigme : il dispose de différents types, opérateurs, objets natifs et méthodes. Sa syntaxe s'inspire des langages Java et

C et de nombreuses structures de ces langages s'appliquent également à JavaScript. JavaScript permet la programmation orientée objet avec les prototypes. JavaScript permet également la programmation fonctionnelle car ses fonctions sont des objets et on peut donc stocker ces fonctions dans des variables et les transmettre comme n'importe quel objet.

- **SQL**

Le langage SQL (Structured Query language) est un langage normalisé d'interrogation de bases de données. Puisqu'il est normalisé, il est indépendant du type des bases de données : les mêmes commandes peuvent être exploitées quelle que soit la base utilisée (Access, MySQL ...). Les commandes SQL peuvent ainsi gérer tout type d'action sur le serveur de bases de données MySQL, depuis la simple manipulation des enregistrements jusqu'à la création, modi_cation ou suppression d'une base, d'une table ou d'un champ.

- **Java**

Java est un langage de programmation et une plate-forme informatique qui ont été créés par Sun Microsystems en 1995. Beaucoup d'applications et de sites Web ne fonctionnent pas si Java n'est pas installé et leur nombre ne cesse de croître chaque jour. Java est rapide, sécurisé et fiable. Les objectifs de Java sont d'être multiplateformes et d'assurer la sécurité aussi bien pendant le développement que pendant l'utilisation d'un programme Java. Son point fort qui le démarque des autres est sa portabilité due a sa bibliothèque de classes indépendances de la plateforme ce qui est le point essentiel de la programmation sur internet ou plusieurs machines dissemblables sont interconnectées

V.2.2- Outils et bibliothèque

o React JS

React est une bibliothèque JavaScript déclarative, efficace et flexible pour construire des interfaces utilisateurs (UI). Elle vous permet de composer des UI complexes à partir de petits morceaux de code isolés appelés « composants ».

o Spring framework

Spring est le framework Java le plus populaire pour la création d'applications pour entreprises. Le framework Spring fournit un écosystème de projet riche pour répondre aux besoins des applications modernes, telles que la sécurité, l'accès simplifié au stockage de données relationnelles et NoSQL, le traitement par lots, l'intégration avec les sites de réseaux sociaux et le traitement massif de flux de données. Étant donné que Spring est un framework très flexible et personnalisable, il existe généralement plusieurs façons de configurer les applications.

o Bootstrap

Bootstrap est un framework CSS/JS sous licence MIT développé par la société californienne Twitter.Bootstrap embarque également des composants HTML et JavaScript. Il comporte un système de grille simple et efficace pour mettre en ordre l'aspect visuel d'une page web. Il apporte du style pour les boutons, les formulaires, la navigation ... Il permet ainsi de concevoir un site web rapidement et avec peu de lignes de code ajoutées.

o Axios

Axios est un client HTTP basé sur des promesses pour node.js et le navigateur. Il est isomorphe (= il peut s'exécuter dans le navigateur et nodejs avec la même base de code). Côté serveur, il utilise le module http natif node.js, tandis que sur le client (navigateur), il utilise XMLHttpRequests.

- **Eclipse**

Eclipse est un IDE, Integrated Development Environment (EDI environnement de développement intégré en français), c'est-à-dire un logiciel qui simplifie la programmation en proposant un certain nombre de raccourcis et d'aide à la programmation. Il est développé par IBM5. Il est gratuit et disponible pour la plupart des systèmes d'exploitation.

- **Visual Studio Code**

Visual Studio Code est un éditeur de code multiplateforme édité par Microsoft. Cet outil est destiné aux développeurs supporte plusieurs dizaines de langages de programmation comme le HTML, C++, PHP, Javascript, CSS, etc. parmi ces fonctionnalités la prise en charge du débogage, la mise en évidence de la syntaxe, la complétion intelligente du code, la refacturation du code et Git intégré.

V.3- Interfaces de l'application

Dans ce qui suit, nous allons présenter quelques interfaces de notre application.

V.3.1 Interface d'authentification

Cette interface permet aux utilisateurs de s'identifier afin d'accéder à leur section.

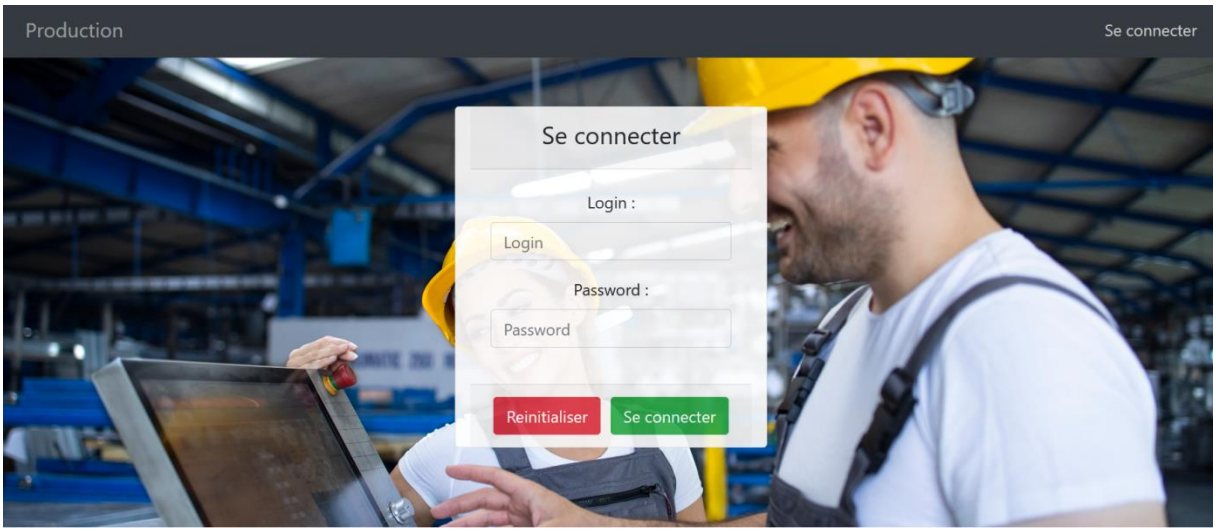


Figure 22 : Interface « S'authentifier ».

V.3.2 Interface d'accueil

Cette interface s'affiche lorsque l'utilisateur accède à sa section après s'être authentifié.

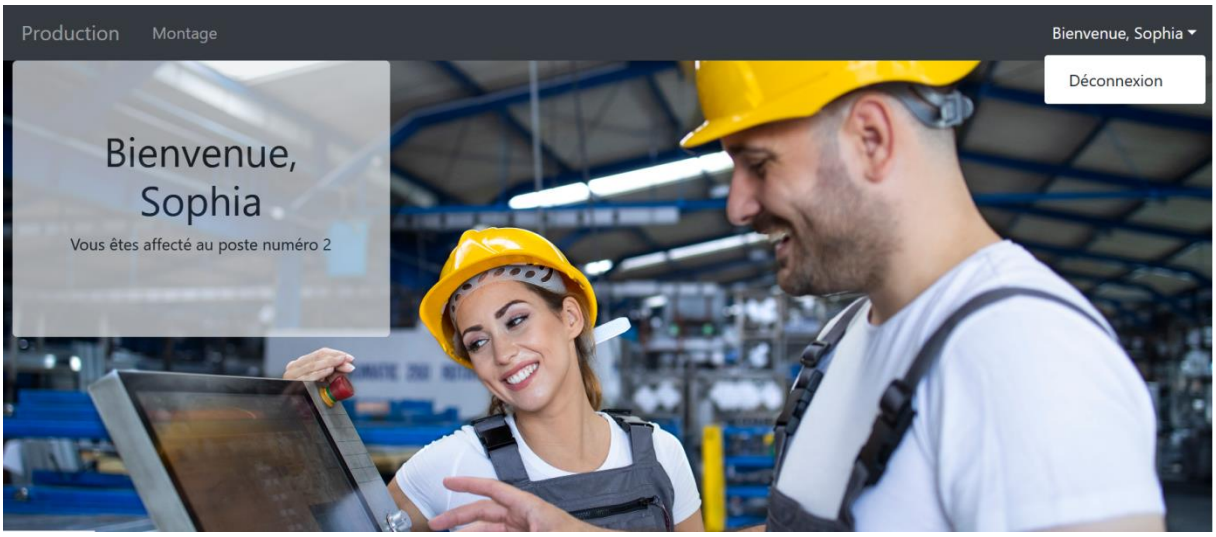


Figure 23 : Interface « Accueil » de l'utilisateur

V.3.3 L'interface de montage (Poste)

Cette interface permet aux chefs de poste d'enregistrer les montages qu'il effectue.

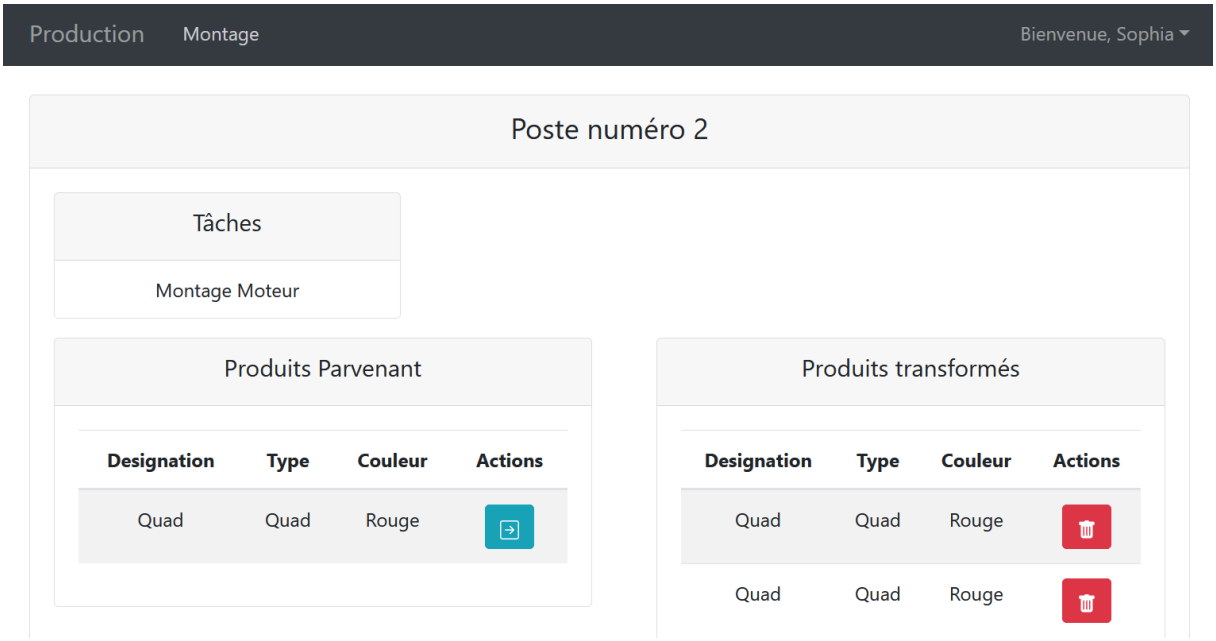


Figure 24 : Interface de montage

V.3.4 L'interface CRUD de postes

Cette interface permet d'ajouter, consulter, modifier ou supprimer (CRUD) des postes de montage.

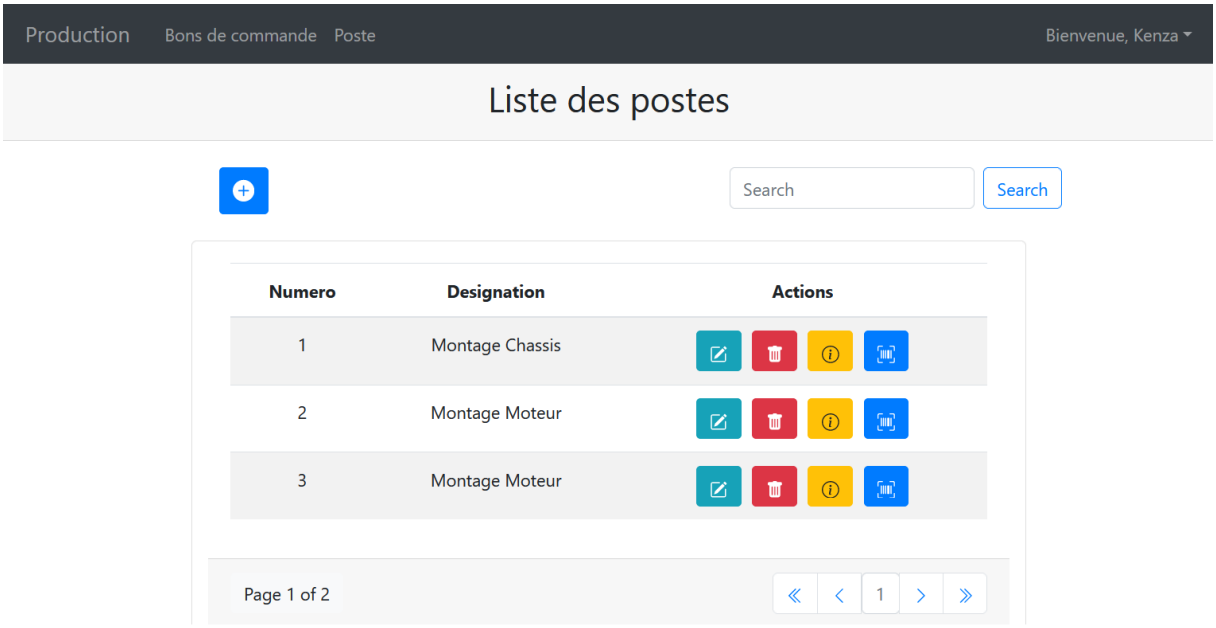


Figure 25 : Interface " Créer poste "

V.4- Conclusion

Dans ce chapitre, nous avons présenté les aspects pratiques liés à la réalisation de l'application, à savoir les langages de développement et outils de développement avec lesquels nous avons élaboré notre application. Par la suite, nous avons illustré quelques interfaces de l'application.

Conclusion Generale

Conclusion Generale

Ce projet a été une expérience très enrichissante pour nous, à travers laquelle nous avons pu étudier de près le service « production » de VMS Industrie et réalisé par la suite une application qu'on espère a satisfait leurs exigences.

La première étape de ce projet a été d'assembler les informations qui nous ont permis de comprendre le fonctionnement du service sur lequel on a travaillé qui est le service « Production ».

Par la suite, nous avons présenté la problématique et énuméré les besoins de cette entreprise afin de mieux comprendre ce dont elle a réellement besoin.

L'étape d'après est celle de l'analyse et la spécification des besoins qui a permis de révéler les différents acteurs à prendre en compte et leurs interactions avec notre système.

Ensuite, nous sommes passé à la conception et avons fixés la structure globale de l'application qui nous a permis par la suite d'entamer la partie réalisation ou l'on a présenté les outils du travail et les interfaces les plus importantes de notre application.

Pour finir, nous ne comptons pas nous arrêter ici. Plusieurs fonctionnalités sont à rajouter pour parfaire ce travail et surtout pour satisfaire un maximum notre entreprise.

« Je crois qu'on ne peut mieux vivre qu'en cherchant à devenir meilleur, ni plus agréablement qu'en ayant pleine conscience de son amélioration. » Socrate

Bibliographie

Bibliographie

[01] Laurent Audibert, UML2 De l'apprentissage à la pratique, 1ere Edition, 2006

[02] Claudio Alves, Learning React JS, 2eme Edition, 2020

[03] React, React Une bibliothèque JavaScript pour créer des interfaces utilisateurs. Disponible sur : https://fr.reactjs.org/ , Consulté le 30-06-2021.

[04] Fedosejev Artemij, React JS Essentials, 1ere Edition

[05] KAHOUADJI Wassila, KEZIZ Amel, '*Conception et réalisation d'une Application web pour la gestion des pannes informatiques*, Université de Béjaia, 2019/2020

[06] React, Glossaire des termes React. Disponible sur : https://fr.reactjs.org/docs/glossary.html , Consulté le 30-06-2021.

[07] React, Créer une nouvelle appli React. Disponible sur : https://fr.reactjs.org/docs/create-a-new-react-app.html#nextjs , Consulté le 30-06-2021.

[08] Alexis Renard, React JS pour de superbes applications web. Disponible sur : https://derniercri.io/agence-react-js , Consulté le 30-06-2021.

Résumé

Le but de notre projet est de concevoir et réaliser un module de gestion de la production au sein de l'ERP de VMS Industrie et cela avec deux technologies prisées qui sont React Js de Facebook pour le coté front-end et Spring Framework pour le coté back-end. Pour cela, nous avons présenté dans le premier chapitre ces deux technologies révolutionnaires tout en expliquant les points importants de leur fonctionnement. Nous avons ensuite enchainé avec la spécification et analyse des besoins, la conception détaillée du projet, et la réalisation de notre application a été effectuée sous l'environnement de développement Windows. Nous avons utilisé MYSQL comme serveur de base de données et le Framework Bootsrap pour l'interface.

Mots Clés: Production, Java, JavaScript, ReactJS, Spring, Spring-Boot, UML, HTML, HTTP, web, VMS.

Abstract

The goal of our project is to design and realize a production management module within the ERP of VMS Industrie and this with two technologies taken which are React Js from Facebook for the front-end side and Spring Framework for the Back-end side. For this, we presented in the first chapter these two revolutionary technologies while explaining the important points of their operation. We then proceeded with the specification and analysis of needs, the detailed design of the project, and the realization of our application was carried out under the Windows development environment. We used MYSQL as the database server and the Bootsrap Framework for the interface.

Keywords: Production, Java, JavaScript, ReactJS, Spring, Spring-Boot, UML, HTML, HTTP, web, VMS.